ABSTRACT

Title of Dissertation: TOWARDS IMMERSIVE VISUAL CONTENT WITH MACHINE LEARNING

Brandon Yushan Feng
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Amitabh Varshney
Department of Computer Science

Extended reality technology stands poised to revolutionize how we perceive, learn, and engage with our environment. However, transforming data captured in the physical world into digital content for immersive experiences continues to pose challenges. In this dissertation, I present my research on employing machine learning algorithms to enhance the generation and representation of immersive visual data.

Firstly, I address the issue of recovering depth information from videos captured using 360-degree cameras. I propose a novel technique that unifies the representation of object depth and surface normal utilizing double quaternions. Experimental results demonstrate that training with a double-quaternion-based loss function improves the prediction accuracy of a neural network using 360-degree video frames as input.

Secondly, I examine the problem of efficiently representing 4D light fields using the emerging concept of neural fields. Light fields hold significant potential for immersive visual applications; however, their widespread adoption is hindered by the substantial cost associated with storing and transmitting such high-dimensional data.

I propose a novel approach for representing light fields. Deviating from previous approaches, I treat the light field data as a mapping function from pixel coordinates to color and train a neural network to accurately learn this mapping function. This functional representation enables high-quality interpolation and super-resolution for light fields while achieving state-of-the-art results in light field compression.

Thirdly, I present neural subspaces for light fields. I adapt the ideas of subspace learning and tracking and identify the conceptual relationship between neural representations of light fields and the framework of subspace learning. My method considers a light field as an aggregate of local segments or multiple local neural subspaces. A set of local neural networks are trained to encode each subset of viewpoints. Since each local network specializes in a specific region, this specialization allows for smaller networks without compromising accuracy.

Fourthly, I introduce primary ray-based implicit function to represent geometric shapes. Traditional implicit shape representations, such as the signed distance function, describe a shape by its relationship to each spatial point. Such a point-based representation of shapes often necessitates the costly iterative sphere tracing to render a surface hit point. I propose a ray-based approach to implicit neural shape modeling, wherein the shape is implicitly described by its relationship with each ray in the 3D space. To render the hit point, my method only requires a single inference pass, considerably reducing the computational cost of rendering.

Lastly, I describe a technique to generate novel view rendering without relying on any 3D structure or camera pose information. I harness the power of neural fields to encode individual images without estimating their camera poses. My method

learns a latent code for each image in the multi-view collection, and then produces plausible and photorealistic novel view renderings by interpolating their latent codes. This entirely 3D-agnostic approach avoids the computational cost incurred by 3D representations, offering a promising outlook on employing image-based neural fields for image manipulation tasks beyond fitting and super-resolving known images.

TOWARDS IMMERSIVE VISUAL CONTENT WITH MACHINE LEARNING

by

Brandon Yushan Feng

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Amitabh Varshney, Chair/Advisor
Assistant Professor Furong Huang
Assistant Professor Christopher A. Metzler
Associate Professor Jia-Bin Huang
Professor Joseph F. JaJa

© Copyright by
Brandon Yushan Feng
2023

Acknowledgments

My first thanks go to my dissertation advisor, Amitabh Varshney. My doctoral studies have been incredibly intellectually rewarding and enlightening due to his thoughtful guidance and constant encouragement. Having the opportunity to conduct my doctoral training under his tutelage has been both a privilege and a pleasure

I appreciate the time spent by all my committee members, namely Amitabh Varshney, Furong Huang, Christopher A. Metzler, Jia-Bin Huang, and Joseph F. JaJa. They all served on my committee, offered valuable suggestions for improvement, and asked insightful questions during my dissertation defense.

Beyond the work on this dissertation, I am privileged to have enjoyed fruitful collaborations with various research groups at UMD and beyond. I learned much about matrix sketching from Furong Huang. I was fortunate to work with Brian Pierce on studying antibody-antigen docking algorithms. I owe much of my knowledge of and joy in studying computational imaging and optics to Chris Metzler. I gained invaluable lessons on formulating research problems and managing projects from my interaction with Jia-Bin Huang. I also gathered countless insights and valuable experience from my many project collaborators, including (in roughly chronological order) Susmija Jabbireddy, David Li, Rui Yin, Tahseen Rabbani, Mingyang Xie, Haiyun Guo, Vivek Boominathan, Manoj K. Sharma, Ashok Veeraraghavan, Ruofei Du, Zhenyi He, Keru Wang, Yinda Zhang, Danhang Tang, Zhiwen Fan, Chenxin Li, Sazan Mahbub, Hadi AlZayer, Kevin Zhang, Michael Rubinstein, and William T. Freeman. I thank the essential support provided by many staff members at UMIACS

and UMD CS, including Tom Hurst, Jonathan Heagerty, Barbara Brawn, Sida Li, Eric Lee, Vivian Lu, Janice Perrone, and Jodie Gray.

I thank all my family, friends, and mentors for their support and impact over the years. My parents have always believed in me and would do anything for me to succeed. I feel fortunate that my grandmother is still around to celebrate my achievements, and for the blessings given by my other grandparents who have passed. I am grateful to Erzhen for the unwavering support and encouragement during this time. I thank Xue Feng, Lu Feng, and Marc Santugini for their kindness and indispensable guidance when I applied for doctoral programs. I thank Kaiming for the friendship that has powered us through all the ups and downs of our academic endeavors. I am especially thankful to Gregory C. Robbins for instilling grit in me and changing the way I read, write, and think. Greg is my best teacher, and what he taught me during my teenage years have shaped the person I am today.

Most of the work on this dissertation occurred during an aberrant and unsettling time, marked by the pandemic and its repercussions. The unprecedented and unexpected challenges of this era have made me more determined and resilient. I must acknowledge the unique challenges faced by international students, and I offer my sympathies to those unable to begin or complete their studies during this period due to obstacles and distress caused by various contingent social and geopolitical factors. I hope that future students will not have to endure similar circumstances.

Table of Contents

Acknowledgements ii

Table of Contents iv

1 Introduction 1
- 1.1 Deep Depth Estimation on 360-degree Images with a Double Quaternion Loss . . . 1
- 1.2 Efficient Neural Representation for Light Fields . . . 4
- 1.3 Neural Subspaces for Light Fields . . . 8
- 1.4 Primary Ray-based Implicit Function . . . 12
- 1.5 View Interpolation with Implicit Neural Representations of Images . . . 16

2 Deep Depth Estimation on 360-degree Images with a Double Quaternion Loss 19
- 2.1 Introduction . . . 19
- 2.2 Related Work . . . 23
  - 2.2.1 Depth Estimation on 360-degree Images . . . 23
  - 2.2.2 Joint Estimation of Depth and Normal . . . 25
  - 2.2.3 Use of Quaternions . . . 26
- 2.3 Method . . . 27
  - 2.3.1 CNN Architecture . . . 28
  - 2.3.2 Depth Refinement based on Normal . . . 28
  - 2.3.3 Aggregation with Confidence Scores . . . 30
  - 2.3.4 Double Quaternion Approximation of Depth and Normal in the Loss function . . . 32
    - 2.3.4.1 Constructing Double Quaternion . . . 32
    - 2.3.4.2 Loss Function Based on Double Quaternions . . . 34
  - 2.3.5 Stereo Consistency . . . 34
  - 2.3.6 Overall Loss Function . . . 35
- 2.4 Experiments . . . 35
  - 2.4.1 Training Details . . . 36
  - 2.4.2 Comparison with Other Methods . . . 38
  - 2.4.3 Ablation Studies . . . 38
- 2.5 Limitations and Conclusion . . . 39

3 Efficient Neural Representation for Light Fields 43
3.1 Introduction . . . 43
3.2 Related Work . . . 46
3.3 Overview . . . 49
3.3.1 Light Fields as Functions . . . 49
3.3.2 Function Approximations . . . 51
3.3.3 MLP for Approximation . . . 52
3.3.4 Towards Multi-dimensional Input . . . 53
3.3.5 Gegenbauer Basis . . . 55
3.4 Methods . . . 55
3.4.1 Proposed Framework . . . 55
3.4.2 Comparative Evaluation . . . 56
3.4.3 Data and Training Setup . . . 58
3.5 Results . . . 59
3.5.1 Static Light Field Reconstruction . . . 59
3.5.2 Extension to Light Field Videos . . . 62
3.5.3 Light Field Super-Resolution . . . 62
3.5.4 Ablation Studies . . . 64
3.6 Discussion and Limitations . . . 67
3.7 Conclusion . . . 68

4 Neural Subspaces for Light Fields 69
4.1 Introduction . . . 69
4.1.1 Scope . . . 74
4.2 Related Work . . . 74
4.2.1 Light Field Compression for Streaming . . . 74
4.2.2 Neural Light Field . . . 76
4.2.3 Subspace Learning and Tracking . . . 78
4.3 Method . . . 79
4.3.1 Neural Light Fields with MLP . . . 79
4.3.2 Constructing Light Field Segments . . . 80
4.3.3 Adaptive Weight Sharing in MLP . . . 81
4.3.4 Soft-Classification for RGB Prediction . . . 83
4.4 Experiments . . . 84
4.4.1 Training Setup . . . 85
4.4.2 Quantitative Metrics . . . 86
4.4.3 Sub-aperture Light Fields . . . 87
4.4.3.1 Comparison with the Residual Approach . . . 88
4.4.3.2 Improved Accuracy with Soft Classification . . . 88
4.4.4 Volumetric Light Fields . . . 89
4.4.4.1 Extension to Dynamic Content . . . 95
4.4.4.2 Possible Flickering Across Subspaces . . . 96
4.4.5 Hyper-parameter Analysis . . . 96
4.5 Discussions . . . 97
4.5.1 End-to-end versus Two-stage Learning . . . 97

4.5.2 Limitations . . . 98
4.5.3 Future directions . . . 99
4.6 Conclusion . . . 99

5 Primary Ray-based Implicit Function 103
5.1 Introduction . . . 103
5.2 Related Work . . . 106
5.2.1 3D Shape Representations . . . 107
5.2.1.1 Functional Representations. . . . 107
5.2.1.2 Global versus Local Representations . . . 108
5.2.2 Ray-based Neural Networks . . . 109
5.3 Method . . . 110
5.3.1 Background . . . 111
5.3.2 Describing Geometry with Perpendicular Foot . . . 112
5.3.3 Background Mask . . . 113
5.3.4 Outlier Points Removal . . . 114
5.4 Experiments . . . 115
5.4.1 Single Shape Representation . . . 116
5.4.2 Shape Generation . . . 117
5.4.3 Shape Denoising and Completion . . . 119
5.4.4 Analysis and Ablations . . . 120
5.4.4.1 Complexity Analysis . . . 120
5.4.4.2 Stress Testing . . . 123
5.4.4.3 Ablations . . . 123
5.4.5 Further Applications . . . 123
5.4.5.1 Learning Camera Poses . . . 123
5.4.5.2 Neural Rendering with Color . . . 125
5.5 Additional Details and Discussion . . . 126
5.5.1 Training Setup . . . 126
5.5.1.1 Optimization . . . 126
5.5.1.2 Network Architecture . . . 126
5.5.2 Runtime Discussions . . . 127
5.5.2.1 Rendering Speed . . . 127
5.5.2.2 Meshing Speed . . . 128
5.5.3 More Ablations . . . 128
5.5.3.1 Comparison to Plücker Coordinates . . . 128
5.5.3.2 Varying Model Complexity . . . 129
5.5.3.3 Different Noise Levels . . . 129
5.5.3.4 Limitation . . . 130
5.6 Conclusion . . . 130

6 View Interpolation with Implicit Neural Representations of Images 131
6.1 Introduction . . . 131
6.2 Related Work . . . 134
6.2.1 Implicit Neural Representations . . . 134

6.2.1.1 3D Reconstruction . . . 135
6.2.1.2 Image Fitting . . . 135
6.2.2 Image-based Rendering . . . 136
6.3 Method . . . 137
6.3.1 INR for Image Fitting . . . 137
6.3.2 Extension to Multiple Images . . . 138
6.3.3 Direct Regularization . . . 139
6.3.4 Indirect Regularization . . . 141
6.4 Experiments . . . 145
6.5 Discussion . . . 147
6.6 Supplementary Information . . . 150
6.6.1 Training Details . . . 150
6.6.1.1 Hyperparameters . . . 150
6.6.1.2 Training with CLIP-based Features . . . 151
6.6.1.3 Dataset Details . . . 152
6.6.1.4 Beyond Interpolating Between Two Views . . . 153
6.6.1.5 Extending to Frame Interpolation. . . . 154
6.7 Conclusion . . . 163

7 Conclusion and Future Work 164

Bibliography 166

Chapter 1: Introduction

In this dissertation, I present a series of innovative machine learning algorithms designed to enhance the creation and representation of immersive visual data for extended reality applications. Firstly, I propose a method that simultaneously addresses depth and normal information to improve depth predictions from 360-degree videos. Secondly, I construct a streamlined technique for representing 4D light fields using neural fields, also known as implicit neural representations (INR). Thirdly, I put forward the concept of neural subspaces to optimize the quality and efficiency of neural light field representations. Fourthly, I introduce the idea of primary ray-based implicit functions for effective geometric shape modeling. Lastly, I explore a 3D-agnostic approach using image-based neural fields to generate novel view renderings without 3D structure or camera pose data. In this chapter, I provide brief overview of these advances before detailing them in later chapters.

1.1 Deep Depth Estimation on 360-degree Images with a Double Quaternion Loss

Depth information for 360° content enables 3D rendering based on the viewer's position and allows scene editing effects like relighting and object insertion. Obtaining

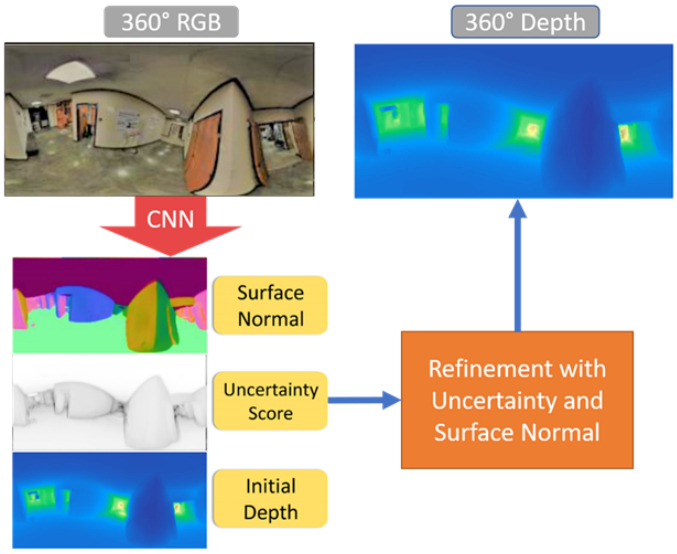


Figure 1.1: Method Overview. The convolutional neural network (CNN) takes in a 360° image and for each pixel estimates its depth, normal, and uncertainty of that depth estimate. These three estimates are used by a refinement module to produce the final depth estimate for each pixel. I train the CNN using a novel loss based on the double quaternion representation of the depth and normal.

depth information for 360° videos has several solutions, such as adding an active depth sensor or using stereo correspondence. However, both methods are cumbersome, costly, and challenging for average users. Recent deep learning advances have enabled a data-driven approach to the 360° depth problem, demonstrating that a neural network can predict 360° depth with monocular input only. In Chapter 2, I present a novel method to improve existing deep learning methods on monocular 360° depth estimation, as illustrated by Figure 2.1. The proposed method unifies the representations of depth and normal based on the concept of double quaternions.

My method enables the conversion of predicted and ground-truth depth and

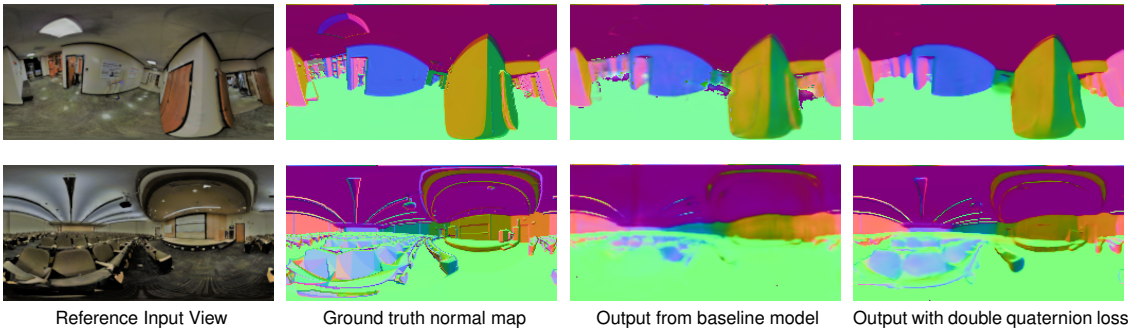


Figure 1.2: Surface Normal Prediction. Training with double quaternion loss enables the network to produce better normal estimates. In the third column, I show predictions from the baseline model which separately calculates depth and normal loss without combining them into a double quaternion form.

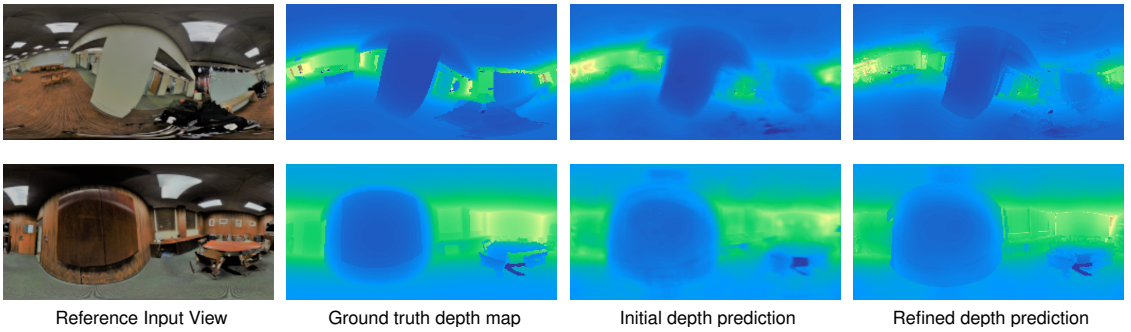


Figure 1.3: Depth Refinement Results. I compare initial depth estimates produced by the network and the refined output based on surface normal. In the first column, I show the input image for reference.

normals into two double quaternions, from which I derive a new loss for joint depth and surface normal estimation. I also use the double quaternion representation to measure the discrepancy between two CNN estimates from a stereo image pair. Experimental results show that training with a double-quaternion-based loss function improves prediction accuracy for neural networks with 360-degree video frames as

Method	RMSE	Log10	AbsRel	δ_1	δ_2	δ_3
UResNet [1]	2.037	0.326	16.906	0.213	0.399	0.560
RectNet [1]	1.738	0.291	16.132	0.240	0.453	0.634
FCRN [2]	0.672	0.101	7.448	0.806	0.932	0.966
PSMNet [3]	0.393	0.059	5.641	0.953	0.975	0.980
SepUNet [4]	0.495	0.042	1.779	0.944	**0.987**	**0.993**
SepUNetS [4]	0.614	0.072	1.841	0.835	0.966	0.985
SepUNetDD [5]	0.392	0.036	2.120	**0.960**	0.987	0.992
Ours	**0.389**	**0.031**	**0.413**	0.954	0.984	0.990

Table 1.1: Performance Comparison on the ODS dataset [4]. Evaluation statistics for row 1-7 are directly taken from Lai *et al.* [4] and Xie *et al.* [5]. My method produces superior results in most metrics.

input. In Figure 1.2, I show normal estimates produced by our trained network. In Figure 1.3, I demonstrate the benefit of refining the initial depth predictions with predicted normals. My method outperforms state-of-the-art methods in most metrics as shown in Table 1.1.

1.2 Efficient Neural Representation for Light Fields

Light field content holds great potential for immersive visual applications. However, a primary obstacle to widespread adoption is the extreme cost of storing and transmitting such high-dimensional data. Past research has proposed numerous

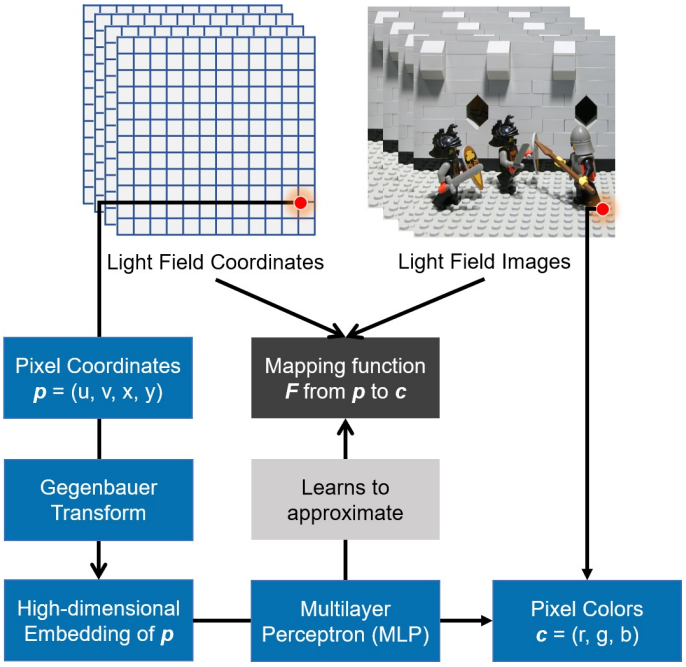


Figure 1.4: Overview of SIGNET. I train a MLP to approximate the mapping function from each pixel's coordinates to its color values. My input transformation strategy based on the Gegenbauer polynomials enables the MLP to more accurately learn the high-dimensional mapping function.

methods to compress light fields, but these efforts have limited success in making light field content compact enough for casual streaming.

In Chapter 3, I introduce SIGNET, a novel approach for representing light fields with neural networks, as illustrated in Figure 1.4. Instead of treating light fields as pixel collections, I consider them as functions mapping pixel coordinates to colors. I also introduce a new transformation based on Gegenbauer polynomials for input coordinates, enabling our network to successfully represent dense light fields.

The ability to accurately represent a huge light field with a compact neural network automatically achieves compression with high fidelity. As shown in Ta-

Table 1.2: Compression Performance Compared to Other Methods. *Size* denotes the storage in megabytes(MB) for each method without further quantizations.

	Static Light Fields						Light Field Videos			
Scene	Lego		Bracelet		Tarot		Painter		Trains	
Method	Size	PSNR	Size	PSNR	Size	PSNR	Size	PSNR	Size	PSNR
SIGNET	**9.0**	**41.26**	**12.0**	38.70	**9.0**	37.47	**144**	**39.56**	**144**	**39.73**
AMDE [6]	29.3	40.90	18.1	**39.90**	44.2	38.54	941	38.25	809	37.00
KSVD [7]	29.3	38.39	18.1	36.73	44.3	**38.81**	942	38.12	807	35.06
HOSVD [6]	29.3	37.24	18.0	33.98	44.3	34.53	942	36.91	807	35.29
5D DCT [6]	29.4	37.29	18.1	32.31	44.2	33.03	941	36.79	807	35.20
CDF 9/7 [8]	29.0	33.71	18.2	31.98	44.3	29.17	941	31.69	1116	29.80

ble 1.2, SIGNET outperforms other compression methods on multiple light field scenes. Moreover, such a functional representation easily achieves interpolation and super-resolution on light fields. In Figure 1.5, I demonstrate the result of spatial upsampling with SIGNET, achieved by evaluating the network on a denser set of coordinates. SIGNET produces acceptable output without being trained for this task. In Figure 1.6, I present SIGNET's performance on angular upsampling, or novel view synthesis. In Figure 1.7, I present the result of temporal upsampling with SIGNET. Although SIGNET has only been trained on content from the first frame at t_0 and the third frame at $t_0 + 1$, when evaluated at the unseen intermediate time step $t_0 + \frac{1}{2}$, SIGNET is able to preserve the motion trajectory.

SIGNET's functional design allows random access to encoded pixels, while

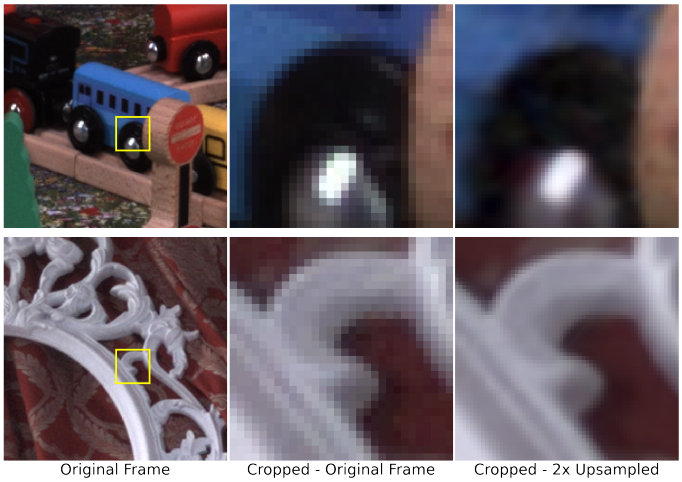


Figure 1.5: Spatial Upsampling with SIGNET. I evaluate the trained SIGNET on dense sampling grid points in the spatial dimensions. I show zoomed-in details in the cropped region bounded by the yellow rectangle.

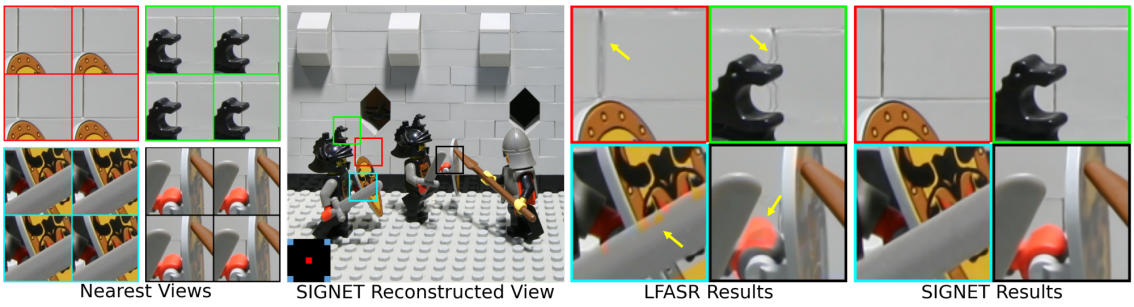


Figure 1.6: Angular Upsampling with SIGNET. At the bottom left corner of the reconstructed view, I show the relative positions of the reconstructed view (red square) and its four nearest views (blue squares) in the original light field. I compare with results from the deep-learning-based method, LFASR [9], which is trained specifically for light field angular upsampling.

traditional codecs like JPEG or MPEG require decoding an entire image patch to access a single pixel. I envision that this property would be highly beneficial for foveated rendering, enabling substantial rendering and streaming speedups by

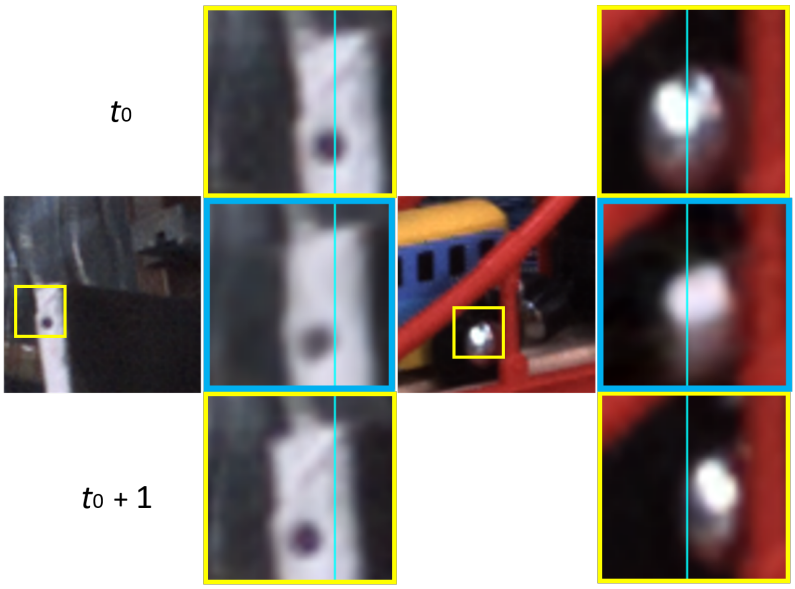


Figure 1.7: Temporal Upsampling with SIGNET. t_0 and $t_0 + 1$ are consecutive frames in the original video. The blue boxes contain output from frames evaluated at $t_0 + \frac{1}{2}$, which is not present in the original video. The vertical lines are drawn for easier observation of the motion trajectory.

adaptively selecting pixel portions based on the viewer's gaze location.

1.3 Neural Subspaces for Light Fields

SIGNET is a neural field, also known as implicit neural representation (INR), that compactly encodes multiple viewpoints from a light field scene (hundreds of megabytes) into a single neural network's weights (a few megabytes). These weights are the only necessary information for storage and streaming. This unified design of training a single neural network to cover the entire light field has simplistic appeal as a concept. However, it is not ideal in practical scenarios that emphasize efficient transmission and rendering. Such scenarios generally involve only rendering a subset

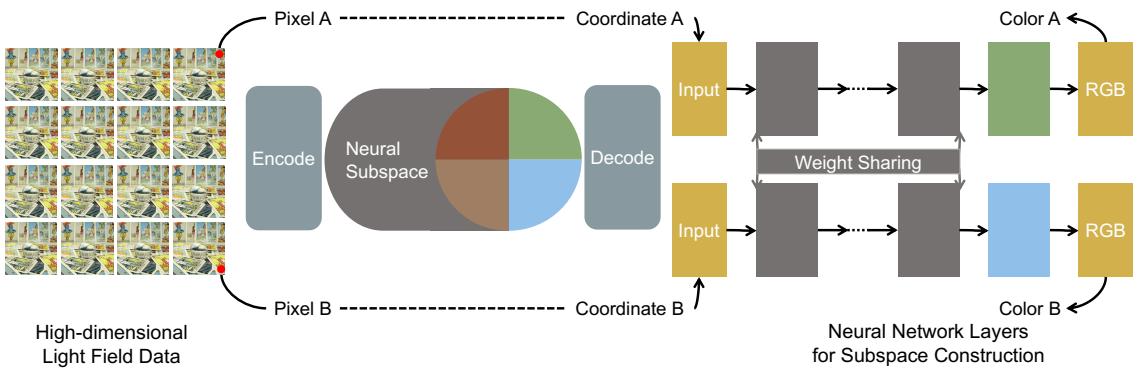


Figure 1.8: Concept of Neural Subspaces for Light Fields. Given a light field scene, I divide it into multiple local segments and construct a neural subspace for each segment. The neural subspace construction is equivalent to training the network parameters to learn accurate coordinate-to-color mappings within each segment. The adaptive weight sharing strategy utilizes the similarity among nearby subspaces and reduces the total number of parameters to represent the entire light field.

of light field viewpoints. Therefore, unnecessary costs are incurred to transmit and evaluate a single neural network that contains other unrelated views.

In Chapter 4, I discuss the conceptual connection between the neural representation of light fields and subspace learning, a signal processing concept for dimensionality reduction of high-dimensional data. Inspired by subspace learning, I show that it is not necessary to treat light fields only as one unified entity - light fields can be regarded as a composite collection of local segments or neural subspaces. Such a perspective is meaningful in practice since only a subset of the light field might be relevant at a particular moment for streaming or rendering.

The approach introduced in Chapter 4 trains a set of local neural networks that

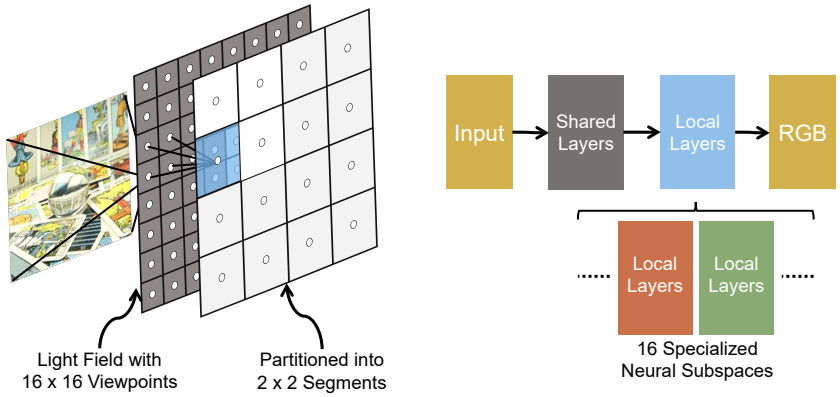


Figure 1.9: Illustration of the Weight Sharing Strategy. The light field is partitioned into segments, each containing 2×2 viewpoints. I construct a neural subspace for each segment that summarizes its pixel-to-color mapping relationship. Each subspace shares a set of network layers while possessing its local layers that enable network specialization on its corresponding data segment.

each encodes only a subset of viewpoints, unlike SIGNET which is one global network representing the entire light field. An overview of this approach is illustrated in Figure 1.8. As each local network specializes in a particular region, this specialization permits smaller networks without sacrificing accuracy. Furthermore, recognizing the similarity among nearby subspaces, I propose a weight-sharing strategy for those local networks to enhance overall parameter efficiency while maintaining network capacity within each subspace. Effectively, this proposed strategy, illustrated in Figure 1.9, achieves the tracking of implicit neural subspaces. As shown in Table 4.1, experimental results indicate that the proposed framework leads to better efficiency and accuracy than the original SIGNET and a range of previous methods. With this work invoking the classic idea of subspace learning, I take neural light field

Scene	Method	PSNR(↑)	SSIM(↑)	EncMB(↓)	DecMB(↓)
Lego 909 MB	AMDE	40.90	0.973	29.3	29.3
	KSVD	38.39	0.960	29.3	29.3
	SIGNET	41.26	0.976	9.0	9.0
	Ours	41.95	0.982	22.9	2.2
Tarot 909 MB	AMDE	38.54	0.973	44.2	44.2
	KSVD	38.81	0.980	44.3	44.3
	SIGNET	37.47	0.975	9.0	9.0
	Ours	38.21	0.975	22.9	2.2
Bracelet 568 MB	AMDE	39.90	0.980	18.1	18.1
	KSVD	36.73	0.973	18.1	18.1
	SIGNET	38.70	0.973	12.0	12.0
	Ours	39.64	0.985	22.9	2.2

Table 1.3: We provide results on sub-aperture light fields. Comparisons to previous methods AMDE [6], KSVD [7], and SIGNET [10]. *DecMB* is the memory (in MB) required in streaming to decode any frame, and *EncMB* is the memory (in MB) required to encode the entire light field in storage.

representation introduced by SIGNET to the next step by making this representation more compact and streaming-friendly.

1.4 Primary Ray-based Implicit Function

Learning accurate and efficient 3D object representations is vital for applications in graphics, vision, and robotics. Recent advances in machine learning involve training neural fields of signed distance functions (SDF) as implicit shape representations. However, rendering and extracting shapes from trained SDF networks can be computationally expensive and typically limited to watertight shapes. Moreover, the shape quality is ultimately constrained by the converging criteria of sphere tracing or grid resolution of the marching cubes extraction.

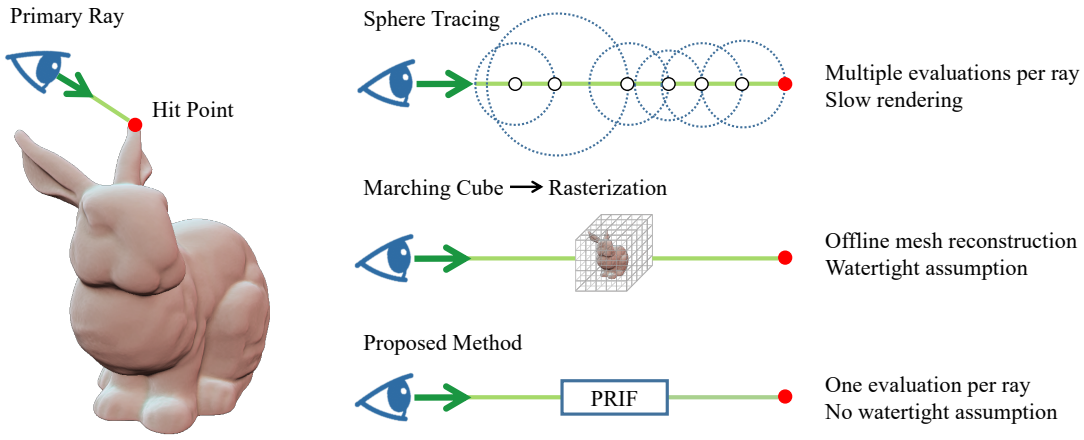


Figure 1.10: Comparing various implicit shape representations. Rendering from common implicit neural shape representations, such as signed distance functions (SDF) and occupancy functions (OF), requires either sphere tracing or rasterizing a separately extracted mesh. My new representation, PRIF, directly maps each primary ray to its hit point. A network encoding PRIF is more efficient and convenient for rendering, since it requires only one evaluation for each ray, avoids the watertight constraint in conventional methods, and easily enables differentiable rendering.

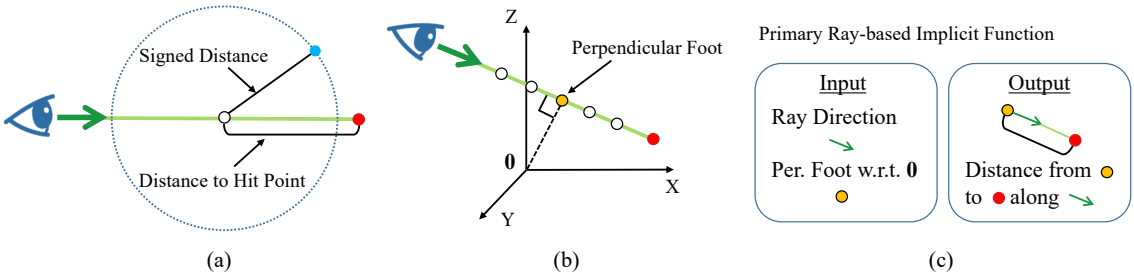


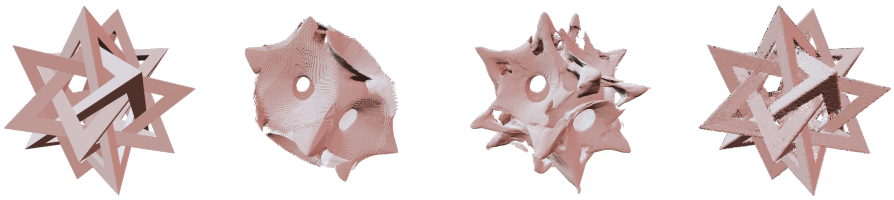
Figure 1.11: Formulation of PRIF. (a) Signed distance at a sampling position (white) reveals the sphere (blue dots) where its nearest surface point (blue) exists, when we really want to know the hit point (red) *along a specific direction*. Thus, multiple samples are required. (b) To obtain the surface hit point, PRIF uses only one sample (yellow) along the ray: the perpendicular foot between the given ray and the coordinate system's origin $\mathbf{O}$. (c) PRIF takes in the ray's direction and its sampling point, and returns the distance from that point to the actual surface hit point.

Chapter 5 presents a novel implicit geometric representation that is efficient, accurate, and innately compatible for downstream tasks involving reconstruction and rendering. I break away from the conventional point-wise implicit functions like SDF, and propose to encode 3D geometry into a ray-based implicit function called PRIF. Specifically, PRIF operates on the realm of oriented rays $r = (\mathbf{p}_r, \mathbf{d}_r)$, where $\mathbf{p}_r \in \mathbb{R}^3$ is the ray origin and $\mathbf{d}_r \in \mathbb{S}^2$ is the normalized ray direction. Unlike SDF that only outputs the distance to the nearest but undetermined surface point, I formulate this representation such that its output directly reveals the surface hit point of the input ray. Figure 1.10 presents the overview of PRIF.

Figure 1.11 shows the formulation of PRIF. I train an MLP to learn $\Phi(\mathbf{f}_r, \mathbf{d}_r) = s_r$. In effect, the objective is equivalent to finding a simple affine transformation

Method	Armadillo	Bunny	Buddha	Dragon	Lucy
SDF	1.905|1.260	1.717|1.147	6.119|2.258	5.184|1.946	3.387|1.417
OF	4.805|1.624	1.704|1.133	17.279|3.113	19.577|3.014	3.396|1.427
PRIF	**0.978|0.706**	**1.169|0.835**	**1.443|0.821**	**1.586|0.913**	**0.846|0.519**

Table 1.4: Quantitative results on single shape representation. The left and right numbers represent the mean and median Chamfer Distance (multiplied by 10^{-4}).



Ground Truth OF SDF Ours - Points

Figure 1.12: Comparing PRIF with SDF and OF. I test on a Tetrahedron grid that is self-intersecting and non-watertight. I obtain the SDF and OF values based on the ground truth geometry, and extract the mesh by Marching Cubes. While the level-set representations fail, PRIF reliably preserves the shape.

$f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, with the input $\mathbf{x} = \mathbf{d}_r$, $\mathbf{A} = s_r I_3$, and $\mathbf{b} = \mathbf{f}_r$. I also avoid a major limitation in previous sphere-tracing-based methods, which is having to sample multiple points and perform multiple network evaluations to obtain a hit point.

Chapter 5 also presents various experiments that verify the efficacy of PRIF for shape representation and demonstrate the applications enabled by using PRIF as the underlying neural shape representation. Table 1.4 and Figure 1.12 offer a preview

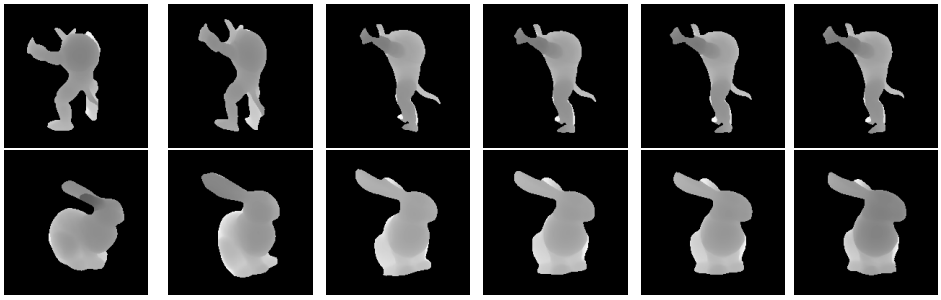


Figure 1.13: Learning Camera Poses. The *Initial* camera pose is optimized based on the difference between PRIF output rendered at the current pose and the PRIF output rendered at an unknown *Target* pose. The PRIF output is depth images. The camera pose gradually converges to the correct *Target* pose.

showing that PRIF significantly outperform SDF and OF in accurately preserving the fine details of the 3D shapes. Figure 1.13 show the successful recovery of camera pose enabled by using PRIF as a hitpoint renderer.

In summary, Chapter 5 introduces PRIF, a new 3D shape representation based on the relationship between a ray and its perpendicular foot with the origin. I demonstrate that neural networks can successfully encode PRIF to achieve accurate shape representations. This new representation avoids multi-sample sphere tracing and obtain the hit point with a single network evaluation. Neural networks trained to encode PRIF inherit such advantages and can represent shapes more accurately than common neural shape representations using the same network architecture.

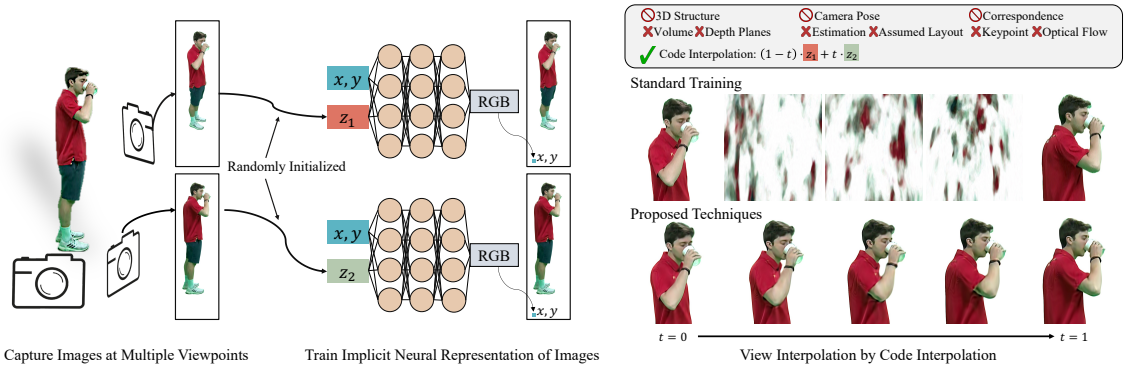


Figure 1.14: Overview of VIINTER. After each image is randomly assigned a code vector z, the codes are then jointly trained with the neural network to produce the RGB color given coordinate (x, y). With standard training, the INR fails to decode coherent images at new interpolated codes, but VIINTER enables smooth transition between two known viewpoints. Contrary to common methods for view interpolation, VIINTER does not use 3D structure, camera poses, or pixel correspondence.

1.5 View Interpolation with Implicit Neural Representations of Images

Neural fields, also known as implicit neural representations (INR), have been successful in representing visual signals such as images, videos, signed distance fields, and radiance fields. In scenarios where only 2D images are available, two prominent applications are 2D image fitting and 3D view synthesis. INRs achieve impressive visual results along these two orthogonal directions. On the one hand, the quality of fitting images is improved by incorporating traditional signal processing techniques. On the other hand, the quality of view synthesis is improved by augmenting INRs with well-established 3D graphics techniques.

In Chapter 6, I explore a different direction and ask a new question: Given

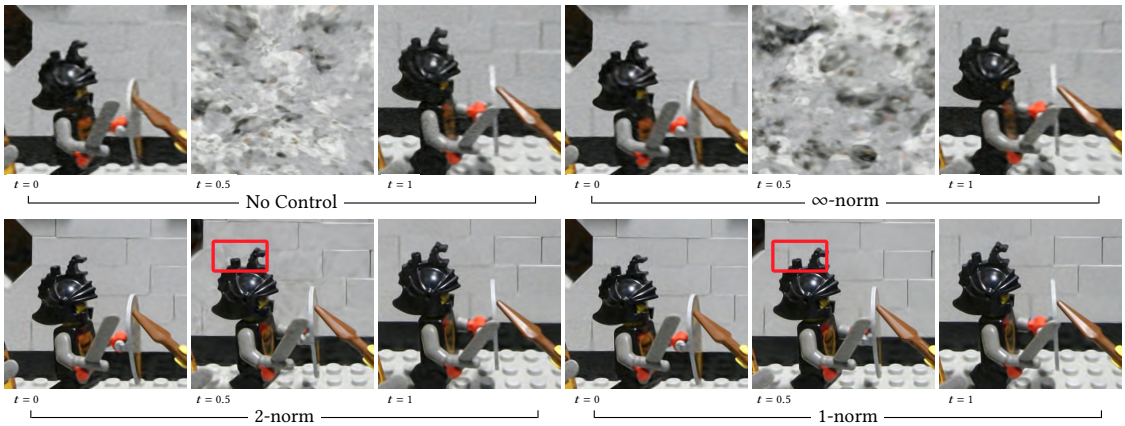


Figure 1.15: Effect of Controlling Latent Codes $\|z\|_p = 1$ with Different p-norm. For each condition, we show the INR output given z_i (left), $0.5z_i + 0.5z_j$ (center), and z_j (right). "No Control" does not control the latent codes, leading to proper reconstruction at known views (left and right) but complete failure in interpolation (center). "∞-norm" scales each z with its maximum norm, but still does not interpolate well. "2-norm" significantly improves interpolation and reconstructs known views better, but "1-norm" is much better at interpolation (see red boxes).

multiple 2D image views of a 3D scene, can we use the INR of those 2D images alone to do view synthesis without any 3D reconstruction, pose, or correspondence? With randomly initialized INR weights and code vectors for individual images, I modify the standard INR training process such that the trained INR can both faithfully reproduce the given images and synthesize plausible novel views when interpolating between those learned image codes. This method is called VIINTER, and its overview is shown in Figure 1.14.

Chapter 6 includes various analysis and experiments. For example, I show that it is important to regularize the magnitude of latent codes, as shown in Figure 1.15.



Figure 1.16: Unstructured Light Field Results. We interpolate learned codes of views i and j as $(1-t)\cdot z_i + t \cdot z_j$. The camera movement between the two known views includes rotation and translation. The interpolation through INR smoothly transforms the perspective, despite having no knowledge of 3D scene structure or camera pose. Images are zoomed in for easier evaluations.

Chapter 6 further includes evaluation results of VIINTER on different types of multi-view scenes. Example results are presented in Figure 1.16.

VIINTER takes an important step toward revealing new potential of neural fields or INRs. With careful modifications, they can perform view interpolation without 3D structures. This work offers a promising outlook on employing them for image manipulation tasks beyond simply representing known images.

Chapter 2: Deep Depth Estimation on 360-degree Images with a Double Quaternion Loss

2.1 Introduction

Traditional depth estimation uses binocular or multi-view stereo image inputs [11–14]. Based on explicit geometric constraints, most of these stereo methods infer relative depth through computing stereo disparity, i.e., the distance between a pixel's location in one image to its corresponding location in the other image. The rise of deep learning enables direct training of convolutional neural networks (CNN) for depth estimation by implicitly computing the matching cost between pixels in stereo images.

However, since stereo images are not easily accessible, depth estimation on monocular images serves as a valuable alternative. Deep CNNs for this problem have shown promising results. A unique advantage for this approach is that monocular CNNs can be trained on both monocular image datasets and stereo image datasets.

As virtual and augmented reality (VR and AR) become more commoditized and panoramic cameras become ubiquitous, 360° visual content is becoming more relevant [15–17]. The interactive nature of VR and AR, fosters an urgent need for

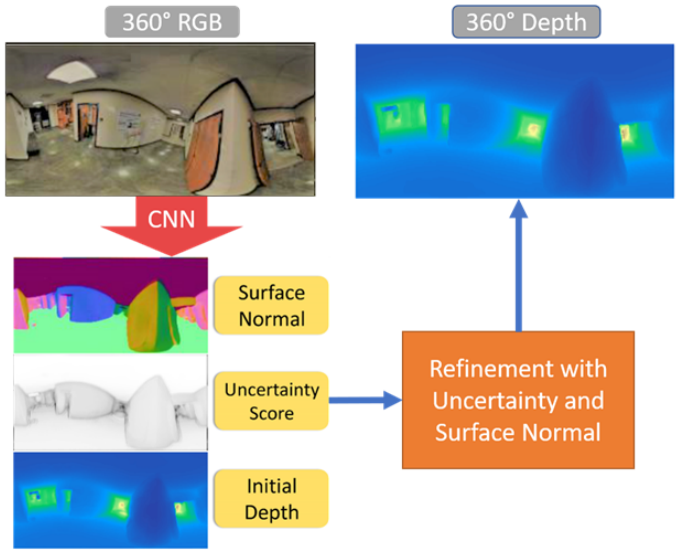


Figure 2.1: Method Overview. The CNN takes in a 360° image and for each pixel estimates its depth, normal, and uncertainty of that depth estimate. These three estimates are used by a refinement module to produce the final depth estimate for each pixel. We train the CNN using a novel loss based on the double quaternion representation of the depth and normal.

methods that estimate depth information from 2D views to instill more creative freedom in content rendering and interaction, including reconstructing the original 3D scenes and synthesizing views from novel angles [18, 19]. However, most of the previous research on depth estimation targets traditional perspective images.

Unlike typical photographs captured on a planar sensor, 360° images have a spherical layout. For 360° stereo images, traditional depth-estimation methods based on binocular disparity are not directly applicable due to the spherical singularity at the stereo epipoles. Moreover, CNNs trained on narrow-field-of-view images for monocular depth estimation perform poorly on 360° monocular images because

of the significant domain shift from traditional perspective to wide-field-of-view equirectangular images.

Zioulis *et al.* [20] and Lai *et al.* [4] have recently released separate datasets for depth estimation on 360° images. While both datasets provide multi-view stereo images, their choices of baseline distance between cameras are vastly different. This difference signifies a severe drawback of training a CNN that simply takes in a stereo image pair: networks directly trained on stereo images with a particular baseline cannot adapt to different baseline configurations at test time. Moreover, such networks require a fixed baseline in training, making it difficult to aggregate training data from multiple datasets. Therefore, training a depth estimation CNN with monocular input seems more favorable.

Among methods that train CNN for depth estimation, joint estimation of depth and normal is commonly adopted as an augmentation technique. However, to the best of our knowledge, all previous networks that jointly estimate normal and depth consider the errors from depth and normal separately. While Qi *et al.* [21] and Yang *et al.* [22] have proposed depth refinement methods that explicitly link surface normal estimates with depth estimates, their methods are based on the planar sensor camera model for traditional narrow-field-of-view images and do not map well to 360° images. Moreover, their refinement procedures modify all pixel points uniformly in the estimated depth map and do not consider the varying quality across different regions.

In this chapter, we present a new framework for 360° depth estimation. We start from a generic CNN that jointly estimates depth and surface normal based

on monocular RGB images. We develop a new loss for this joint estimation task, which combines depth and surface normals into a 4D hyperspherical space with a double quaternion approximation. We implement depth refinement using the normal estimates produced by this network. In contrast with previous normal-based refinement methods on perspective images, our new method adaptively adjusts the refinement to the initial depth estimates by an uncertainty score map which is also estimated by the CNN. This uncertainty construct allows us to identify image regions where further refinement could be helpful and avoid unnecessary changes to estimates that the network expects to be accurate. Furthermore, to make full use of available image data, we introduce a stereo loss when training the CNN on stereo-image pairs. After producing two separate monocular estimates of depth and normal for a stereo image pair, the CNN learns to minimize their hyperspherical angular difference. By this design, the monocular network can take advantage of stereo training data without being restricted by a particular stereo baseline distance. Experiments show the improved performance of our proposed framework compared to previous methods on 360° depth estimation.

In summary, our contributions include:

- An adaptive depth refinement framework for 360° images using normal estimates and uncertainty scores.
- A new way to incorporate depth and surface normal estimates for a 3D point into a hyperspherical 4D space using a double quaternion approximation.
- A stereo loss that enables the CNN to learn stereo consistency and remain

flexible across datasets with different stereo baseline distances.

2.2 Related Work

We first present learning-based methods for monocular and stereo depth estimation on 360° images, followed by previous work on using the surface normal to refine depth from perspective images. We then present previous approaches that incorporate quaternion representations in estimating surface normals and approximating 3D motions.

2.2.1 Depth Estimation on 360-degree Images

Several methods have been used to perform depth estimation [2, 23–29] and surface normal estimation [28, 30–32] on perspective images. Unfortunately, 360° images are distorted by equirectangular projection and contain irregular disparity pattern due to the spherical singularity at the stereo epipoles. Therefore, depth estimation on 360° images requires special adaptations.

One approach for learning on 360° images is to project pixels onto rectified cubemaps and then perform inference using pre-trained CNNs. Huang *et al.* [33] apply the traditional structure-from-motion (SfM) algorithm [34] in 3D scene reconstruction by projecting each 360° video frame onto a cubemap. Monroy *et al.* [35] obtain 360° saliency maps following this approach, but the distortion and discontinuity among cubemap patches are not handled by their method. Cube padding [36, 37] was introduced to help resolve cubemap distortion problem by padding each patch

with features from adjacent cubemap patches.

Another approach for 360° depth estimation is to transfer models for perspective images to 360° images. To account for the distortion from equirectangular projection, Su and Graumann [38] modified a CNN trained on perspective images by varying the kernel shape based on its location on the sphere. Su and Graumann [39] improve the previous method by learning a transformation function for kernels pre-trained on perspective images without separately training new kernels for each location. Zioulis *et al.* [1] directly train CNNs on 360° images using rectangular kernels of varying resolutions along with traditional square kernels to cover different distortion levels. They also adopt dilated convolutions [40] to increase the receptive field and enable the networks to gather more global information. Lee *et al.* [41] use a spherical polyhedron to represent 360° images and devise special convolution and pooling kernels for image pixels after they are projected on the polyhedron. Tateno *et al.* [42] deform the kernel sampling grid to compensate for distortions in spherical images. For the similar task of saliency detection on 360° videos, Zhang *et al.* [43] also define kernels on the 360° sphere and resample the kernels on the grid points for every location in the equirectangular projection.

Unsupervised learning through view synthesis has also been exploited to solve depth estimation [22, 44]. De La Garanderie *et al.* [45] use the stereo consistency of perspective images to achieve unsupervised depth estimation on panoramic images. Wang *et al.* [37] explore self-supervised depth estimation from 360° images through cubemap projection. Zioulis *et al.* [20] introduced the view-synthesis approach into the realm of omnidirectional 360° images. Aware of the distortion problem of 360°

images, they also adaptively weight the loss contribution of each pixel based on its coordinates on the image grid.

While most previous work on 360° depth estimation focuses on monocular input, Lai *et al.* [4] present a framework for stereo depth estimation on 360° images with a CNN which produces a depth map for a horizontally displaced pair of images. Xie *et al.* [5] further extend this stereo depth estimation framework to include deformable convolution and correlation convolution. Wang *et al.* [46] propose a learnable cost volume approach for spherical stereo depth estimation which also shows promising results.

2.2.2 Joint Estimation of Depth and Normal

Motivated by the inherent geometric relationship between depth and normal estimates of points on the same surface, several methods include the surface normal information into depth estimation. Wang *et al.* [47] deploy a dense conditional random field on initial estimates of normal and depth, which produces more regularized depth and normal outputs with better geometric consistency. Eigen and Fergus [48] also simultaneously estimate depth, surface normal, and semantic segmentation for perspective images.

Furthermore, the depth-normal relationship can be explicitly constructed. Two spatially close points with similar surface normal estimates are approximately co-planar, and thus they form a vector that is orthogonal to the surface normal. Building upon this assumption, Qi *et al.* [21] introduce a module that refines the

depth estimates produced by a CNN using its normal estimates. Likewise, Yang *et al.* [22] formulate this depth-normal relationship as a quadratic minimization problem for a set of linear equations constructed by the local depth and normal estimates in a small region. However, these methods do not consider the varying quality of CNN estimates across different regions.

Lai *et al.* [4] also use the information of surface normal to improve depth estimation. To the best of our knowledge, this is the first work that implements a joint estimation of depth and normal on 360° images. However, their method only includes normal as an auxiliary task of the CNN, without further exploiting the explicit geometric relationship between depth and surface normal.

2.2.3 Use of Quaternions

Quaternions are widely used in computer graphics to represent rotation transformation of 3D points. By representing surface normal as a pure quaternion, Karakottas *et al.* [49] calculate the angular loss of normal predictions based on the quaternion product of the estimated and ground-truth normal vectors.

As a natural extension to quaternions, double quaternions integrate the rotation and translation components for motion interpolation [50]. Unlike traditional 3D point representation where spatial displacements are separately characterized into translation and rotation, double quaternions provide a unified framework to approximate 3D displacements as a rotation in the 4D space. In other words, the difference between two 3D spatial displacements can be described by their angular

distance in 4D.

In this chapter, we introduce a method that directly unifies depth and surface normal information into a single measurement based on a double quaternion approximation. With this novel construct, the predicted and the ground-truth depth and normals can be converted into two double quaternions. We thus derive a new loss specifically for joint estimation of depth and surface normals.

We also take advantage of this double quaternion representation to measure the discrepancy between two CNN estimates from a stereo image pair. After transforming the two separate estimates into a homogeneous coordinate system, we derive a stereo loss based on the double quaternion angular distance between these two sets of estimates.

2.3 Method

Our goal is to train a CNN for 360° depth estimation. To exploit the information from surface normals, the CNN produces a normal map and an uncertainty map for initial depth estimates, which we feed into a refinement procedure to produce a final depth map. We derive a loss function based on double quaternions to facilitate better depth-normal joint learning. To further use datasets containing stereo pairs, we introduce a stereo loss also based on double quaternions.

2.3.1 CNN Architecture

We adopt the commonly used U-Net architecture with skip connections, as shown in Figure 2.1. For an input RGB image of size $h \times w \times 3$, the CNN produces three separate outputs: 1) $h \times w \times 1$ depth map, 2) $h \times w \times 1$ uncertainty map for depth, and 3) $h \times w \times 3$ normal map. These three output maps are fed into a refinement step detailed in Section 2.3.2.

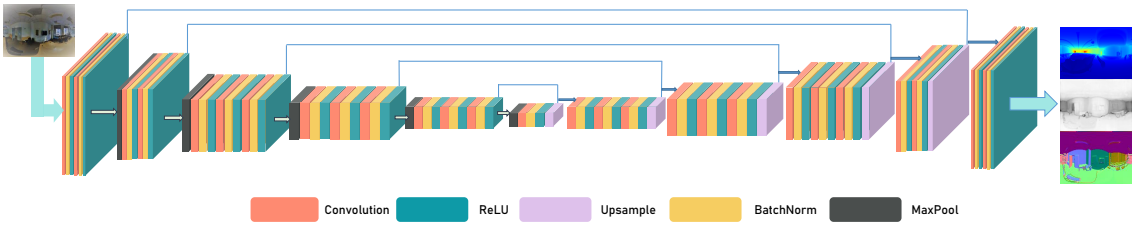


Figure 2.2: Network Architecture. We adopt the commonly used U-Net architecture for end-to-end per-pixel estimation. The first six blocks in the encoding part are based on the VGG-16 model [51]. The decoding part is symmetric to the encoding part, and it outputs a depth map, an uncertainty map, and a normal map. These three maps are combined to produce a final depth map using the method described in Section 2.3

2.3.2 Depth Refinement based on Normal

In general, image-based depth estimation aims to recover the depth value of a 3D point (x, y, z) given its projected pixel location (u, v) in an image.

The depth value for a pixel in a 360° image is defined as the distance of its

corresponding 3D point from the camera.

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.1}$$

Moreover, the pixel coordinates (u, v) of a 360° image with width w and height h directly correspond to the spherical coordinates (θ, ϕ) of its corresponding 3D point.

$$\phi = 2\pi u \qquad \theta = \pi \frac{2v - 1}{2} \qquad u, v \in [0, 1] \tag{2.2}$$

The direct conversion between spherical and Cartesian coordinates in 3D is given as follows

$$x = r \sin\phi \sin\theta \qquad y = r \cos\theta \qquad z = r \cos\phi \sin\theta \tag{2.3}$$

Using equations (2.2) and (2.3), we can obtain the relationship that maps 2D grid coordinates to 3D Cartesian coordinates for 360° depth maps.

Using the normal estimates (n_{ix}, n_{iy}, n_{iz}) also produced by the CNN, we can further formulate the following equation based on the orthogonality between surface normal vector and in-plane vector among points (x_i, y_i, z_i) and (x_j, y_j, z_j):

$$n_{ix}(x - x_i) + n_{iy}(y - y_i) + n_{iz}(z - z_i) = 0 \tag{2.4}$$

$$\frac{n_{ix}x_j + n_{iy}y_j + n_{iz}z_j}{n_{ix}x_i + n_{iy}y_i + n_{iz}z_i} = 1 \tag{2.5}$$

Then, using an assumption similar to Qi *et al.* [21], for pixels within a small region, we treat their corresponding 3D points as co-planar if their surface normal estimates are also similar. Thus, we obtain an approximately co-planar neighborhood N_i for each image pixel P_i using spatio-angular measures defined as follows:

$$N_i = \{(x_j, y_j, z_j) \mid n_j^\intercal n_i > \alpha, |u_i - u_j| < \beta, |v_i - v_j| < \beta\} \tag{2.6}$$

where (u_i, v_i) and (u_j, v_j) are the 2D grid coordinates of pixels P_i and P_j, β is the parameter that controls the size of the spatial neighborhood, and α controls the size of the angular neighborhood. A larger value of $n_j^\intercal n_i$ implies a greater likelihood that the corresponding 3D points for P_i and P_j are co-planar.

For each neighbor $P_j \in N_i$, we may obtain an estimate r_{ij} for the depth of r_i of P_i by plugging in the spherical coordinates with equations (2.3) and (2.5):

$$r_{ij} = \frac{n_{ix}x_j + n_{iy}y_j + n_{iz}z_j}{n_{ix}\sin\phi_i \sin\theta_i + n_{iy}\cos\theta_i + n_{iz}\cos\phi_i \sin\theta_i} \tag{2.7}$$

where θ_i and ϕ_i are determined by Eq (2.2). Note that the calculation in Eq (2.7) suffers from instability when the denominator is close to zero, producing abnormal values. Thus, we leave out any depth estimate that violates the following constraints:

$$0 < r_{ij} < 255 \qquad max(\frac{r_{ij}}{r_i}, \frac{r_i}{r_{ij}}) < 10 \tag{2.8}$$

For any r_{ij} that violates the constraints in Eq (2.8), we set it as r_i, the original depth estimate of P_i.

2.3.3 Aggregation with Confidence Scores

For each pixel P_i, we aggregate the estimates of its depth r_i from its neighbors $P_j \in N_i$ by using normalized weights. These weights have two components. First, we use the uncertainty score q_j of pixel P_j from the CNN output to compute its

confidence value $\boldsymbol{C}(P_j) = 1 - q_j^2$.

An example of the uncertainty score output maps can be seen in Figure 2.6. Second, the neighbor P_j's contribution is also weighted by $\boldsymbol{W}(P_i, P_j)$, the dot product between their respective normals, n_i and n_j.

Specifically, we aggregate the depth estimates for each pixel P_i with its neighbors as:

$$r_i^N = \frac{\sum_{P_j \in N_i} \boldsymbol{C}(P_j) \cdot \boldsymbol{W}(P_i, P_j) \cdot r_{ij}}{\sum_{P_j \in N_i} \boldsymbol{C}(P_j) \cdot \boldsymbol{W}(P_i, P_j)} \tag{2.9}$$

with $\boldsymbol{C}(P_j) = 1 - q_j^2$ and $\boldsymbol{W}(P_i, P_j) = n_j^\intercal n_i$.

Finally, the refined $\hat{r_i}$ for P_i is calculated as:

$$\hat{r_i} = \boldsymbol{C}(P_i) \cdot r_i + (1 - \boldsymbol{C}(P_i)) \cdot r_i^N \tag{2.10}$$

In other words, for a pixel with higher uncertainty and lower confidence, we place a greater reliance on its neighbors to refine its initial depth estimate. On the other hand, if a pixel has a low uncertainty score, the CNN believes this depth estimate is likely accurate, and so the neighbor estimates are less informative. This formulation allows us to adaptively refine the initial CNN estimates and reduce the unnecessary modifications of the already robust estimates.

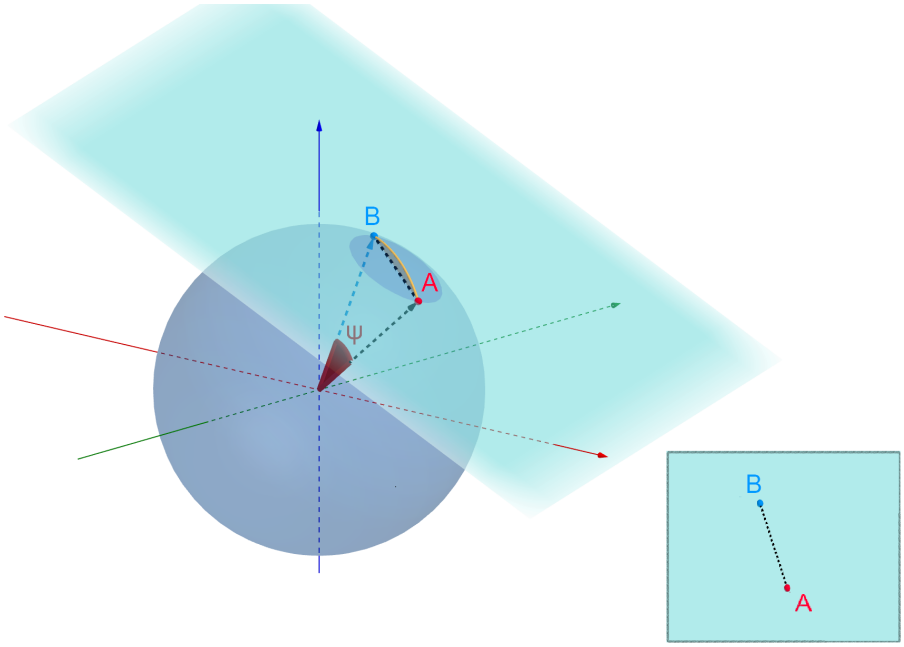


Figure 2.3: Hyperspherical Rotation Approximation. This figure illustrates that a displacement in 2D (shown at bottom right) can be regarded as a rotation around the center of a 3D sphere. Similarly, a double quaternion approximates a 3D displacement as a rotation on a suitably large 4D sphere.

2.3.4 Double Quaternion Approximation of Depth and Normal in the Loss function

2.3.4.1 Constructing Double Quaternion

Since a point's spatial coordinate (x, y, z) represents a translation from the coordinate origin $(0, 0, 0)$, a pixel's corresponding depth and surface normal orientation can be viewed as 3D translation and rotation, respectively.

The translation component of a 2D spatial displacement can be viewed as a rotation with respect to the origin of the 3D coordinate system. In fact, similar

approximation can be done from 3D to 4D. McCarthy [52,53] has shown that the homogeneous transform of 3D spatial displacements with rotation and translation is the limiting case of a 4D rotation as the radius of the 4D sphere R approaches infinity. Thus, we combine the 3D depth (translation) and normal (rotation) into one 4D measurement, which is represented by a double quaternion. Specifically, a 3D translation d can be approximated by a rotation on a 4D sphere of radius $R, \lim R \to \infty$ by an angle ψ as $\lim_{\psi \to 0} \sin(\psi) = \psi = \frac{d}{R}$. The double quaternion representing this 4D rotation is:

$$\mathbf{D} = \cos(\frac{\psi}{2}) + \sin(\frac{\psi}{2})\frac{d}{|d|} \tag{2.11}$$

Therefore, the 3D translation can be represented by a double quaternion $(\mathbf{D}, \mathbf{D^*})$, and the 3D rotation is represented as $(\mathbf{Q}, \mathbf{Q})$, where

$$\mathbf{Q} = (0, n_x, n_y, n_z) \tag{2.12}$$

Two double quaternions, ($\mathbf{G_1}$, $\mathbf{H_1}$) and ($\mathbf{G_2}$, $\mathbf{H_2}$) can be composed into a new double quaternion ($\mathbf{G_3}$, $\mathbf{H_3}$), where

$$\mathbf{G_3} = \mathbf{G_1}\mathbf{G_2} \qquad \mathbf{H_3} = \mathbf{H_1}\mathbf{H_2} \tag{2.13}$$

Moreover, following Ge *et al.* [50], we can compute the spatial distance between two double quaternions $(\mathbf{G_1}, \mathbf{H_1})$ and $(\mathbf{G_2}, \mathbf{H_2})$ as the angle between the respective double quaternion components:

$$\alpha = \cos^{-1}(\mathbf{G_1} \cdot \mathbf{G_2}) \qquad \beta = \cos^{-1}(\mathbf{H_1} \cdot \mathbf{H_2}) \tag{2.14}$$

2.3.4.2 Loss Function Based on Double Quaternions

Based on Eq (2.13), we combine the double quaternions representing translation and rotation (Eqs (2.11) and (2.12)) into a double quaternion representation for a depth and normal estimation pair:

$$\mathbf{G} = \mathbf{DQ} \qquad \mathbf{H} = \mathbf{D}^{*}\mathbf{Q} \tag{2.15}$$

We thus derive a loss function based on the angular distance between the two double quaternions: predicted ($\mathbf{G^{Pred}}$, $\mathbf{H^{Pred}}$) and ground-truth ($\mathbf{G^{GT}}$, $\mathbf{H^{GT}}$) as

$$L_{DQ} = \sqrt{\alpha_{DQ}^2 + \beta_{DQ}^2} \tag{2.16}$$

where α_{DQ} and β_{DQ} are calculated as in Eq (2.14)

2.3.5 Stereo Consistency

While training on datasets with stereo pairs, we further impose a stereo loss to minimize the discrepancy between the estimates from the two horizontally displaced images.

Given a known baseline distance b between a horizontal stereo pair, a pixel in one image L can be mapped onto the other image R with the following equation:

$$\phi_R = \phi_L + b \cdot \frac{\cos(\phi_L)}{r_L \cdot \sin(\theta_L)} \qquad \theta_R = \theta_L + b \cdot \frac{sin(\phi_L)cos(\theta_L)}{r_L} \tag{2.17}$$

Following the procedure presented in Section 2.3.2, we combine depth and normal estimates from the stereo pair images into two double quaternions ($\mathbf{G^L}$, $\mathbf{H^L}$) and ($\mathbf{G^R}$, $\mathbf{H^R}$), from which we calculate the stereo loss:

$$L_{\text{Stereo}} = \sqrt{\alpha_{\text{Stereo}}^2 + \beta_{\text{Stereo}}^2} \tag{2.18}$$

where α_{Stereo} and β_{Stereo} are also calculated as in Eq (2.14).

2.3.6 Overall Loss Function

With the double-quaternion-based losses derived above, we present the overall loss function for network training:

$$L_{\text{total}} = L_{\text{berHu}} + L_{DQ} + L_{\text{Stereo}} \tag{2.19}$$

Here, L_{berHu} is the reverse Huber loss function for both the depth and normal estimates compared to their respective ground truth [2]. In effect, this loss is equivalent to the mean absolute error for errors below a threshold c, and equivalent to a weighted mean squared loss for errors larger than c. We follow Laina *et al.* [2] and set c as 20% of the maximal error among all images of the current batch. We follow Lai *et al.* [4] and place extra weight for error at boundary pixels in calculating L_{berHu}.

2.4 Experiments

We have trained and evaluated the performance of our method on the ODS dataset [4]. It contains $40,000$ frames of indoor scenes from the Stanford 2D-3D-Semantics Dataset [54] with ground truth depth and surface normal. We adopt the same training-validation data split and evaluation metrics as Lai *et al.* [4]. We have also evaluated our method on the 360D dataset provided by Zioulis *et al.* [20].

2.4.1 Training Details

We initialize the encoding blocks of the CNN shown in Figure 2.1 with the commonly used VGG-16 [51] pre-trained weights. We use the Adam optimizer with its default parameters. We follow the data augmentation procedures detailed in Lai *et al.* [4] to introduce more variability in data. To be consistent with previous work,

Algorithm 1: Steps to Compute Training Loss

Input: Horizontally Displaced Image Pair I, $I^{'}$

Parameters: Weights θ of CNN

Label: Depth Map D_I and Surface Normal Map S_I

Output: Initial depth $\tilde{\mathrm{D}}_{\mathrm{init}}$, refined depth $\tilde{\mathrm{D}}_I$, estimated normal $\tilde{\mathrm{S}}_I$, total loss L_{total}

For each training iteration

- $\tilde{\mathrm{D}}_{\mathrm{init}}, \tilde{\mathrm{S}}_I = \mathrm{CNN}(I)$ (Section 3.1)
- $\tilde{\mathrm{D}}_{\mathrm{init}'}, \tilde{\mathrm{S}}_{I'} = \mathrm{CNN}(I')$ (Section 3.1)
- $\tilde{\mathrm{D}}_I = \mathrm{Refine}(\tilde{\mathrm{D}}_{\mathrm{init}}, \tilde{\mathrm{S}}_I)$ (Sections 3.2 - 3.3)
- $\tilde{\mathrm{D}}_{I'} = \mathrm{Refine}(\tilde{\mathrm{D}}_{\mathrm{init}'}, \tilde{\mathrm{S}}_{I'})$ (Sections 3.2 - 3.3)
- $L_{\mathrm{DQ}} = \mathrm{DQLoss}(\tilde{\mathrm{D}}_I, \tilde{\mathrm{S}}_I, \mathrm{D}_I, \mathrm{S}_I)$ (Section 3.4)
- $L_{\mathrm{Stereo}} = \mathrm{StereoLoss}(\tilde{\mathrm{D}}_I, \tilde{\mathrm{D}}_{I'})$ (Section 3.5)
- $L_{\mathrm{berHu}} = \mathrm{berHuLoss}(\tilde{\mathrm{D}}_I, \mathrm{D}_I)$ (Section 3.6)
- $L_{\mathrm{total}} = L_{\mathrm{berHu}} + L_{\mathrm{DQ}} + L_{\mathrm{Stereo}}$
- $\theta = \mathrm{Update}(L_{\mathrm{total}}, \theta)$

we train our networks for 40 epochs on this dataset to enable direct comparison of method performance. We adopt the conventional depth estimation metrics [1,2,4,55]. We denote the absolute prediction error of a pixel i as $\mathrm{E}_i = |y_i - \hat{y}_i|$, where y_i is the ground truth depth and $\hat{y}_i$ is the predicted depth. δ_j refers to the percentage of pixels with $\max(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}) < 1.25^j$. The other metrics used and their definitions are listed below:

$$\text{RMSE} : \sqrt{\frac{\sum_{i=1}^{N} {\mathrm{E}_i}^2}{N}} \quad \text{Abs. Rel.} : \frac{\sum_{i=1}^{N} \frac{\mathrm{E}_i}{\hat{y}_i}}{N}$$

$$\text{RMLSE} : \sqrt{\frac{\sum_{i=1}^{N} |\ln y_i - \ln \hat{y}_i|^2}{N}} \quad \text{Sq. Rel.} : \frac{\sum^{N} \frac{{\mathrm{E}_i}^2}{\hat{y}_i}}{N}$$

$$\text{Log10} : \frac{\sum_{i=1}^{N} |\log_{10} y_i - \log_{10} \hat{y}_i|}{N}$$

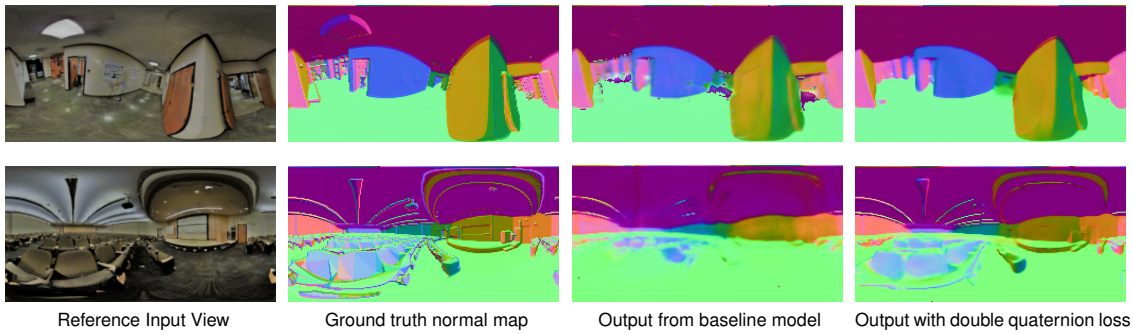


Figure 2.4: Surface Normal Prediction. Training with double quaternion loss enables the network to produce better surface normal estimates. In the first column, we show the input image for reference. In the third column, we show predictions from the traditional baseline model in which we separately calculate depth and normal loss without combining them into a double quaternion form. In the fourth column, we show results from our full model.

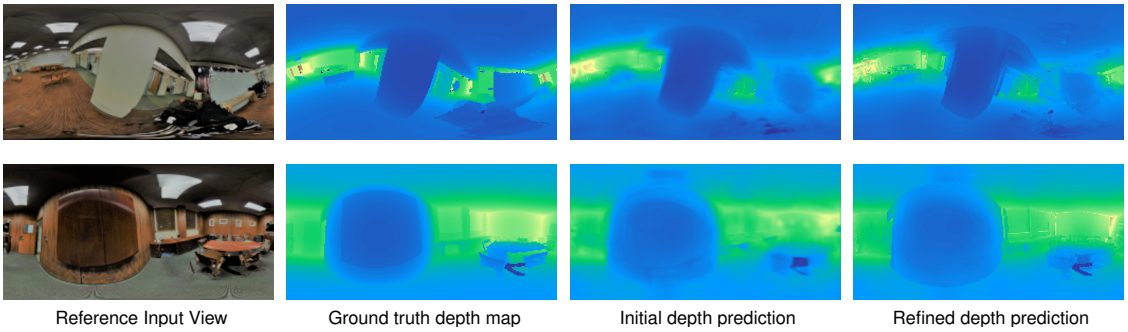


Figure 2.5: Depth Refinement Results. We compare initial depth estimates produced by the network and the refined output based on surface normal. In the first column, we show the input image for reference.

2.4.2 Comparison with Other Methods

The network performance of a CNN trained with our method is shown in Tables 2.1 and 2.2. Compared to other methods in Tables 2.1 and 2.2, our network shows improved performance in almost all metrics. In Figure 2.7 we show an example where our method better preserves the geometric detail of the scene. We believe this is because our model is aware of the surface normals and can use it to improve depth estimation.

2.4.3 Ablation Studies

We present the performance comparisons in Table 2.3. We observe decreased estimation accuracy with the removal of each component of the loss function. Figures 2.4, 2.5, and 2.6 further illustrate the impact of our method. It is worth noting that the network trained with double quaternions shows smoother normal esti-

Method	RMSE	Log10	AbsRel	δ_1	δ_2	δ_3
UResNet [1]	2.037	0.326	16.906	0.213	0.399	0.560
RectNet [1]	1.738	0.291	16.132	0.240	0.453	0.634
FCRN [2]	0.672	0.101	7.448	0.806	0.932	0.966
PSMNet [3]	0.393	0.059	5.641	0.953	0.975	0.980
SepUNet [4]	0.495	0.042	1.779	0.944	**0.987**	**0.993**
SepUNetS [4]	0.614	0.072	1.841	0.835	0.966	0.985
SepUNetDD [5]	0.392	0.036	2.120	**0.960**	0.987	0.992
Ours	**0.389**	**0.031**	**0.413**	0.954	0.984	0.990

Table 2.1: Performance Comparison on the ODS dataset [4]. Evaluation statistics for row 1-7 are directly taken from Lai *et al.* [4] and Xie *et al.* [5]. Our method produces superior results in most metrics.

mates, which could explain the increase in estimate accuracy since the normal-based refinement method relies on accurate normal estimates.

2.5 Limitations and Conclusion

We have shown how double-quaternion loss is useful in reducing the geometric inconsistency and improving estimation accuracy. Our results indicate that a double quaternion construct could have a meaningful potential for other tasks that involve processing 360° images. We hope our work will bring a new hyperspherical perspective to analyzing omnidirectional visual data, as a complement to the traditional Cartesian

Method	RMSE	RMLSE	AbsRel	SqRel	δ_1	δ_2	δ_3
UResNet [1]	0.3374	0.1204	0.0835	0.0416	0.9319	0.9889	0.9968
RectNet [1]	0.2911	0.1017	**0.0702**	0.0297	0.9574	0.9933	0.9979
monoDepth [44]	7.2097	0.8200	0.4747	2.3783	0.2970	0.7900	0.7510
FCRN [2]	0.9410	0.3760	0.3181	0.4469	0.4922	0.7792	0.9150
DCRF [25]	1.1596	0.4400	0.4202	0.7597	0.3889	0.7044	0.8774
Ours	**0.2373**	**0.0907**	0.0859	**0.0213**	**0.9690**	**0.9954**	**0.9988**

Table 2.2: Performance Comparison the 360D dataset [20]. Evaluation statistics for row 1-5 are directly taken from Zioulis *et al.* [1]. Our method surpasses other methods in all metrics except for AbsRel.



Figure 2.6: Uncertainty Estimates. The network learns to produce meaningful uncertainty maps by effectively grasping the object's geometric outline. It places higher uncertainty near object edges, where depth predictions tend to be overly smooth and prone to error.

(or equirectangular) perspective.

Our method achieves good performance on the testing scenes in the given datasets. One of the assumptions our method makes is that the normals can be

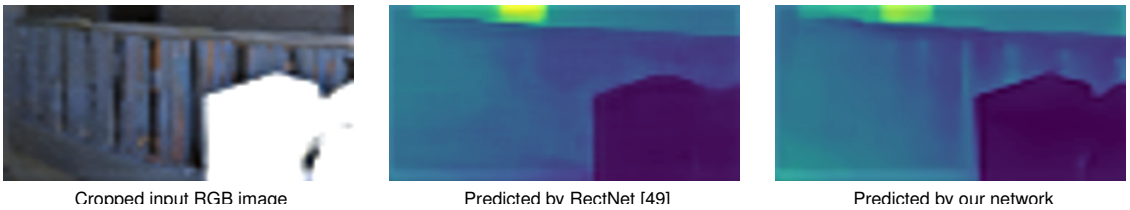


Figure 2.7: More Qualitative Comparisons. Here we show an example from a test image from the 360D dataset [1]. Note that our result largely preserves the geometry of the hallway railings.

Method	RMSE	RMLSE	AbsRel	SqRel	δ_1	δ_2	δ_3
Full model	**0.3894**	**0.2572**	**0.4130**	**0.6872**	**0.9543**	**0.9836**	**0.9904**
w/o L_{DQ}	0.4731	0.3452	0.5830	0.9012	0.9257	0.9718	0.9880
w/o Refinement	0.4114	0.3190	0.5535	0.9220	0.9313	0.9780	0.9903
w/o L_{Stereo}	0.3953	0.2622	0.4562	0.7068	0.9530	0.9801	0.9904

Table 2.3: Ablation Results. Evaluation statistics are based on prediction results on the ODS dataset [4]. Results in rows 2-4 show the network performance when trained without the double quaternion loss, depth refinement step, and stereo consistency loss, respectively. Results show that each component in our proposed method contributes to better estimation accuracy.

estimated well and provide meaningful guidance for depth refinement. Also, the quality of our depth estimation on real world 360° images is dependent on their domain similarity to the training dataset on which the model is trained. Our method does not perform well if either of these assumptions do not hold.

Furthermore, as previously discussed, direct learning on 360° images suffers

from image distortion, which is not explicitly addressed by our method. In particular, we directly deploy a 2D CNN with regular, square kernels without any modification. Thus, it would be worthwhile to incorporate methods that alleviate the distortion problem, such as modifying convolutional kernels to account for distortion, and directly performing convolution on spheres instead of images with equirectangular projection.

In summary, we present a new framework for 360° depth estimation using CNN. We use the double quaternion formulation to integrate depth and surface normal in loss calculations. Experiments show superior results for the joint depth and normal estimation task. We also extend the double quaternion formulation to establish stereo consistency from the training data without restricting the network to a fixed baseline. We demonstrate quantitative and qualitative results that confirm the benefits of our new approach.

Chapter 3: Efficient Neural Representation for Light Fields

3.1 Introduction

Light fields offer an information-rich medium for static and dynamic scenes. However, a significant barrier to their widespread adoptions is a lack of sufficiently compact representations of such high-dimensional data, making it impractical for efficient storage, editing, and streaming. For example, a 1080p 60-fps light field video captured on a 10×10 camera grid easily requires several gigabytes of storage space for every second of content.

A straightforward solution to compressing light fields is to apply existing, widely used compression methods such as JPEG and MPEG. However, due to the sheer amount of images captured in a light field, the compression rate of these single-view-based methods are far from satisfactory [56, 57]. Therefore, it is imperative to have a compact way to represent light fields by taking advantage of the overlapping and repetitive visual patterns in light fields.

Extensive research has been devoted to designing compact light field representations based on the patch-based compression strategy manifest in the JPEG standard. These methods represent each image patch as a weighted sum of a small dictionary of basis functions, and the goal is finding new ways to construct dictionaries of basis

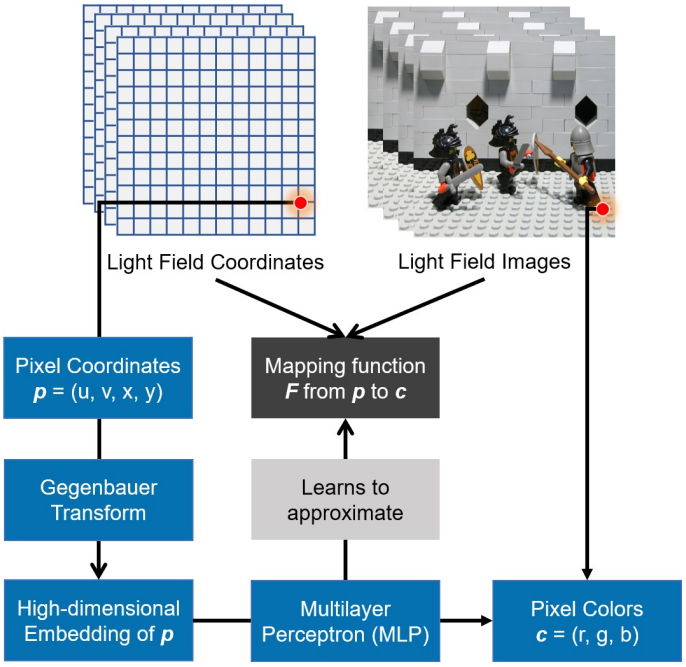


Figure 3.1: Overview of SIGNET. We train a MLP to approximate the mapping function from each pixel's coordinates to its color values. My input transformation strategy based on the Gegenbauer polynomials enables the MLP to more accurately learn the high-dimensional mapping function.

functions that achieve better compression results. Yet, previous efforts have limited success in enabling easy transmission and manipulation of light field content.

Recent advances in deep learning have led to impressive results in representing data like images and volumes [58–60] with neural networks. A common thread among these methods is incorporating Fourier-inspired modifications to the classical neural network design called multilayer perceptron (MLP). Specifically, the SIREN network [59] uses a sinusoidal activation function between the MLP layers, while neural radiance field (NeRF) networks [58] designed for volumetric radiance data show the effectiveness of applying cosine and sine transformations on input coordinates.

The improvement brought by the Fourier basis used in NeRF is further analyzed and formalized by Tancik *et al.* [60], who also successfully extend the neural representation to data like 2D images and 3D shapes.

The proven capability of MLPs to express visual content with high fidelity implies that we could potentially compress a gigapixel light field within a few megabytes. However, as shown in Section 3.4, the previous techniques fall short of representing light fields without visible artifacts.

In this chapter, we present a new framework that efficiently and accurately represents light field content using neural networks. Crucially, we introduce a novel input transformation strategy of the multi-dimensional light field coordinate based on the orthogonal Gegenbauer polynomials, which in our experiments work very well with the sinusoidal activation functions between the MLP layers. We call this network SIGNET (SInusoidal Gegenbauer NETwork), and we show its superiority for light field neural representation over a variety of Fourier-inspired input transformation strategies. SIGNET also achieves outstanding reconstruction quality with a higher compression rate than state-of-the-art dictionary-based light field compression methods. We further demonstrate how our MLP-based approach easily allows for view synthesis and super-resolution on the encoded light field scenes.

In summary, our contributions are as follows:

- We present a neural representation of light fields which achieves high reconstruction quality and compression rate and offers pixel-level random access to the encoded light field.

- We introduce an input transformation strategy for coordinate-input MLPs using Gegenbauer polynomials, which outperforms other recently proposed techniques on light field data.

- We show such a neural representation enables high-quality decoding at novel coordinates without additional training, achieving super-resolution along spatial, angular, and temporal dimensions on light fields.

3.2 Related Work

Light Field Compression. Traditional compression relies on classical coding strategies that typically involve analytical basis functions such as the Fourier basis and wavelets. Prior research has augmented this analytical approach with disparity [61–64] and geometry information [65]. Some sophisticated applications of light field video [66–68] also integrate motion prediction and build on existing video codec algorithms such as HEVC (H.265) [69] and VP9 [70]. More recently, Le Pendu *et al.* [71] present a Fourier Disparity Layer representation for light fields, which allows upsampling [72] and compression [73, 74] in the Fourier domain.

A different approach towards light field compression involves learning a dictionary of basis functions, which is inspired by progress in sparse coding from machine learning, where dictionaries learned with data-driven algorithms have been shown to outperform analytical basis functions [75–78]. However, the dictionaries learned with conventional algorithms such as K-SVD [7] still contain too much redundancy and have a high storage cost. The current state-of-the-art methods [6, 79] for light field

compression improve this approach by learning an ensemble of orthogonal dictionaries with a novel pre-clustering strategy.

We present a novel approach to this task by learning a neural representation of light fields. While our approach is rooted in the idea of basis functions, we fundamentally differ from the previous methods as we use the expressive power of neural networks with non-linear activation functions to combine the basis functions into the desired output.

Light Field Interpolation. Most approaches rely on proxy information such as depth or optical flow [12, 19, 80–83]. Recently, deep learning methods have been used to infer depth and optical flow from light fields, and render novel viewpoints [84–88]. These methods warp the original frames to a novel viewpoint. While the results are impressive, they require access to the original light field data at run-time, incurring additional, sometimes prohibitive, costs to the light field processing pipeline.

In this chapter, we show how our neural light field representation naturally enables interpolation from the compressed data without explicit learning or proxy information. Although our presented network is not specifically designed for light field super-resolution or view synthesis, our results show its promising potential to be adapted for such tasks.

Coordinate-input MLP. Recent research [58–60] has shown the potential of using coordinate-input MLP networks to represent various data. The Fourier-inspired transformation achieves state-of-the-art free viewpoint synthesis on static scenes [58].

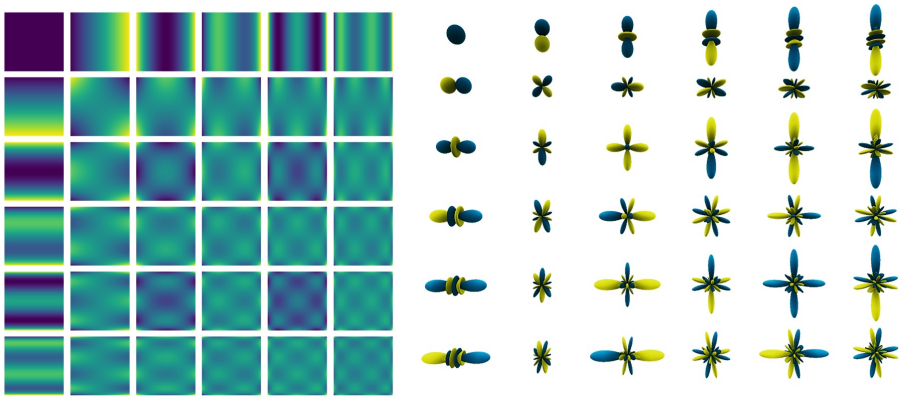


Figure 3.2: Illustration of Gegenbauer (Ultraspherical) Polynomials. We evaluate the 2D Gegenbauer basis functions on a 2D Cartesian grid (left) and a 3D polar grid (right). Only the first six orders of the basis are selected for illustration purposes.

The sine activation, introduced in SIREN [59], allows a simple MLP with raw coordinate inputs to accurately model the coordinate-to-color mapping of data including images and videos. However, our experimental results show that these Fourier-inspired methods are unable to accurately model the coordinate-to-color mapping in light fields. We present a new transformation that allows the MLPs to successfully represent dense light fields, and we show its applicability for compactly representing high-resolution light fields.

Gegenbauer Polynomials. Previous research in applied mathematics has shown the effectiveness of Gegenbauer polynomials, also known as ultraspherical polynomials, in addressing the Gibbs phenomenon [89], which is a commonly observed artifact in MRI reconstruction using Fourier-based approximations [90, 91]. It has been shown by Gottlieb *et al.* [89] that the finite Gegenbauer expansion of such functions provides a better convergence and usually resolves the Gibbs artifact using fewer basis functions

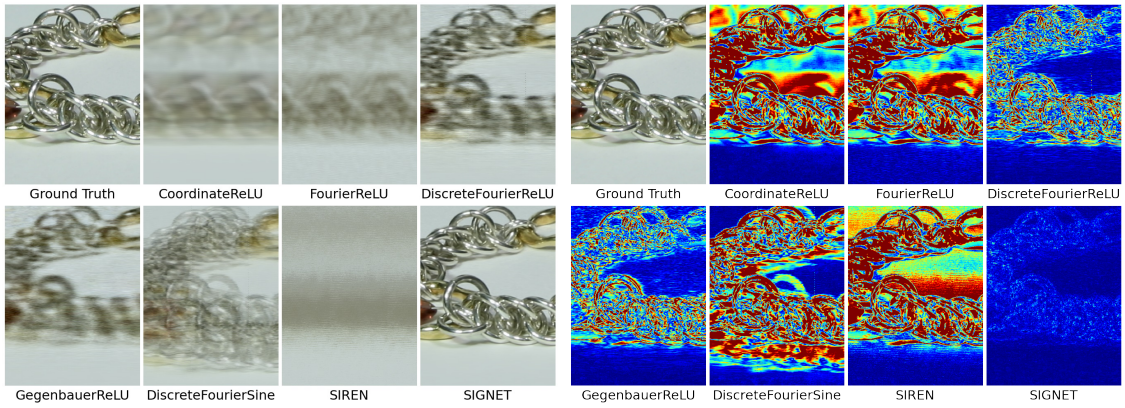


Figure 3.3: We show examples of reconstructed images (left) and absolute errors (right). SIGNET achieves good accuracy while other methods find encoding this scene challenging.

than the Fourier approach. Specifically, they show that given the first N Gegenbauer expansion coefficients, we can construct an exponentially convergent approximation to the point values of $f(x)$ in any sub-interval where f is analytic [92].

Moreover, recent studies on machine learning also show the usefulness of Gegenbauer kernels [93, 94] or hyperspherical loss functions [95, 96]. Our experiments show applying Gegenbauer transformation on light field coordinates not only improves reconstruction quality but also results in a faster neural network convergence.

3.3 Overview

3.3.1 Light Fields as Functions

Our goal is to find an accurate approximation to the mapping function $\mathcal{F}$ for a given light field, from which we could retrieve the color value of any pixel. Moreover, such a functional approximation could be parameterized using fewer bits than the

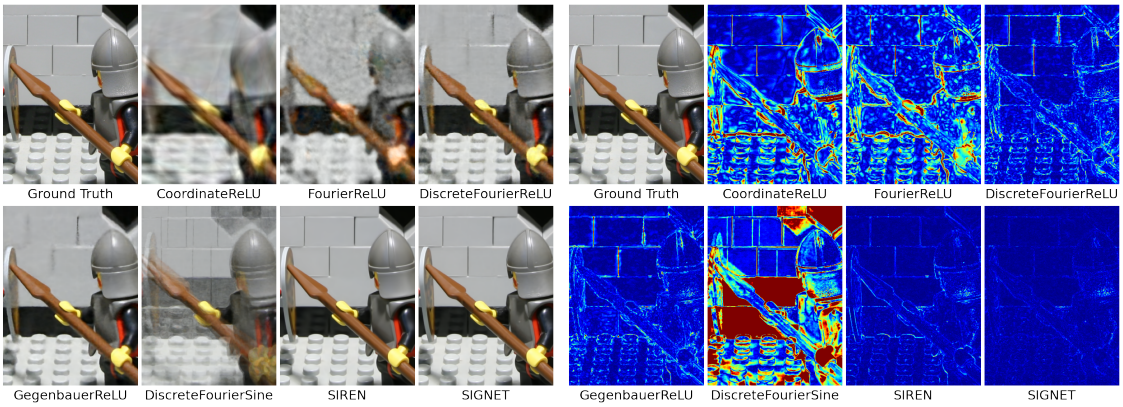


Figure 3.4: We show examples of reconstruction results (left) and absolute errors (right) on the *Lego* scene. Our SIGNET method with Gegenbauer input transformation produces the best result. Notice the *SIREN* method is relatively inaccurate in capturing the discontinuity near the edges.

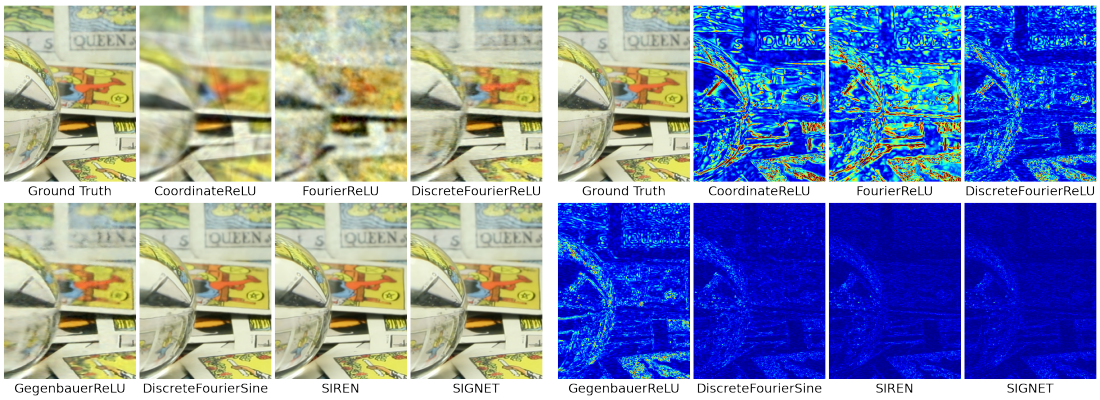


Figure 3.5: Examples of reconstruction results (left) and absolute errors (right) on the *Tarot* scene. The last three methods (DiscreteFourierSine, SIREN, and SIGNET) achieve similarly accurate reconstruction, with most of the residual errors coming from the extreme refraction at the surface of the crystal ball.

original light field content, providing us with a compressed representation of the light field.

A noteworthy feature of this functional approach towards light field representation is that we could arbitrarily decode any pixel within the given light field, providing random access to the compressed data which ensures efficiency in content retrieval and streaming. Most previous compression methods discussed in Section 3.2 involve encoding and decoding blocks of pixels, and many video compression methods even require information from the previous frame to decode the current frame.

3.3.2 Function Approximations

A single-channel image represented as a 2D function $\mathcal{F}(x, y)$, could be approximated by the weighted sum of N orthogonal basis functions $\Theta(x, y)$:

$$\tilde{\mathcal{F}}(x, y) = \sum_{i=1}^{N} a_i \Theta_i(x, y) \tag{3.1}$$

Assuming the set of orthogonal functions is known, an image could be recovered by just using the coefficients $\{a_i\}_{i=1}^{N}$. Thus, image compression is reduced to compressing this set of coefficients. In the case of JPEG compression, cosine functions are used as the Θ_i's, and the coefficients $\{a_i\}_{i=1}^{N}$ are quantized and entropy-encoded.

If we use a similar, analytical formulation for 4D light fields, we would obtain the following approximation:

$$\tilde{\mathcal{F}}(u, v, x, y) = \sum_{i=1}^{N} a_i \Theta_i(u, v, x, y) \tag{3.2}$$

However, instead of analytically calculating the coefficients a_i, we propose using an L-layer MLP to compute:

$$\tilde{\mathcal{F}}(u, v, x, y) = \phi_L \circ \phi_{L-1} \circ ... \circ \phi_1([\Theta_i(u, v, x, y)]_{i=1}^{N}) \tag{3.3}$$

Here, ϕ_l stands for the l-th layer of the neural network with a weight matrix W_l, bias vector b_l, and an activation function σ . The output from each layer is $\phi_l(x) = \sigma(W_l x + b_l)$. We next discuss why this approach is preferred over computing analytic coefficients.

3.3.3 MLP for Approximation

This MLP-based formulation shares several similarities to the classic Fourier expansion method. In fact, for the 1D function case, a MLP could be constructed that has the same representation capacity as a Fourier expansion:

$$f_N(x) = \sum_{n=-N}^{N} a_n \cdot \exp(i \cdot 2\pi n x), x \in [0, 1], N \in \mathbb{Z} \tag{3.4}$$

This Fourier expansion is equivalent to a special two-layer MLP with activation function $\sigma(x) = \exp(ix)$. The first layer of this MLP would be a $1 \times 2N$ matrix with values $\{2\pi n\}_{n=-N}^{N}$, while the second layer would contain the $2N \times 1$ Fourier coefficients $\{a_n\}_{n=-N}^{N}$.

The same analogy generalizes to multi-dimensional inputs. For example, the Fourier expansion of a 2D function is of the following form:

$$f_{N,M}(x, y) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{m,n} \cdot \exp[i \cdot 2\pi(nx + my)] \tag{3.5}$$

with $x, y \in [0, 1]$ and $N, M \in \mathbb{Z}$. Notably, the Fourier coefficients $\{a_{m,n}\}$ are grouped together as a 2D matrix. While this seems incompatible with the MLP described earlier, we could flatten this 2D matrix into 1D and fit this expansion into the form of a two-layer MLP. For example, the input to the MLP would be $[x, y]^T$, the first

layer could be a $2 \times 4NM$ matrix with the columns populated by every combination of (n, m), and the second layer could be a $4NM \times 1$ matrix that contains the Fourier coefficients $\{a_{m,n}\}$.

Given the examples in 1D and 2D, we can see how the same derivation naturally extends to higher dimensions. However, increasing the dimensions would lead to a combinatorial growth of number of coefficients to consider. More importantly, it would be hardly meaningful to have a MLP with even more parameters than the Fourier coefficients. Therefore, we need a MLP with far fewer parameters while ensuring that it can approximate the multi-dimensional function for light fields.

3.3.4 Towards Multi-dimensional Input

Recent works [59, 60, 97] have shown that typical MLPs with coordinate inputs suffer from a spectral bias when trying to approximate multi-dimensional functions like images. Two recent techniques modify coordinate-input MLPs to enable them to successfully learn on data with high-frequency details such as natural images and volumes. SIREN [59] uses sine function as the activation function between network layers, while FourierMLP [60] shows the effectiveness of transforming the input ($[x, y]$ in the 2D image case) using the cosine and sine functions as a basis.

Recalling Euler's formula, $e^{ix} = cos(x) + i \cdot sin(x)$, we observe that these two techniques have similar elements to Equation 3.5, where we show the equivalence between 2D discrete Fourier transform and a particular MLP. It is therefore not surprising that the coordiate-input MLP finally achieves accurate representation

when it is empowered by the periodicity behind Fourier expansions, either from the sine activation used in SIREN or from the sine and cosine input transformations in FourierMLP.

Moreover, in NeRF and FourierMLP, the coordinate along each dimension is independently transformed, and the respective high-dimensional embeddings are concatenated together as input to the MLP. This concatenation is crucial since it elegantly avoids the combinatorial explosion of multi-dimensional basis.

For instance, for a 2D image with cosine bases of orders N and M along each dimension, it is sufficient to simply calculate $\{\cos(nx)\}_{n=-N}^{N}$ and $\{\cos(my)\}_{m=-M}^{M}$ in 1D and then concatenate them into a $1 \times 2(N+M)$ input, instead of the $1 \times 4NM$ input from $\{\cos(nx+my)\}_{n,m}$.

This modification enables MLPs to make use of multi-dimensional orthogonal basis functions without the quadratic increase in the input size. Therefore, we adopt this concatenation strategy to avoid training an exceptionally wide MLP.

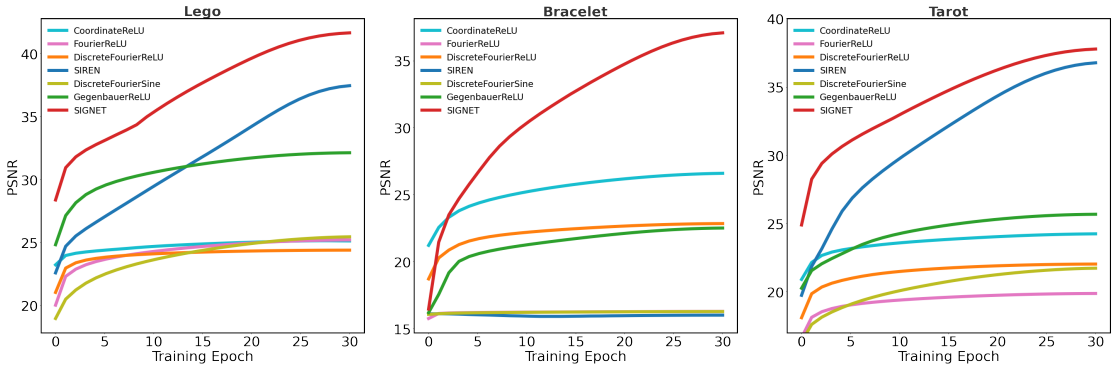


Figure 3.6: We show the training PSNR on the static light field scenes. Compared to other variants of coordinate-input MLP, SIGNET clearly has a faster convergence and a higher quality in representing the 4D light fields.

3.3.5 Gegenbauer Basis

While these Fourier-inspired transformation techniques are shown to work really well for images and volumes, we are unable to get satisfactory results when using them with MLPs to represent light fields as discussed in Section 3.4. Instead, inspired by the benefits of Gegenbauer reconstruction over Fourier-based reconstruction discussed in Section 3.2, we develop an input transformation strategy using Gegenbauer polynomials as basis functions. The n-th order Gegenbauer polynomials could be computed recursively as:

$$G_{n+1}^{(\alpha)}(x) = \frac{1}{n}[2x(n+\alpha-1)G_n^{(\alpha)}(x) - (n+2\alpha-2)G_{n-1}^{(\alpha)}(x)], \tag{3.6}$$

where $-1 \leq x \leq 1$, $G_0^{(\alpha)}(x) = 1$, and $G_1^{(\alpha)}(x) = 2\alpha x$.

3.4 Methods

3.4.1 Proposed Framework

To reconstruct pixels of a 4D light field from SIGNET, a coordinate-input vector $p = [u, v, x, y]$ is transformed with a set of functions $\mathcal{S}_i$ as

$$\mathcal{E}(p) = [\mathcal{S}_1(u), ..., \mathcal{S}_{C_u}(u), \mathcal{S}_1(v), ..., \mathcal{S}_{C_v}(v), ...,$$
$$\mathcal{S}_1(x), ..., \mathcal{S}_{C_x}(x), ..., \mathcal{S}_1(y), ..., \mathcal{S}_{C_y}(y)]$$

C_u, C_v, C_x, C_y are the maximum orders of basis functions used to map the coordinates along each dimension. In our case, we use the Gegenbauer basis functions to transform the input by setting $\mathcal{S}_n(z) = G_n^{\alpha}(z)$ as defined in Section 3.3.5. We adopt the sine

Table 3.1: Different MLP used in experiments. Definitions of input transformation methods are in Section 3.4.2

Network	Transformation	Activation
CoordinateReLU	None	ReLU
FourierReLU	Gaussian Fourier	ReLU
DiscreteFourierReLU	Discrete Fourier	ReLU
SIREN	None	Sine
DiscreteFourierSine	Discrete Fourier	Sine
GegenbauerReLU	Gegenbauer	ReLU
SIGNET	Gegenbauer	Sine

activation presented by Sitzmann *et al.* [59] as it enhances the ability of a MLP to approximate functions even without an input transformation.

3.4.2 Comparative Evaluation

Having discussed the motivations for using MLP for representing light fields and introducing the Gegenbauer input transform, we need to validate this approach through qualitative results for MLP light field reconstruction. To objectively evaluate our network, we compare it with several other networks with different transformation strategies and activation functions (see Table 3.1).

The specific input transformation strategies in the second column in Table 3.1.

None transformation means $\mathcal{S}_n(z) = z$ and that $C_u = C_v = C_x = C_y = 1$. With *Discrete Fourier*, the transformation function returns a tuple as $S_n(x) = [cos(2\pi nx), sin(2\pi nx)]$. With *Gaussian Fourier*, we adopt the Fourier-based transformation presented in FourierMLP [60], with the scale set at 5. This is equivalent to having $\mathcal{E}(p) = [cos(Bp^\top), sin(Bp^\top)]$ with B being a $\frac{C}{2} \times 4$ matrix with entries randomly initialized from a Gaussian distribution $\mathcal{N}(0, 5)$, and $C = C_u + C_v + C_x + C_y + C_t$.

Table 3.2: Compression Performance. Values in the column *Size* denote the storage in megabytes(MB) for each method without further quantization. Details of the other listed methods may be found in Miandji *et al.* [6]. The storage cost of SIGNET is calculated based on the number of MLP parameters required (see Table 3.3) to reconstruct all pixels in each scene.

	Static Light Fields						Light Field Videos			
Scene	Lego		Bracelet		Tarot		Painter		Trains	
Method	Size	PSNR	Size	PSNR	Size	PSNR	Size	PSNR	Size	PSNR
SIGNET	**9.0**	**41.26**	**12.0**	38.70	**9.0**	37.47	**144**	**39.56**	**144**	**39.73**
AMDE [6]	29.3	40.90	18.1	**39.90**	44.2	38.54	941	38.25	809	37.00
KSVD [7]	29.3	38.39	18.1	36.73	44.3	**38.81**	942	38.12	807	35.06
HOSVD [6]	29.3	37.24	18.0	33.98	44.3	34.53	942	36.91	807	35.29
5D DCT [6]	29.4	37.29	18.1	32.31	44.2	33.03	941	36.79	807	35.20
CDF 9/7 [8]	29.0	33.71	18.2	31.98	44.3	29.17	941	31.69	1116	29.80

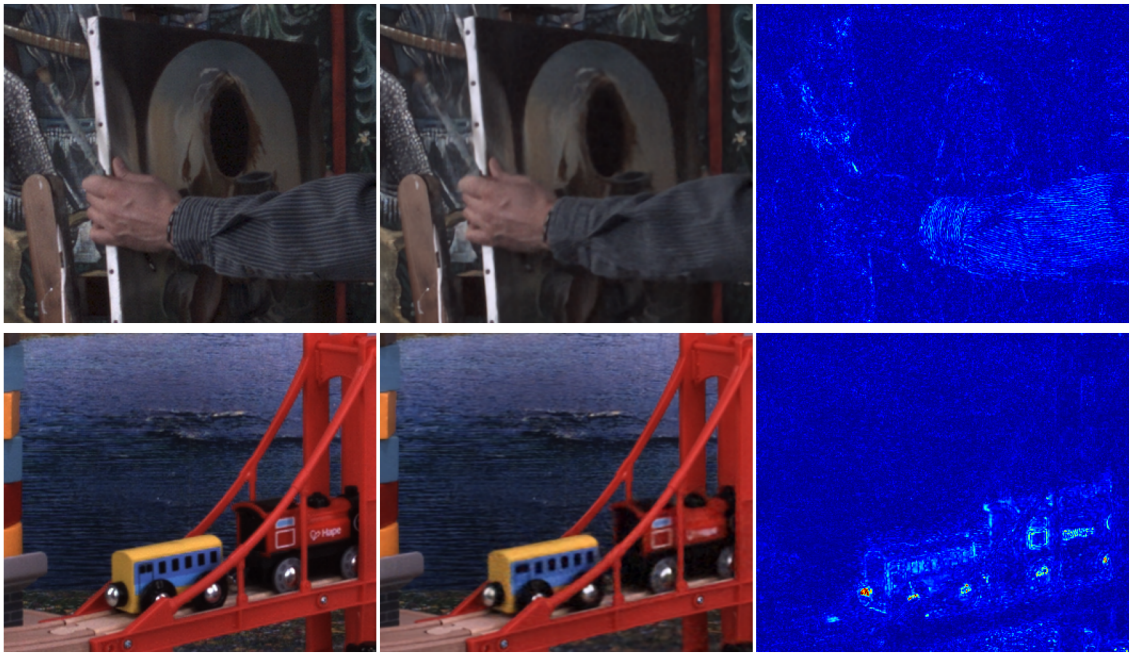


Figure 3.7: Light Field Video Reconstruction. We show examples of reconstruction on light field video scenes *Painter* and *Trains*.

3.4.3 Data and Training Setup

For static light fields, we use the Stanford Light Field Archive [98]. For light field videos, we choose the Technicolor Natural Light Field Video dataset [99]. We select these specific scenes used by Miandji *et al.* [6] for a fair comparison of performance.

We implemented the networks in PyTorch and follow the same training scheme and random seed to ensure reproducibility. We train all networks for 30 epochs, each taking around 12 minutes on an NVIDIA GeForce RTX 2080 Ti.

3.5 Results

3.5.1 Static Light Field Reconstruction

We test the effectiveness of SIGNET and compare it against the other configurations listed in Table 3.1 with the same model size. For a fair comparison, we use the same number of basis functions (C_u, C_v, C_x, C_y) for the three input transform strategies: **GaussianFourier**, **DiscreteFourier** and **Gegenbauer**. We train each type of MLP on the three static light field scenes and present example results in Figure 3.3.

We observe that SIGNET leads to higher quality reconstructions. The network with Gegenbauer transformed input not only produces more accurate reconstruction than with other transform strategies, its results are also more visually stable. Even without the sine activation (GegenbauerReLU), the Gegenbauer transformation improves the performance of the basic ReLU MLP (CoordinateReLU), and it even achieves better performance averaged over the three scenes than Fourier-based MLPs (FourierReLU, DiscreteFourierReLU, and DiscreteFourierSine). We also show the PSNR values during the training progression for each type of MLP in Figure 3.3. The PSNR curves further corroborate the superiority of SIGNET, which clearly shows faster convergence and higher accuracy. Another takeaway from these results is that the success of our method is not entirely due to the sinusoidal activation (see SIREN versus SIGNET), and that the Gegenbauer transformation is indeed necessary and effective for more accurate light field representations. Therefore, for

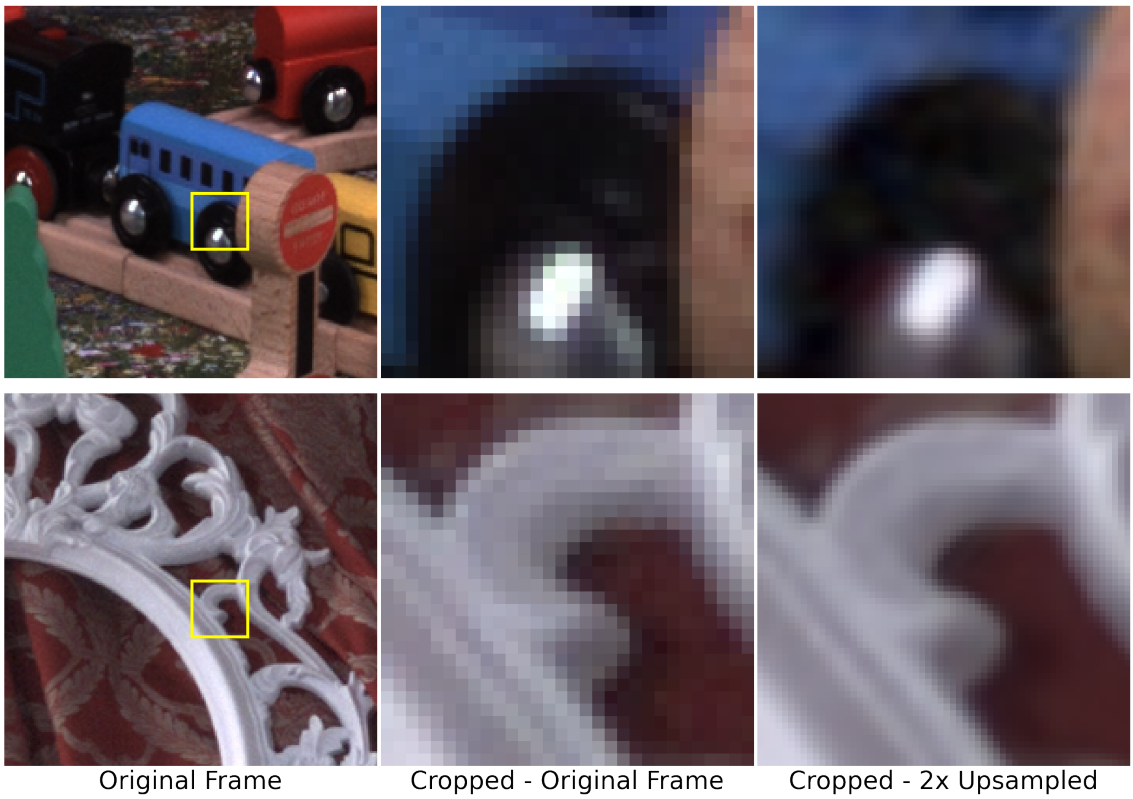


Figure 3.8: Spatial Upsampling. We evaluate the trained SIGNET on dense sampling grid points in the spatial dimensions. We show zoomed-in details in the cropped region bounded by the yellow rectangle.

the rest of the chapter, we adopt SIGNET as our default MLP setup.

Furthermore, SIGNET not only accurately reproduces the RGB values at each pixel location, but it is also parsimonious in storage cost. In Table 3.1 we compare the compression and reconstruction result of SIGNET against previous methods. On all three static light field scenes, we achieve reconstruction quality on par with the state-of-the-art methods while requiring much less storage.

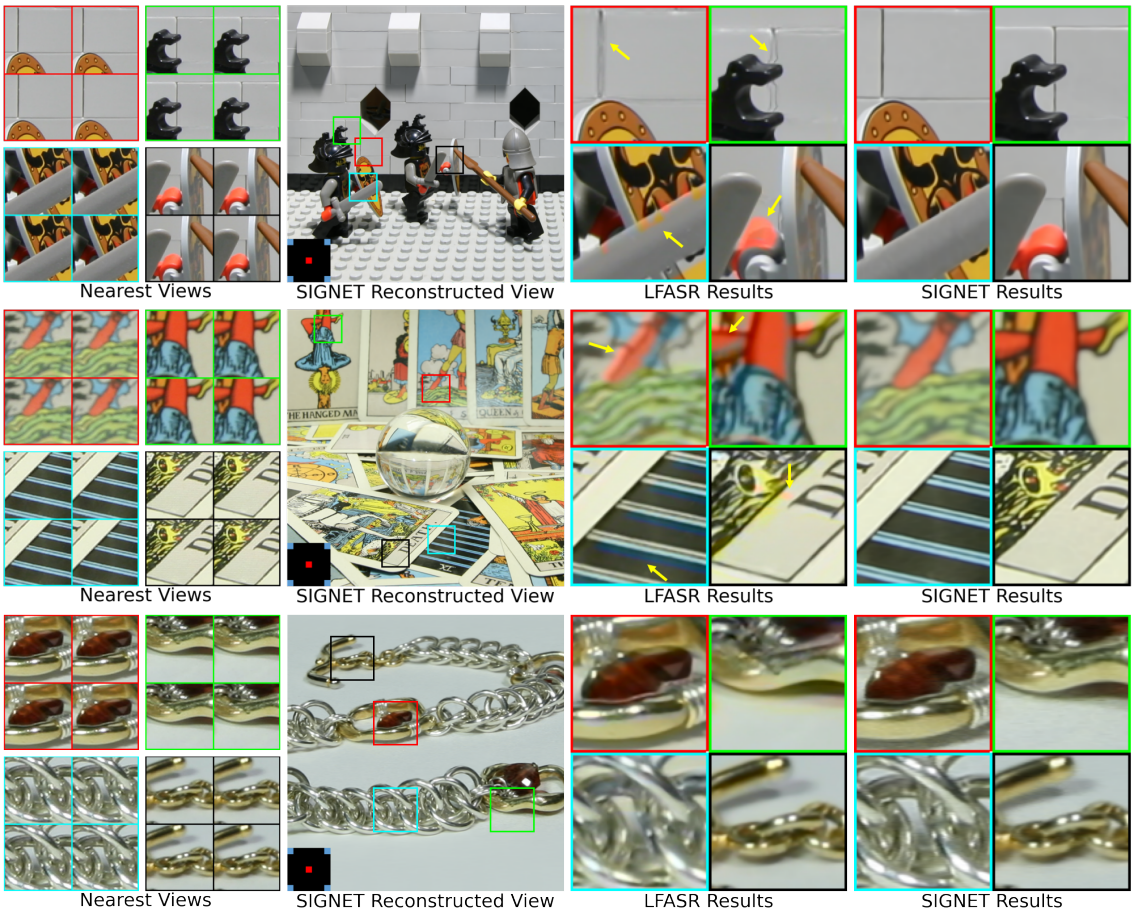


Figure 3.9: Angular Upsampling. At the bottom left corner of the reconstructed view, we show the relative positions of the reconstructed view (red square) and its four nearest views (blue squares) in the original light field. We present reconstructions at novel viewpoints from the three static scenes, and we also show results from the deep-learning-based method, LFASR [9], which is trained specifically for light field angular upsampling. Notice the LFASR results show visible artifacts pointed to by the yellow arrows, such as distorted geometry and ghosting.

3.5.2 Extension to Light Field Videos

After observing the good performance of SIGNET on static 4D light fields, we extended the same framework to light field videos, where effective compression is even more important. Since a light field video contains many more images than a static light field, it is challenging to only train one network to approximate the 5D function that covers the entire video. Therefore, we divide a light field video into smaller blocks such that each block is independently learned by a small SIGNET.

We show example results in Figure 3.7 and compare the compression results with previous methods in Table 3.2. SIGNET has a clear-cut advantage both in terms of reconstruction quality and compression size.

3.5.3 Light Field Super-Resolution

SIGNET's unique advantage of is it can evaluate arbitrary coordinates, thus allowing interpolation of viewpoint, as well as spatial and temporal coordinates.

In Figure 3.8, we show the results of spatial upsampling with the trained SIGNET. We observe that the results do not have any perceptible artifacts. Furthermore, these results suggest that SIGNET does not merely memorize the training samples, but it also gracefully interpolates among the unsampled coordinates.

In Figure 3.9, we further show the upsampling results for the angular dimensions. We compare our results with LFASR [9], a state-of-the-art method that is representative of most learning-based methods that rely on depth estimation. Note that our novel views are generated without any depth information, and we do not

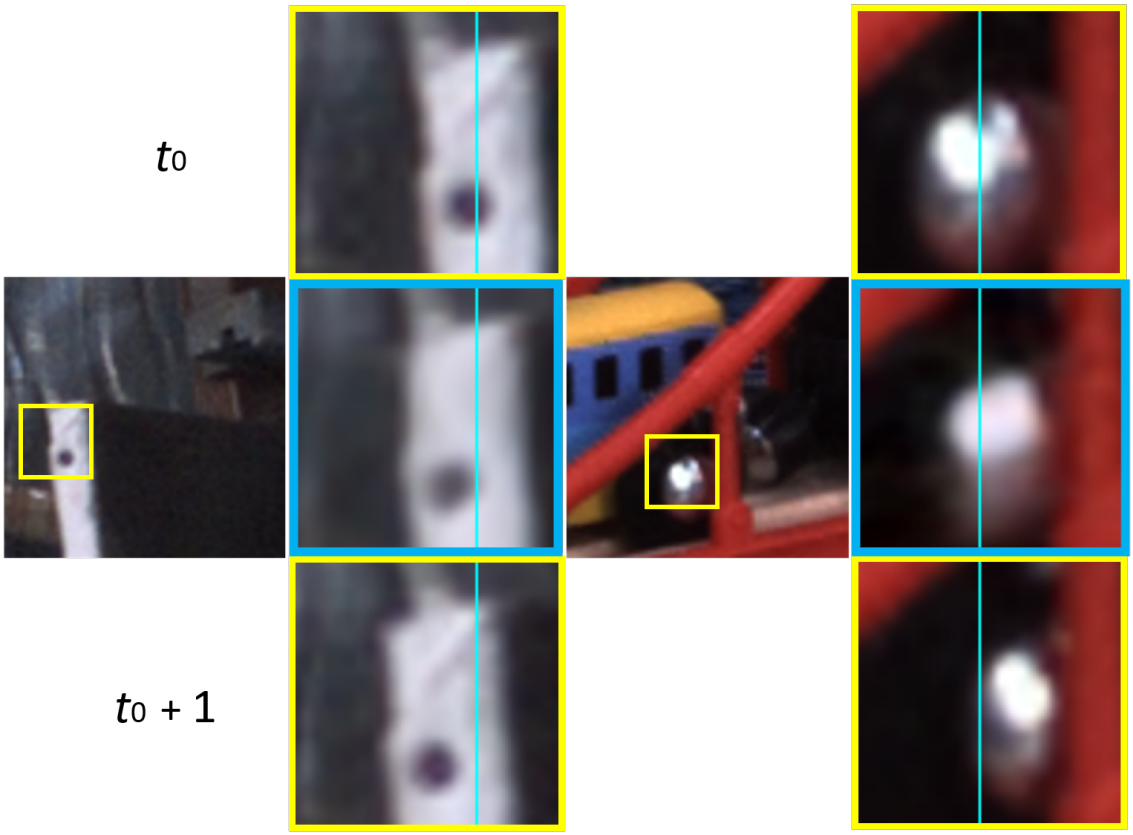


Figure 3.10: Temporal Upsampling. t_0 and $t_0 + 1$ are consecutive frames in the original video. The blue boxes contain output from frames evaluated at $t_0 + \frac{1}{2}$, which is not present in the original video. The vertical lines are drawn for easier observation of the motion trajectory.

explicitly use any information from adjacent images. SIGNET achieves a similar level of visual quality to the learning-based method equipped with deep CNNs, while avoiding the mismatching artifacts that stem from inaccurate depth or optical flow estimation. The fact that SIGNETs can generate views at unseen viewpoints implies that they actually store far more images than just the original data, which could significantly increase the effective compression rate.

Finally, we show in Figure 3.10 the result of upsampling along the temporal

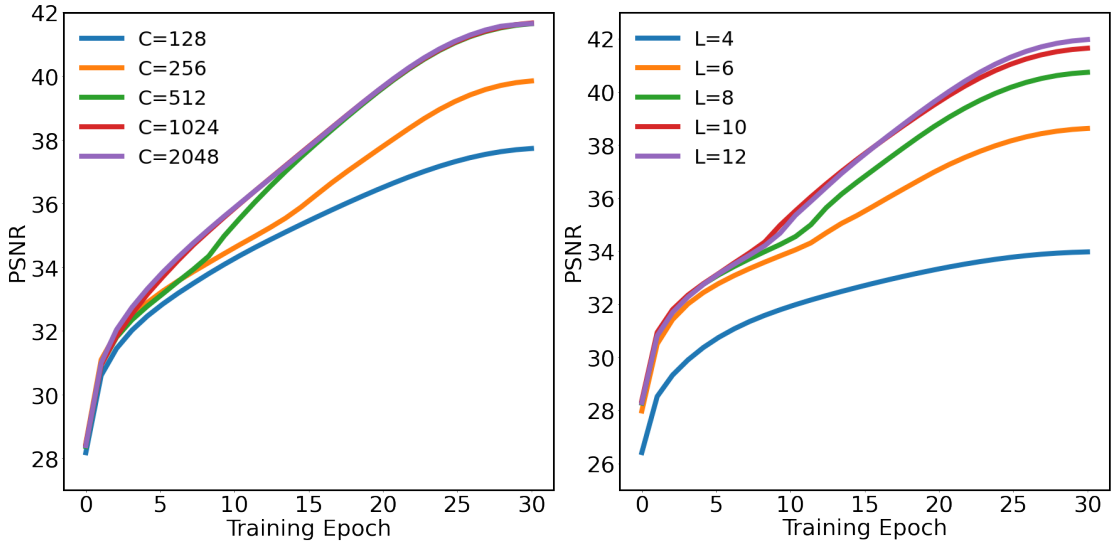
dimension on light field videos. Such results again demonstrate the trained network not only memorizes the training samples, but also implicitly derives the motion pattern between two frames. SIGNET produces realistic motion trajectories between frames, despite the fact that every pixel coordinate is independently evaluated. These results also suggest the potential of extending this MLP-based approach to interpolation on conventional videos, 3D volumes, and point clouds, which is out of the scope of this work.

3.5.4 Ablation Studies

For simplicity, in this section we use the static light field scene, *Lego*, to train our networks. We first examine the effect on network performance when we modify C, the total number of Gegenbauer basis orders. We set hidden layer length $L = 10$ and test network performance with different input sizes $C = \beta C_u + \beta C_v + \beta C_x + \beta C_y$ with $\beta = 0.25, 0.5, 2, 4$ and the Cs same as in Table 3.3. In Figure 3.11a we show the training curves, and the performance gain from increase in the input basis size apparently diminishes after $C > 1024$.

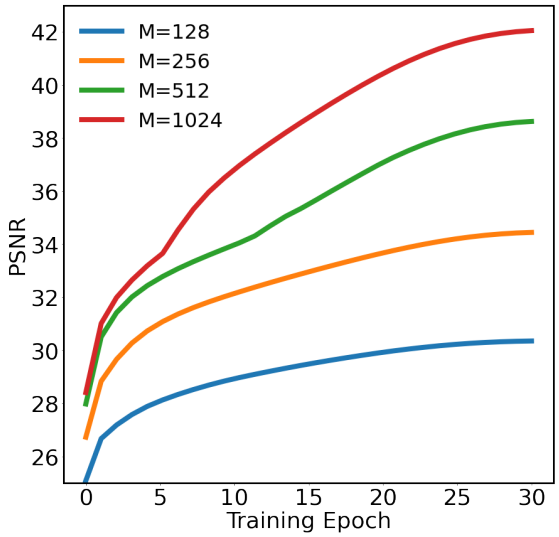
We then investigate how the network performance changes as we modify the number of network layers, L. We set the dimension of all intermediate layers to be 512×512. In Figure 3.11b, we show how the reconstruction quality changes for different configurations. The network accuracy saturates as the network expands to 10 layers.

In Figure 3.11c we show the effect of changing the matrix dimension size M

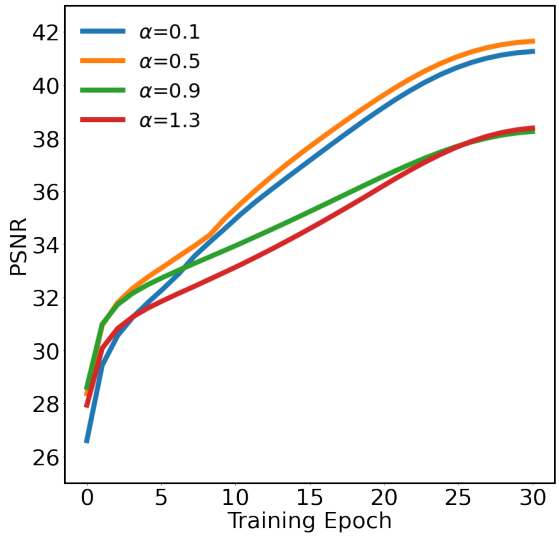


(a) Network Input Size C.

(b) Network Depth L



(c) Network Width M



(d) Gegenbauer's α

Figure 3.11: Ablation Studies. We change different aspects of SIGNET as discussed in Section 3.5.4.

for all hidden layers. We set the hidden layer length as $L = 4$ and the number of input basis functions as $C = 512$. As expected, we do see higher quality with more bases, although the quality also saturates to a certain ceiling. Increasing L seems to

Table 3.3: We provide network parameters used to achieve results shown in Table 3.2. $C = C_u + C_v + C_x + C_y + C_t$ as defined in Section 3.4. M is the dimension of the MLP layers. For simplicity we use $M \times M$ square matrices for all intermediate layers, where $M = 512$. L refers to the number of MLP layers.

Scene	L	C	C_u	C_v	C_x	C_y	C_t
Lego	10	512	16	16	240	240	0
Tarot	10	512	16	16	240	240	0
Bracelet	10	2048	144	144	1080	680	0
Painter	10	512	0	0	260	180	72
Trains	10	512	0	0	260	180	72

be a more effective way to increase SIGNET capacity and approximation accuracy, since it only moderately increases the storage cost; incresing M barely improves performance but significantly increases the storage cost.

We also examine the impact of varying α, a hyperparameter of Gegenbauer polynomials. In addition to our default choice of $\alpha = 0.5$, we test several other values and we show the results in Figure 3.11d. The results suggest that our default choice is reasonable, since the network performance seems to decline as α becomes larger than 0.5.

3.6 Discussion and Limitations

Compared to prior work on light field compression, SIGNET distinguishes itself with the ability to decode at coordinates not captured in the original data, thanks to its functional design.

Our results could be further enhanced by traditional image coding schemes which quantize and further encode spectral coefficients and residuals. As a quick check, we implemented quantization and weight pruning on our network weights. Our preliminary results show a further 15-fold bitrate reduction while PSNR decreased by less than 0.1. Additional weight compression techniques such as knowledge distillation could further enhance our compression rates. In contrast to prior work, SIGNET would not require sending the original light field images to the users; only the MLP weights are sufficient for decoding a high-resolution and densely sampled light field at arbitrary viewpoints.

While SIGNET achieves high reconstruction quality, a limitation is we need to retrain the network for every new scene. Latest advances [100, 101] have presented promising results in using meta-learning to speed up the training of coordinate-input MLPs. We believe that meta-learning strategies are likely to drastically reduce the training cost, and we leave explorations on this idea for future work.

Although we briefly showcase the ability of the trained networks to perform upsampling, we do not focus on further exploiting its potential in this direction. Our method is not guaranteed to achieve satisfactory view synthesis results on sparsely sampled light fields (e.g. light field with 3×3 viewpoints with a large disparity), since

there is insufficient data along the angular dimensions for the network to approximate smooth interpolations. In the future, it would be desirable to design neural networks that can generalize across different scenes utilizing learned prior knowledge.

3.7 Conclusion

We present SIGNET, a novel framework to represent light fields with neural networks, which achieves high-fidelity reconstruction and state-of-the-art compression performance. We hope SIGNET could motivate more research into utilizing neural networks for processing light fields and other high-dimensional visual data.

Chapter 4: Neural Subspaces for Light Fields

4.1 Introduction

Light field content can display scenes at unprecedented levels of realism and immersion. However, the vast amount of data in high quality light fields exceeds the storage, transmission, and interactive rendering capabilities of current graphics pipelines. Previous efforts to compactly represent light field content were mostly built on established concepts in image and video compression. Nonetheless, they still fall short of enabling large-scale, real-time dissemination of high-resolution light field content.

Recently, neural representation of light fields [10] has emerged as a viable alternative to conventional compression methods for efficiently storing light fields with high fidelity. For a given light field, a neural network is trained to learn the mapping function between each light field pixel's coordinates and color values. The key to achieving high accuracy of this neural network is to transform the input coordinates with a set of basis functions so that the network can more easily capture the high-frequency details. For instance, Fourier basis and Gegenbauer polynomials have been shown to be effective in transforming light field coordinates for a more accurate network [10, 58–60]. After training, multiple viewpoints from a light field

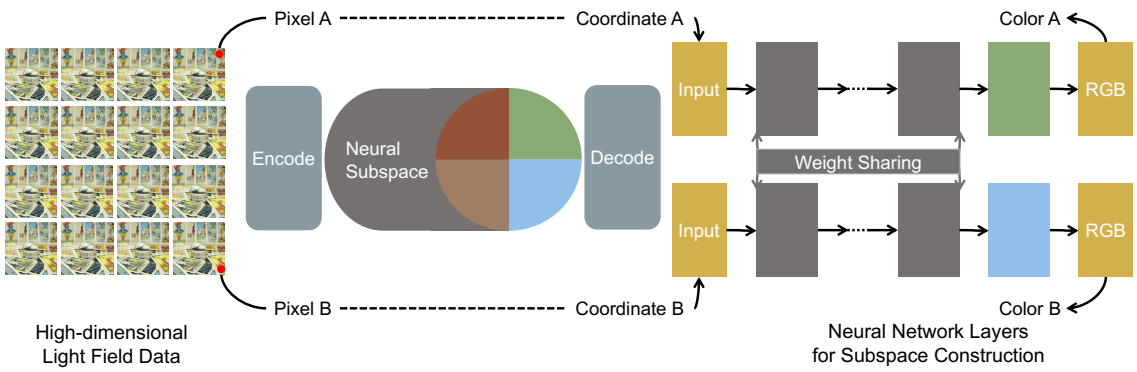


Figure 4.1: We illustrate our proposed concept of neural subspaces for light fields. Given a light field scene, we divide it into multiple local segments and construct a neural subspace for each segment. This figure shows an example of a light field partitioned into four segments in space; the same concept could be similarly applied to light field videos with segments across different periods. Each neural subspace of a light field is parameterized by a multi-layer neural network, which outputs the color value given its coordinates as input. The neural subspace construction is equivalent to training the network parameters to learn accurate coordinate-to-color mappings. We further introduce an adaptive weight sharing strategy, which implicitly utilizes the similarity among nearby subspaces and reduces the total number of parameters to represent the entire light field. Compared to previous methods, this new framework achieves a higher light field compression rate for storage and streaming without sacrificing reconstruction accuracy.

scene (hundreds of megabytes) are compactly encoded into such a neural network's weights (a few megabytes). These weights are the only necessary information for storage and streaming. Moreover, this trained neural network can be evaluated

at arbitrary viewpoints, thus quickly enabling interpolation for a smooth viewing experience.

This unified design of training a single neural network to cover the entire light field has simplistic appeal as a concept. However, it is not ideal in practical scenarios that emphasize efficient transmission and rendering. Such scenarios generally involve only rendering a subset of light field viewpoints. Therefore we incur unnecessary costs to transmit and evaluate a single neural network that contains other unrelated views. This question naturally arises: how can we improve the neural representation of light fields to accelerate decoding/rendering at a particular viewpoint without sacrificing visual quality?

In signal processing research, subspace learning is a well-known concept for dimensionality reduction of high-dimensional data. Its most notable example involves using principal component analysis (PCA) to find a small number of orthogonal basis vectors (principal components) with most data variability. Naively constructing a single subspace without adequately considering the structural attributes of the data would be suboptimal. If the data contains patterns that evolve over a long period, or the data are simultaneously captured by multiple sensors at different locations, constructing a global subspace may require many parameters to adequately represent the data, which would defeat the purpose of dimensionality reduction. It would be more sensible to divide such inherently composite data into multiple local subspaces such that each corresponds to a small segment or region of the data. The subsequent problem of tracking the changing subspace of data with such composite structures is often referred to as *subspace tracking*.

We first recognize the conceptual connection between the neural representation of light fields and the construct of subspace learning. When a neural network is trained to encode a given light field scene, it reduces dimensionality on the high-dimensional light field data. Specifically, the neural network succinctly describes the pixel coordinate to color mapping relationship. The fact that basis functions such as the Fourier basis or Gegenbauer polynomials improve network accuracy is deeply connected to how an appropriate basis function forms a more suitable subspace to analyze the data. Effectively, this network is trained to parametrize through its weights a *neural subspace* of the light field data.

Moreover, in the initial neural representation of light fields, one global neural subspace is learned for every part of a light field scene, leading to potential inefficiencies in scenarios that concern only a subset of the scene. This is similar to the issues that motivate subspace learning for composite data, prompting us to construct multiple local neural subspaces for light field data. Indeed, it is not technically necessary to treat light fields as one unified entity and encode the viewpoints altogether - light fields can be regarded as a composite collection of local segments. Such a perspective is meaningful in practice since only a subset of the light field might be relevant at a particular moment for streaming or rendering. Moreover, it is plausible that networks specialized in local segments could represent the data more accurately than a global network while using fewer parameters overall.

This paper presents a novel approach to constructing neural subspaces for light fields, with main ideas illustrated in Fig. 4.1. With experimental results, we demonstrate improving parameter efficiency and reconstruction accuracy under this

novel perspective. Unlike the initial attempt that trains one global network for the entire light field, we train a set of local neural networks that encode only a subset of viewpoints. As each local network specializes in a particular region, this specialization permits smaller networks without sacrificing accuracy, if not further boosting accuracy. Next, recognizing the similarity among nearby subspaces, we propose a weight-sharing strategy for those local networks to enhance overall parameter efficiency while maintaining network capacity within each subspace. Effectively, this proposed strategy achieves the tracking of implicit neural subspaces. Furthermore, we introduced a new soft-classification technique for neural representations to produce the RGB color predictions, which further improves the reconstruction accuracy. Experimental results on various light field scenes indicate that the proposed framework leads to better efficiency and accuracy than the original neural representation of light fields and a range of previous methods.

In summary, our contributions are as follows:

- We introduce neural subspaces for light fields, a general framework for improving the compactness of neural representation of light fields via subspace learning.
- We propose a soft-classification technique to predict the pixel color values, which further improves the reconstruction quality.
- We propose an adaptive weight sharing strategy to improve parameter efficiency by leveraging the layered structure of neural networks.
- We achieve results superior to the state-of-the-art compression on various light fields with our proposed light field encoding framework.

4.1.1 Scope

The work is focused on accurately encoding and decoding pixels captured in a light field. The related but orthogonal tasks of novel view synthesis or 3D volume reconstruction is beyond the scope of this work.

4.2 Related Work

4.2.1 Light Field Compression for Streaming

Traditional light field compression relies on classic coding strategies that typically involve analytical basis functions such as the Fourier basis and wavelets. Prior research has augmented this analytical approach with disparity [61–63] and geometry information [65]. Some sophisticated applications of light field video [66,67] also integrate motion prediction [102] and build on existing video codec algorithms such as HEVC (H.265) [69] and VP9 [70]. More recently, Le Pendu *et al.* [71] propose a Fourier Disparity Layer representation specific for sub-aperture light fields, which allows upsampling [72] and compression [73, 74] in the Fourier domain.

A different approach towards light field compression involves learning a dictionary of basis functions. This idea is inspired by progress in sparse coding from machine learning, where dictionaries learned with data-driven algorithms have been shown to outperform analytical basis functions [75–78]. However, the dictionaries learned with conventional algorithms such as K-SVD [7] still contain too much redundancy and have a high storage cost. The state-of-the-art dictionary-based

method [6, 79] for light field compression improves this approach by learning an ensemble of orthogonal dictionaries with a novel pre-clustering strategy.

Recently, a hierarchical tree structure of light fields has been introduced [64], where a root node stores a representative key view image for a given light field, and the child nodes store the sparse residuals (difference from their parent node's image) for each remaining image. Compression is achieved by only encoding the sparse residuals of those children's views instead of their full-color values. Specifically, they use the standard JPEG2000 to encode the representative key view and Bounded Integer Sequence Encoding [103] for the sparse residual views. Pratapa *et al.* [102] later extend this hierarchical method to incorporate the motion prediction and compensation concept widely used in traditional video encoding techniques.

While previous research has made substantial progress, most methods remain conceptual extensions to traditional image compression methods, which independently treat each image patch or camera view. Such methods essentially achieve compression by taking advantage of the redundancy within each patch or viewpoint without exploring potentially more optimal strategies that further utilize the redundancy between views or patches and between different time steps in the case of dynamic videos.

This paper presents a method based on the neural subspace that improves the compression rate through an adaptive weight sharing strategy. Specifically, we utilize the implicit redundancy between viewpoints or time steps to explicitly reduce the number of parameters to represent the entire light field scene.

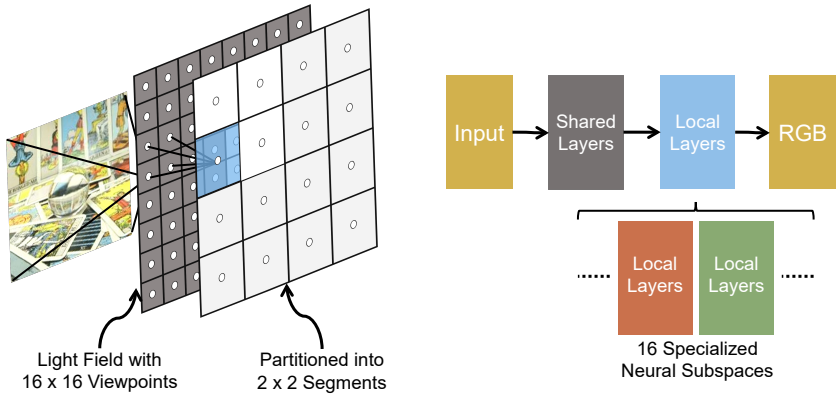


Figure 4.2: We illustrate the proposed weight sharing strategy for a classic 4D light field. The light field is partitioned into segments, in this example 16 segments, each containing 2×2 viewpoints. We construct a neural subspace for each segment that summarizes its pixel-to-color mapping relationship. Each subspace shares a set of network layers while possessing its local layers that enable network specialization on its corresponding data segment.

4.2.2 Neural Light Field

Recent research [58–60] has shown the potential of using coordinate-input MLP networks to represent multidimensional data such as audio signals, images, videos, volumes, signed distance functions, and wave equations. Fourier-inspired techniques are the key to their breakthroughs in accurately approximating content with high-frequency details. The neural radiance field (NeRF) network [58] achieves state-of-the-art free viewpoint synthesis on static scenes by transforming the input coordinates with cosine and sine functions. Tancik *et al.* [60] further analyze this transform strategy, which is closely related to the Fourier basis functions, and show its effectiveness on 2D images. Concurrently, SIREN networks [59] replace the ReLU

activation in conventional MLPs with a sine activation, also achieving accurate coordinate-to-color mappings for images and videos.

While it is straightforward to re-purpose those Fourier-inspired MLPs into light field representation by setting light field coordinates as the network input, they achieve less than ideal results [10]. Feng *et al.* introduce SIGNET, an MLP that encodes light field with high fidelity through a novel coordinate transformation strategy based on the Gegenbauer polynomials. They further demonstrate the compression capability of such a neural representation of light fields and show SIGNET's superior performance over the state-of-the-art compression methods based on sparse dictionaries.

This paper extends the initial success of neural representation of light fields and shows that its formation generalizes to learning a concept called neural subspace. Moreover, we show how the concept of subspace learning reveals the inefficiencies of not adapting to the different compositions of light field data. We then present how our techniques significantly optimize the parameter efficiency compared to using the original architectures of SIGNET.

Several recent works closely relate to light field representations with neural networks, but they focus on different problems than this paper. Kalantari et al. [86] and Bemana et al. [84] use neural networks to interpolate between nearby light field viewpoints, but the networks do not represent the known light field images themselves. Recent developments of neural light fields have also achieve successes in modeling appearance [104] and geometry [105] in unbounded 3D scenes, as well as view interpolation without geometric structures [106]. In the context of video compression

without considering multi-view redundancies, implicit neural representations [107] have also been shown to outperform established neural video codecs.

4.2.3 Subspace Learning and Tracking

Subspace learning and tracking have been instrumental for signal processing and computer vision applications. In the traditional settings for learning subspaces from multidimensional data, the purpose is to detect meaningful patterns or correlations using a few parameters constructing a reduced number of dimensions. This technique is useful for applications such as object tracking from radar or video [108, 109], image denoising [110], and video background removal [111]. Since these tasks generally only require a coarse representation of the data points, the traditional subspace learning or dimension reduction methods are sufficient and have led to proven successes. However, the problem of constructing neural light fields requires decoding each light field pixel as accurately as possible, which is hard to achieve by traditional subspace learning methods, which only provide a rough estimation of the general patterns in the data.

We propose a framework that utilizes the expressive power of neural networks to construct a highly accurate and condensed representation of the subspace of our light field data. Compared to previous methods for light field compression, we tackle this problem from a different perspective based on the newly proposed neural representation paradigm. We break away from previous methods that use various basis functions to construct an analytically defined subspace. Instead, we propose

learning a neural subspace, which is realized by multiplying the input basis functions through a sequence of matrices with non-linear activations (MLP network), whose parameters (network weights) are adaptively learned for each light field scene. More importantly, we introduce a strategy to divide the neural subspace into segments properly. Then, starting from the learned neural subspace at one segment, we devise a method to update it for another segment without re-learning every parameter from scratch. This specialization allows us to downsize the total representation and the memory required to decode each pixel.

4.3 Method

4.3.1 Neural Light Fields with MLP

A neural light field representation [10] uses an N-layer neural network (MLP) to encode a light field (sub-aperture in the original case). Formally, it learns the following function $\mathcal{F}$:

$$\mathcal{F}(u, v, x, y) = \phi_L \circ \phi_{L-1} \circ ... \circ \phi_1([\Theta_i(u, v, x, y)]_{i=1}^{N}) \tag{4.1}$$

Here, (u, v, x, y) stands for the coordinate of a pixel within a sub-aperture light field, Θ_i stands for the orthogonal basis functions used to transform the coordinates, ϕ_l stands for the l-th layer of the neural network with a weight matrix W_l, bias vector b_l, and an activation function $\sigma(x) = sin(x)$. The output from each layer is $\phi_l(x) = \sigma(W_l x + b_l)$.

The MLP predicts the pixel color separately as $\mathcal{F}(u, v, x, y) = \tilde{\mathbf{c}}_p$. The training

objective directly minimizes the regression error between the predicted color $\tilde{\mathbf{c}}_p$ and ground truth $\mathbf{c}_p$ across all known pixels in a light field:

$$\mathcal{L}_{Regress} = \sum_{p} \|\tilde{\mathbf{c}}_p - \mathbf{c}_p\|^2. \tag{4.2}$$

4.3.2 Constructing Light Field Segments

A global MLP is responsible for learning the coordinate-to-color mapping for all the pixels in a sub-aperture light field in the original neural representations for light fields. As discussed in Section 4.1, such a unified construct would be suboptimal in other camera configurations. Instead, we first partition a given light field into different segments in our method. For each segment's content, we then train an MLP to encode its coordinate-to-color mapping $\mathcal{F}$.

There are two aspects in which we could partition a given light field: space and time. To partition a light field in space means that we group camera viewpoints into separate clusters. To partition a light field in time means dividing a light field video into separate time intervals. In prior subspace learning and tracking methods, this partitioning strategy is key to achieving efficient dimension reduction of high-dimensional data. In our case of light field encoding, we conjecture that this partitioning strategy would significantly improve the parameter efficiency of light fields compared to SIGNET.

For example, consider a sub-aperture static light field with 16×16 viewpoints. Suppose we train an MLP that specializes in the particular segment of the top-left 4×4 viewpoints. In that case, it could require much smaller weight matrices to

achieve the same reconstruction accuracy compared to training a global MLP for all 16×16 camera viewpoints. In the case of light field videos, we could similarly extend the partition strategy to divide the temporal dimension into several time intervals.

4.3.3 Adaptive Weight Sharing in MLP

It is not hard to imagine that if we partition a large light field into segments, we could use a much smaller MLP to encode each segment because it now needs to encode a simpler neural subspace. Nonetheless, is that the best we could do to make the neural subspaces as compact as possible?

A distinctive feature of the parametrization of the neural subspace is that each neural network contains a series of layers. Suppose every network contains N layers, and we partition the light field into D segments. Then if we naively train an MLP for each segment, we would need to learn and store ND weight matrices for those layers.

We observe that this approach leads to possible parameter redundancy. More concretely, although each MLP is constructing a different neural subspace, nearby subspaces inevitably contain similar patterns. The redundancy occurs when we use too many parameters to represent those similar patterns. Therefore, it makes sense to let MLPs covering nearby segments share a portion of their parameters. For instance, we may let the D segments share the first K matrices in their neural subspace construction, and thus we could only learn and store $(N-K)D+K$ weight matrices instead of ND matrices. In Fig. 4.2, we illustrate our modification to

Scene	Method	PSNR(↑)	SSIM(↑)	EncMB(↓)	DecMB(↓)
Lego 909 MB	AMDE	40.90	0.973	29.3	29.3
	KSVD	38.39	0.960	29.3	29.3
	SIGNET	41.26	0.976	9.0	9.0
	Ours	41.95	0.982	22.9	2.2
Tarot 909 MB	AMDE	38.54	0.973	44.2	44.2
	KSVD	38.81	0.980	44.3	44.3
	SIGNET	37.47	0.975	9.0	9.0
	Ours	38.21	0.975	22.9	2.2
Bracelet 568 MB	AMDE	39.90	0.980	18.1	18.1
	KSVD	36.73	0.973	18.1	18.1
	SIGNET	38.70	0.973	12.0	12.0
	Ours	39.64	0.985	22.9	2.2

Table 4.1: We provide results on sub-aperture light fields. Comparisons to previous methods AMDE [6], KSVD [7], and SIGNET [10]. *DecMB* is the memory (in MB) required in streaming to decode any frame, and *EncMB* is the memory (in MB) required to encode the entire light field in storage.

the MLP layers in the classic case of a sub-aperture light field. These modifications could be easily extended to light field videos by sharing weights among adjacent time intervals instead of nearby viewpoints. Initially, we train the network at Level 0, optimizing all N layers for all light-field pixels. After the first round of training,

we pick the $(N-K)$-th layer as the starting point of fine-tuning. Specifically, when fine-tuning at Level 1, we freeze the first $N-K$ layers trained at Level 0 and only optimize the last K layers. In the example shown in Fig. 4.2, this step is equivalent to creating four copies of the last three matrices and then separately fine-tuning each copy on one patch of pixels at Level 1. When finetuning Level 2, we additionally freeze the $(N-K+1)$-th layer (green matrix in Fig. 4.2) and train the remaining $K-1$ layers.

4.3.4 Soft-Classification for RGB Prediction

RGB color values are the common network output in neural representations of light fields, images, videos, and radiance fields. Visual information is often stored as 8-bit color with 256 possible discretized values, but are usually normalized to be within 0 to 1, before being used to train those neural representations. As a result, the final layer of the neural network is often designed to directly output three scalar values corresponding to the normalized RGB values. Effectively, the network weights, along with the final activation function (such as linear or sigmoid), perform a direct regression on the color values.

While this standard direct regression approach is sensible and works well in practice, it is not the only valid approach. A naive alternative would be to let the network classify, out of all possible colors, which specific color the input pixel belongs to. However, the 8-bit RGB format essentially covers $(2^8)^3 = 2^{24}$ individual colors, which would require the final network layer to have $M \times 2^{24}$ parameters and defeat

the purpose of using this network for compression.

Instead, we let the network classify each of the three color channels separately. Specifically, its final layer outputs a vector with 768 values, which we then reshape into a 3×256 matrix. We then apply the softmax function separately for each of the three rows. As a result, the 256 values in each row represent the probability of those 256 possible colors being the true color of this channel.

Common machine learning applications adopt hard classification, where the final prediction is determined as the class with the highest corresponding probability. This is appropriate when the classes are discrete and have no ordered relationship (e.g. predicting dog/cat/chicken). However, in our setting, the possible classes are by definition ordered, and thus we do not have to adopt hard classification in practice. Instead, we multiply the predicted probability with their actual value (from 0 to 255), and then sum them up as the final output. As shown in later experimental results, our soft-classification approach enhances the reconstruction quality of our neural representation. Moreover, compared to hard classification, it does not restrict the output to be discrete color levels; it has the same flexibility as direct regression to produce any floating numbers with comparable numeric precision.

4.4 Experiments

In this section, we provide detailed analysis of our method through quantitative experiments. First, we confirm its overall validity by comparing its compression performance with several prior methods on traditional sub-aperture light fields.

Second, we validate our use of the adaptive weight-sharing strategy, juxtaposing it with an alternative approach that seems potentially more effective but achieves inferior results in practice. Third, we demonstrate the benefit of predicting RGB color using our proposed soft-classification strategy. Fourth, we show the efficacy of our framework on dynamic light field scenes where its streaming-friendly property flourishes. Finally, we analyze the impact of the varying the hyper-parameters of the network architecture.

4.4.1 Training Setup

We train the MLPs in PyTorch using the Adam optimizer with default parameters in all our experiments. We supervise the network based on the mean squared error loss between reconstructed and ground truth colors. We train and test the networks under the automatic mixed precision mode in PyTorch using 16-bit floats. We adopt the cosine-annealing learning rate scheduler, with the initial learning rate as 1×10^{-5} and the final learning rate as 1×10^{-8}. We randomly shuffled all the training samples and divided them into batches of size equal to 2048. To ensure reproducibility, we set the random seed as 0.

Unless otherwise noted, we set all networks to have ten layers with residual connections, where the intermediate layers have a dimension size of 512×512. We apply layer normalization after all MLP layers except the final one.

For networks with sine activation functions, we follow the initialization method suggested by Sitzmann *et al.* [59].

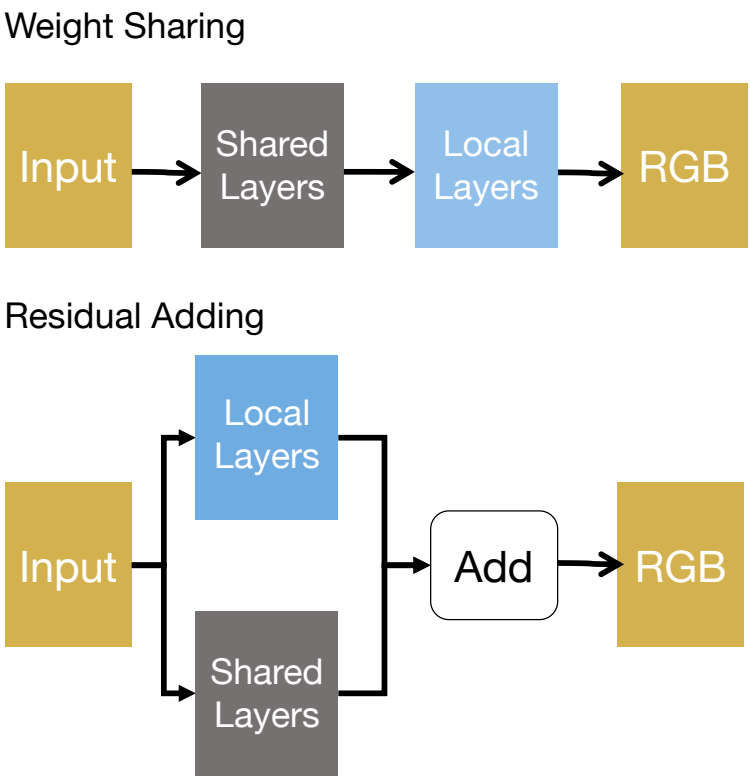


Figure 4.3: We show two possible strategies to incorporate local subspaces into network architecture. Weight Sharing fine-tunes the later network layers, while Residual Adding adjusts the features through addition for different subspaces. Our empirical results show Weight Sharing achieves better quality while using fewer parameters.

4.4.2 Quantitative Metrics

To evaluate the reconstruction accuracy, we compute PSNR and SSIM, two commonly used metrics for measuring the fidelity of image data. To holistically evaluate the parameter efficiency, we compute two metrics: *DecMB*, the memory required to decode any frame in megabytes completely, and *EncMB*, the memory

Method	PSNR(↑)	SSIM(↑)	DecMB(↓)
SIGNET	39.14	0.975	9.0
+ Residual Adding	35.30	0.943	2.3
+ Weight Sharing	38.91	0.972	1.8
+ Hard Classification	39.26	0.979	2.2
+ Soft Classification	39.93	0.982	2.2

Table 4.2: Quantitative Comparison of Different Strategies. Weight Sharing achieves subspace specialization with better quality than Residual Adding, despite using fewer parameters. Soft Classification achieves better performance in improving prediction accuracy over Hard Classification and Direct Regression (used by SIGNET).

needed to encode the entire light field for storage in megabytes. In the context of streaming applications, *DecMB* better reflects the cost for real-time transmission and on-device decoding and rendering. In contrast, *EncMB* captures the total storage cost of the server and is less relevant to the cost of large-scale, repetitive content transmissions.

4.4.3 Sub-aperture Light Fields

We first deploy our framework on the classic sub-aperture light fields from the Stanford Light Field Archive [98]. We report the performance of our methods compared with previous methods in Table 4.1, using the same set of scenes selected by Miandji et al. [6] for a direct comparison. Results indicate that our method achieves

a significant (ranging from 5.2x to 12.5x) reduction in the decoding bandwidth cost with improved reconstruction accuracy.

4.4.3.1 Comparison with the Residual Approach

While the above preliminary results may serve as a proof of concept of the overall framework, we now demonstrate the specific importance of our weight-sharing strategy in enabling subspace specialization. We present an alternative approach with a residual adding setup, shown in Fig. 4.3. For this approach, instead of finetuning the later layers for a local segment, we train a parallel path of layers that learns to produce the residual error of the original path. This residual adding approach is similar to many conventional modifications to neural network architectures [112–114], and on the surface, it might seem more promising than the weight sharing setup. However, as the results in Table 4.2 and 4.7 and Fig. 4.9 show, it is much less effective in accurately constructing a local neural subspace, even if we allow it to use more parameters than the weight sharing approach.

4.4.3.2 Improved Accuracy with Soft Classification

Using the sub-aperture scenes, we also analyze the impact of using different strategies to obtain the final color prediction. We train our network with Soft Regression, Hard Classification, and Soft Classification as discussed in Section 3. Soft Regression directly outputs the predicted color values as done in [10]. Hard Classification uses the color level with the highest predicted probability as the

output for each channel. Soft Classification aggregates the predicted probability with multiply-then-sum approach introduced in Section 3. We measure the performance on the scenes as in Table 4.1, and the averaged results are provided in Table 4.2.

4.4.4 Volumetric Light Fields

Scene	Method	PSNR(↑)	SSIM(↑)	EncMB(↓)	DecMB(↓)
Cup (370MB)	SIGNET	38.04	0.955	9.0	9.0
	Naive	29.73	0.898	0.38	0.38
	Ours	40.64	0.967	5.2	0.46
Violin (370MB)	SIGNET	37.69	0.953	9.0	9.0
	Naive	28.99	0.891	0.38	0.38
	Ours	40.40	0.966	5.2	0.46
Training (370MB)	SIGNET	39.29	0.970	9.0	9.0
	Naive	30.38	0.911	0.38	0.38
	Ours	42.81	0.982	5.2	0.46

Table 4.3: Results on Volumetric Light Fields. Naive refers to naively reducing the width of the MLP networks without specializing on subspace segments as described in Section 4.3. Our results achieves higher accuracy and better parameter efficiency than SIGNET. Compared to the Naive method, our subspace-based method better balances the trade-off between streaming and storage efficiency.

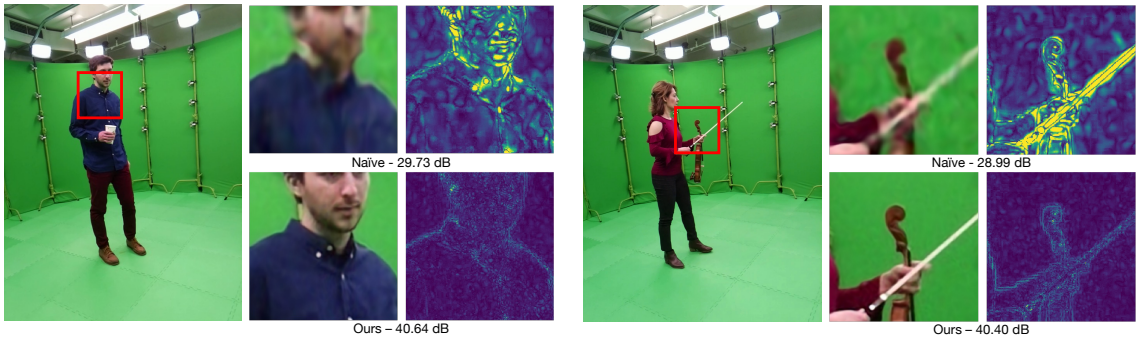


Figure 4.4: We provide comparisons on the cylindrical light field scenes. We show the original image (left), reconstructed images using the Naive method (top right), and our method (bottom right). The Naive method only uses one MLP network to encode the content (similar to a slim SIGNET), but without the subspace specialization and weight sharing described in Section 4.3. We design the MLP such that the Naive method incurs a similar streaming bandwidth cost. The residual errors are visualized in yellow. Our subspace setup achieves superior visual quality.

The results thus far validate our method's usefulness on classic light field scenes, and they also approve the specific implementation of our weight-sharing strategy. Nonetheless, the neural subspace method truly shines in the case of large-scale light fields with irregular camera layout and even dynamic content. In this scenario, it is reasonable to assume that only a small portion of the light field scene must be streamed and displayed in real-time. It makes more sense to encode each local subspace separately to reduce the streaming bandwidth.

Therefore, we try to relax the assumption of only focusing on classic light fields with the two-plane parameterization. We create our dataset of room-scale light field scenes captured using 64 synchronized motion cameras at the resolution

of 1640×1232 and 30 frames per second. Unlike the classic light fields where all the viewpoints exist on a single 2D plane and are restricted to the same front-facing orientation, our 64-camera setup contains viewpoints distributed on a cylindrical surface. It provides a 360° all-around coverage of the scene. For each scene, we record 210 frames and capture various human movements.

Compressing high-resolution scenes with such a structural layout is not straightforward for prior methods. Many prior methods for light field compression are tailored to light fields with viewpoints restricted to a 2D planar grid [64, 87, 102] . The non-planar layout of the cameras results in both translation and rotation among the viewpoints, which is ill-posed for these compression methods that utilize translation between viewpoints to represent repetitive patterns compactly. On the other hand, some other methods [6, 7] explicitly construct a dictionary for encoding a new set of content, and they require sending this sizable dictionary for decoding, which is not ideal for streaming applications. On the contrary, neural-network-based methods like SIGNET and ours are flexible towards different camera layouts without a dictionary requirement. Considering their superior compression performance over previous methods already shown in Table 4.1, we thus primarily focus on the neural network-based methods in this section.

Since the light field scenes, in this case, cannot be parametrized by two planes, we label the cameras using cylindrical coordinates, which make up the angular part of each light field pixel's input coordinate to the MLP network.

Let us consider encoding a static moment captured by the volumetric light field. First, we directly deploy the original SIGNET design on our dataset by training

Scene	Method	PSNR(↑)	SSIM(↑)	MB/s(↓)	Δf(↓)
Vid-Cup	MPEG	37.12	0.959	0.15	44.89
	Ours	38.21	0.968	0.15	45.25
Vid-Violin	MPEG	37.54	0.957	0.15	38.39
	Ours	38.11	0.962	0.15	38.31
Vid-Training	MPEG	37.61	0.955	0.15	39.95
	Ours	38.14	0.962	0.15	40.12

Table 4.4: Comparison to MPEG on Dynamic Light Field Videos. Controlling for the same level of bitrates, our method achieves higher fidelity without the blocking artifacts commonly seen (e.g. Fig. 4.5) in traditional, patch-based methods represented by MPEG. Δf measures the level of flickering as discussed in Sec. 4.4.2.

one MLP to encode all 64 images of each scene. Then, to achieve a streaming-friendly file size, we train a slim SIGNET version on our data by reducing SIGNET's network width by four times, which effectively cuts the number of parameters by sixteen times. We call this the Naive method since it merely reduces the network's representation capacity compared to a full SIGNET without adjusting to compensate for its representation quality.

Finally, we apply our neural subspace framework on these light field scenes. Each scene is divided into 16 segments. Each segment contains four viewpoints vertically aligned. We construct the neural subspaces across these 16 segments by training their corresponding MLP network weights. The MLP network's architecture



Figure 4.5: We provide results on Dynamic Volumetric Light Field scenes. For each example frame, we show zoomed-in comparisons between MPEG (top) and our method (bottom). At a similar bitrate level, our method avoids the blocking artifacts commonly seen in MPEG-compressed content.

is specifically designed to incur a streaming bandwidth cost similar to the Naive method. Specifically, the first four layers are shared across the subspace, while the remaining layers are adapted for each subspace. More architectural details are in Table 4.6.

In Table 4.3 and Fig. 4.4 we show the quantitative and qualitative results. Compared to SIGNET, our method considerably reduces storage and streaming costs while improving the reconstruction accuracy due to the local subspace specialization.

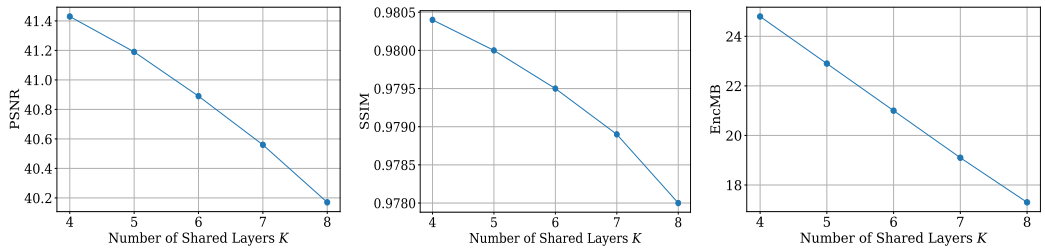


Figure 4.6: Impact of Changing the Number of Shared Layers K. As the number of shared layers increases, there are fewer subspace-specific layers as we maintain the total number of layers, reducing the benefit of subspace specialization. On the other hand, having more shared layers reduces the total storage cost *EncMB.*

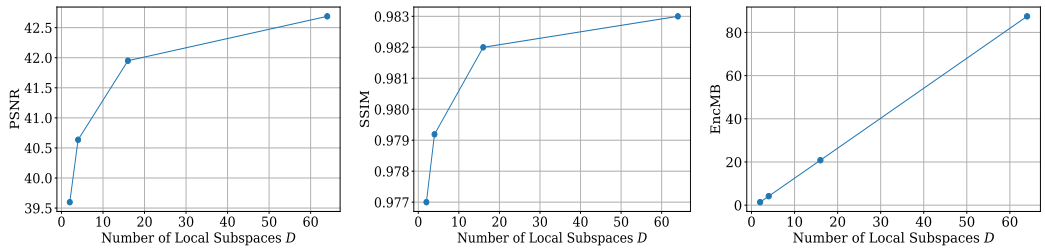


Figure 4.7: Impact of Changing the Number of Local Subspaces D. As the number of local subspaces increases, each subspace covers a smaller region in the light field, therefore reducing the network fitting difficulty and hence higher reconstruction quality. On the other hand, having more individual subspaces entails higher storage cost *EncMB.*

The Naive method, while achieving a lightweight file size suitable for streaming, fails to accurately encode the data, which shows the importance of our proposed strategy of subspace specialization.

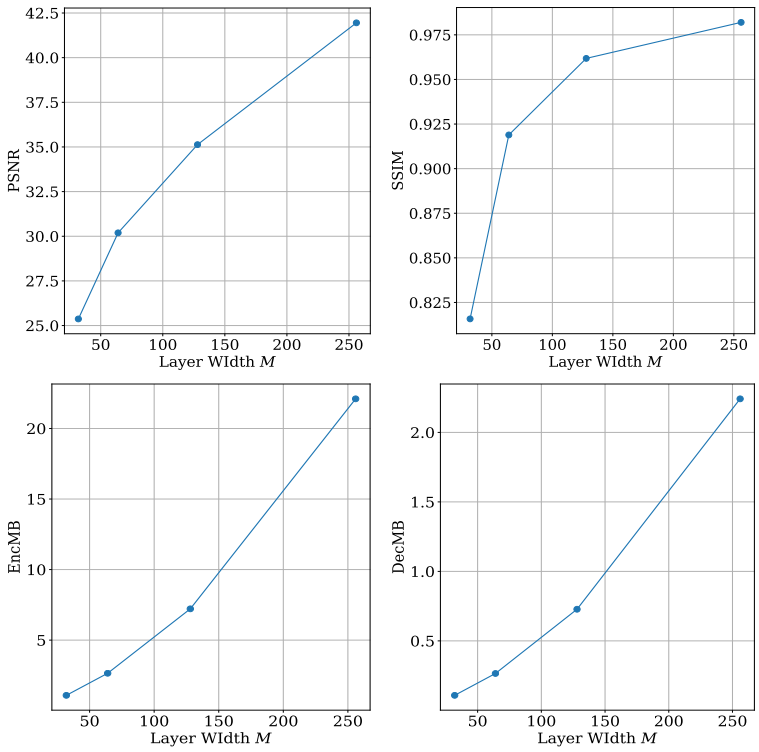


Figure 4.8: Impact of Changing the Layer Width M. As the width of the intermediate layers increases, the network has more capacity to fit the data and achieves higher reconstruction quality. However, the number of parameters required in storage *(EncMB)* and decoding *(DecMB)* increases at the same time.

4.4.4.1 Extension to Dynamic Content

We further apply our framework on dynamic sequences captured with our volumetric setup. We partition each light field scene into 14 temporal subspaces (each encompassing 0.5 seconds of content) and train the neural network weights for each subspace with the same layer allocation as in the static case. We provide the quantitative results in Table 4.4, and in Fig. 4.5 we show example frames reconstructed from our network juxtaposed with frames processed using the conventional MPEG

codec. We control the MPEG bitrates to be as low as possible without sacrificing the frame rate. Quantitatively, our method achieves higher fidelity and avoids the typical blocking artifacts of MPEG as in Fig. 4.5.

4.4.4.2 Possible Flickering Across Subspaces

To quantitatively measure the possible flickering when moving across subspaces, we adopt the Flicker metric [115, 116], which measures the temporal flickering across consecutive frames. Unlike previous works [115, 116] that measure the flickering across entire videos, we are interested in how the subspace division strategy affects the level of flickering. Specifically, assuming the previous subspace ends at time $t-1$ and the next subspace starts at time t, we compute the flicker $f_{t-1\to t+1}$. For all subspace initial time steps t, we record the difference in flickering between the reconstructed frames and original ground truth frames

$$\Delta f_t = |f^{Recon}_{t-1\to t+1} - f^{True}_{t-1\to t+1}|, \tag{4.3}$$

and report the average Δf in Table 4.4. The level of flickering exhibited between subspaces is similar to using MPEG.

4.4.5 Hyper-parameter Analysis

Our strategy to achieve subspace specializations through the network design involved several hyper-parameters, such as the number of shared layers K among the subspaces, the number of subspaces D to divide the light field, and the width M of the network layers. In Figs. 4.6, 4.7, and 4.8, we present the results of varying

each of those hyper-parameters while fixing all others, using the same dataset (*Lego*) and the same training batch size.

Changing K or D does not change the number of parameters required to decode any view, *DecMB* , while changing M affects both the storage cost *EncMB* and *DecMB*. When changing the number of subspaces D, $D = 2$ refers to separating all viewpoints into the left and right halves, $D = 4$ divides the viewpoints into quadrants, $D = 16$ is our adopted setup dividing the two viewpoint axes by 4, and $D = 64$ divides the two axes by 8.

4.5 Discussions

4.5.1 End-to-end versus Two-stage Learning

Our method essentially performs a two-stage learning, with the global stage of training the network to represent the entire light field, followed by a local stage where we finetune the layers specific to each subspace. A conceptually viable alternative is to combine the two separate stages and directly train the method end-to-end. Specifically, at each training iteration, we may dynamically select the subspace-specific layers for the current training batch.

However, we observe inferior reconstruction accuracy from such an end-to-end learned network. We believe a major reason is that, under the end-to-end learning scenario, each training batch comes from the same subspace. In contrast, in the implementation of this paper and SIGNET [10], all light field pixels are **randomly shuffled** and divided into batches at the first (global) stage. Compared

to just sampling 2048 pixels randomly **from one randomly selected image**, this completely randomized data preparation was adopted because it led to a better global accuracy. Therefore, when we use the end-to-end scheme with batches from the same subspace, it is not surprising that it learns an inferior global solution than training with completely randomized batches. If we use global randomized batches while dynamically selecting the subspace layers at each training iteration, we would need to split up the batch on-the-fly in every forward pass, sort the pixels by their subspaces, and then separately infer them using their own local subspace layers. This leads to approximately 20× slower run-time using PyTorch. In other words, this potentially helpful approach is unfortunately not computationally efficient enough under the current deep learning framework.

4.5.2 Limitations

Our method's encoding and decoding speed are inherently dependent on neural networks' training and inference speed on the available hardware. We train the MLP network weights from scratch to encode a new scene, which takes over an hour to achieve acceptable quality. Decoding a 1024×1024 image, the network takes around 0.258 seconds to complete the forward pass on an NVIDIA A4000 GPU. Another limitation is that our method currently only achieves lossy compression. It remains uncertain whether such neural-network-based methods could ever come close to lossless encoding under a prolonged training regime or if they could theoretically establish an error bound.

4.5.3 Future directions

It would be worthwhile for a future study to explore incorporating meta-learning or few-shot learning [117,118] to speed up the encoding, potentially utilizing the prior knowledge or re-using the weights learned from earlier light field scenes. Moreover, since the implicit neural representation naturally allows random access to individual pixels, another interesting direction is to combine it with foveated or view-dependent rendering techniques [119] to enable more efficient rendering. Considering the lossy nature of neural representation, it would therefore be a promising research direction to explore advancing such a neural-network-based method towards lossless compression.

4.6 Conclusion

As the intersection among neural networks, graphics, and multimedia processing is rapidly expanding, content creation and application possibilities based on high-resolution light field data are on the horizon. Compactly representing such densely sampled data is crucial for the light field's wider dissemination and impact on people's activities. Our work builds on the recent progress of neural computing. It invokes the classic idea of subspace learning, taking neural light field representations to the next step by making them more compact and streaming-friendly. We hope the new framework of neural subspace for light fields inspires more progress in bringing light field content closer to a ubiquitous component of our daily lives.

Scene	Method	PSNR(↑)	SSIM(↑)	EncMB(↓)	DecMB(↓)
Amethyst	SIGNET	43.23	0.972	9.0	9.0
	Ours	44.53	0.977	22.9	2.2
Beans	SIGNET	37.94	0.951	9.0	9.0
	Ours	46.63	0.986	22.9	2.2
Bulldozer	SIGNET	36.65	0.935	9.0	9.0
	Ours	40.23	0.971	22.9	2.2
Bunny	SIGNET	39.29	0.970	9.0	9.0
	Ours	41.36	0.979	22.9	2.2
Chess	SIGNET	38.09	0.972	9.0	9.0
	Ours	40.35	0.979	22.9	2.2
Flowers	SIGNET	40.29	0.965	9.0	9.0
	Ours	42.03	0.975	22.9	2.2
Treasure	SIGNET	38.35	0.965	9.0	9.0
	Ours	40.71	0.975	22.9	2.2
Truck	SIGNET	40.13	0.965	9.0	9.0
	Ours	42.72	0.972	22.9	2.2

Table 4.5: We provide additional results on sub-aperture light field data.

Method	L	K	C	M	D
SIGNET	10	0	512	512	1
Naive	10	0	128	128	1
Ours	14	4	512	128	16

Table 4.6: We provide network parameters for the cylindrical light field data.

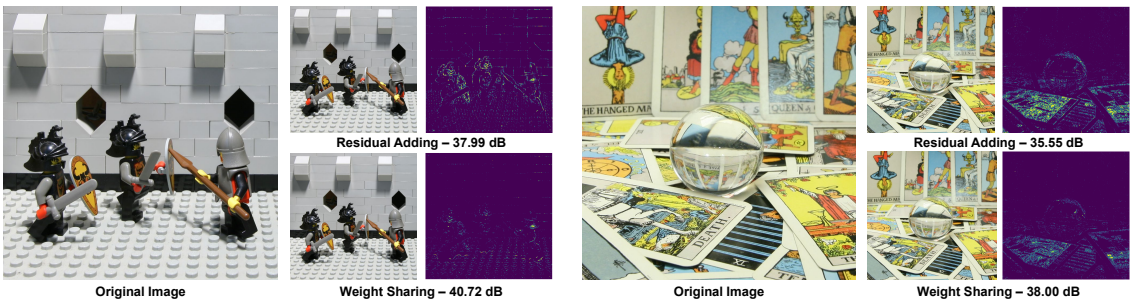


Figure 4.9: We show comparisons between Residual Adding (RA) and Weight Sharing (WS). Here, we show the reconstruction results (left) and error map (right) from two sub-aperture scenes using the two strategies described in Sec. 4.4.3.1. The Weight Sharing setup is more effective since it achieves a much lower level reconstruction error.

Scene	Method	PSNR(↑)	SSIM(↑)	EncMB(↓)	DecMB(↓)
Lego (909 MB)	SIGNET	41.26	0.976	9.0	9.0
	+ Residual Adding	37.99	0.962	9.9	2.3
	+ Weight Sharing	40.72	0.972	16.4	1.8
	+ Hard Classification	40.95	0.980	22.9	2.2
	+ Soft Classification	41.95	0.982	22.9	2.2
Tarot (909 MB)	SIGNET	37.47	0.975	9.0	9.0
	+ Residual Adding	35.55	0.968	9.9	2.3
	+ Weight Sharing	38.00	0.975	16.4	1.8
	+ Hard Classification	38.15	0.975	22.9	2.2
	+ Soft Classification	38.21	0.975	22.9	2.2
Bracelet (568 MB)	SIGNET	38.70	0.973	12.0	12.0
	+ Residual Adding	32.37	0.906	9.9	2.3
	+ Weight Sharing	38.00	0.971	16.4	1.8
	+ Hard Classification	38.67	0.984	22.9	2.2
	+ Soft Classification	39.64	0.985	22.9	2.2

Table 4.7: We provide detailed results on quantitative comparison of different strategies.

Chapter 5: Primary Ray-based Implicit Function

5.1 Introduction

Learning an accurate and efficient geometric representation of a 3D object is an important problem for computer graphics, computer vision, and robotics. Recent advances in machine learning have inspired a growing trend of implicit neural shape representations, where a neural network learns to predict the signed distance function (SDF) for an arbitrary location in the 3D space. Moreover, in addition to the 3D location (x, y, z), the neural SDF network may take in a latent vector that describes the object identity, thus enabling the generative modeling of multiple objects. Such an implicit neural representation (INR) not only produces fine-grained geometry, but also enables a plethora of applications [120], *e.g.*, shape completion, pose estimation, via a differentiable rendering based optimization.

However, rendering and extracting the shape from a trained neural SDF network are computationally expensive and often limited to watertight shapes. The direct approach to rendering from SDF requires sphere tracing, which needs access to the SDF values at multiple locations along each pixel ray [121]. The indirect approach computes and stores the SDF values at predefined 3D grid points, from which the shape can be rendered with sphere-tracing or extracted as a polygon mesh through

Marching Cubes [122]. Both cases demand a large number of network evaluations since they require sampling SDF values at locations far away from the surface. Moreover, the shape quality is ultimately constrained by the converging criteria of sphere tracing or the 3D resolution of the grid.

In this work, we present a novel geometric representation that is efficient, accurate, and innately compatible for downstream tasks involving reconstruction and rendering. We break away from the conventional point-wise implicit functions and propose to encode 3D geometry into a novel ray-based implicit function. Specifically, our representation operates on the realm of oriented rays $r = (\mathbf{p}_r, \mathbf{d}_r)$, where $\mathbf{p}_r \in \mathbb{R}^3$ is the ray origin and $\mathbf{d}_r \in \mathbb{S}^2$ is the normalized ray direction. Unlike SDF that only outputs the distance to the nearest but undetermined surface point, we formulate our representation such that its output directly reveals the surface hit point of the input ray. The rays whose surface intersection we care about are specifically known in computer graphics as *primary* rays, whose origins are the rendering viewpoint, as opposed to *secondary* rays that originate from the object surface [123]. Therefore, we name our representation as Primary Ray-based Implicit Function (PRIF). In effect, a neural network trained to encode PRIF represents the manifestation of the object's geometry at any viewpoint that it is observed.

Modeling the object from the ray-based perspective, rather than 3D-point-based, has huge implications for efficient application in any task that involves rendering the object from a viewpoint. In Section 5.4, we show that PRIF outperforms common functions such as SDF and OF, and we further demonstrate successful applications of PRIF to various tasks using neural networks.

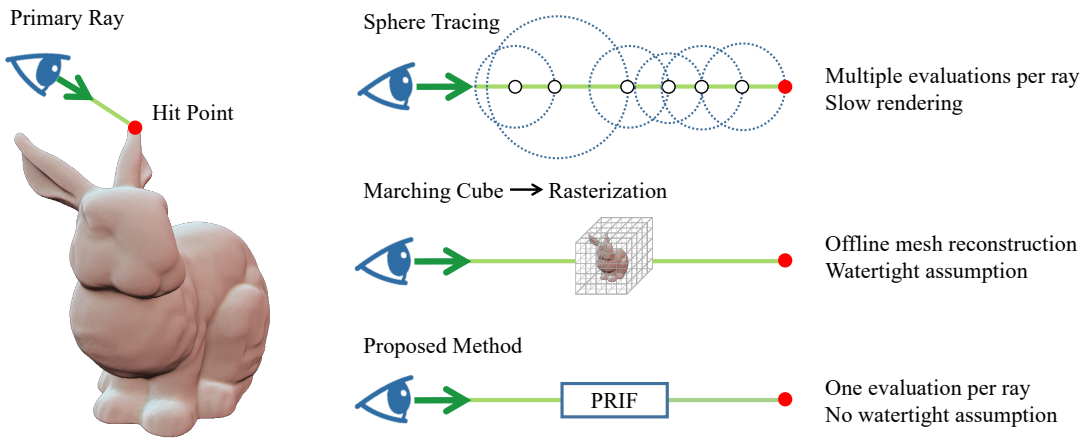


Figure 5.1: Overview. Common neural shape representation methods learn the level-set functions implicitly describing the object geometry, such as the signed distance functions (SDF) and occupancy functions (OF). Rendering from these implicit networks requires either sphere tracing or rasterizing a separately extracted mesh. Sphere tracing is inefficient since it requires multiple network evaluations for a single ray. Rasterization induces a separate meshing step (often through Marching Cubes), which hinders end-to-end gradient propagation in differentiable applications, and the shape quality is ultimately restricted by the meshing algorithm's limitation, such as grid resolutions and shape watertightness. Our new representation, PRIF, directly maps each primary ray to its hit point. A network encoding PRIF is more efficient and convenient for rendering, since it requires only one evaluation for each ray, avoids the watertight constraint in conventional methods, and easily enables differentiable rendering.

Properly formulating the rays is nontrivial. While it may be intuitive to let PRIF output the distance from the ray origin $\mathbf{p}_r$ to the hit point $\mathbf{h}_r$, this formulation leads to ray aliasing - *i.e.*, if we move $\mathbf{p}_r$ along the ray direction $\mathbf{d}_r$, the distance

to the hit point $\mathbf{h}_r$ needs to change. However, the actual surface intersection point would not change to translation along the view direction. Therefore, it is undesirable to have such a potential variance as it adds unnecessary complexity to the network output.

To avoid the aliasing, we reparametrize the ray r by replacing the view position $\mathbf{p}_r$ with perpendicular foot $\mathbf{f}_r$ from the coordinate origin $\boldsymbol{O}$ to r. Wherever we move $\mathbf{p}_r$ along the ray direction $\mathbf{d}_r$, the perpendicular foot $\mathbf{f}_r$ stays the same. Furthermore, we can easily define the surface hit point $\mathbf{h}_r$ by a single scalar value: its distance from $\mathbf{f}_r$. Thus, we formulate PRIF so that it outputs this distance for an input ray. More details are in Section 5.3.

In summary, our main contributions are the followings:

- We present PRIF, a novel formulation for geometric representation using ray-based neural networks.
- We show that PRIF outperforms common level-set-based methods in shape representation accuracy and extraction speed.
- We demonstrate that PRIF enables generative modeling, shape completion, and camera pose estimation.

5.2 Related Work

We discuss prior art on neural representations for 3D shapes and scene rays.

5.2.1 3D Shape Representations

5.2.1.1 Functional Representations.

Traditional 3D shape representations include polygon meshes, point clouds, and voxels. In recent years, as deep neural networks achieve remarkable success on various vision-related tasks, there has been a growing interest in developing implicit neural representations (INRs) of 3D shapes. Following seminal works [124–126] showing successful applications of neural network to encode 3D shapes, many methods have been introduced to solve various vision and graphics tasks using INRs of 3D shapes [120, 127–145].

INRs usually use the multilayer perceptron (MLP) architecture to encode geometric information of a 3D shape by learning the mapping from a given 3D spatial point and a scalar value. Typically, the output scalar value denotes either the occupancy at the given point, or the signed distance from the given point to the nearest point on the shape. On one hand, networks that are trained to encode the occupancy function [126] (OF) essentially learns the binary classification problem, where the output equals 0 if the point is empty, and equals 1 if occupied. Therefore, the decision boundary where the network predictions equal to 0.5 represents the surface of the encoded shape. On the other hand, for networks trained to encode the signed distance function [124, 125] (SDF), the surface is represented by the decision boundary where the network predictions equal to 0. A 3D surface determined in such fashions is also known as an isosurface, which is a level set of a continuous 3D

function. In practice, however, obtaining 3D meshes from these isosurfaces extracted from INRs still requires an additional meshing step often through the Marching Cubes algorithm.

In this paper, we introduce a new shape representation which is not determined by an isosurface. Instead, our representation encodes a 3D shape by learning the PRIF associated with the shape. Outputs of such a function directly correspond to points on the surface, and the shape can be extracted without an additional meshing step that could inject inaccuracies to the final representation.

5.2.1.2 Global versus Local Representations

The initial works of INRs for 3D shapes inspire many techniques to improve its the rendering efficiency [120, 146] and representation quality [59, 147]. Among many techniques, a common thread is the idea of spatial partitions which manifest in two main approaches. One approach divides the surfaces of shapes into different local patches, reducing the difficulty of globally fitting a complex surface with a single network [148–153]. Another approach divides the 3D volume into small local regions (often based on the spatial octree structure), and then train INRs to encode the geometric information within each local region [154–161].

In this paper, we only focus on global representations, where a shape is represented by a single network without any spatial partitions. While the aforementioned works largely focus on improving the performance of INRs based on SDF, our main contribution is a new functional representation in place of isosurface-based representa-

tions like SDF. Nonetheless, the idea of spatial partitions explored in previous works has the potential to improve the performance of our currently global representation, and we see local specializations as a promising future direction to explore.

5.2.2 Ray-based Neural Networks

Our method is closely related to an emerging concept called Ray-based Neural Networks. Rays are a common construct from computer graphics, and can be denoted by a tuple consisting of a 3D point (ray origin) and a 3D direction (ray direction). Due to their closeness to rendering 3D scenes, rays have become a central component in the problem formulation of many recent works using neural networks to model 3D scenes [162, 163]. Feng *et al.* [164] and Attal *et al.* [165] have further demonstrated that for front-facing light field scenes, MLPs can be trained to accurately map camera rays to their observed colors in highly detailed real-world scenes, but their networks only consider rays restricted within two parallel planes. Most recently, Sitzmann *et al.* [104] successfully train a MLP to encode the observed color of unrestricted rays with arbitrary origins and directions, and the key is to parametrize rays using the Plucker coordinates [166].

We similarly adopt rays as input to the neural network. However, instead of the Plucker parametrization which replaces the ray position by its moment vector about the origin, we replace the moment vector by the perpendicular foot between the ray and the origin. As discussed in Sec. 5.3, our formulation allows the network to simply produce an affine transformation to its input.

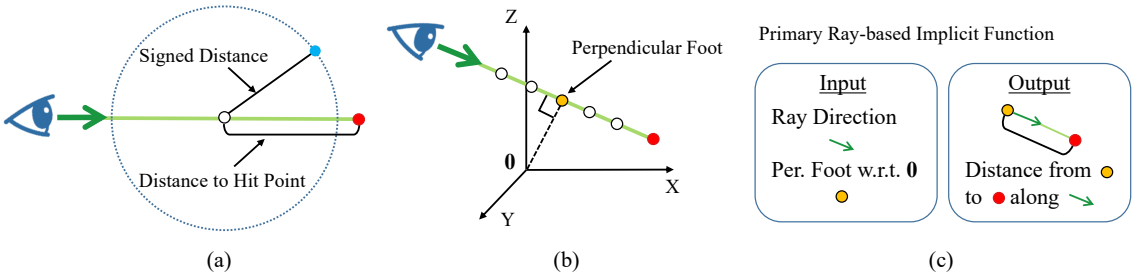


Figure 5.2: Representation. (a) Signed distance at a sampling position (white) reveals the sphere (blue dots) where its nearest surface point (blue) exists, but that may be irrelevant when we really want to know the hit point (red) *along a specific direction*. Thus, multiple samples are often required. (b) Our new representation uses only one sample (yellow) along the ray to obtain its surface hit point. The sampling position is the perpendicular foot between the given ray and the coordinate system's origin $\boldsymbol{O}$. (c) The proposed function takes in the ray's direction and its sampling point, and returns the distance from that point to the actual surface hit point. We train neural networks to encode geometry by learning this new function, and we demonstrate the capability of our trained networks in various tasks involving shape representation and rendering.

5.3 Method

In this section, we introduce our new representation for 3D shapes. We first discuss the motivation behind ray parametrization used in prior works, and then we present our new formulation specifically designed for shape representations.

5.3.1 Background

Sitzmann *et al.* [104] encode light fields by training an MLP Φ_ϕ with parameters ϕ on a set of observed rays to learn the mapping of $r = (\mathbf{p}_r, \mathbf{d}_r) \rightarrow c_r$, where c_r denotes the color of the observed radiance. However, naively concatenating $\mathbf{p}_r$ and $\mathbf{d}_r$ as input to the network is not ideal due to ray aliasing. If we move the position of a ray r along the ray direction $\mathbf{d}_r$, we would obtain a ray $r' = (\mathbf{p}_{r'}, \mathbf{d}_r)$ that is really an aliased version of r. There is no guarantee that the trained network would produce the same output for different aliases r' of the ray r.

To resolve ray aliasing, Sitzmann *et al.* [104] reparametrize the ray $r = (\mathbf{p}_r, \mathbf{d}_r)$ into Plucker coordinates as $r = (\mathbf{m}_r, \mathbf{d}_r)$, where $\mathbf{m}_r = \mathbf{p}_r \times \mathbf{d}_r$ is also known as the moment vector of $\mathbf{p}_r$ about the origin $\boldsymbol{O}$. Plucker coordinates represent all oriented rays in space without singularity or special cases, and they are invariant to changes in the ray position along the ray direction. To better understand this property, consider moving the ray position to some other point $\mathbf{p}'_r$ at a fixed ray direction $\mathbf{d}_r$. Then, for a certain $\lambda \in \mathbb{R}$, $\mathbf{p}'_r = \mathbf{p}_r - \lambda \mathbf{d}_r$, and

$$\begin{aligned} \mathbf{p}'_r \times \mathbf{d}_r &= (\mathbf{p}_r - \lambda \mathbf{d}_r) \times \mathbf{d}_r \\ &= \mathbf{p}_r \times \mathbf{d}_r - \lambda \mathbf{0} \\ &= \mathbf{p}_r \times \mathbf{d}_r. \end{aligned} \tag{5.1}$$

Therefore, $\mathbf{m}_r = \mathbf{p}_r \times \mathbf{d}_r$ is invariant to any change of λ along $\mathbf{d}_r$.

5.3.2 Describing Geometry with Perpendicular Foot

Our goal is to train a neural network to encode the mapping from a ray to its hit point on the 3D shape's surface. Although replacing ray position with the moment vector $\mathbf{m}_r$ allows Sitzmann et al. [104] to train networks to encode light fields, it is hard to geometrically relate a ray's moment vector to its surface hit point. We propose an alternative way to parameterize a ray as input to the network. which has an intuitive and intrinsic relationship to its hit point.

Specifically, we consider the perpendicular foot $\mathbf{f}_r$ between the ray r and the coordinate system's origin $\boldsymbol{O}$, which may be computed by

$$\mathbf{f}_r = \mathbf{d}_r \times (\mathbf{p}_r \times \mathbf{d}_r). \tag{5.2}$$

Similar to $\mathbf{m}_r$ in Plucker coordinates, $\mathbf{f}_r$ is also invariant to changing the ray position along the ray direction. Specifically, let $\mathbf{p}'_r$ be the translated ray position defined as before, we can then write $\mathbf{p}_{r'} = \mathbf{p}_r - \lambda \mathbf{d}_r$, and

$$\begin{aligned}
\mathbf{f}_{r'} &= \mathbf{d}_r \times (\mathbf{p}_{r'} \times \mathbf{d}_r) \\
&= \mathbf{d}_r \times ((\mathbf{p}_r - \lambda \mathbf{d}_r) \times \mathbf{d}_r) \\
&= \mathbf{d}_r \times (\mathbf{p}_r \times \mathbf{d}_r - \lambda \mathbf{0}) \\
&= \mathbf{d}_r \times (\mathbf{p}_r \times \mathbf{d}_r) \\
&= \mathbf{f}_r
\end{aligned} \tag{5.3}$$

In other words, $\mathbf{f}_r$ is invariant to moving $\mathbf{p}_r$ along the direction $\mathbf{d}_r$.

As a result, we can represent any ray $r = (\mathbf{f}_r, \mathbf{d}_r)$, and we can further establish

the following relationship for each ray r:

$$\mathbf{h}_r = s_r \cdot \mathbf{d}_r + \mathbf{f}_r, \tag{5.4}$$

where $s_r \in \mathbb{R}$ denotes the signed displacement between the ray's hit point $\mathbf{h}_r$ and its perpendicular foot $\mathbf{f}_r$ *w.r.t.* the world origin $\boldsymbol{O}$.

To encode a 3D shape, we propose the mapping function $(\mathbf{f}_r, \mathbf{d}_r) \rightarrow s_r$, which we call Primary Ray-based Implicit Function (PRIF). Different than SDF or OF, which implicitly encodes the geometry through the distance from any given point to its nearest surface point *in any direction*, PRIF operates on oriented points with *a specific ray direction*.

In practice, we train an MLP to learn

$$\Phi(\mathbf{f}_r, \mathbf{d}_r) = s_r. \tag{5.5}$$

With this new representation, we are able to represent the surface hit point of a ray with a single value. In effect, our objective is equivalent to finding a simple affine transformation $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, with the input $\mathbf{x} = \mathbf{d}_r$, $\mathbf{A} = s_r I_3$, and $\mathbf{b} = \mathbf{f}_r$. We also avoid a major limitation in previous sphere-tracing-based methods, which is having to sample multiple points and perform multiple network evaluations to obtain a hit point.

5.3.3 Background Mask

For 3D functions like SDF or OF, every 3D point has a well-established scalar value denoting the distance to surface or the occupancy at that point. In contrast,

our function would take in rays that never intersect with the shape and therefore do not even have a hit point.

To address this issue, we let the network additionally produce

$$\Phi(\mathbf{f}_r, \mathbf{d}_r) = a_r, \tag{5.6}$$

where $a_r \in [0, 1]$ denotes the probability in which the ray r hits the foreground. We compute the cross-entropy loss

$$\mathcal{L}_a = \sum_r -a_r^{gt} \log(a_r) - \left(1 - a_r^{gt}\right) \log(1 - a_r), \tag{5.7}$$

where $a_r^{gt} = 0$ for background rays and $a_r^{gt} = 1$ for foreground rays.

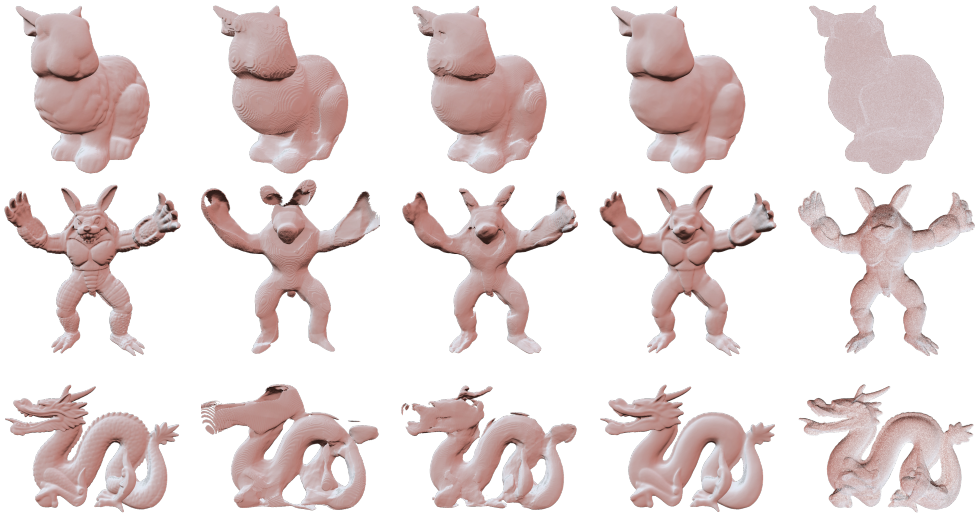
For the signed displacement s_r, we supervise the learning by computing its absolute difference to the ground truth as $\mathcal{L}_s = \sum_{r \in \mathcal{F}} \|s_r - s_r^{gt}\|$, given the set of foreground rays $\mathcal{F}$. As a result, the total loss function to train our network is $\mathcal{L} = \mathcal{L}_a + \mathcal{L}_s$ and is averaged among all rays in a training batch

5.3.4 Outlier Points Removal

In rare cases where sharp surface discontinuities exist between two neighboring rays, the network would likely produce continuous predictions when interpolating between those two rays, resulting in undesirable outlier points. Fortunately, since our network is fully differentiable, for each prediction s_r we can compute its gradient with respect to the changes in ray position. We discard all predictions that satisfy the threshold: $\left\| \frac{\partial s_r}{\partial \mathbf{p}_r} \right\| \geq \delta$. In our experiments, δ is set equal to 5.

5.4 Experiments

In this section, we first verify the efficacy of neural PRIF for shape representation. Then, we show verious applications achieved by using PRIF as the underlying neural shape representation. Note that the scope of our experiments is to compare these functional shape representations *encoded by neural networks*.



Reference OF SDF Ours - Mesh Ours - Points

Figure 5.3: Single Shape. To examine the representation capability of PRIF, we train networks with the same architecture to encode the occupancy function (OF), signed distance function (SDF), and our proposed function (PRIF). Here, we visualize the extracted shapes from the trained neural representations. For OF and SDF, we follow the convention and extract the shapes by Marching Cubes. Our method directly outputs hit points, and we also apply the point-based Screened Poisson algorithm and present the resulting mesh for comparison.

5.4.1 Single Shape Representation

We select five models (*Armadillo*, *Bunny*, *Buddha*, *Dragon*, *Lucy*) from the Stanford 3D Scanning Repository [167, 168] and train a neural network to fit the PRIF for each 3D model. We also fit the signed distance function (SDF) and the 3D occupancy function (OF) for these models. For a fair comparison between these functions, we adopt the same network architecture as Park et al. [125], containing eight layers with 512 hidden dimensions and ReLU activation.

Compared to SDF and OF which are trained on individual spatial points (x, y, z), PRIF requires an inherently different strategy to generate the training data since it takes in individual rays. In our experiments, for each 3D model, we select 50 virtual camera locations oriented towards the origin, and we capture 200×200 rays at each location. For SDF and OF, we follow Park et al. [125] and sample $500,000$ points with more aggressive sampling near the object surface.

For all three functions, we train the neural representation for 100 epochs with the learning rate initialized as 10^{-4} and decayed to 10^{-7} with a cosine annealing strategy. After training, the SDF- and OF-based shape representation are obtained by evaluating the neural network at uniform 256^3 volume grid and extracted using Marching Cubes. On the other hand, with our PRIF-based representation, we can evaluate the neural network at those virtual camera rays in the training set and directly obtain a dense set of surface points.

To evaluate the representation quality, for SDF- and OF-based representations, we first follow conventions [125] and sample $8,192$ points on the mesh extracted with

Method	Armadillo	Bunny	Buddha	Dragon	Lucy
SDF	1.905|1.260	1.717|1.147	6.119|2.258	5.184|1.946	3.387|1.417
OF	4.805|1.624	1.704|1.133	17.279|3.113	19.577|3.014	3.396|1.427
PRIF	**0.978|0.706**	**1.169|0.835**	**1.443|0.821**	**1.586|0.913**	**0.846|0.519**

Table 5.1: Quantitative results on single shape representation on 3D models from the Stanford 3D Scanning Repository. The left and right numbers represent the mean and median Chamfer Distance (multiplied by 10^{-4}). After extracting shapes from each representation, 30,000 points are sampled for evaluation.

Marching Cubes. For the point set produced by the PRIF-based representation, we apply the point-based meshing algorithm Screened Poisson [169] in MeshLab [170] and then sample $8,192$ points from the reconstructed mesh. Then, we obtain $8,192$ Poisson-disk samples of the ground truth surface points from the original 3D model. Finally, we compute the mean and median Chamfer Distance (CD) between the ground truth point set and point sets sampled from those three representations. In Table 5.1 and Fig. 5.3, we provide quantitative and qualitative comparisons among the three representations. PRIF significantly outperform SDF and OF in accurately preserving the fine details of the 3D shapes.

5.4.2 Shape Generation

Having established the representation power of PRIF for single shapes, we now examine its capability for generative shape modeling. We adopt the strategy



Figure 5.4: Shape Generation. We further examine the representation power on generative modeling over multiple shapes. After training the neural networks to learn shapes from a category, we use auto-decoding to find latent vectors that best represents novel shapes within this category. Here, we visualize the extracted shapes from networks trained to encode the three different functions. Note that *Points* are the immediate output of the PRIF networks.

from Park *et al.* [125] and enable multi-shape generation from a single network by concatenating a latent code for each object with the original network input.

In line with Park *et al.* [125], we select five categories of 3D objects from ShapeNetCore [171]. Within each category, we select the first 180 objects as the training set and the next 20 as the testing set. We maintain the same network architecture and training schedule when training neural networks to fit OF, SDF, and PRIF. We evaluate the trained networks on unseen shapes from the test set

Method	Car	Chair	Table	Plane	Lamp	Sofa
SDF	2.315|0.495	2.649|0.407	7.213|0.441	2.728|0.170	32.571|3.475	6.427|0.218
OF	2.820|0.587	4.589|0.835	6.427|1.296	2.999|0.169	143.377|145.753	12.672|0.184
PRIF	**1.961|0.347**	**0.982|0.267**	**4.532|0.315**	**0.389|0.125**	**3.276|0.534**	**1.236|0.222**

Table 5.2: Quantitative results on generative representation on unseen 3D shapes of six categories from ShapeNetCore. The left and right numbers represent the mean and median CD ($\times 10^{-3}$) averaged over the test set. After extracting shapes from each representation, 30,000 points are sampled for evaluation.

using auto-decoding [125] and provide quantitative and qualitative comparisons in Table 5.2 and Fig. 5.4. The PRIF-based representation extends its success from representing single shapes and remains effective in the generative task.

5.4.3 Shape Denoising and Completion

As depth sensors have become available on mobile and AR/VR devices, there are an array of applications for persistent geometric reconstructions [172]. Given real-world data of sparse and noisy observations, the capability of generative modeling could enable shape recovery from noisy and incomplete point clouds.

In Fig. 5.5, we demonstrate the denoising and completion ability of a trained generative PRIF network. Specifically, after training a PRIF network on an object category as described before, we provide the network with an unseen object's point cloud observations, which are either incomplete or contain noise.



Noisy Data Reconstruction Incomplete Data Reconstruction

Figure 5.5: Noisy and Incomplete Observations. Networks trained for shape generation can reconstruct unseen shapes in challenging scenarios where only noisy or incomplete observations are available. Here we visualize the observations and reconstructions as raw point cloud data.

5.4.4 Analysis and Ablations

5.4.4.1 Complexity Analysis

Shapes are commonly extracted from networks encoding SDF or OF using meshing algorithms like Marching Cubes. This meshing step requires evaluating the network at many 3D grid sampling points, and the grid resolution affects the final quality of the mesh. In our case, we can extract the shapes by rendering the scene from multiple positions and change the number of evaluations by varying the number and resolution of virtual cameras. For a more objective comparison against prior arts that require meshing, we further apply Screened Poisson to mesh our output

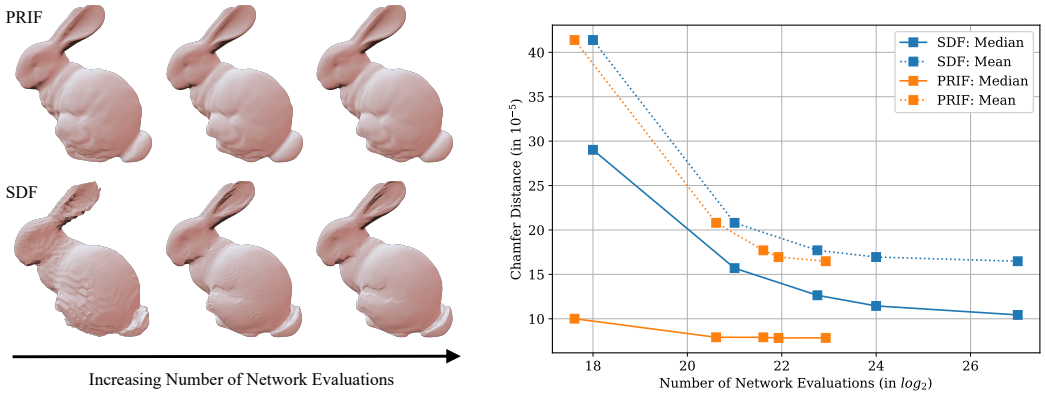
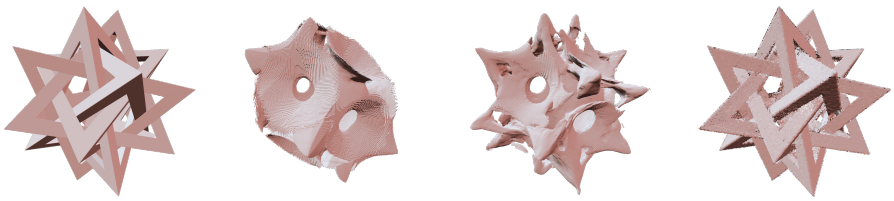
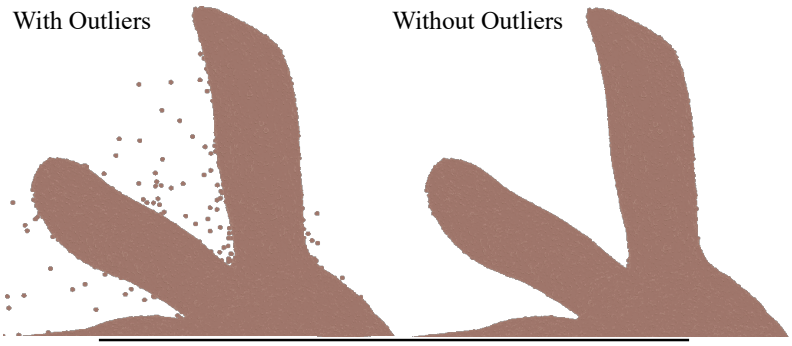


Figure 5.6: Complexity Analysis. We evaluate the trained SDF network at grid resolutions of 64^3, 128^3, 192^3, 256^3, 512^3 and obtain final meshes by Marching Cubes. We evaluate the trained PRIF network at resolutions of 5×200^2, 10×400^2, 20×400^2, 25×400^2, 50×400^2 (# of Cameras $\times$ Resolution), and we apply Screened Poisson [169] to the output hit points to obtain the final mesh. The reconstructed mesh quality is measured by mean and median CD ($\times 10^{-5}$). The PRIF network achieves better quality with fewer evaluations.



Ground Truth OF SDF Ours - Points

Figure 5.7: Stress Testing. We test on a Tetrahedron grid that is self-intersecting and non-watertight. We obtain the SDF and OF values with the scanning method [173], and extract the mesh by Marching Cubes. While the level-set representations fail as expected, our method reliably preserves the shape.



Setup	Trained	Novel
Proposed	**12.07**|**7.95**	**20.81**|**12.05**
$\mathbf{f}_r \Rightarrow \mathbf{m}_r$	12.15|8.05	28.38|13.53
$\mathbf{f}_r \Rightarrow \mathbf{p}_r$	1.16|0.77	1924.4|1067.7
w/ Outliers	12.14|8.06	21.32|12.05

Figure 5.8: Ablations. *Top*: Impact of outlier removal described in Sec. 5.3. While the outliers have little impact on the quantitative metrics since they are scant, removing them improves visual consistency. *Bottom*: Mean and median CD ($\times 10^{-5}$) of network prediction on trained rays and novel rays under different setup. Notice that changing $\mathbf{f}_r \Rightarrow \mathbf{p}_r$ causes ray aliasing discussed in Sec. 5.3, leading the network to overfit on trained rays and collapse on novel rays.

hit points. In Fig. 5.6, we analyze the trade-off between the number of network evaluations and the reconstructed mesh quality measured in mean and median CD. Evidently, PRIF produces reconstruction with a better quality with fewer network evaluations.

5.4.4.2 Stress Testing

Fig. 5.7 shows an example of encoding a self-intersecting and non-watertight shape, which is expected to be challenging for neural networks trained to encode conventional functional representations like SDF and OF. In contrast, the shape is well preserved by the network trained to encode our proposed functional representation PRIF.

5.4.4.3 Ablations

We present ablation studies validating the effectiveness of our proposed techniques in Fig. 5.8 and Table 3. The results verify that: 1) not reparametrizing $r = (\mathbf{p}_r, \mathbf{d}_r)$ leads to severe overfitting on the training data, and 2) replacing the Plucker moment vector $\mathbf{m}_r$ with our proposed $\mathbf{f}_r$ improves the encoding quality, possibly because the network only has to learn a simple affine transform matrix as mentioned in Sec. 3.2.

5.4.5 Further Applications

5.4.5.1 Learning Camera Poses

Evaluating the PRIF network on some input camera rays is essentially performing differentiable rendering of the underlying scene represented by the network. With a PRIF network trained on a 3D shape, given a silhouette image at an unknown camera pose, our task is to recover that camera pose based on the silhouette image.

In this task, the weights of the PRIF network are frozen, and the only learnable parameters are the camera pose matrix. At each iteration, we render the scene through the PRIF network with camera rays defined by the current estimate of the camera pose, and we adjust the estimate based on the silhouette difference between the observed and rendered images. In Fig. 5.9, we present the rendered image during the optimization steps. The estimated camera pose gradually converges to the correct solution as the rendered image becomes more similar to the observed image.

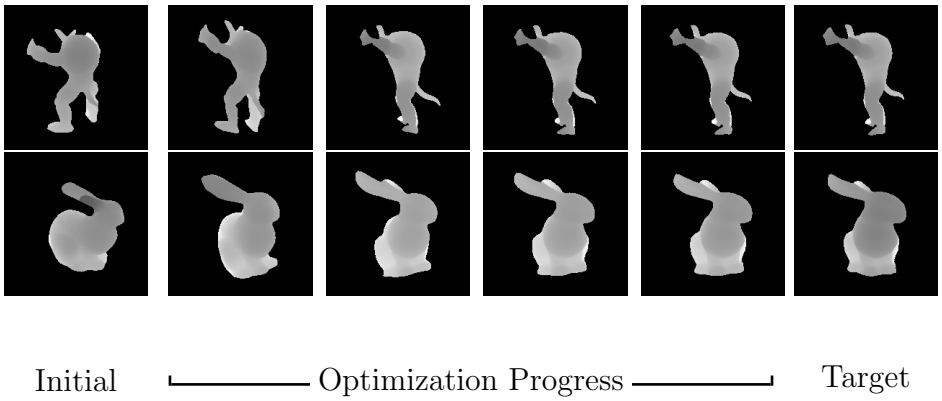


Figure 5.9: Learning Camera Poses. Starting from the *Initial* camera pose, we optimize the learnable pose parameters based on the difference between PRIF output rendered at the current pose and the PRIF output rendered at an unknown *Target* pose. Here, we render the PRIF output as depth images. PRIF-based rendering successfully facilitates the differentiable optimization progress as the camera pose gradually converges to the correct *Target* pose.



Figure 5.10: Incorporating Color Rendering. After training the PRIF network to encode the geometry (rendered as depth images in the first row), we can further train a second-stage network that takes in the hit point produced by PRIF and produces its corresponding color (rendered in the second row).

5.4.5.2 Neural Rendering with Color

The PRIF network behaves as a geometry renderer since it effectively returns the surface hit point given a viewing ray. Here we show that the PRIF network can be further extended to rendering color at the surface. We select a 3D model of human from the Renderpeople dataset [174] and virtually capture the hit points and RGB colors at 25 locations surrounding the model. We then train the PRIF network to fit the geometric shape, similar to the single shape experiments. Finally, we extend the network to produce the observed RGB color given the input ray while fixing its hit point prediction. Fig. 5.10 shows the rendered results of geometry and appearance of the 3D model.

5.5 Additional Details and Discussion

5.5.1 Training Setup

5.5.1.1 Optimization

In all of our experiments, we train the MLPs in PyTorch using the Adam optimizer with the default parameters. We supervise the network based on the mean squared error loss between reconstructed color and ground truth color. We adopt the cosine annealing learning rate scheduler, with initial learning rate as 10^{-4} and final learning rate as 10^{-7}. We randomly shuffled all the training samples and divided them into batches of size equal to 1,024. To ensure reproducibility, we set the random seed as 0. On a machine with NVIDIA GeForce RTX 2080 TI, training 10 epochs takes around 4 minutes for the SDF and OF condition, and 13 minutes for PRIF.

5.5.1.2 Network Architecture

We set all networks to have 10 layers with residual connections, where the 8 intermediate layers having a dimension size of 512×512. We apply layer normalization after all MLP layers except the final one. When we further extend the network to model color or background mask, we simply invoke a second network with the same architecture, as network compression rate or total parameter count is not the main focus of the current work.

Method	Steps	Queries	Time (s)
DeepSDF + Sphere Tracing	100	2,453,700	1.01070
DeepSDF + DIST Acceleration	100	382,325	0.30539
PRIF	**1**	**262,144**	**0.00578**

Table 5.4: Quantitative results on the cost to render a 512×512 depth image from trained neural shape representations. *Queries* refer to the number of network evaluations, and *Steps* refer to the maximum sphere tracing steps specified in the actual implementation. *Steps* $= 100$ is in line with the original DIST implementation from Liu *et al.* [120].

5.5.2 Runtime Discussions

5.5.2.1 Rendering Speed

The formulation of PRIF inherently allows us to bypass the inefficient multi-sample issue of the previous methods that adopt level-set functional representations. To more quantitatively measure the speed-up, we measure the number of network evaluations (Queries) and the actual time to render a 512×512 depth image from a trained neural representation on the same machine with an NVIDIA GeForce RTX 2080 Ti graphics card. We first consider a baseline method where a trained DeepSDF network is used to perform naive sphere tracing. We then consider an improved baseline where the DeepSDF-based sphere tracing is sped up by various techniques proposed by Liu *et al.* [120]. Both baselines are adopted from the implementation

of Liu *et al.*, and correspond to two conditions (trivial and recursive) in their implementation.

5.5.2.2 Meshing Speed

The overall meshing speed depends on *network inference + surface reconstruction*. We examine the trade-off by decoding the same shape with PRIF and SDF trained with networks of equivalent capacity. Our method spends 1.01s at inference, then 2.65s to obtain the triangle mesh from MeshLab (Poisson with normal estimation). The SDF-based network (similar to DeepSDF) spends 6.97s for inference, then 1.19s for Marching Cubes. We also note that many practical scenarios do not require the meshing process, such as camera pose registration and neural rendering.

5.5.3 More Ablations

5.5.3.1 Comparison to Plücker Coordinates

We use the original Plücker coordinates for the shape generation task for the *Car* category. Below are the results (with mean/median Chamfer distance):

Method	Car	Chair	Table	Plane	Lamp	Sofa
Plain Plücker	2.397|0.547	2.603|0.531	5.054|0.554	0.766|0.199	4.838|1.094	1.738|0.305
Proposed	**1.961|0.347**	**0.982|0.267**	**4.532|0.315**	**0.389|0.125**	**3.276|0.534**	**1.236|0.222**

Although the plain Plücker formulation works well for single shape representation (Section 5 Table 3), it performs much worse in the generative task. We find the

network trained with Plücker coordinates often fails to predict correct background masks, leading to many noisy points and worse performance overall. Compared to the Plücker moment vector, our proposed reference points is better geometrically related to the surface hitpoint, which is our overarching goal of shape representation and rendering.

5.5.3.2 Varying Model Complexity

We examine the effect of varying the model capacity (by changing MLP layer width) for the generation task on shapes from the *Car* category.

# Param (in Millions)	2.96	2.23	1.58	1.35	SDF (2.10)
Mean|Median CD	1.961|0.347	2.026|0.364	2.058|0.368	2.055|0.386	2.315|0.495

Results show we can decrease the number of parameters and still achieve better performance than SDF.

5.5.3.3 Different Noise Levels

We perform additional denoising experiments (as Fig. 5 in the paper) and present the impact of different noise levels on the shape representation accuracy on the *Car* category (measured by Chamfer distance):

Noise Level	0.01	0.02	0.04	0.08	0.1
Mean|Median CD	1.929|0.368	1.931|0.428	2.005|0.590	2.471|0.951	2.842|1.123

5.5.3.4 Limitation

This paper is focused on 3D shape encoding and decoding through hit points of rays from known views. PRIF can render at novel views by sampling new rays, but to guarantee consistent novel view results would likely require further modifications such as multi-view consistency loss or denser training views.

5.6 Conclusion

We propose PRIF, a new 3D shape representation based on the relationship between a ray and its perpendicular foot with the origin. We demonstrate that neural networks can successfully encode PRIF to achieve accurate shape representations. With this new representation, we avoid multi-sample sphere tracing and obtain the hit point with a single network evaluation. Neural networks trained to encode PRIF inherit such advantages and can represent shapes more accurately than common neural shape representations using the same network architecture. We further extend the neural PRIF networks to enable various downstream tasks including generative shape modeling, shape denoising and completion, camera pose optimization, and color rendering. Promising future directions include using spatial partitions to improve the network accuracy, jointly learning the geometry and view-dependent color directly from images, speeding up network training, and real-time inference for robotic applications.

Chapter 6: View Interpolation with Implicit Neural Representations of Images

6.1 Introduction

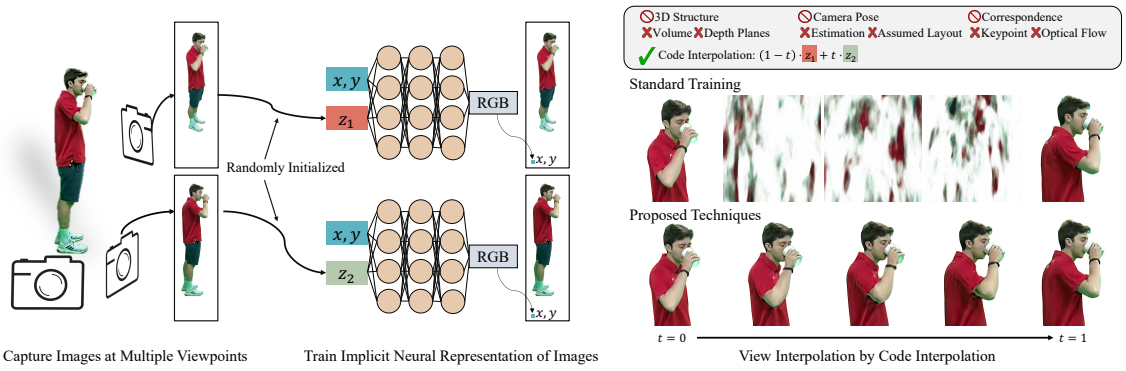


Figure 6.1: We propose a new method for view interpolation through implicit neural representations (INR) of images. After each image is randomly assigned a code vector z, the codes are then jointly trained with the neural network to produce the RGB color given coordinate (x, y). With standard training, the INR fails to decode coherent images from new codes interpolated by two trained codes, but our method enables smooth transition between two known viewpoints. Contrary to common methods for view interpolation, our method does not use 3D structure, camera poses, or pixel correspondence during training.

Neural networks have become a prevalent component in various computational systems over the past decade. For the graphics and vision community, they have been an effective tool in tasks involving visual data, such as recognition, segmentation, and 3D reconstruction. In these classic tasks, neural networks are often deployed as a feature extractor from the input visual signal (e.g. image), but more recently, coordinate network has emerged as a new concept. Instead of extracting features from the signal, the network takes in a coordinate and produces the signal value at that coordinate. Such a network learns a continuous function that maps signal coordinates to values, and it is often referred to as an implicit neural representation (INR) of the signal. INR has led to remarkable success in representing visual signals such as images, videos, signed distance fields, and radiance fields.

In scenarios where only 2D images are available, INR has found two prominent applications. One of them is image fitting, where INRs are trained to produce the color of each known image pixel. Along this line, much progress has been made to improve the accuracy and speed of fitting INR on images, as well as its ability for compression and super-resolution. The other prominent application is reconstructing 3D scenes from 2D images. Here, INRs produce the attribute values (e.g. radiance and opacity) at each spatial coordinate, which are then differentiably rendered into pixels. In this case, the INRs are optimized such that these rendered pixels reproduce the known image pixels. Once sufficiently trained, these INRs can synthesize plausible novel views outside the training set.

On both fitting and view synthesis, INRs achieve impressive visual results that closely resembled the original 2D images. However, it also appears that the

development of INRs has gone into two orthogonal directions. On one hand, the quality of fitting images with INRs is improved by incorporating traditional signal processing techniques like multi-scale subsampling and filtering. On the other hand, the quality of view synthesis is improved by augmenting INRs with well-established 3D graphics techniques, such as spatial subdivision, parametric modeling, and level-set methods.

Although the exciting advancements towards these two directions are rapidly pushing the state of the art, we like to explore a different direction and ask a new question: Given multiple 2D image views of a 3D scene, can we use the INR of those 2D images alone to do view synthesis without any 3D reconstruction, pose, or correspondence? Driven by this question, we present an initial exploration towards view interpolation with INR of images (VIINTER). With randomly initialized INR weights and code vectors for individual images, we modify the standard INR training process such that the trained INR can both faithfully reproduce the given images and synthesize plausible novel views when we interpolate between those learned image codes.

It is nontrivial to obtain sensible novel views through code interpolation with standard training of INR. We experiment on a range of changes to the training of INR and provide details in Section 6.3. We present further evaluation results on different types of multi-view scenes in Section 6.4. Our work takes an important early step toward revealing new potential of INR of images, and we summarize our main contributions as follows:

- We present a novel approach to view interpolation by interpolating INRs trained to fit 2D images without any knowledge of 3D structure, pose, or correspondence.
- We introduce several modifications to the common process of training image-fitting INRs, which significantly improve the view interpolation quality.
- We show that the proposed non-3D approach achieves smooth and photorealistic interpolation across several scenes with a variety of viewpoint layout and scene content.

6.2 Related Work

In this section, we review recent work on implicit neural representation, as well as prior techniques for view interpolation.

6.2.1 Implicit Neural Representations

Following seminal works [124–126] showing successful applications of neural network to encode 3D shapes, many methods have been introduced to solve various vision and graphics tasks using INRs of 3D shapes. These INRs usually use the multilayer perceptron (MLP) architecture to encode geometric information of a 3D shape by learning the mapping from a given 3D spatial point and a scalar value denoting either the signed distance or occupancy.

6.2.1.1 3D Reconstruction

As differentiable rendering becomes more practical, researchers have succeeded in training INRs to learn, not just fit, the geometry and appearance of a 3D scene based on 2D image observations. The most prominent works is Neural Radiance Fields (NeRF) [162], which learns an INR of the view-dependent radiance volume inside a 3D scene and naturally enables view synthesis. The success of NeRF sparked an enthusiastic trend of improving INRs for highly photorealistic view synthesis in terms of their training speed, rendering speed, and rendering quality. A wide range of techniques have been studied and incorporated to 3D INRs, including spatial subdivision or octree [175, 176], parametric modeling with human body shape prior [177, 178], level-set methods for more accuracy geometry [118, 179], caching and distillation for faster rendering‘ [176, 180], camera pose refinement [181–183], and lighting and camera variation during capture to better extract physical attributes [184, 185]. Convolutional neural networks [84, 186–188] have also been trained to take camera pose as input and produce 2D renderings of simple 3D scenes.

6.2.1.2 Image Fitting

Images are arguably the most dominant form of visual data, and many efforts on INRs are to make them fit 2D images as accurately and quickly as possible [117, 160]. ACORN [158] applies spatial subdivision to more efficiently train INR of a single gigapixel image (with 1 billion pixels). Various signal processing techniques are also shown useful in making INRs fit images more accurately, such as image pyramids [161],

sinusoidal and Fourier basis functions [59, 147], and multiplicative filtering with Fourier or Gabor wavelet basis functions [159, 189, 190]. Many of these techniques are applicable for other signals like 3D MRI data or 3D signed distance fields, which are beyond the scope of this paper.

While most of these methods focus on training a single network as INR of a single image, a single network may also serve as INR of multiple images. For consecutive images at a fixed viewpoint, an extra time dimension is added to the coordinate input [59]. For structured 4D light fields where the views lie on a 2D plane, those 2D coordinates can be re-parameterized as input [164, 165]. A single network can further serve as a generalizable INR of arbitrary images, by concatenating the code with the 2D pixel coordinate as input to the INR [191]. An alternative approach is to modulate network activation based on the code [191, 192], which has success in fitting arbitrary image patches. We find the simple code concatenation is sufficient for our problem, and it has been successfully used to train an INR of light rays from different scenes [104, 193]. Unlike modulation, it avoids the cost of additionally training an encoder and a modulator network, allowing us to study of INR with its most basic form.

6.2.2 Image-based Rendering

The early approaches of image-based rendering (IBR) achieve novel view synthesis through explicitly blending relevant pixels from known images [194–196]. The visual quality of IBR is heavily dependent on the strategy of deciding the blending weights of images, and researchers have developed a line of techniques

improving blending weights selection, such as ray-space proximity [195, 197], proxy geometry [196, 198, 199], optical flow [200, 201], soft blending [202, 203], and neural-network-assisted blending [204–207]. These techniques often require an approximate 3D structure (proxy geometry or depth) of the scene so that pixels can be re-projected to the novel view. For methods that do not involve 3D re-projection [195, 208], many still assume the knowledge of the 3D camera locations and orientations of each image and leverage the spatial relationship among the cameras to decide the blending weights. In contrast, we explore a different and more challenging problem setting which does not involve 3D reconstruction nor the knowledge of 3D locations and camera orientations. Our problem setup is similar to prior work on image morphing [200, 209–211], but we achieve the morphing effect without finding pixel-wise correspondences between images.

6.3 Method

We provide details on the INR parametrization adopted in our study, and we introduce the proposed modifications to INR training.

6.3.1 INR for Image Fitting

Let $\mathcal{F}$ denote the INR of images. In the case of a single image, for all pixels p of the image, the INR $\mathcal{F}$ defines

$$\mathcal{F}(p_x, p_y) = p_c, \tag{6.1}$$

where (p_x, p_y) denotes the coordinate of the pixel p, with $p_x \in \mathbb{R}$ and $p_y \in \mathbb{R}$. $p_c \in \mathbb{R}^3$ denotes the value (often the RGB vector) associated with the pixel p. In itself, the INR formulation is invariant to different numeric ranges of (p_x, p_y) or p_c, and for simplicity we rescale the pixel coordinates and values to be within [0, 1].

We adopt the conventional MLP architecture to parameterize $\mathcal{F}$ as a chain of fully connected layers, with activation function usually set as a ReLU or sinusoidal function. Various embedding functions of the input coordinate (x, y) have been proposed, but in this work we apply no embedding and use sinusoidal activation [59], which are sufficient for fitting single 2D images.

The primary training objective of INR $\mathcal{F}$ for single 2D images is to minimize the reconstruction error between the predicted p_c and ground truth p_c^{GT} across all known pixels in a single image, namely

$$L_{SingleRecon} = \sum_p \|p_c - p_c^{GT}\|^2. \tag{6.2}$$

6.3.2 Extension to Multiple Images

Our goal is to use a single network $\mathcal{F}$ as the INR for multiple images from the same scene. Prior methods assume the camera layout (for planar light fields [164]) or known camera poses in the pipeline (for general light fields [104, 165]), but we are interested in pushing the limit to where the camera pose of each image is unknown.

In our 3D-agnostic setup which does not consider camera poses, we assign a randomly initialized vector $z \in \mathbb{R}^M$ for each image, which serves as its identity code. We then modify the INR setup so that the operation on each pixel coordinate

(p_x, p_y) is now conditional on the code z of length M. In practice, we concatenate z with (p_x, p_y) to form the input vector to the network. Formally, with N images and $n = 1, ..., N$,

$$\mathcal{F}(p_x, p_y \mid z_n) = p_{c|n}, \tag{6.3}$$

where $p_{c|n}$ stands for the predicted value of pixel p in image I_n.

The training loss function can be easily modified as

$$L_{Recon} = \sum_n \sum_p \|p_{c|n} - p_{c|n}^{GT}\|^2, \tag{6.4}$$

such that the INR $\mathcal{F}$ fits the pixels among all N images.

While it is easy to optimize $\mathcal{F}$ and $\mathrm{Z}_N = \{z_n\}_{n=1}^N$ to reach a low L_{Recon} across all known pixels, it remains unclear how the learned $\mathcal{F}$ would perform given a novel $z \notin \mathrm{Z}_N$. Of course, it would be too demanding to expect $\mathcal{F}$ to always produce sensible results for any random $z \in \mathbb{R}^M$. Nonetheless, we believe it is fair to inquire $\mathcal{F}$ under a more relaxed setting: With $z_i, z_j \in \mathrm{Z}_N$, can $\mathcal{F}$ produce sensible results given a novel $z_{Inter} = \alpha z_i + \beta z_j$? In other words, as $\mathcal{F}$ is trained to produce good results with z_i and z_j, what would it produce with a weighted combination of z_i and z_j?

6.3.3 Direct Regularization

The reason we are interested in the above problem is that, if $\mathcal{F}$ would produce good results on weighted combinations of known $z_i, z_j \in \mathrm{Z}_N$, it could produce a smooth transition from image I_i to image I_j. Effectively, it could achieve view interpolation without using any correspondence point or 3D information. In this

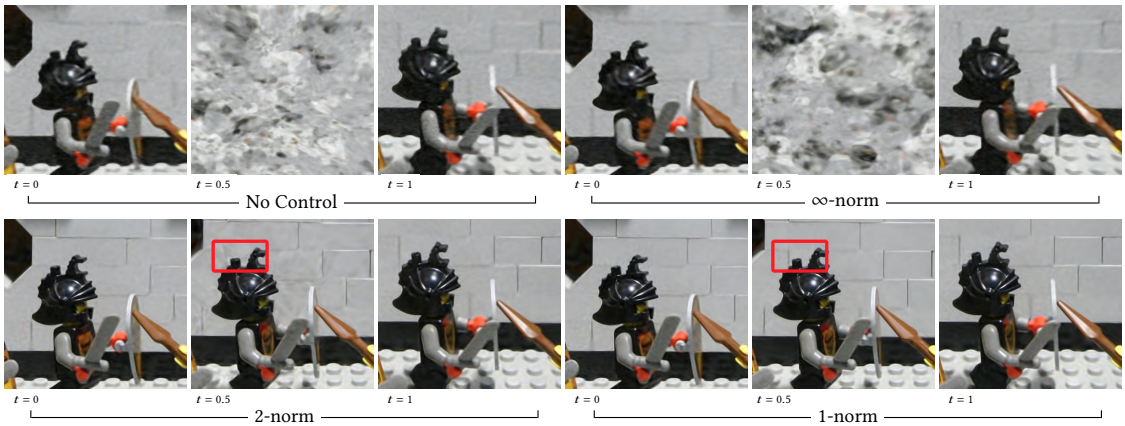


Figure 6.2: We illustrate the effect of controlling $\|z\|_p = 1$ with different p. For each condition, we show the INR-produced images given z_i (left), $0.5z_i + 0.5z_j$ (center), and z_j (right). "No Control" trains INR $\mathcal{F}$ and all codes for known images without controlling their scales, showing proper reconstruction at known views (left and right) but complete failure in interpolation (center). "∞-norm" scales each z with its maximum norm, but still does not interpolate well. "2-norm" significantly improve interpolation and reconstructs known views better, but "1-norm" is much better at interpolation (see red boxes).

paper, we select the interpolation weights α, β with the simple linear interpolation, inducing $\alpha = 1-t, \beta = t$ with $0 \leq t \leq 1$. Unfortunately, as shown in Fig. 6.2, linearly interpolating between the two learned codes fails to let $\mathcal{F}$ produce any meaningful result.

The initial failure is not really a surprise. The codes in Z_N are optimized only towards minimizing L_{Recon}. It would make sense for them to end up with different scales so that $\mathcal{F}$ can better distinguish them and reduce L_{Recon}. Thus, interpolating between them would likely produce a noise vector which $\mathcal{F}$ cannot meaningfully

decode.

To address this issue, we directly regularize the Z_N during training. In particular, we prevent the codes from having different scales by explicitly enforcing the learnable code z as unit p-norm. For our method we select $p = 1$ and enforce

$$z = \frac{z}{\|z\|_1}, \forall z \in \mathrm{Z}_N \tag{6.5}$$

in addition to training $\mathcal{F}$ and Z_N based on L_{Recon}.

Although a more instinctive option is to rescale z based on its Euclidean norm (sum of squares), this Euclidean norm is only a special case for the general p-norm when $p = 2$, or $\|z\|_2$. We investigate the effect of varying p while computing the norm of z, and we present the results in Table 6.1 and in Fig. 6.2. With rescaling based on 1-norm, the interpolation is more natural and stable than with 2-norm. Moreover, as indicated by Table 6.1, 1-norm even leads to more accurate reconstruction at the known image views, which is the original task of fitting INR of images.

6.3.4 Indirect Regularization

Although the unit norm constraint significantly improves visual quality, artifacts are observable as shown in Fig. 6.4. This shortcoming does not come as a surprise, because there must be a limit as to how smoothly $\mathcal{F}$ can interpolate, given how little prior knowledge it has about the content. After all, $\mathcal{F}$ is only trained on a small set of images, and, unlike powerful generative networks, it does not possess domain knowledge (e.g. faces, cars) from a huge dataset.

However, there is still room for improvement if we more proactively alter



Figure 6.3: We show the effect of the code length M. When trained with shorter code vectors, the INR can still produce good results at known views ($t = 0, 1$) where the code zs are well-trained to reconstruct the pixel color. However, the output given interpolated codes ($t = 0.5$) rapidly decreases as the code length decreases.

the training process. Since the fundamental issue is that $\mathcal{F}$ produces poor results when decoding novel interpolated z_{Inter} unseen during training, we should explicitly encourage good results with interpolated codes during training. As we only have access to those N images and do not have ground truth interpolated frames, we propose computing the loss of interpolated results through an off-the-shelf pre-trained network $\mathcal{E}$. Specifically, we hope the features extracted from the interpolated output are similar to the features extracted from the two source images I_i and I_j.

Formally, with $z_{Inter} = (1 - t) \cdot z_i + t \cdot z_j$, $\triangle$ denoting all pixels in a full image frame, the interpolated output image is $I_{Inter} = \mathcal{F}(\triangle | z_{Inter})$. We then use feature extractor $\mathcal{E}$ to compute

$$L_{Inter} = \|\mathcal{E}(I_{Inter}) - [(1 - t) \cdot \mathcal{E}(I_i) + t \cdot \mathcal{E}(I_j)]\|^2, \tag{6.6}$$

and train the INR towards minimizing this loss.

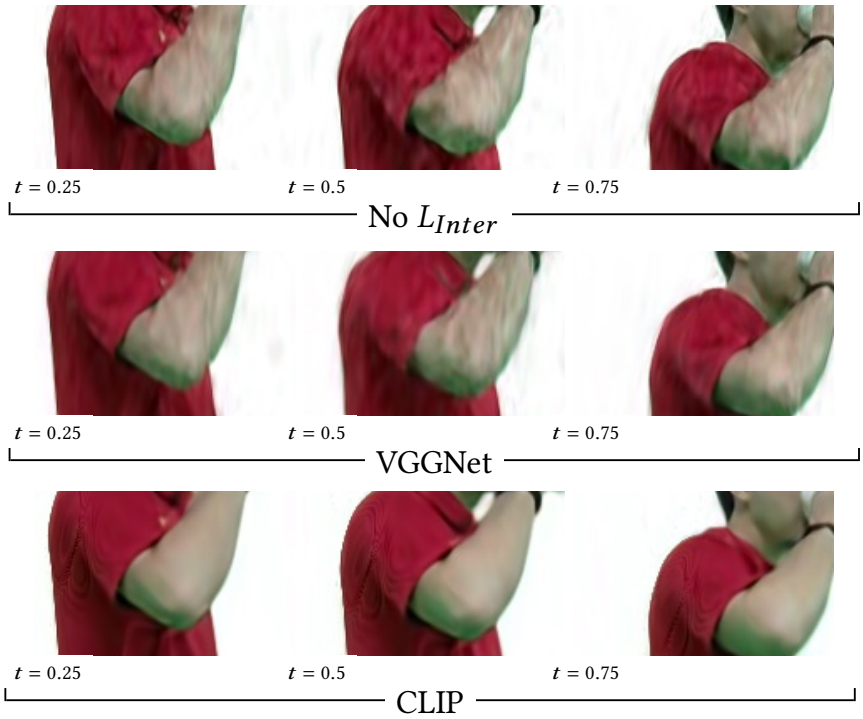


Figure 6.4: We show the effect of the L_{Inter} loss. Without considering L_{Inter}, the interpolation contains visible artifact. With L_{Inter} computed based on the common VGGNet-based perceptual features, the results are worsened by over-smoothing artifacts. We propose using CLIP-extracted features to compute L_{Inter}, which significantly reduces the artifacts during interpolation.

While this setup is similar to the VGGNet-based perceptual loss widely used in tasks like style transfer and image reconstruction, we find that using VGGNet as the feature extractor $\mathcal{E}$ is not adequate for our problem, as shown in Fig. 6.4. Instead, we employ the recently released CLIP network [212] as $\mathcal{E}$. As shown by Fig. 6.4, we discover that the CLIP-based extractor significantly outperforms VGGNet, likely because CLIP benefits from being trained to extract semantically-consistent features from images (see [212] for more details), whereas VGGNet is trained to extract

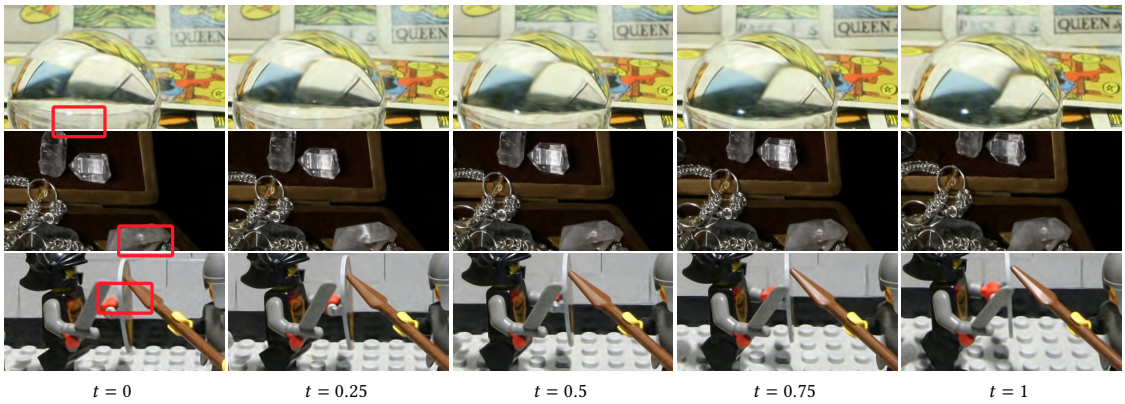


Figure 6.5: Results on Stanford 4D Light Fields. We interpolate learned codes of views i and j as $(1-t)\cdot z_i + t \cdot z_j$. The INR preserves the image details from the known views and smoothly transitions between them, such as bright speckles on the ball (row 1) and view-dependent reflections (rows 2 and 3). Images are zoomed in for easier evaluations.

features mainly for image classification. Our results are analogous to previous findings [213] that show CLIP improves NeRF training on sparse views.

In short, in addition to the original objective of minimizing L_{Recon}, we introduce rescaling with unit norm and the CLIP-guided interpolation loss to the process of training INR of images. To compute L_{Inter}, the interpolation endpoints z_i and z_j are randomly selected from Z_N. We refrain from specifically sampling neighboring or adjacent viewpoints to avoid using the camera pose information and keep training as 3D-agnostic. We would only knowingly select endpoints based on their viewpoint locations during evaluation or demonstration of the interpolation results between different viewpoints, after training is finished.



Figure 6.6: Results on Unstructured Light Field. We interpolate learned codes of views i and j as $(1-t)\cdot z_i + t \cdot z_j$. The camera movement between the two known views includes rotation and translation. The interpolation through INR smoothly transforms the perspective, despite having no knowledge of 3D scene structure or camera pose. Images are zoomed in for easier evaluations.

6.4 Experiments

In this section, we provide more results on view interpolation and ablation studies on the techniques introduced in Section 6.3. We train VIINTER to encode real-world scenes captured under two different regimes: 4D light fields (viewpoints are on a 2D plane with the same orientation) and unstructured light fields (viewpoints

are not aligned on a 2D grid and orientations might be rotated).

4D Planar Light Fields We use scenes from Stanford Light Field Archive [98], with 17×17 camera viewpoints on a 2D grid. We use a 5×5 subset by taking every 4-th image horizontally and vertically. We render new views by selecting two trained codes and linearly interpolate them. The resulting interpolation results are shown in Fig. 6.5.

Unstructured Light Fields To test VIINTER on scenes with irregular camera layout, we test on the LLFF dataset [206] and our own volumetric dataset. The LLFF scenes are captured in natural indoor environments, while our own scenes come from a volumetric studio for human body captures. We present the interpolation results in Fig. 6.6.

Quantitative Evaluation The unique challenge in evaluating our method is we cannot explicitly specify a camera pose to render at. Nonetheless, to provide a quantitative evaluation, we approximately render at testing viewpoints by interpolating the codes from nearby known viewpoints. For example, for the Stanford Light Field scenes, we select two viewpoints in the 5×5 training set, viewpoints $(4, 4)$ and $(4, 8)$, and interpolate their learned codes with $t = 0.5$. Then we render the full image with the interpolated code and compare it against the actual test image (withheld from training) captured at viewpoint $(4, 6)$. Thanks to the well-aligned structure of these 4D scenes, we can compute metrics like peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) against a reasonable ground truth image.

Table 6.1 provides the quantitative impact of our proposed techniques on known and novel viewpoints in the *Lego* scene. In Table 6.2 we present the evaluation results aggregated from all scenes. Although our method is not meant to outperform methods which use 3D information, we still provide quantitative comparisons with two recent methods: NeRF [162], the most prevalent INR method for view synthesis, and LFN [104], which uses camera pose information to train an INR of 5D rays. For results at novel viewpoints, although we can reasonably approximate the viewpoint in the *4D Light Field* scenes, the approximation is very inaccurate in the *Unstructured* scenes due to sparse and irregular camera layouts. For more comprehensive assessment of the quality, please refer to the supplements.

6.5 Discussion

In this section, we provide further discussion on the significance of the proposed method and presented results.

Why not NeRF (or 3D approach)? In general, image-based methods avoid certain issues unique to 3D approaches (e.g. properly setting 3D bounding box, # of samples per ray) that often complicate the training. However, this paper is not intended to present a better method than the state of the art in view interpolation and synthesis, but rather to explore a new direction where a classic image manipulation problem meets the modern implicit neural representation. We believe that 3D approaches like NeRF are currently still more appropriate in production, due to the abundance of techniques and optimizations developed to improve their performance.

For sake of transparency and thoroughness, we provide comparisons in Section 6.4 with representative methods and datasets, and we hope they help readers better contextualize this new and untested approach.

How is this different than image morphing? Although many image morphing techniques do not invoke explicit 3D knowledge about the structure or viewpoints, they rely on finding correspondence points between the images being interpolated. Our proposed method is correspondence-free, and the interpolation happens in the space of the z codes, rather than the space of image pixels. Moreover, image morphing often applies to images of different scenes or identities (e.g. face morphing between two people), but this paper is concerned with multi-view images from the same scene. In our setting, if we do find correspondences like most morphing methods, we would essentially do keypoint matching.

How is this different than GAN interpolation? The code interpolation is seemingly similar to the latent space interpolation of GANs, but our method fundamentally differs from GANs in three ways. First, VIINTER does not train a discriminator that provides adversarial guidance to a generator. Second, GANs usually model a continuous Gaussian latent space while we only consider the pairwise interpolation between codes in Z_N. Third, GANs are trained for domain-specific data and are unlikely to work for out-of-distribution data. For example, although we could project face images into the latent space of a powerful GAN trained on aligned human faces, but we cannot use it to interpolate the various categories of

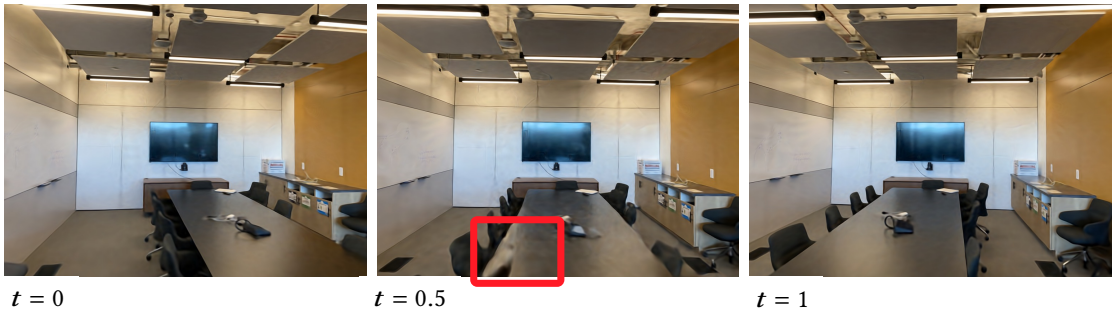


Figure 6.7: Limitations. When the disparity is too large due to insufficient viewpoint density, our method might not produce plausible interpolation, likely because the training views do not provide the INR with enough information to perform implicit 2D interpolation without 3D knowledge.

images used in Section 6.4. Our method is applicable for each separate scene, and the CLIP-based feature extraction is shown to generalize well both in prior work [213] and our experiments with scenes containing vastly different visual attributes.

What is the implication of the norm of z? We point out that our strategy is not adding a penalty on the norm of z, but rather strictly enforcing the norm of $z = 1$. Controlling the norm of z ensures that the learned codes exists on a well-defined region in $\mathbb{R}^M$. In the case of $M = 2$, 2-norm ensures all 2D points lie on a circle, whereas 1-norm ensures all 2D points lie on a square inside that circle. As we increase M to higher dimensions, the difference between 1-norm and 2-norm intensifies as the gap between that "circle" and "square" enlarges. As a result, the codes learned with 1-norm is more compact and likely more conducive for interpolation.

What are the limitations? The method may produce obvious artifacts when interpolating scenes with large disparity (Fig. 6.7). Additional experiments on more varied data would be necessary for a comprehensive assessment of its performance on challenging data Another limitation is it has no sense of 3D locations or camera pose. As a result, it can only interpolate known viewpoints and cannot render at arbitrary locations. Nonetheless, this limitation would not be a deal breaker for many practical use cases that only interpolate between known viewpoints, like 4D light field rendering where viewpoints are fixed on a 2D plane. Another notable use case is event replay for TV viewers, which produces a fly-through effect by interpolating between known cameras. Finally, the proposed method is limited by the training speed of INR, and training the basic INR implemented in this work (with loss from feature extractor) takes a few hours. We believe future work can significantly alleviate this limitation by incorporating recent techniques [158, 160] to speed up INR training.

6.6 Supplementary Information

6.6.1 Training Details

6.6.1.1 Hyperparameters

We use SIREN [59] as the network backbone and randomly initialize the weights and the codes z by default PyTorch settings. In the main experiments, we set length of z as $M = 128$. All networks have eight intermediate layers with dimension of

512, except the three volumetric scenes where the hidden dimension of 256, since we found that increasing the dimension to 512 did not meaningfully improve the quality. We optimize the all parameters (network weights and image code z) with the Adam optimizer, with a learning rate of 10^{-6} which decays to 10^{-7} with the cosine annealing schedule. At each training iteration, the pixel batch size is 8192. For the 4D Planar scenes, the network is trained for $300,000$ iterations, while it is trained for $200,000$ iterations for the other scenes, since we did not find quality improvements even if we trained the network for more iterations. For the LLFF scenes, we $\alpha = 0.05$ except for *Fortress* where $\alpha = 0.01$. For the Stanford Light Field scenes with narrow camera baseline, we find it not necessary to apply L_{inter} and thus set $\alpha = 0$. For the volumetric scenes, we set $\alpha = 0.1$. Scenes with more between-view content disparities would benefit from stronger semantic regulations through CLIP. The Stanford Light Field scenes do not necessarily need such regulation because the difference between adjacent views is small.

6.6.1.2 Training with CLIP-based Features

Without invoking CLIP to compute L_{inter}, training on the Stanford Light Field scenes takes around 5 hours. For the other scenes which involve L_{inter}, the training time varies from 8 to 13 hours depending on the number of pixels for each scene. We only compute L_{inter} once every two iterations to reduce the training time.

To extract the CLIP-based features, we use the public implementation provided by the original authors [212]. To address the issue that pre-trained networks require

the input image to have a specific size which is different than our images, we reshape the full image into patches of 224×224 in a sliding window fashion, namely "*torch.functional.unfold(im, kernel_size = (224, 224), stride=224*". Then, the features of the entire image are formed by concatenating the feature embedding of each 224×224 patch.

6.6.1.3 Dataset Details

For the Stanford Light Field scenes, we train and test at the original resolution with a total of 25 images. For the LLFF scenes, we use their $4\times$ downsampled version provided by the original authors. For our own volumetric dataset, we train 1261×1612 for *M1*, 658×2246 for *M2*, and 1059×182 for *W1* cropped based on their bounding boxes. Each scene contains 30 images, and those images are captured and included with consent from the three human participants.

To train NeRF on the Stanford Light Field scenes, we follow [165] and use their setup to train NeRF on these scenes. We use NeRF with 8 hidden layers and dimension as 256 (for both coarse and fine networks), and we train for $200,000$ iterations with the batch size of $1,024$ pixels. To train LFN, we adapt the implementation from [104] to train a single network for each scene. We use LFN with 8 hidden layers and dimension as 512, and we train for 500 epochs with the batch size of $65,536$ pixels. The results for all methods (including ours) would likely improve further with longer training, and we tried to obtain fair results under a limited resource budget. We note that the purpose of these comparisons is for reference, not for competition.

Algorithm 2: VIINTER Training Procedure

Data: N images of different views $\{I_n\}_{n=1}^{N}$ each with pixels $\mathcal{P}_n$. Each $p \in \mathcal{P}_n$ has coordinate (p_x, p_y) and color $p_{c|n}^{GT}$. Feature extracting network $\mathcal{E}$.

Parameters: Weights of $\mathcal{F}$ and $\{z_n \in \mathbb{R}^M\}_{n=1}^{N}$.

Prepare: Extract features $\{\mathcal{E}(I_n)\}_{n=1}^{N}$ of known images.

For each training iteration:

- Randomly select $i, j = \{1, ..., N\}$
- 1-norm constraint $z_i, z_j = \frac{z_i}{\|z_i\|_1}, \frac{z_j}{\|z_j\|_1}$
- Sample B pixels as $\mathcal{P}_i^{batch}$ from $\mathcal{P}_i$. $\forall p \in \mathcal{P}_i^{batch}$, get $\mathcal{F}(p_x, p_y \mid z_i) = p_{c|i}$ and loss L_{Recon} with $p_{c|i}^{GT}$
- Randomly select an interpolation weight $t = [0, 1]$ such that $z_{Inter} = (1-t) \cdot z_i + t \cdot zj$
- $\forall p \in \mathcal{P}_i$, compute $\mathcal{F}(p_x, p_y \mid z_{Inter}) = p_{c|Inter}$
- Reshape the output $\{p_{c|Inter}\}_{\forall p \in \mathcal{P}_i}$ as 2D image I_{Inter}
- Extract features $\mathcal{E}(I_{Inter})$ and compute loss L_{Inter} with $(1-t) \cdot \mathcal{E}(I_i) + t \cdot \mathcal{E}(I_j)$
- Update $\mathcal{F}, z_i, zj$ based on loss terms L_{Recon} and L_{Inter}

6.6.1.4 Beyond Interpolating Between Two Views

Our proposed setting obtains the novel view AB between two known views A and B by interpolating the latent codes associated with A and B. We could do the same for two other known views, C and D, and obtain another novel view CD. At



Figure 6.8: Interpolation between interpolated latent codes. After interpolating the latent codes for two training views (Column 1 and 4), we can further interpolate between those interpolated latent codes (Column 2 and 3). This additional step effectively leads to more viewpoints that can be expressed by our INR.

this stage, we can further interpolate between AB and CD, and in Figure 6.8 we present some example results of interpolation between the two interpolated latent codes.

6.6.1.5 Extending to Frame Interpolation.

Our method can be potentially modified and improved for frame interpolation using only two input images [214]. As a proof of concept, we take two human portrait images of the same person and deploy our method on those images. In Figure 6.9, we present the intermediate frames produced by our method. We also test our method on the data used in X-Fields [84]. In Figure 6.10, we show the intermediate frames produced by our method.



Figure 6.9: Frame interpolation with human faces. Inspired by [214], we deploy VIINTER on pairs of human portrait images with different expressions (Column 1 and 5). We then render the INR with interpolated latent codes between those two training views, and the resulting images (Column 2, 3 4) exhibit smooth transitions between the two expressions.

Table 6.1: Top: Effect of varying p for code rescaling. Quantitative results reaffirm that rescaling based on 1-norm achieves the best quality. Bottom: Effect of varying the length of code M. Results indicate that the code length cannot be arbitrary, as a small M can be detrimental to the interpolation quality. Setting M too large is also not helpful, as the quality appears to peak at $M = 128$. More details about the experimental setting are in Sec. 6.4.

p-norm	($M = 128$)	No	∞	2	1.5	1
Known	SSIM	0.901	0.903	0.951	0.955	0.962
	PSNR	27.49	27.57	31.63	31.92	33.38
Novel	SSIM	0.595	0.583	0.937	0.952	0.958
	PSNR	11.02	11.04	29.15	31.25	32.39

M	($p = 1$)	16	32	64	128	256	512
Known	SSIM	0.953	0.958	0.961	0.962	0.961	0.960
	PSNR	32.64	33.03	33.37	33.38	33.16	32.95
Novel	SSIM	0.891	0.951	0.958	0.958	0.958	0.956
	PSNR	25.27	31.37	32.56	32.49	32.39	32.10

Table 6.2: Quantitative results on real-world scenes with different viewpoint layouts. Our method can only render at the approximate viewpoints of ground truth, as discussed in Section 6.4, leading to lower PSNR and SSIM values for novel views of "Unstructured" where the viewpoint mismatch is severe. See visual results for more comprehensive quality assessments.

		4D Planar			Unstructured		
	Method	NeRF	LFN	Ours	NeRF	LFN	Ours
Known	SSIM	0.926	0.977	0.978	0.911	0.920	0.885
	PSNR	33.62	37.67	37.28	29.04	30.11	28.32
Novel*	SSIM	0.917	0.944	0.975	0.905	0.788	0.664
	PSNR	33.28	30.67	35.77	27.15	21.35	16.80



Figure 6.10: Frame interpolation on general images. We deploy VIINTER on two images (Column 1 and 5) captured at different timesteps. We then render the INR with interpolated latent codes between those two training views, and the resulting images (Column 2, 3, 4) exhibit smooth transitions between the two expressions.

	4D Planar				Unstructured			
	NeRF	LFN	Ours	Ours-Ft	NeRF	LFN	Ours	Ours-Ft
SSIM	0.917	0.944	0.975	0.977	0.905	0.788	0.664	0.802
PSNR	33.28	30.67	35.77	36.84	27.15	21.35	16.80	24.03

Table 6.3: Quantitative results on novel views with an additional condidtion *Ours-Ft*, where we render our INR with the latent code obtained after optimizing it against the ground truth test image (while freezing the network weights). Results suggest that the trained INR is capable of achieving better quantitative novel view results, but is handicapped by our inability input exact camera poses due to non-3D nature of our method.

Method	NeRF	LFN	Ours	NeRF	LFN	Ours
Scene	SSIM			PSNR		
Amethyst	0.976	0.979	0.982	37.43	38.92	39.32
Beans	0.984	0.996	0.997	37.51	46.61	47.85
Bracelet	0.986	0.992	0.992	35.32	38.89	38.44
Bulldozer	0.967	0.971	0.966	35.36	36.81	34.36
Bunny	0.985	0.986	0.986	41.73	42.52	41.52
Chess	0.987	0.988	0.987	39.71	41.39	39.17
Flowers	0.962	0.977	0.970	33.80	38.28	35.24
Knights	0.976	0.947	0.980	34.69	30.73	36.14
Tarot-L	0.605	0.960	0.962	17.52	31.37	30.92
Tarot-S	0.763	0.973	0.982	21.73	33.08	34.99
Treasure	0.948	0.971	0.956	30.66	34.42	30.65
Truck	0.977	0.980	0.981	38.00	39.05	38.63

Table 6.4: Detailed known views results on Stanford Light Field Scenes.

Method	NeRF	LFN	Ours	NeRF	LFN	Ours
Scene	SSIM			PSNR		
Amethyst	0.977	0.960	0.979	37.56	32.67	38.14
Beans	0.970	0.994	0.994	32.91	42.48	43.74
Bracelet	0.989	0.962	0.992	36.46	29.22	37.81
Bulldozer	0.969	0.937	0.963	35.59	28.44	33.56
Bunny	0.985	0.975	0.984	41.77	35.61	40.83
Chess	0.987	0.967	0.985	40.05	31.62	37.35
Flowers	0.965	0.950	0.971	34.45	30.10	35.71
Knights	0.976	0.782	0.976	34.11	19.26	34.54
Tarot-L	0.521	0.933	0.937	15.68	27.00	25.90
Tarot-S	0.738	0.971	0.980	21.09	32.33	34.04
Treasure	0.954	0.931	0.962	31.28	26.72	31.62
Truck	0.978	0.961	0.979	38.42	32.60	36.01

Table 6.5: Detailed novel view results on Stanford Light Field Scenes.

Method	NeRF	LFN	Ours	NeRF	LFN	Ours
Scene	SSIM			PSNR		
Fern	0.860	0.838	0.814	25.61	25.02	24.53
Flower	0.921	0.949	0.900	29.11	33.02	28.06
Fortress	0.943	0.945	0.892	32.14	32.13	30.27
Horns	0.898	0.893	0.847	28.15	29.27	26.51
Leaves	0.790	0.833	0.674	21.65	23.84	20.01
Orchids	0.794	0.832	0.816	22.55	24.67	23.79
Room	0.975	0.962	0.949	34.24	30.87	30.28
Trex	0.929	0.945	0.914	27.78	30.14	27.67
M1	0.958	0.971	0.953	29.49	32.30	30.39
M2	0.987	0.988	0.989	35.75	36.48	35.78
W1	0.963	0.968	0.975	32.99	33.46	34.27

Table 6.6: Detailed known views results on Unstructured Light Field Scenes.

Method	NeRF	LFN	Ours	NeRF	LFN	Ours
Scene	SSIM			PSNR		
Fern	0.832	0.721	0.529	24.20	20.63	15.72
Flower	0.905	0.827	0.529	28.21	23.72	15.929
Fortress	0.948	0.810	0.734	32.39	25.09	23.90
Horns	0.907	0.811	0.687	27.69	22.75	20.22
Leaves	0.776	0.603	0.284	20.87	16.72	12.49
Orchids	0.786	0.360	0.346	21.72	11.96	13.83
Room	0.962	0.844	0.790	29.94	21.59	19.16
Trex	0.940	0.875	0.816	28.34	23.78	22.35
M1	0.946	0.894	0.844	25.43	18.13	13.57
M2	0.986	0.968	0.897	32.63	25.85	15.30
W1	0.964	0.958	0.854	27.23	24.66	12.37

Table 6.7: Detailed novel view results on Unstructured Light Field Scenes.

6.7 Conclusion

Images have been an indispensable data primitive in graphics and vision. Exciting recent developments are advancing INR of images towards two goals: image fitting and view synthesis. Instead of pushing further ahead along either direction with increasingly specialized techniques, we look sideways to explore a new possibility of fusing those two directions together. Results from our study show that with careful modifications, INRs can perform view interpolation through code interpolation in appropriate scenarios. Although the method is limited by its inherent lack of 3D knowledge, our study presents a proof of concept revealing an unrealized potential of INR of images. Our success in adapting CLIP, a pre-trained deep network, to guide the INR training also suggests that future developments of INR could further benefit from absorbing concurrent progresses in other areas of deep learning. As INRs of images evolve to be more accurate and efficient, with this paper, we offer a promising outlook on employing INRs for image manipulation tasks beyond fitting and super-resolving known images.

Chapter 7: Conclusion and Future Work

In this dissertation, I have presented my research on designing machine learning algorithms to enhance the creation and representation of immersive visual data. The topics include depth estimation from 360-degree images, effective compression of light fields, efficient rendering of implicit 3D shapes, and photorealistic view interpolations. The contributions I made on these topics collectively advance the state-of-the-art in generating immersive visual experiences from real-world data.

In the future, I foresee machine learning playing an increasingly significant role in processing and generating visual data. With the development of increasingly powerful algorithms and innovative paradigms, I expect a substantial shift towards automation in many aspects of content creation that still rely on manual labor today. At the time of writing this dissertation, the impressive capabilities of generative AI have started to gain recognition beyond the confines of academia, rapidly permeating into the broader society. This shift is likely to drive further research and investment into AI technologies, accelerating the progress in this field and leading to even more sophisticated and versatile tools for visual data generation and manipulation. The integration of these advancements into everyday life will not only streamline workflows and enhance creative possibilities but also contribute to the democratization

of content creation, making it more accessible for people with diverse backgrounds and skill sets.

Moreover, beyond content creation, I anticipate machine learning to play a pivotal role in unveiling previously hidden or invisible aspects of the physical world. Extended reality, in my view, is not merely about superimposing visually appealing graphics on a near-eye display for users to enjoy immersive experiences. Instead, the ultimate goal of extended reality is to empower people with the ability to perceive previously unseen elements and gain a deeper understanding of the world around them. Achieving significant progress towards this vision necessitates a synergistic co-design and understanding between imaging hardware and machine learning. By integrating advanced sensors, cameras, and other data acquisition devices with cutting-edge machine learning algorithms, we can unlock new possibilities for interpreting and analyzing complex visual data. This synergy will enable us to reconstruct and visualize hidden structures, detect subtle patterns, and reveal insights that were previously unattainable. Such advancements will not only enrich our visual experience of the physical world but also facilitate important breakthroughs in various fields such as life and biomedical sciences, environmental protection, physics research, and astronomy and space exploration, ultimately transforming the way we interact with and comprehend our surroundings, across both macro and nano scales.

Bibliography

[1] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 448–465, 2018.

[2] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[3] J. Chang and Y. Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[4] P. K. Lai, S. Xie, J. Lang, and R. Laganière. Real-time panoramic depth maps from omni-directional stereo images for 6 dof videos in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 405–412. IEEE, 2019.

[5] S. Xie, P. K. Lai, R. Laganiere, and J. Lang. Effective convolutional neural network layers in flow estimation for omni-directional images. In *2019 International Conference on 3D Vision (3DV)*, pages 671–680. IEEE, 2019.

[6] E. Miandji, S. Hajisharif, and J. Unger. A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos. In *ACM Transactions on Graphics (TOG)*, volume 38, pages 1–18, 2019.

[7] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[8] A. Cohen, I. Daubechies, and J. Feauveau. Biorthogonal Bases of Compactly Supported Wavelets. *Communications on pure and applied mathematics*, 45(5):485–560, 1992.

[9] J. Jin, J. Hou, H. Yuan, and S. Kwong. Learning Light Field Angular Super-Resolution via a Geometry-Aware Network. In *Proceedings of the AAAI Conference on Artificial Intelligence 2020*, 2020.

[10] Brandon Yushan Feng and Amitabh Varshney. SIGNET: Efficient Neural Representation for Light Fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14204–14213, 2021.

[11] S. Bista, Í. da Cunha, and A. Varshney. Kinetic depth images: Flexible generation of depth perception. *The Visual Computer*, 33(10):1357–1369, 2017.

[12] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney. Montage4d: Real-time seamless fusion and stylization of multiview video textures. *Journal of Computer Graphics Techniques*, 8(1), 2019.

[13] R. Patro, C. Y. Ip, S. Bista, and A. Varshney. Social snapshot: A system for temporally coupled social photography. *IEEE Computer Graphics and Applications*, 31(1):74–84, 2010.

[14] A. C. Sankaranarayanan, R. Patro, P. Turaga, A. Varshney, and R. Chellappa. Modeling and visualization of human activities for multicamera networks. *EURASIP Journal on Image and Video Processing*, 2009:1–13, 2009.

[15] R. Du, D. Li, and A. Varshney. Project geollery.com: Reconstructing a live mirrored world with geotagged social media. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019.

[16] R. Du, D. Li, and A. Varshney. Geollery: A mixed reality social media platform. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[17] R. Du, D. Li, and A. Varshney. Interactive fusion of 360 images for a mirrored world. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 900–901. IEEE, 2019.

[18] R. Du and A. Varshney. Social street view: Blending immersive street views with geo-tagged social media. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 77–85, 2016.

[19] R. Du, S. Bista, and A. Varshney. Video fields: Fusing multiple surveillance videos into a dynamic virtual environment. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 165–172, 2016.

[20] N. Zioulis, A. Karakottas, D. Zarpalas, F. Alvarez, and P. Daras. Spherical view synthesis for self-supervised 360° depth estimation. In *2019 International Conference on 3D Vision (3DV)*, pages 690–699. IEEE, 2019.

[21] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet:geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291, 2018.

[22] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[23] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1253–1260. IEEE, 2010.

[24] M. Liu, M. Salzmann, and X. He. Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2014.

[25] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2015.

[26] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016.

[27] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, volume 18, pages 1–8, 2005.

[28] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015.

[29] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5354–5362, 2017.

[30] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016.

[31] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv preprint arXiv:1702.06506*, 2017.

[32] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features

and hierarchical crfs. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

[33] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pages 37–44. IEEE, 2017.

[34] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[35] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic. Salnet360:saliency maps for omni-directional images with cnn. *Signal Processing: Image Communication*, 69:26–34, 2018.

[36] H. Cheng, C. Chao, J. Dong, H. Wen, T. Liu, and M. Sun. Cube padding for weakly-supervised saliency prediction in 360° videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2018.

[37] F. Wang, H. Hu, H. Cheng, J. Lin, S. Yang, M. Shih, H. Chu, and M. Sun. Self-supervised learning of depth and camera motion from 360° videos. In *Asian Conference on Computer Vision*, pages 53–68. Springer, 2018.

[38] Y. Su and K. Grauman. Learning spherical convolution for fastfeatures from 360° imagery. *Advances in Neural Information Processing Systems*, 30:529–539, 2017.

[39] Y. Su and K. Grauman. Kernel transformer networks for compact spherical convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9442–9451, 2019.

[40] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 472–480, 2017.

[41] Y. Lee, J. Jeong, J. Yun, W. Cho, and K. Yoon. Spherephd: Applying cnns on a spherical polyhedron representation of 360° images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9181–9189, 2019.

[42] K. Tateno, N. Navab, and F. Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 707–722, 2018.

[43] Y. Zhang, F. Dai, Y. Ma, H. Li, Q. Zhao, and Y. Zhang. Saliency detection in 360° videos. *IEEE Journal of Selected Topics in Signal Processing*, 14(1):27–37, 2019.

[44] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[45] G. Payen de La Garanderie, A. A. Abarghouei, and T. P. Breckon. Eliminating the blind spot: Adapting 3d object detection and monocular depth estimation to 360 panoramic imagery. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 789–807, 2018.

[46] N. Wang, B. Solarte, Y. Tsai, W. Chiu, and M. Sun. 360sd-net: 360 stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 582–588. IEEE, 2020.

[47] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. Surge: Surface regularized geometry estimation from a single image. *Advances in Neural Information Processing Systems*, 29:172–180, 2016.

[48] D. Eigen and R. Fergus. Eigen2014. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[49] A. Karakottas, N. Zioulis, S. Samaras, D. Ataloglou, V. Gkitsas, D. Zarpalas, and P. Daras. 360° surface regression with a hyper-sphere loss. In *2019 International Conference on 3D Vision (3DV)*, pages 258–268. IEEE, 2019.

[50] Q. Ge, A. Varshney, J. Menon, and C. Chang. Double quaternions for motion interpolation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 80340, page V004T04A021. American Society of Mechanical Engineers, 1998.

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[52] J. M. McCarthy. Planar and spatial rigid motion as special cases of spherical and 3-spherical motion. *Journal of Mechanisms, Transmissions, and Automation in Design*, 1983.

[53] J. M. McCarthy. The generalization of line trajectories in spatial kinematics to trajectories of great circles on a hypersphere. *Journal of Mechanisms, Transmissions, and Automation in Design*, 1986.

[54] I. Armeni, S. Sax, A. R Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[55] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014.

[56] I. Viola, M. Řeřábek, and T. Ebrahimi. Comparison and Evaluation of Light Field Image Coding Approaches. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1092–1106, 2017.

[57] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu. Light Field Image Processing: An Overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017.

[58] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV 2020)*, pages 405–421, 2020.

[59] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations With Periodic Activation Functions. volume 33, pages 7462–7473, 2020.

[60] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, NeurIPS 2020, 2020.

[61] C. Chang, X. Zhu, P. Ramanathan, and B. Girod. Light Field Compression Using Disparity-compensated Lifting and Shape Adaptation. *IEEE Transactions on Image Processing*, 15:793–806, 2006.

[62] A. Jagmohan, A. Sehgal, and N. Ahuja. Compression of lightfield rendered images using coset codes. *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, 1:830–834, 2003.

[63] M. Magnor and B. Girod. Data compression for light-field rendering. *IEEE Trans. Circuits Syst. Video Technol.*, 10:338–343, 2000.

[64] S. Pratapa and D. Manocha. RLFC: Random Access Light Field Compression using Key Views and Bounded Integer Sequence Encoding. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2019.

[65] X. Zhu, A. Aaron, and B. Girod. Distributed Compression for Large Camera Arrays. *IEEE Workshop on Statistical Signal Processing, 2003*, pages 30–33, 2003.

[66] M. Broxton, John Flynn, R. Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39:86:1–86:15, 2020.

[67] X. Jiang, M. Le Pendu, R. A. Farrugia, and C. Guillemot. Light Field Compression With Homography-Based Low-Rank Approximation. *IEEE Journal of Selected Topics in Signal Processing*, 11:1132–1145, 2017.

[68] 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. *ACM Transactions on Graphics (TOG)*, 37:1–15, 2018.

[69] G. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1649–1668, 2012.

[70] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje. The Latest Open-source Video Codec VP9 - An Overview and Preliminary Results. *2013 Picture Coding Symposium (PCS)*, pages 390–393, 2013.

[71] M. Le Pendu, C. Guillemot, and A. Smolic. A Fourier Disparity Layer Representation for Light Fields. *IEEE Transactions on Image Processing*, 28:5740–5753, 2019.

[72] M. Le Pendu and A. Smolic. High Resolution Light Field Recovery with Fourier Disparity Layer Completion, Demosaicing, and Super-Resolution. *2020 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12, 2020.

[73] E. Dib, M. Le Pendu, and C. Guillemot. Light Field Compression Using Fourier Disparity Layers. *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3751–3755, 2019.

[74] M. Le Pendu, C. Ozcinar, and A. Smolic. Hierarchical Fourier Disparity Layer Transmission For Light Field Streaming. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2606–2610. IEEE, 2020.

[75] A. Abdi, A. Payani, and F. Fekri. Learning Dictionary for Efficient Signal Compression. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3689–3693, 2017.

[76] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro. Online Dictionary Learning for Sparse Coding. In *Proceedings of the 26th annual International Conference on Machine Learning (ICML '09)*, pages 689–696, 2009.

[77] E. Miandji, J. Kronander, and J. Unger. Learning-Based Compression of Surface Light Fields for Real-Time Rendering of Global Illumination Scenes. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, 2013.

[78] J. Sulam, V. Papyan, Y. Romano, and M. Elad. Multilayer Convolutional Sparse Modeling: Pursuit and Dictionary Learning. *IEEE Transactions on Signal Processing*, 66:4090–4104, 2018.

[79] S. Hajisharif, E. Miandji, P. Larsson, K. Tran, and J. Unger. Light Field Video Compression and Real Time Rendering. *Computer Graphics Forum*, 38(7):265–276, 2019.

[80] J. Chai, S. Chan, H. Shum, and X. Tong. Plenoptic Sampling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 307–318, 2000.

[81] G. Chaurasia, O. Sorkine, and G. Drettakis. Silhouette-Aware Warping for Image-Based Rendering. In *Computer Graphics Forum*, volume 30, pages 1223–1232. Wiley Online Library, 2011.

[82] X. Meng, R. Du, J. F. JaJa, and A. Varshney. 3D-Kernel Foveated Rendering for Light Fields. *IEEE Transactions on Visualization and Computer Graphics*, 27(8):3350–3360, 2021.

[83] H. Shum, S. Chan, and S. Kang. *Image-based rendering*. Springer Science & Business Media, 2008.

[84] M. Bemana, K. Myszkowski, H. Seidel, and T. Ritschel. X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation. *ACM Trans. Graph.*, 39(6), November 2020.

[85] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deep Stereo: Learning to Predict New Views from the World's Imagery. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5524, 2016.

[86] N. K. Kalantari, T. Wang, and R. Ramamoorthi. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics (TOG)*, 35:1–10, 2016.

[87] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu. Light Field Reconstruction Using Deep Convolutional Network on EPI. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1638–1646, 2017.

[88] 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. In *Proceedings of the European Conference on Computer Vision (ECCV 2018)*, 2018.

[89] D. Gottlieb and C. Shu. On the Gibbs Phenomenon and its Resolution. *SIAM Review*, 39:644–668, 1997.

[90] R. Archibald and A. Gelb. A Method to Reduce the Gibbs Ringing Artifact in MRI Scans While Keeping Tissue Boundary Integrity. *IEEE Transactions on Medical Imaging*, 21:305–319, 2002.

[91] S. Gottlieb, J. Jung, and S. Kim. A Review of David Gottlieb's Work on the Resolution of the Gibbs Phenomenon. *Communications in Computational Physics*, 9:497–519, 2011.

[92] D. Gottlieb and C. Shu. On the Gibbs phenomenon IV: recovering exponential accuracy in a subinterval from a Gegenbauer partial sum of a piecewise analytic function. *Mathematics of Computation*, 64:1081–1095, 1995.

[93] A. A. Afifi and E. A. Zanaty. Generalized Legendre Polynomials for Support Vector Machines (SVMs) Classification. *International Journal of Network Security & Its Applications*, 11:87–104, 2019.

[94] L. C. Padierna, M. Carpio, A. R. Domínguez, H. J. P. Soberanes, and H. J. Fraire. A Novel Formulation of Orthogonal Polynomial Kernel Functions for SVM Classifiers: The Gegenbauer Family. *Pattern Recognition*, 84:211–225, 2018.

[95] B. Y. Feng, W. Yao, Z. Liu, and A. Varshney. Deep Depth Estimation on 360° Images with a Double Quaternion Loss. In *2020 International Conference on 3D Vision (3DV)*, pages 524–533, 2020.

[96] A. Karakottas, N. Zioulis, S. Samaras, D. Ataloglou, V. Gkitsas, D. Zarpalas, and P. Daras. 360° Surface Regression with a Hyper-Sphere Loss. In *2019 International Conference on 3D Vision (3DV)*, pages 258–268, 2019.

[97] N. Rahaman, A. Baratin, D. Arpit, F. Dräxler, M. Lin, F. Hamprecht, Y. Bengio, and A. C. Courville. On the Spectral Bias of Neural Networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[98] A. Adams. The Stanford Light Field Archive, 2008. http://lightfield.stanford.edu/lfs.html.

[99] N. Sabater, G. Boisson, B. Vandame, P. Kerbiriou, F. Babon, M. Hog, R. Gendrot, T. Langlois, O. Bureller, A. Schubert, and V. Allie. Dataset and Pipeline for Multi-view Light-Field Video. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1743–1753, 2017.

[100] V. Sitzmann, E. Chan, R. Tucker, N. Snavely, and G. Wetzstein. MetaSDF: Meta-learning Signed Distance Functions. In *Advances in Neural Information Processing Systems*, NeurIPS 2020, 2020.

[101] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. Barron, and R. Ng. Learned Initializations for Optimizing Coordinate-Based Neural Representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, pages 2846–2855, 2021.

[102] S. Pratapa and D. Manocha. HMLFC: Hierarchical Motion-Compensated Light Field Compression for Interactive Rendering. *Computer Graphics Forum*, 38, 2019.

[103] J. Nystad, A. Lassen, A. Pomianowski, S. Ellis, and T. Olson. Adaptive Scalable Texture Compression. In *EGGH-HPG'12*, 2012.

[104] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. In *Advances in Neural Information Processing Systems*, NeurIPS 2021, 2021.

[105] Brandon Yushan Feng, Yinda Zhang, Danhang Tang, Ruofei Du, and Amitabh Varshney. PRIF: Primary Ray-based Implicit Function. In *Proceedings of the European Conference on Computer Vision (ECCV 2022)*, 2022.

[106] Brandon Yushan Feng, Susmija Jabbireddy, and Amitabh Varshney. VIINTER: View Interpolation With Implicit Neural Representations of Images. In *ACM SIGGRAPH Asia 2022 Conference Proceedings*, pages 1–9, 2022.

[107] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit Neural Video Compression. *arXiv preprint arXiv:2112.11312*, 2021.

[108] G. W. Stewart. An Updating Algorithm for Subspace Tracking. *IEEE Trans. Signal Processing*, 40:1535–1541, 1992.

[109] F. D. L. Torre and M. J. Black. A Framework for Robust Subspace Learning. *International Journal of Computer Vision*, 54:117–142, 2004.

[110] B. Wang and Z. Tu. Sparse Subspace Denoising for Image Manifolds. *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 468–475, 2013.

[111] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy. Robust Subspace Learning: Robust PCA, Robust Subspace Tracking, and Robust Subspace Recovery. *IEEE Signal Processing Magazine*, 35:32–55, 2018.

[112] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[113] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual Dense Network for Image Super-Resolution. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2018.

[114] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. DeepView: View Synthesis With Learned Gradient Descent. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2362–2371, 2019.

[115] Stefan Winkler, Elisa Drelie Gelasca, and Touradj Ebrahimi. Toward Perceptual Metrics for Video Watermark Evaluation. In *Applications of Digital Image Processing XXVI*, volume 5203, pages 371–378. SPIE, 2003.

[116] David Li, Ruofei Du, Adharsh Babu, Camelia D Brumar, and Amitabh Varshney. A Log-Rectilinear Transformation for Foveated 360-degree Video Streaming. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2638–2647, 2021.

[117] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned Initializations for Optimizing Coordinate-based Neural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2846–2855, 2021.

[118] Alexander Bergman, Petr Kellnhofer, and Gordon Wetzstein. Fast Training of Neural Lumigraph Representations Using Meta Learning. *Advances in Neural Information Processing Systems*, 34:172–186, 2021.

[119] Charalampos Koniaris, Maggie Kosek, David Sinclair, and Kenny Mitchell. Compressed Animated Light Fields with Real-time View-dependent Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1666–1680, 2018.

[120] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering Deep Implicit Signed Distance Function With Differentiable Sphere Tracing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2020.

[121] John C Hart. Sphere Tracing: a Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[122] William E Lorensen and Harvey E Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. *ACM Siggraph Computer Graphics*, 21(4):163–169, 1987.

[123] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. AK Peters/CRC Press, 2019.

[124] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[125] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[126] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[127] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks. In *European Conference on Computer Vision*, pages 523–540. Springer, Springer, 2020.

[128] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6968–6979, 2020.

[129] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations Without 3D Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[130] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D Objects With Differentiable Rendering of SDF Shape Priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12224–12233, 2020.

[131] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1251–1261, 2020.

[132] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. Curriculum Deepsdf. In *European Conference on Computer Vision*, pages 51–67. Springer, Springer, 2020.

[133] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-Learning Signed Distance Functions. *Advances in Neural Information Processing Systems*, 33:10136–10147, 2020.

[134] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[135] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction. *Advances in Neural Information Processing Systems*, 32, 2019.

[136] Matan Atzmon and Yaron Lipman. Sal: Sign Agnostic Learning of Shapes From Raw Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[137] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6970–6981, 2020.

[138] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to Infer Implicit Surfaces Without 3D Supervision. *Advances in Neural Information Processing Systems*, 32, 2019.

[139] Danny Driess, Jung-Su Ha, Marc Toussaint, and Russ Tedrake. Learning Models As Functionals of Signed-Distance Fields for Manipulation Planning. In *Conference on Robot Learning*, pages 245–255. PMLR, 2022.

[140] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution Deep Implicit Functions for 3D Shape Representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13087–13096, 2021.

[141] Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, et al. Deep Implicit Volume Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1293–1303, 2020.

[142] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation. *ArXiv Preprint ArXiv:2112.05124*, 2021.

[143] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction. In *European Conference on Computer Vision*, pages 311–329. Springer, Springer, 2020.

[144] Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan D Jepson. Representing 3d shapes with probabilistic directed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19343–19354, 2022.

[145] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry Processing With Neural Fields. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[146] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. *Advances in Neural Information Processing Systems*, 34, 2021.

[147] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren

Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[148] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. Patchnets: Patch-Based Generalizable Deep Implicit 3D Shape Representations. In *European Conference on Computer Vision*, pages 293–309. Springer, Springer, 2020.

[149] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning Shape Templates With Structured Implicit Functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019.

[150] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Niessner, Thomas Funkhouser, et al. Local Implicit Grid Representations for 3D Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[151] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local Deep Implicit Functions for 3D Shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[152] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, Springer, 2020.

[153] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural Parts: Learning Expressive 3D Shape Abstractions With Invertible Neural Networks. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3203–3214, 2021.

[154] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021.

[155] Shun Yao, Fei Yang, Yongmei Cheng, and Mikhail G Mozerov. 3D Shapes Local Geometry Codes Learning With SDF. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2110–2117, 2021.

[156] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-Net: Generating Compact Meshes Via Binary Space Partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020.

[157] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable Convex Decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020.

[158] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive Coordinate Networks for Neural Scene Representation. *ArXiv Preprint ArXiv:2105.02788*, 2021.

[159] David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: Band-Limited Coordinate Networks for Multiscale Scene Representation. *ArXiv Preprint ArXiv:2112.04645*, 2021.

[160] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives With a Multiresolution Hash Encoding. *ArXiv Preprint ArXiv:2201.05989*, 2022.

[161] Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G. Baraniuk, and Ashok Veeraraghavan. MINER: Multiscale Implicit Neural Representations, 2022.

[162] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes As Neural Radiance Fields for View Synthesis. In *ECCV 2020: Computer Vision – ECCV 2020*, pages 405–421. Springer International Publishing, 2020.

[163] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene Representation Networks: Continuous 3d-Structure-Aware Neural Scene Representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[164] Brandon Yushan Feng and Amitabh Varshney. SIGNET: Efficient Neural Representation for Light Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14224–14233, 2021.

[165] Benjamin Attal, Jia-Bin Huang, Michael Zollhoefer, Johannes Kopf, and Changil Kim. Learning Neural Light Fields With Ray-Space Embedding Networks. *ArXiv Preprint ArXiv:2112.01523*, 2021.

[166] Yan-Bin Jia. Plucker Coordinates for Lines in the Space. https://faculty.sites.iastate.edu/jia/files/inline-files/plucker-coordinates.pdf, 2020.

[167] Greg Turk and Marc Levoy. Zippered Polygon Meshes From Range Images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 311–318, 1994.

[168] Brian Curless and Marc Levoy. A Volumetric Method for Building Complex Models From Range Images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 303–312, 1996.

[169] Michael Kazhdan and Hugues Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[170] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, volume 2008, pages 129–136. Salerno, Italy, 2008.

[171] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: an Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[172] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST, pages 829–843. ACM, Oct. 2020.

[173] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. In Alexander Wilkie and Francesco Banterle, editors, *41st Annual Conference of the European Association for Computer Graphics, Eurographics 2020 - Short Papers, Norrköping, Sweden, May 25-29, 2020 [online only]*, pages 41–44. Eurographics Association, 2020.

[174] Renderpeople. renderpeople.com, 2022.

[175] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[176] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for Real-time Rendering of Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[177] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural Actor: Neural Free-view Synthesis of Human Actors With Pose Control. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021.

[178] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021.

[179] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, 2021.

[180] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[181] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[182] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: GAN-based Neural Radiance Field Without Posed Camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6351–6361, 2021.

[183] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF–: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[184] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep Reflectance Volumes: Relightable Reconstructions From Multi-view Photometric Images. In *European Conference on Computer Vision*, pages 294–311. Springer, 2020.

[185] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse Rendering With Spherical Gaussians For Physics-based Material Editing And Relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021.

[186] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural Scene Representation and Rendering. *Science*, 360(6394):1204–1210, 2018.

[187] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3D Models From Single Images With A Convolutional Network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016.

[188] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to Generate Chairs, Tables and Cars with Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):692–705, 2016.

[189] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative Filter Networks. In *International Conference on Learning Representations*, 2020.

[190] Zhichun Huang, Shaojie Bai, and J Zico Kolter. Implicit Layers for Implicit Representations. *Advances in Neural Information Processing Systems*, 34, 2021.

[191] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated Periodic Activations for Generalizable Local Functional Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14214–14223, 2021.

[192] Emilien Dupont, Hyunjik Kim, SM Eslami, Danilo Rezende, and Dan Rosenbaum. From Data to Functa: Your Data Point Is A Function And You Should Treat It Like One. In *ICML*, 2022.

[193] Brandon Yushan Feng, Yinda Zhang, Danhang Tang, Ruofei Du, and Amitabh Varshney. Prif: Primary ray-based implicit function. In *European Conference on Computer Vision (ECCV)*, 2022.

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

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

[196] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and Rendering Architecture From Photographs: A Hybrid Geometry-and Image-based Approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996.

[197] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, 2000.

[198] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and L Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*, pages 94–101. Springer, 1999.

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

[200] Shenchang Eric Chen and Lance Williams. View Interpolation for Image Synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[201] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4d: Interactive seamless fusion of multiview video textures. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–11, 2018.

[202] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[203] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020.

[204] Justus Thies, Michael Zollhofer, and Matthias Niessner. Deferred Neural Rendering: Image Synthesis Using Neural Textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[205] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning Multi-view Image-based Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[206] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local Light Field Fusion: Practical View Synthesis With Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[207] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14356–14366, 2021.

[208] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. *Light Field Photography With A Hand-held Plenoptic Camera*. PhD thesis, Stanford University, 2005.

[209] George Wolberg. Image Morphing: A Survey. *The visual Computer*, 14(8):360–372, 1998.

[210] Steven M Seitz and Charles R Dyer. View Morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30, 1996.

[211] Jing Liao, Rodolfo S Lima, Diego Nehab, Hugues Hoppe, Pedro V Sander, and Jinhui Yu. Automating Image Morphing Using Structural Similarity on A Halfway Domain. *ACM Transactions on Graphics (TOG)*, 33(5):1–12, 2014.

[212] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[213] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF On A Diet: Semantically Consistent Few-shot View Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[214] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *The European Conference on Computer Vision (ECCV)*, 2022.