Neural Scene Representations for 3D Reconstruction and Generative Modeling

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Michael Niemeyer
aus Kleve

Tübingen
2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.09.2023
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr. Andreas Geiger
2. Berichterstatter/-in:	Prof. Dr. Gerard Pons-Moll
3. Berichterstatter/-in:	Prof. Dr. Shubham Tulsiani

Abstract

With the increasing technologization of society, we use machines for more and more complex tasks, ranging from driving assistance to video conferencing, to exploring planets. The scene representation, i.e., how sensory data is converted to compact descriptions of the environment, is a fundamental property for enabling the success but also the safety of such systems. A promising approach for developing robust, adaptive, and powerful scene representations are learning-based systems that can adapt themselves from observations. Indeed, deep learning has revolutionized computer vision in recent years. In particular, better model architectures, large amounts of training data, and more powerful computing devices enabled deep learning systems with unprecedented performance, and they now set the state-of-the-art in many benchmarks, ranging from image classification, to object detection, to semantic segmentation. Despite these successes, the way these systems operate is still fundamentally different from human cognition. In particular, most approaches operate in the 2D domain, while humans understand that images are projections of the three-dimensional world. In addition, they often do not follow a compositional understanding of scenes, which is fundamental to human reasoning. In this thesis, our goal is to develop scene representations that enable autonomous agents to navigate and act robustly and safely in complex environments while reasoning compositionally in 3D. To this end, we first propose a novel output representation for deep learning-based 3D reconstruction and generative modeling. We find that, compared to previous representations, our neural field-based approach does not require 3D space to be discretized achieving reconstructions at arbitrary resolution with a constant memory footprint. Next, we develop a differentiable rendering technique to infer these neural field-based 3D shape and texture representations from 2D observations and find that this allows us to scale to more complex, real-world scenarios. Subsequently, we combine our novel 3D shape representation with a spatially and temporally continuous vector field to model non-rigid shapes in motion. We observe that our novel 4D representation can be used for various discriminative and generative tasks, ranging from 4D reconstruction to 4D interpolation, to motion transfer. Finally, we develop an object-centric generative model that can generate 3D scenes in a compositional manner and that allows for photorealistic renderings of generated scenes. We find that our model not only improves image fidelity but also enables more controllable scene generation and image synthesis than prior work while training only from raw, unposed image collections.

Kurzfassung

Mit der zunehmenden Technologisierung der Gesellschaft nutzen wir Maschinen für immer komplexere Aufgaben, die sich von Fahrassistenz über Videokonferenzen zur Erkundung neuer Planeten erstrecken. Die Repräsentation einer Szene, i.e., wie Sensordaten in kompakte Beschreibungen der Umwelt umgewandelt werden, ist eine grundlegende Eigenschaft für den Erfolg wie auch die Sicherheit solcher Systeme. Ein vielversprechender Ansatz für die Entwicklung robuster, adaptiver und leistungsstarker Repräsentationen sind lern-basierte Systeme, die sich auf Grundlage von Beobachtungen selbst anpassen können. In der Tat hat Deep Learning den Bereich des maschinellen Sehens in den letzten Jahren revolutioniert. Besonders haben bessere Modellarchitekturen, mehr Trainingsdaten und bessere Rechengeräte lern-basierte Methoden mit noch nie dagewesenen Leistungen ermöglicht und sie setzen nun die neuen Standards in Benchmarks, die von Bildklassifizierung über die Objekterkennung bis zur semantischen Segmentierung reichen. Trotz dieser Erfolge unterscheidet sich die Funktionsweise dieser Systeme meistens grundlegend von der menschlichen Erkenntnis. Insbesondere operieren die meisten Ansätze zweidimensional, während der Mensch Bilder als Projektionen der dreidimensionalen Welt versteht. Darüber hinaus folgen sie oft nicht dem kompositorischen Verständnis von Szenen, das für das menschliche Denken grundlegend ist. Das Ziel dieser Arbeit ist, Repräsentationen von Szenen zu entwickeln, die es autonomen Agenten ermöglichen, in komplexen Umgebungen robust und sicher zu navigieren und zu handeln, während sie ihre Umwelt in 3D und kompositorisch begreifen können. Mit diesem Ziel entwickeln wir zunächst eine neuartige 3D Repräsentation für Deep Learning-basierte 3D Rekonstruktion und generative Modellierung. Wir stellen fest, dass unser auf neuronalen Netzen basierender Ansatz im Vergleich zu vorherigen Repräsentationen keine Diskretisierung des 3D Raums erfordert und Rekonstruktionen mit beliebiger Auflösung und konstantem Speicherbedarf ermöglicht. Im Anschluss entwickeln wir ein differenzierbares Rendering-Verfahren, um diese 3D Geometrie und Texturrepräsentationen von 2D Informationen zu inferieren. Wir zeigen auf, dass dieser Ansatz eine Skalierung auf komplexere, reale Szenen ermöglicht. Anschließend kombinieren wir unsere neuartige 3D Geometriedarstellung mit einem räumlich und zeitlich kontinuierlichen Vektorfeld, um nicht-starre 3D Strukturen in Bewegung zu modellieren. Unsere neuartige 4D Darstellung kann für verschiedene diskriminative und generative Anwendungen verwendet werden, welche von der 4D Rekonstruktion über die 4D Interpolation bis hin zur Bewegungsübertragung reichen. Darauf folgend entwickeln wir ein objektzentriertes generatives Modell, das 3D Szenen auf kompositorische Weise erzeugen kann und zudem fotorealistische Renderings der erzeugten Szenen ermöglicht. Wir stellen fest, dass unser Modell nicht nur die Bildqualität verbessert, sondern auch eine besser kontrollierbare Generierung von Szenen und anschließende Bildsynthese dieser Szenen ermöglicht, während nur Datensätze von Bildern ohne Annotationen für das Training benötigt werden.

Acknowledgments

Advisor I would like to thank my supervisor Andreas Geiger. First, I am deeply grateful for providing me the opportunity to start as a Ph.D. student in such a top-tier research group despite having barely any experience in the field. Andreas seeks talent over experience or reputation which I admire. Further, I am deeply grateful for him being so involved and supportive in all the research projects I was part of. He is not only incredibly knowledgeable in the field but also enjoys pushing himself to produce research of the highest quality. Finally, he supports and supervises students according to their individual standards, always trying to facilitate their own personal or career development. I am deeply impressed by the research environment he creates and deeply thankful for having been a part of it.

Co-Authors I would like to thank Lars Mescheder and Michael Oechsle for their incredible collaborations. I am deeply grateful to Lars for taking me on board to such an impactful research project right at the start of my Ph.D. I not only learned so much, but it also shaped my entire Ph.D. which would have been significantly less successful without his guidance and support. Similarly, I am deeply grateful for sharing more than half of my time as a Ph.D. student with Michi - we went to my first conference together, shared offices, had many coffee breaks together, and in general supported each other during the Ph.D. I consider myself very lucky to not only have worked with both of you but also to call you friends.

Next, I would like to thank Songyou Peng - my Ph.D. would have been only half the fun without you. Thanks a lot for being so accepting during my first steps in a more advisory role and for being such a fantastic co-author. I already know how much fun it will be to run into you at the next conference, and I am very grateful for you as a friend.

I would also like to thank Katja Schwarz, Yiyi Liao, and Axel Sauer. I truly enjoyed our times together, especially great conversations over way too many coffee breaks with great Latte art - and of cause, also pushing hard for another deadline only a few days away. I learned so much from you, and I am very happy that our paths crossed.

Further, big thanks to Zehao Yu for being such a great office companion and for helping me test myself in a more advisory position.

Additionally, I want to thank Jon Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, and Noha Radwan for making my internship experience so great. I am very grateful not only for all the advice you provided but also for all the support long after the end of the internship.

Finally, I want to thank Chiyu Jiang, Marc Pollefeys, Thilo Strauss, Torsten Sattler, and Sebastian Nowozin for the great collaborations.

Colleagues While co-authors are captured on papers, most colleagues are not but I would not have survived my Ph.D. without such a supportive and great research group. Many thanks (alphabetically) to Aditya Prakash, Apratim Bhattacharyya, Aseem Behl, Benjamin Coors, Bernhard Jaeger, Bozidar Antic, Carolin Schmitt, Christian Reiser, Despoina Paschalidou, Eshed Ohn-Bar, Haiwen Huang, Joel Janai, Joo Ho Lee, Katrin Renz, Kashyap Chitta, Markus Flicke, Niklas Hanselmann, Seungjun Nah, Simon Donné, Stefan Baur, Xu Chen, Zhaoyang Lv, Zijian Dong.

TAC Committee I am very thankful to the TAC Committee members Isabel Valera and Michael Black for their guidance and great advice during different stages of my Ph.D.

Family, Friends, Partner The biggest support I have received from my family, friends, and partner, and I cannot express how grateful I am for having you in my life. I want to thank in particular Sina who supported me during the good and, more importantly, the hardest times of my Ph.D. I consider myself incredibly lucky to share my visions, ideas, and in general the journey of life with you - I could not do any of it without you.

Administrative Team and Institutions I want to thank Kerstin McGaughey for her great support. You are such an important person for the group, and the amount and pace of the group's research output would truly not be possible without your supportive nature.

I further want to thank the IMPRS-IS coordinators Leila Masri and Sara Sorce. You are one of the driving forces of making the Tübingen area such a great and impactful research environment. Thanks a lot for supporting me during my Ph.D.

I want to additionally thank all the people behind the International Max Planck Research School for Intelligent Systems, the Max Planck Institute for Intelligent Systems, and the University of Tübingen for enabling such fantastic research centers.

Notation and Symbols

General Notation

Scalars	Regular (greek) lower case	a, b, c, λ
Scalar-valued functions	Regular (greek) lower case	f, g, h
Vectors	Bold (greek) lower cases	$\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\boldsymbol{\lambda}$
Vector-valued functions	Bold (greek) lower cases	$\mathbf{f}$, $\mathbf{g}$, $\mathbf{h}$
Matrices	Bold upper case	$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, Σ, Λ
Sets	Calligraphic upper case	$\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$
Distributions	Calligraphic upper case	$\mathcal{U}(\cdot)$, $\mathcal{N}(\cdot)$

Indexing

i	First-order index $i \in \{1,\dots,N\}$
j	First-order index $j \in \{1,\dots,M\}$
k	First-order index $k \in \{1,\dots,K\}$

Spaces

$\mathbb{N}$	Natural numbers
$\mathbb{N}^+$	Natural numbers greater than 0
$\mathbb{R}$	Real numbers
$\mathbb{R}^+$	Real numbers greater than 0
$[a,b]$	Closed interval subset of $\mathbb{R}$
(a,b)	Open interval subset of $\mathbb{R}$
$[a,b),(a,b]$	Half-open interval subset of $\mathbb{R}$
$\mathcal{X}$	Input data space
$\mathcal{Z}$	Latent space

Analysis and Linear Algebra

∇f	Gradient of function f

$\frac{df}{dt}$	Total derivative of f wrt. t
$\frac{\partial f}{\partial t}$	Partial derivative of f wrt. t
$\det(\cdot)$	Determinant

Deep Learning and Deep Generative Modeling

θ, ϕ	Learnable network parameters
G_θ	Deep generator model with learnable parameters θ
D_ϕ	Deep discriminator model with learnable parameters ϕ
E_ϕ	Deep encoder model with learnable parameters ϕ

Probability

$p(\cdot)$	Probability
$\log p(\cdot)$	Log probability
$\mathcal{N}(\cdot)$	Gaussian distribution
$\mathcal{U}(\cdot)$	Uniform distribution
$\mathrm{KL}(\cdot\|\|\cdot)$	Kullback–Leibler (KL) divergence
$p_0(\mathbf{z})$	Prior distribution on $\mathcal{Z}$
$p_D(\mathbf{x})$	Data distribution on $\mathcal{X}$
$q_\phi(\mathbf{z}\|\mathbf{x})$	Posterior distribution on $\mathcal{Z}$ defined via the encoder $E_\phi(\mathbf{x}) = \mathbf{z} \sim q_\phi(\mathbf{z}\|\mathbf{x})$
$p_\theta(\mathbf{x})$	Generator distribution on $\mathcal{X}$ defined via the generator $G_\theta(\mathbf{z}) = \mathbf{x} \sim p_\theta(\mathbf{x}\|\mathbf{z})$

Neural Fields

$\mathbf{f}$	Generic field
$\mathbf{f}_\theta$	Generic neural field with learnable parameters θ
f_θ	Occupancy Network with learnable parameters θ
$\mathbf{t}_\theta$	Texture Field with learnable parameters θ
$\mathbf{v}_\theta$	Velocity Network with learnable parameters θ
$\mathbf{h}_\theta$	Feature Field with learnable parameters θ

Geometry

$\mathbf{p}$	3D Point in $\mathbb{R}^3$
$\mathbf{n}(\cdot)$	Unit surface normal in $\mathbb{R}^3$

$\mathcal{V}$	Scalar Voxel Grid
$\mathcal{P}$	Point Cloud
$\mathcal{M}$, $\mathcal{E}$, $\mathcal{F}$	Mesh, mesh edges, mesh faces
$\mathcal{S}_f$	Implicit surface defined by function f
τ	Level set value for implicit surfaces

Rendering

π	Generic rendering operator
π^{rm}, π^{st}, π^{vol}	Ray marching, sphere tracing, and volume rendering operator
$\pi_\theta^{\text{neural}}$	Neural rendering operator with learnable parameters θ
$\mathbf{K}$	Camera intrinsics matrix in $\mathbb{R}^{3\times3}$
$\xi = [\mathbf{R}\vert\mathbf{t}]$	Camera extrinsics matrix (or camera pose) in $\mathbb{R}^{3\times4}$
$\mathbf{I}$, $\hat{\mathbf{I}}$	ground truth and predicted RGB image
$\mathbf{r}(\cdot), \mathbf{r}_0, \mathbf{d}$	Ray, ray origin, and ray direction
t_n, t_f	Near and far scene bounds

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

With the rapid progress of science, our everyday lives are more and more interconnected with technology. Only 20 years ago, the machines people interacted with were mostly operating independently of their surroundings like an alarm clock that always rings at a specific time or a vending machine that always gives out the same drink for the same input pattern. Today, our society has become more complex and the use of technology along with it, and we now regularly use machines for tasks like video conferences, driving assistance, exploring new planets, and more.

This development requires our machines to solve more complex problems, act more autonomously, and have a better understanding of our world. The *scene representation*, i.e., the process of converting visual sensory data into compact descriptions [67], is crucial for the performance and safety of such systems. It defines the ability not only to parse and reconstruct scenes from input data but also to imagine and generate new, unseen environments. Both capabilities, i.e., imagining possible scenarios as well as accurately understanding the surrounding environment, are equally important for building robust systems that can be employed in the real world.

A promising way to obtain powerful scene representations are *learning-based approaches*, i.e., systems that can learn and adapt themselves from observations [246]. The modern era of deep learning was initiated by the success of Convolutional Neural Networks (CNNs) applied to the task of image classification [138], and since then, CNNs have achieved impressive results in many computer vision tasks ranging from image classification to object detection, to semantic segmentation [143].

While achieving state-of-the-art performance in benchmarks like image classification, these systems are still not able to parse and understand a scene in the way that humans do. One key limitation is that they operate on the two-dimensional image plane. In contrast, humans understand that images are projections of a three-dimensional world. In addition, these systems are trained to predict predefined class labels, per-pixel semantic labels, or object bounding boxes for an input image. They are not capable of imagining how the scene appears from a different viewpoint, what the scene could look like with different lighting and background, or how the scene changes when individual objects are shifted and rotated. While these models lead to success in 2D-based computer vision tasks, they do not follow the compositional understanding of scenes that humans naturally use.

The goal of this thesis is to develop scene representations for learning-based systems which allow for navigating and acting safely in complex environments while reasoning similarly to humans and adhering to fundamental properties of our world and natural scenes. More specifically, we want to use a three-dimensional representation of scenes to be able to perform reasoning in 3D. Next, we want to develop learning algorithms that can be trained from 2D data like single or multi-view images instead of requiring access to accurate 3D ground truth. Further, we want to incorporate an understanding of natural motion that considers smoothness, locality, and correspondences. Finally, we want a system that can not only infer representations from input data but also imagine and generate new scenes in a compositional manner as well as synthesize photorealistic renderings of the generated scenes.

1.2 Applications

Developing such representations enables many applications ranging from autonomous driving to virtual reality applications, to faster and more affordable photorealistic computer graphics. In the following, we discuss these fields of potential applications in greater detail.

Autonomous Driving A key challenge of autonomous driving is handling not only an infinite number of different scene configurations or weather conditions but also critical safety issues. One wrong decision of the autonomous agent can result in life-threatening consequences for the passengers. As a result, such a system should be comprehensible and perform reasoning similar to humans, in contrast to black-box methods that cannot be understood nor dissected in case of wrong behavior. We identify a need for understanding a scene in all three dimensions, and to perform reasoning in 3D as a major component of enabling such a system. As a result, the scene representations that we develop in this thesis can be used in autonomous driving agents that reason in 3D. In addition, the generative model we develop in this thesis can be used as a first step towards a system that allows for automated and fast data generation of realistic training and testing scenarios, which can in turn be used to train and benchmark autonomous driving agents.

Virtual Reality Applications Virtual reality applications become more and more popular across a variety of sectors including entertainment, education, and business areas. However, the use of current systems is still limited mainly due to high costs, too complex pipelines, or lacking realism. In this thesis, we develop techniques for inferring 3D representations from 2D data alone, e.g., single or multi-view images. As a result, the developed systems could be used for reducing the costs and complexity of current applications that might require additional sensory data as input. Further, we aim to develop a system that can produce photorealistic renderings of generated scenes. This can not only increase the realism of current applications but also reduce labor costs as content generation is fully automated.

Photorealistic Computer Graphics The traditional computer graphics pipeline involves designers that first create 3D assets by hand, then arrange 3D objects in the scene, and finally define the camera viewpoint, materials, and lighting to render an image of the scene. This process is labor- and resource-intense, and while potentially achieving photorealistic renderings, a generated video clip might still lack realism, e.g., due to incorrect appearances in 3D or unrealistic motion. The systems we develop in this thesis might serve as a first approach towards achieving a simpler, more affordable, and more realistic content generation pipeline. In addition, the 3D representations that we develop in this thesis can be used to improve individual components of the traditional generation pipeline, e.g., by guiding the 3D asset generation process, helping creators with object arrangements, or speeding up the rendering time.

1.3 Problem Statement

In this thesis, we are interested in developing representations that allow learning-based systems to reconstruct and understand scenes from visual sensory data as well as to generate new, unseen scenes with control over the generative process. Further, we want the system to learn these representations with as little supervision as possible to enable scalability and generalization. To this end, we define the following questions that we address in this thesis:

1. How should 3D geometry be represented in learning-based systems?
2. How can these 3D representations be inferred when only 2D supervision, e.g., in the form of posed single- or multi-view images, is available?
3. How can this representation of 3D geometry be extended to 4D to represent deforming 3D shapes, while considering the smoothness and locality of natural motion?
4. How can we develop an object-centric generative model of 3D representations that allows for controlling the camera viewpoint as well as the location, rotation, appearance and other properties of individual objects in the scene?

1.4 Contributions

The contributions of this thesis are as follows:

- A novel learning-based 3D reconstruction system that uses a neural field representing occupancy in 3D space as 3D representation. Experimental evaluation shows that it compares favorably against previous state-of-the-art methods that use voxel, point cloud, or mesh-based 3D representations.
- A novel differentiable rendering technique that allows training neural fields that represent textured 3D objects from single or multi-view images instead of requiring 3D supervision. We observe that we achieve similar performance to methods having access to 3D supervision and that we outperform other 2D supervised methods. Our method further enables textured 3D reconstruction from real-world multi-view imagery.
- A novel 4D neural field-based representation for 3D shapes in motion. We combine a neural field representing the occupancy in 3D space with a neural temporal-dependent vector field that describes the motion of the shape over time. We find that our novel

learning-based system outperforms baselines and allows for a variety of applications ranging from 4D reconstruction to 4D interpolation, to motion transfer.

- A compositional and 3D-aware generative model that can be trained from raw, unposed image collections. Our method allows for generating new scenes with explicit control over not only the camera viewpoint, but also the shape, size, appearance, position, and rotation of individual objects in the scene. Experimental evaluation shows that we further achieve higher image fidelity than prior state-of-the-art 3D-aware generative models.

Authorship The above-mentioned contributions are part of four peer-reviewed research papers [179, 197, 198, 200] published at computer vision conferences. The author of this thesis additionally made contributions to another nine peer-reviewed and published projects [199, 201, 203, 204, 219, 221, 255, 256, 341] that are not included in the main part of the thesis. Please see Appendix A for a credits discussion and Appendix B for a full publication list and short discussions on specific contribution details.

1.5 Outline

We start by introducing common 3D representations and key concepts of deep generative modeling in Chapter 2. Next, we review related work on 3D reconstruction and generative modeling in Chapter 3. In Chapter 4, we introduce the concept of neural fields and discuss their parameterization and rendering techniques. Subsequently, we investigate how well neural fields are suited for learning-based 3D reconstruction in Chapter 5. We then develop a differentiable surface rendering technique for neural fields in Chapter 6. Next, we extend neural fields to 4D reconstruction in Chapter 7. In Chapter 8, we develop a compositional and 3D-aware deep generative model that can be trained from raw, unposed image collections. We conclude the thesis and discuss broader future research perspectives in Chapter 9.

2 Background

A key property for machines to navigate and act safely in our environments is the capability of accurate three-dimensional reasoning. As a result, the representation of 3D geometry is a crucial part of the systems that we develop in this thesis. Therefore, this background chapter first reviews common 3D representations used in computer vision algorithms in Section 2.1. Note that our focus lies on the 3D representations themselves rather than data structures that can be used to store them. Further, we cover core principles of generative modeling in Section 2.2 as our goal is a model that can not only infer scenes from visual sensory data but also imagine and generate new scenes.

2.1 3D Representations

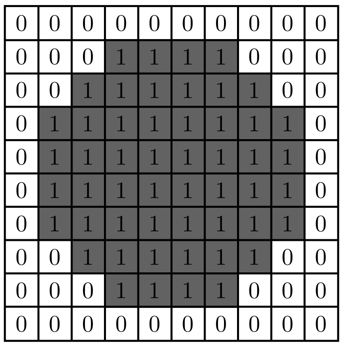
Computer vision investigates how machines can infer information and gain a high-level understanding from images. Hence, imagery data is the most important data type in the field. A property of images that is often underestimated is that the representation of images is almost always the same: color images are represented as a grid of $H \times W \times 3$ values where H indicates the height and W the width of the grid. The three-dimensional vector for each position is called pixel and defines the red, green, and blue components of the color value at that position. In contrast, the representation of 3D geometry is less obvious, and many different approaches are used across the field. In the following, we review the most common 3D representations and analyze their key properties (see Figure 2.1 for an overview). Note that our focus lies on the 3D representations themselves rather than data structures that can be used to store them. For example, a voxel grid can be used to store any quantity in discretized cells in $\mathbb{R}^3$, but here we focus on how it is used to represent 3D geometry in common computer vision applications.

2.1.1 Voxel Grids

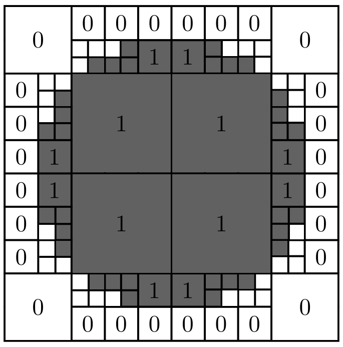
Definition Voxel grids extend the pixel-based image representation to the 3D world. Similar to how RGB color values are stored in a grid structure to represent an image, voxel-based representations store binary or scalar values in a three-dimensional regular grid to represent 3D geometry. More specifically, we define a generic (scalar) voxel grid $\mathcal{V}$ as the set of $H \times W \times D$ one-dimensional scalar values:

$$\mathcal{V} = \left\{ s_{ijk} \in \mathbb{R} \,|\, i = 1, \dots, H \wedge j = 1, \dots, W \wedge k = 1, \dots, D \right\} \tag{2.1}$$

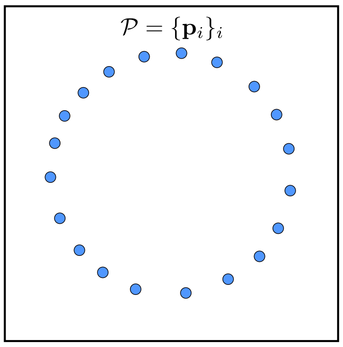
where $H, W, D \in \mathbb{N}^+$ are positive natural numbers. In many cases, H, W, and D are assumed to be equal. The positions of the voxel cells are spaced equal-distantly across 3D space in all



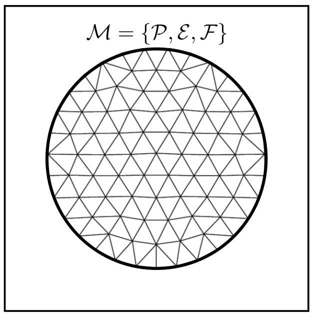
***(a)** Binary Voxel Grid*



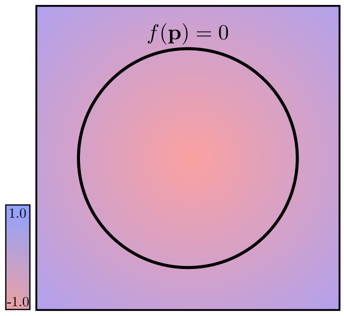
***(b)** Binary Octree*



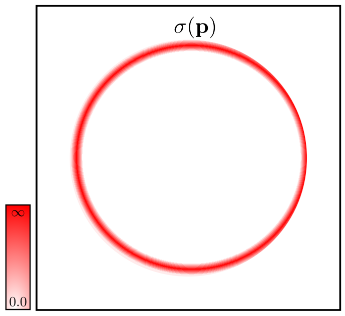
***(c)** Point Cloud*



***(d)** Mesh*



***(e)** Signed Distance Field*

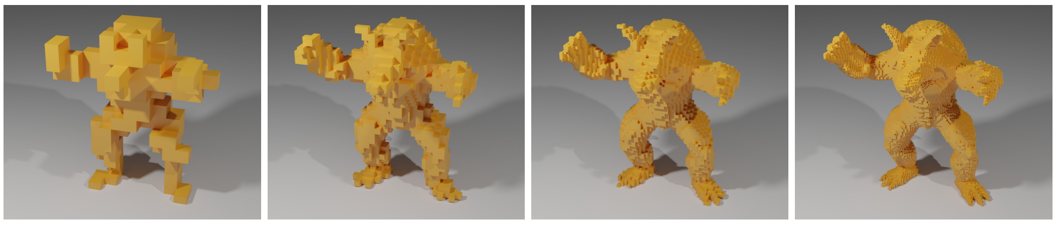


***(f)** Volume Density Field*

Figure 2.1: Common 3D Representations in Computer Vision. *In contrast to image data, there is no canonical 3D representation adopted in the computer vision community. As a result, different 3D representations are commonly used ranging from voxels (2.1a), to octrees (2.1b), to point clouds (2.1c), to meshes (2.1d), to signed distance fields (2.1e), and to volume density fields (2.1f), all representing a cricle in the examples above.*

three directions. As each scalar value s_{ijk} is associated with one voxel cell, their positions do not need to be explicitly stored. From Equation 2.1 we can see that the memory consumption grows cubically with the height, width, and depth dimensions. The most common choices for the type of stored values s_{ijk} are binary occupancy values and signed distances. Binary occupancy grids store a binary value per voxel cell, indicating whether it is empty (zero) or occupied (one). Grids of signed distances store a floating-point number indicating the distance of the cell center to the nearest surface of the 3D geometry together with a sign indicating whether the cell is inside (negative sign) or outside (positive sign) of the 3D shape.

Discussion A key property of voxel grids is that values, e.g., binary occupancies or signed distances, are stored in a regular grid. As a result, space is partitioned into equally-sized cells such that a similar capacity is assigned to each part of space. However, when considering natural 3D objects and shapes, many consist of areas of low frequency with simple structures and areas of high frequency with fine details (see Figure 2.2). Hence, different parts of the 3D object would require different capacities but this cannot be modeled with a regular voxel



(a) 32^3 *(b)* 64^3 *(c)* 128^3 *(d)* 256^3

Figure 2.2: Voxels as 3D Representation. *In a binary voxel-based presentation, 3D space is discretized into a regular grid of voxel cells, and each cell is marked either as occupied or empty. Voxel-based representations require high grid resolutions to be able to represent fine shape details. Above we show path-traced renderings of voxelizations at different grid resolutions for the Armadillo shape from the Standford 3D Scanning Repository [53].*

grid.

2.1.2 Octrees

Definition Inspired by the observation that capacity should be spent adaptively instead of regularly in 3D space, octrees [175] equip regular voxel grids with a mechanism to spend different capacity at different parts of space. It partitions 3D space by recursively subdividing cells into 8 octants. The key idea is that the degree of subdividing can be different for different areas of 3D space. For example, a fine surface area of a 3D shape requires multiple subdivisions, while empty space can be represented by a large empty cell. More specifically, an octree is a tree structure in which each node has either no or exactly 8 children. When representing 3D shapes, its generation process can be described as follows:

1. Initialize the root node.
2. Mark current leave nodes as “partially occupied”, “occupied”, or “empty”.
3. Subdivide leave nodes that are marked as partially occupied into 8 cells.
4. Repeat 2. and 3. until all nodes are marked either “occupied” or “empty”.

See Figure 2.3 for an illustration of this process. In practice, a maximum number of repetitions $d_{\text{max}} \in \mathbb{N}^+$ is set which defines the maximum depth of the tree. We can see that the memory requirements for storing an octree are directly correlated with d_{max} and the complexity of the represented shape itself. This is different from voxels where the memory requirements only depend on the grid resolution. The comparison of Figure 2.1a and Figure 2.1b shows that, in this example, the octree representation stores a similar number of values while representing the shape (a circle) more accurately.

Discussion Although octrees allow for spending different capacities in different areas of 3D space, values are still stored in grids. As a result, the resolution of represented geometry is always bounded by the chosen grid resolution and the maximum depth d_{max} of the octree.

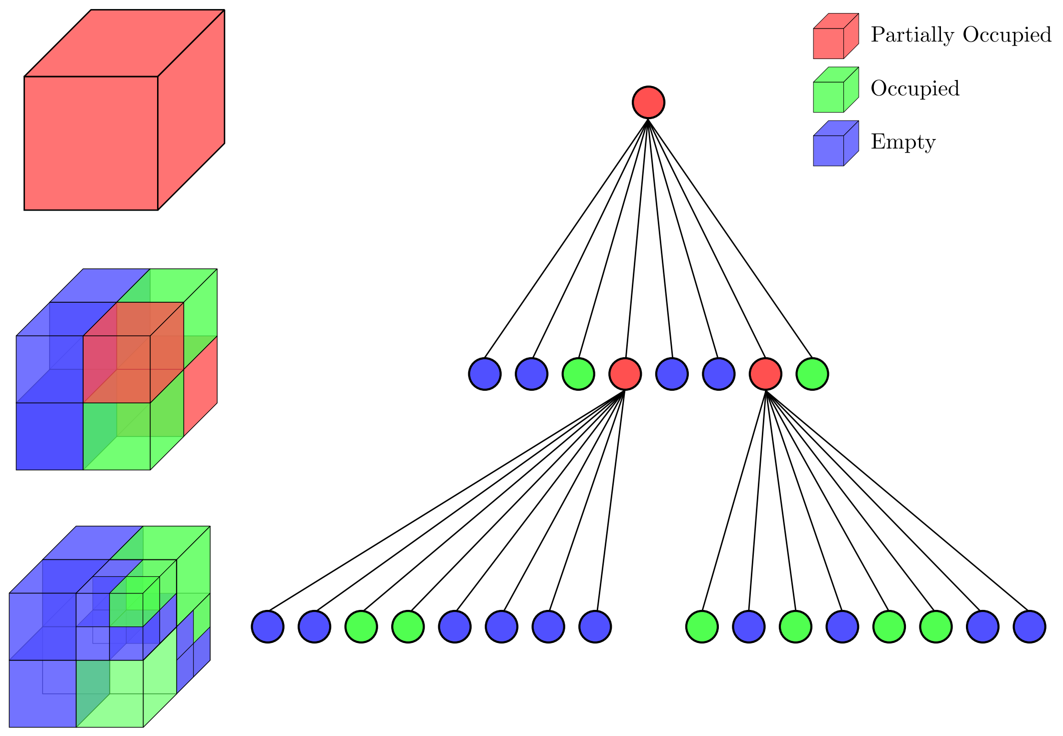


Figure 2.3: ***Illustration of an Octree.*** *An octree in 3D space is initialized with a root node corresponding to a voxel cell. Next, it is subdivided into 8 cells and each cell is marked as "partially occupied" (red), "occupied" (green), or "empty" (blue). In a subsequent step, each cell marked as partially occupied is again subdivided into 8 cells. These marking and subdividing steps are repeated until all cells are either marked as occupied or empty, or until a fixed number of maximum steps $d_{max} \in \mathbb{N}^+$ is reached, where d corresponds to the maximum depth of the tree.*

2.1.3 Point Clouds

Definition Another form of discretization is to represent 3D geometry by a finite set of 3D surface points. More specifically, we define a point cloud $\mathcal{P}$ as

$$\mathcal{P} = \left\{ \mathbf{p}_i \in \mathbb{R}^3 \,|\, i = 1, \dots, N_P \right\} \tag{2.2}$$

where the number of surface points $N_P \in \mathbb{N}^+$ is a positive natural number. The points are assumed to lie on the represented surface and, depending on the application, the point cloud is assumed to approximately cover the entire object.

Discussion The coordinates of the surface points are not required to be in a predefined cell structure and hence capacity can be spent dynamically by design. However, point clouds also discretize the 3D geometry: The number of surface points needs to be finite. Further, as point clouds represent 3D geometry in the form of point sets, they do not store any information about their relationship or how they are connected. This can be problematic, e.g., for graphics applications where surface information is required for calculating accurate light bounces and reflections. Further, trivially rendering a point cloud to the 2D image plane would result in holes and missing regions in the rendering.

2.1.4 Polygon Meshes

Definition A polygon mesh extends the point cloud surface representation with connectivity information. More specifically, a triangle mesh $\mathcal{M}$ is defined as a set of vertices, edges, and faces that describe a 3D object

$$\mathcal{M} = \{\mathcal{P}, \mathcal{E}, \mathcal{F}\} \tag{2.3}$$

where $\mathcal{P}$ is a set of surface points as defined in Equation 2.2 and

$$\mathcal{E} \subset \left\{(i,j) \,|\, \mathbf{p}_i, \mathbf{p}_j \in \mathcal{P} \wedge j \neq k \right\} \tag{2.4}$$

$$\mathcal{F} \subset \left\{(i,j,k) \,|\, \mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k \in \mathcal{E} \wedge i \neq j \wedge j \neq k\right\} \tag{2.5}$$

where we assume a triangle mesh for simplicity. The definition can naturally be extended to polygon meshes by allowing for faces spanned over more than three edges.

Discussion Mesh-based representations are widely used across a variety of computer vision and graphics applications. However, they still require 3D space to be discretized into a finite number of vertices, edges, and faces. Further, as we will see later in Chapter 5, meshes are hard to be expressed by neural networks such that most vision applications require template meshes that are deformed to represent a target shape, leading to topology restrictions and reduced generalizability.

2.1.5 Implicit Surfaces

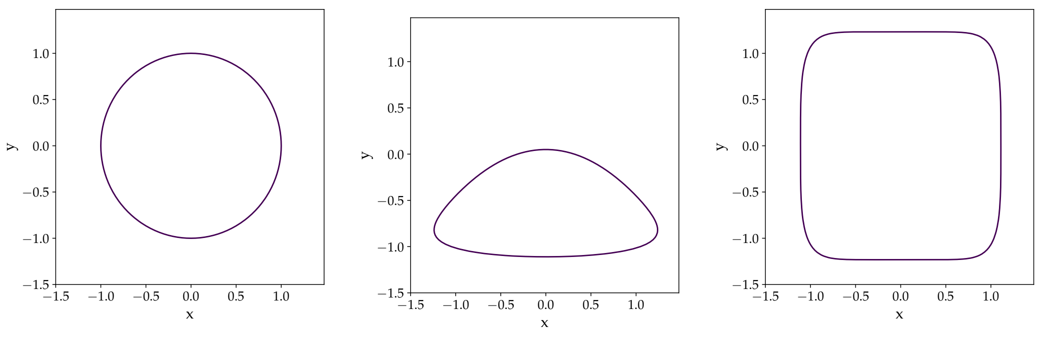
Definition The representations discussed so far have in common that they discretize the 3D shape in some way. In contrast, *implicit surfaces* do not require a discretization step (see Figure 2.4). More specifically, let

$$f : \mathbb{R}^3 \to \mathbb{R} \tag{2.6}$$

be a differentiable function mapping every point in $\mathbb{R}^3$ to a scalar value. We define the implicit surface $\mathcal{S}_f$ as

$$\mathcal{S}_f = \left\{\mathbf{p} \in \mathbb{R}^3 \,|\, f(\mathbf{p}) = \tau\right\} \quad \text{where } \tau \in \mathbb{R} \tag{2.7}$$

Common choices for τ are 0, e.g., when f represents signed distances, or 0.5 when f represents occupancy probabilities. In these cases, the function f is often referred to as a *signed distance field* or an *occupancy field*. As implicit surfaces are described by differentiable functions, their derivatives often provide additional information about the 3D shape. For



(a) $\left\{(x,y) \in \mathbb{R}^2 \,|\, \sqrt{x^2+y^2}-1=0\right\}$ ***(b)*** $\{(x,y) \in \mathbb{R}^2 \,|\, x^2+2y+y^8=0.1\}$ ***(c)*** $\left\{(x,y) \in \mathbb{R}^2 \,|\, e^{-(x^8+y^4)}=0.5\right\}$

Figure 2.4: Examples of Implicit Surfaces. *We show 2D examples of surfaces defined by implicit functions. The first function in Figure 2.4a is a signed distance function and the last function in Figure 2.4c is an occupancy probability function.*

example, the *surface normal* $\mathbf{n}(\mathbf{p}) \in \mathbb{R}^3$, i.e., a unit vector perpenticular to the surface at point $\mathbf{p} \in \mathbb{R}^3$, is given by the normalized gradient when f is a signed distance field

$$\mathbf{n}(\mathbf{p}) = \frac{\nabla f(\mathbf{p})}{||\nabla f(\mathbf{p})||_2} \tag{2.8}$$

In the case of an occupancy field, the normal is given by the negative of the normalized gradient. Further, the chain rule can be used to differentiate implicit functions via implicit differentiation. For example, let's assume our goal is to differentiate the implicit function $f(\mathbf{p})$ given by an equation $g(\mathbf{p}, f(\mathbf{p}))$. Then, we first calculate the *total derivative* of g

$$\frac{dg}{d\mathbf{p}} = \frac{\partial g}{\partial \mathbf{p}} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial \mathbf{p}} = 0 \tag{2.9}$$

where $^{dg}/_{d\mathbf{p}}$ indicates the total and $^{\partial g}/_{\partial \mathbf{p}}$ the partial derivative. We can then rearrange the equation and obtain

$$\frac{\partial f}{\partial \mathbf{p}} = -\frac{\partial g}{\partial \mathbf{p}} \cdot \left(\frac{\partial g}{\partial f}\right)^{-1} \tag{2.10}$$

As we can see, we obtain the derivative of f by using implicit differentiation without needing to find an explicit expression for f. Hence, this technique can be used when it is not possible or very hard to obtain an explicit formula for f. We will see in Chapter 6 that this idea allows us to derive gradient update rules we need for the optimization of neural networks that represent implicit surfaces when only 2D images are used as supervision.

Discussion A key difference between implicit surfaces and the previous representations is that the surface is given implicitly. If an application requires an explicit representation, e.g., in the form of a surface point cloud or a mesh, additional processing, e.g., in the form of marching cubes, needs to be performed. Further, all representations including implicit surfaces assume the existence of solid surfaces, and effects like transparency cannot be modeled. As a result, they can only represent scenes consisting of non-transparent objects and are not able to model phenomena like clouds, fog, flames, or dust.

2.1.6 Volume Density Fields

Definition In contrast to previous representations, volume density fields do not assume that surfaces are perfectly solid and allow for modeling transparent or translucent objects as well. More specifically, in a volume density field, a density $\sigma(\mathbf{p}) \in [0,\infty)$ is assigned to each point in 3D space $\mathbf{p} \in \mathbb{R}^3$:

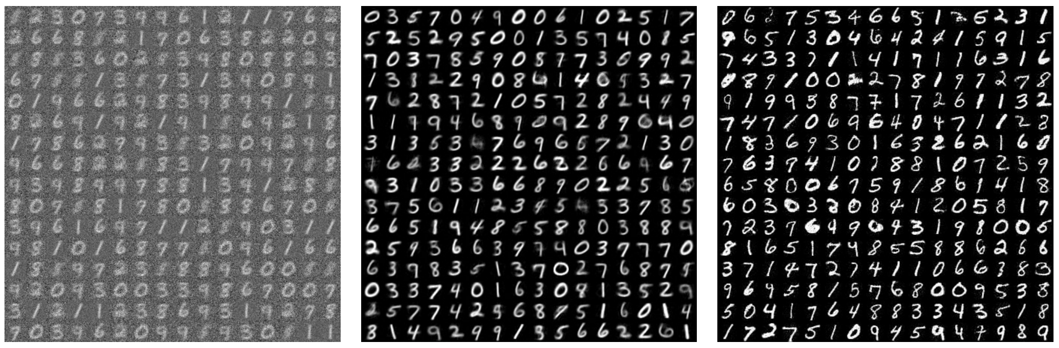
$$\sigma : \mathbb{R}^3 \to [0,\infty), \quad \mathbf{p} \mapsto \sigma(\mathbf{p}) \tag{2.11}$$

This volume density formulation originates from volume scatting in physics [38]. Since its introduction in the computer graphics literature [20, 117, 147], it is usually discussed in the context of volume rendering, i.e., how these volume density-based representations can be rendered to the image plane. The volume density $\sigma(\mathbf{p})$ can be understood as the differential probability of a ray terminating at location $\mathbf{p} \in \mathbb{R}^3$ [183].

Discussion Volume density fields require additional processing in case an explicit shape representation is needed, similar to implicit surfaces. Further, while the density-based representation allows for modeling transparent and translucent effects, it might also introduce more ambiguity when approximating 3D representations from sparse observations.

2.2 Deep Generative Modeling

Representing 3D shapes and inferring them from input data is one of the capabilities we require our model to have. However, purely discriminative models that predict one output or label for an input signal are not our end goal. The tasks our model should solve, e.g., inferring a 3D scene representation from sparse data, are ill-posed, and there exist more valid solutions than only one. Further, the ability to generate new data points would enable powerful applications such as data generation pipelines for other supervised models or automatic content creation for graphics software. This capability of not only discriminating between different data points but also generating unseen instances is also closer to human behaviour [143]. Therefore, generative models are a promising model class for our desired system, in particular deep generative models as they allow, in contrast to more classical statistical models, for drawing realistic samples from high-dimensional data spaces (see Figure 2.5). In the following, we first introduce basic concepts of deep generative modeling and then review two of the most popular approaches.



(a) *GMM (30sec)* ***(b)*** *VAE (3min)* ***(c)*** *GAN (15min)*

Figure 2.5: Samples from a GMM, VAE, and GAN Model. *While classical statistical models like Gaussian Mixture Models (GMMs) have rather limited data generation capabilities in high dimensional spaces, deep generative models like Variational Autoencoders (VAEs) [129] or Generative Adversarial Networks (GANs) [85] allow for drawing realistic samples after optimization. Above we show MNIST [142] samples generated by the respective model with rough optimization times in brackets (implemented in JAX [26]). For the GMM, we use 10 mixture components, and for the VAE and GAN, we use a 10-dimensional latent space and fully-connected neural networks with 5 hidden layers of dimension 512 and ReLU [191] activation.*

2.2.1 Introduction

We define a (deep) generative model as a function

$$G_\theta : \mathcal{Z} \to \mathcal{X} \tag{2.12}$$

parameterized by a neural network that maps network parameters $\theta \in \mathbb{R}^n$ and a often lower-dimensional latent vector $\mathbf{z} \in \mathcal{Z}$ to a data point $\mathbf{x} \in \mathcal{X}$ in some space $\mathcal{X}$ (see Figure 2.6). In computer vision applications, $\mathcal{X}$ is usually a high-dimensional space like the space of 2D images of a specific resolution or the space of 3D shapes. Given a generative model G_θ, we can sample a latent code $\mathbf{z} \sim p_0$ from a prior distribution p_0 that is defined on $\mathcal{Z}$ and obtain a new sample $G_\theta(\mathbf{z}) \in \mathcal{X}$ in the output space. This procedure allows us to define the generator distribution p_θ on $\mathcal{X}$

$$G_\theta(\mathbf{z}) = \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}) \tag{2.13}$$

Our goal when training a generative model is to optimize for optimal network parameters $\theta^\star$ such that

$$p_{\theta^\star} \approx p_{\mathcal{D}} \tag{2.14}$$

where $p_{\mathcal{D}}$ is the data distribution. In computer vision applications, $p_{\mathcal{D}}$ is usually given by a set of samples $\mathcal{D}$ (e.g., a dataset of 2D images or 3D shapes) that are assumed to be i.i.d.

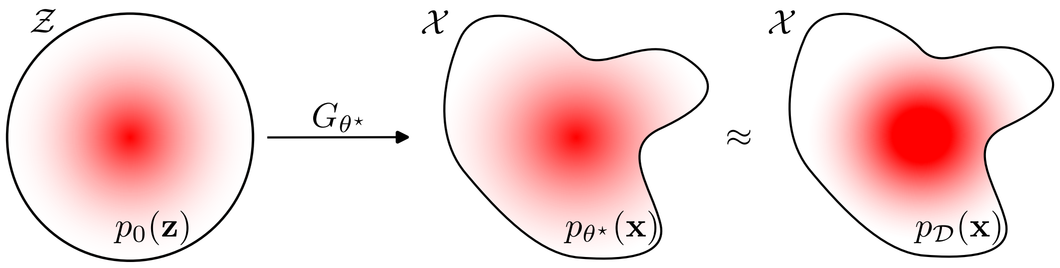


Figure 2.6: Deep Generative Modeling. *In deep generative modeling, a generator model G_θ parameterized by a neural network with parameters θ is optimized to map samples from a latent space $\mathcal{Z}$ equipped with a (simple) prior $p_0(\mathbf{z})$ to the data space $\mathcal{X}$. The goal is find parameters $\theta^\star$ such that the distribution $p_{\theta^\star}$ introduced by $G_{\theta^\star}$ approximiates the data distribution $p_{\mathcal{D}}$. In most cases, the data distribution is only given approximately in the form of a finite set of drawn samples (the dataset) that are assumed to be i.i.d.*

2.2.2 Variational Autoencoders

In a Variational Autoencoder (VAE) [129], the input space is linked to the latent space via a stochastic encoder

$$E_\phi(\mathbf{x}) = \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}) \tag{2.15}$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ is the posterior distribution modeled with a second neural network with parameters ϕ. As directly calculating the marginal likelihood p_θ is intractable, the objective that is maximized in a VAE via stochastic gradient descent is the evidence lower bound (ELBO):

$$\mathrm{ELBO}(\theta,\phi,\mathbf{x}) = \mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\,||\,p_0(\mathbf{z}))\right] \leq \log p_\theta(\mathbf{x}) \tag{2.16}$$

Intuitively, the first part of the ELBO $\log p_\theta(\mathbf{x}|\mathbf{z})$ measures the reconstruction quality on the autoencoded sample $G_\theta(E_\phi(\mathbf{x}))$ and the KL-divergence term measures the closeness of the posterior q_ϕ to the prior distribution p_0.

For training a VAE model, the parametric forms of p_0, p_θ, and q_ϕ have to be specified. In most cases, they are assumed to be Gaussians:

$$p_0 = \mathcal{N}(\mathbf{0}, I) \tag{2.17}$$

$$E_\phi(\mathbf{x}) \sim \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \mathrm{diag}(\sigma_\phi(\mathbf{x}))) \tag{2.18}$$

$$G_\theta(\mathbf{z}) \sim \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}), I) \tag{2.19}$$

where $\mathbf{0}$ indicates the zero vector, I the identity matrix, and μ_ϕ, σ_ϕ mean and standard deviation predictions from E_ϕ. The expectation in Equation 2.16 is usually approximated via a single sample due to computational contraints [129], while higher order Monte Carlo sampling likely leads to better approxmiation [30]. The assumptions allow for simplifying the VAE objective to

$$\mathbb{E}_{x\sim p_{\mathcal{D}}}\left[-\mathcal{L}_{\text{rec}}(\theta,\phi,\mathbf{x})-\mathcal{L}_{\text{KL}}(\phi,\mathbf{x})\right] \tag{2.20}$$

or, when minimizing the objective instead of maximizing it, to

$$\mathcal{L}_{\text{ELBO}}(\theta,\phi)=\mathbb{E}_{x\sim p_{\mathcal{D}}}\left[\mathcal{L}_{\text{rec}}(\theta,\phi,\mathbf{x})+\mathcal{L}_{\text{KL}}(\phi,\mathbf{x})\right] \tag{2.21}$$

with

$$\mathcal{L}_{\text{rec}}(\theta,\phi,\mathbf{x})=\left|\left|G_\theta(E_\phi(\mathbf{x}))-\mathbf{x}\right|\right|_2^2 \tag{2.22}$$

$$\mathcal{L}_{\text{KL}}(\phi,\mathbf{x})=\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})\,||\,p_0(\mathbf{z}))=\frac{1}{2}\left(||\mu_\phi(\mathbf{x})||_2^2+d+\sum_{i=1}^{d}\sigma_\phi(\mathbf{x})_i-\log\sigma_\phi(\mathbf{x})_i\right) \tag{2.23}$$

where d indicates the dimension of the latent space $\mathcal{Z}=\mathbb{R}^d$. Calculating $\mathcal{L}_{\text{rec}}$ requires stochastic sampling which is non-differentiable. In practice, the reparameterization trick [129] is used to obtain gradients such that the VAE objective can be minimized using stochastic gradient descent.

Discussion While the VAE framework possesses desirable properties like stable training [285] or outlier robustness [54], designing the distributions p_θ and q_ϕ as Gaussians is often described as a key limitation [30, 55, 131]. Further, empirically it is observed that generated samples of a vanilla VAE tend to be unrealistic and blurry [62]. In the following, we discuss an alternative model class that does not make these Gaussian assumptions by instead modeling the generator distribution implicitly.

2.2.3 Generative Adversarial Networks

In Generative Adversarial Networks (GANs) [85], the generator distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is not modeled explicitly, e.g., by assuming the parametric form of a Gaussian, but implicitly instead. More specifically, a *discriminator* is defined

$$D_\phi:\mathcal{X}\to[0,1]\subset\mathbb{R} \tag{2.24}$$

which is parameterized as a neural network with network parameters ϕ. The discriminator D_ϕ maps a sample from the input space $\mathcal{X}$ to a probability indicating whether the sample is real or fake. By real we mean that the sample is drawn from the data distribution, and by fake, we mean that the generator G_θ synthesized the sample. Both the generator and the discriminator networks are optimized in a zero-sum game. The objective is defined as

$$\min_{G_\theta}\max_{D_\phi}V(G_\theta,D_\phi)=\mathbb{E}_{x\sim p_{\mathcal{D}}}[\log D_\phi(x)]+\mathbb{E}_{z\sim p_0(z)}[\log(1-D_\phi(G_\theta(z)))] \tag{2.25}$$

and optimized via simultaneous or alternating stochastic gradient descent/ascent.

Discussion The training dynamics of GANs are complex and global convergence criteria are an open research problem [180]. However, it has been shown that regularization techniques such as gradient penalties [178] enable local convergence under a number and assumptions and they are widely adopted in practice. While GANs, even when regularized, can suffer from training instabilities, mode collapse, or poor mode coverage, they are shown to achieve unprecedented results in terms of image fidelity in 2D as well as 3D-aware generative modeling [37, 93, 122, 123, 251].

3 Related Work

After introducing important background concepts, we review related work in this chapter. We discuss important steps of the development of the fields of 3D reconstruction in Section 3.1 and generative modeling in Section 3.2. This enables us to provide context and highlight important prior works before we introduce and investigate our proposed systems in the following chapters.

3.1 3D Reconstruction

3.1.1 Origins and Early 3D Reconstruction

Origins Computer vision originated as building the visual perception component of a system that could mimic human intelligence and that could be used for robotic applications requiring intelligent behavior [271]. In contrast to the related field of digital image processing, recovering the underlying 3D structure of a scene from input images has been a core problem from the start of computer vision [97, 312, 313]. Initially, recovering the 3D structure was seen as a simple problem that could then in turn be used to solve more complex problems like higher-level reasoning or planning [21]. From today's perspective, we can say that the problem is significantly more complex than initially suggested, and to this day, researchers work on improving 3D reconstruction methods. We refer the interested reader to Chapter 1.2 of [271] for a detailed computer vision history overview.

Early 3D Reconstruction Inspired by human biology, correspondence estimation techniques for stereo image pairs are among the first works to estimate the 3D structure of scenes. For example, in their pioneering work [171], Marr and Poggio investigate the computational structure of the problem of finding correspondences between a stereo image pair and estimating disparities. At the same time, the structure-from-motion (SfM) problem, i.e., the recovery of the 3D structure of the scene while simultaneously estimating the camera motion of the input images, gained more attention. For example, Ullman [289] proposes to divide images into groups and test rigid transformation hypotheses to decompose the scene into rigid objects in motion. Most of these early works are heavily inspired by principles of stereopsis [310], i.e., the perception of 3D structure through binocular vision, and related findings from psychology such as the kinematic depth effect [299]. Next to correspondences between image pairs, other cues such as texture [222], focus [192], or shading [18] have been investigated for recovering the 3D geometry. These approaches are also called *Shape-from-X*, indicating 3D shapes can also be recovered from other cues next to correspondences.

3.1.2 Multi-View Stereo

In *multi-view stereo (MVS) algorithms*, the idea of using correspondences as the main cue for 3D geometry is extended to using more than two images [73] (see Figure 3.1). For example, in their pioneering work [257], Seitz and Dyer propose a voxel coloring approach in which the 3D scene space is discretized into voxel cells and a color is assigned to each cell considering the reprojections into the input images and occlusions. In subsequent works, reconstruction systems are developed which can be applied to unordered image collections [252, 263] and complex urban scenes [225]. This allowed researchers to develop methods that can scale to large image sets in the order of thousands [3] and millions [71, 100, 253, 319] of images. In [254], Schönberger et al. revisits the steps of the structure-from-motion pipeline and proposes a robust, multi-purpose 3D reconstruction system together with the implementation named COLMAP that is commonly used to this day.

Learning-based MVS Systems With the advent of large-scale (labeled) datasets [56, 79] and the availability of more computing power, more learning-based 3D reconstruction systems are developed, especially since the breakthrough of AlexNet [138] being the first deep neural network-based approach to win the ImageNet [56] recognition challenge. While first works focus on using, in particular, Convolutional Neural Networks (CNNs) to extract more robust features for tasks like stereo matching [146, 165, 173, 343], later methods propose to learn the entire MVS pipeline end-to-end from data [108, 331, 332]. In contrast to traditional methods, learning-based systems are shown to better handle poorly-textured and reflective surfaces where photometric consistency is not given. While achieving compelling results for multi-view image input, these approaches still rely on a form of feature matching across views or a fusion process to obtain the final 3D shape.

3.1.3 Learning-based 3D Reconstruction

This assumption of being able to match features breaks down for large baselines [163] or sparse inputs with the most extreme case of a single input view. As a consequence, another line of work directly learns the mapping from the input image(s) to the output 3D representation without adhering to the traditional MVS pipeline. While these methods achieve promising results in certain scenarios, their inner mechanisms tend to be harder to be analyzed and dissected [276]. In the following, we categorize and discuss these approaches wrt. the output representation they use.

Voxel Voxel-based systems were among the first methods proposed for discriminative [172, 229, 264] and generative [47, 83, 239, 269, 315, 316] learning-based 3D reconstruction. First works use 2D convolutional encoder together with 3D convolutional decoder models operating on voxel grids [47, 286, 315]. For example, 3D-R2N2 [47] uses a 2D convolutional encoder and a 3D convolutional decoder to predict 32^3 voxel grids from single or multi-view images, where multi-view information is fused with an LSTM model [107]. Due to memory requirements, however, these approaches are limited to relatively small voxel grids. Follow-up works [317, 318, 347] apply 3D convolutional neural networks to resolutions

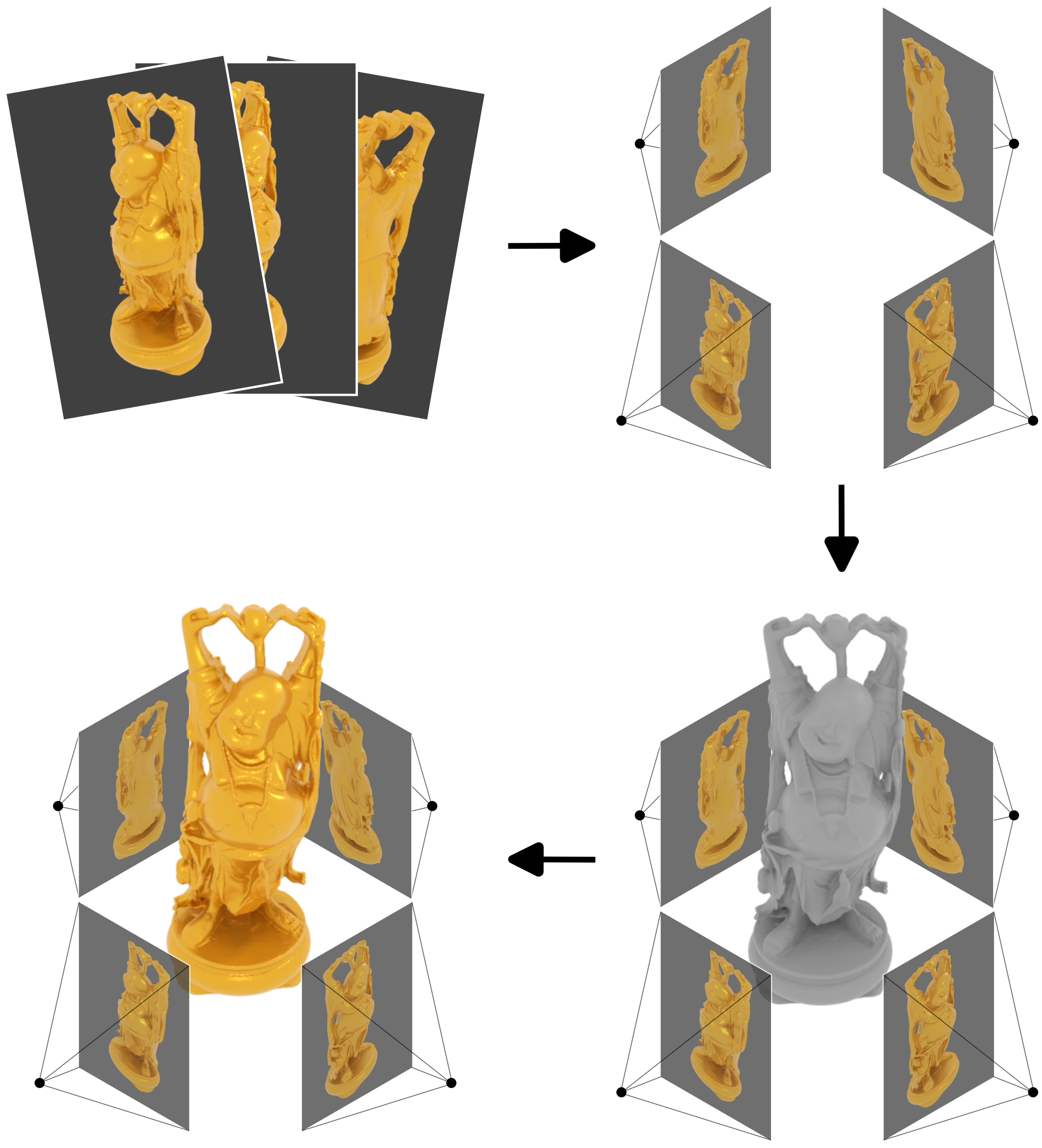


Figure 3.1: Multi-View Stereo Pipeline. *In the traditional multi-view stereo pipeline, the input images (top left) are first used to estimate camera poses (top right). Next, 3D geometry is predicted for the posed input images (bottom right) which is then optionally textured (bottom left) in the last step. The renderings are created with Blender [51] and the Buddha shape from the Standford 3D Scanning Repository [53], and the figure design is inspired by Fig. 1.2 from [73].*

up to 128^3, but only made possible with shallow architectures and smaller batch sizes, which in turn leads to slow training. Another recent line of works that overcomes the memory requirements for high-resolution voxel grids proposes to reconstruct 3D objects in a multi-resolution fashion [96, 275]. However, the resulting methods are often complicated to implement and require multiple passes over the input to generate the final 3D model. Furthermore, they are still limited to comparably small 256^3 voxel grids, such that most of the recent works focus on combining voxels with other representations like implicit functions [187, 258].

Point Representations Point cloud-based systems are widely used both in robotics and computer graphics. In their seminal work PointNet [230, 231], Qi et al. propose a simple model architecture that uses point clouds as a representation for discriminative deep learning tasks. They achieve permutation invariance by applying a fully connected neural network to each point independently followed by a global pooling operation. Fan et al. [69] introduced point clouds as an output representation for 3D reconstruction, and subsequent works propose convolutional-based [90, 148, 280] or transformer-based [338, 340] architectural improvements. However, unlike the other representations, point cloud-based systems require additional non-trivial post-processing steps [16, 32, 126, 127] to generate the final 3D mesh and hence are less-frequently used for the task for learning-based 3D reconstruction.

Mesh Representations Meshes have first been considered for discriminative 3D classification or segmentation tasks by applying convolutions on the graph spanned by the mesh's vertices and edges [28, 94, 302]. More recently, meshes are also used as output representation for 3D reconstruction [92, 118, 136, 303]. As directly learning a mesh-based representation with deep neural networks is hard and prone to degenerate solutions, most works rely on a simple template mesh that is deformed. For example, in Pixel2Mesh [303], features are extracted from an input image to gradually deform the vertices of an initial ellipsoidal mesh. Other works assume to have access to a reference template of the same object class [118, 136, 234]. AtlasNet [92] is in contrast not restricted by the genus of the template mesh as many 2D patches are merged to obtain the final output mesh. However, this approach has been shown to suffer from self-intersections and non-watertight surfaces [179]. As a result, many recent works propose combinations of mesh-based and other representations such as neural fields [110, 329].

Neural Field To mitigate the above-mentioned problems, neural field representations were proposed [41, 179, 212] and quickly gained popularity [322]. By describing 3D geometry implicitly, e.g., as the decision boundary of a binary classifier [41, 179], they do not discretize space and have a fixed memory footprint. While first works use a single, global latent code to represent a 3D shape, subsequent works improve reconstruction quality by combining local and global information [46, 247, 306]. Later approaches [35, 219, 311] scale from single-object to room- or house-level reconstructions by using distributed latent codes. The contributions made in this thesis include neural field-based approaches for 3D and 4D reconstruction.

3.2 Generative Modeling

In the context of machine learning and computer vision, generative modeling is often introduced and discussed as a form of unsupervised learning (UL) in distinction to supervised learning (SL). While in SL, the goal is to approximate a probability distribution $p(\mathbf{y}|\mathbf{x})$ over predefined labels $\mathbf{y} \in \mathcal{Y}$ for data points $\mathbf{x} \in \mathcal{X}$ from some input space $\mathcal{X}$, in UL, an unconditional density model $p(\mathbf{x})$ is learned only from unlabeled data $\mathbf{x} \in \mathcal{X}$. While early deep learning breakthroughs were dominated by SL, UL is often considered "far more important in the longer term" [143] mostly due to the vast abundance of unlabeled data.

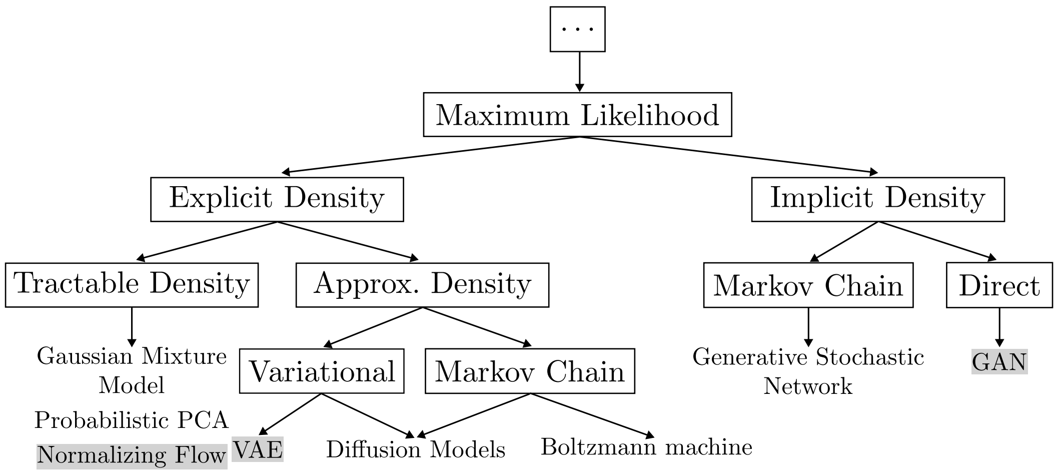


Figure 3.2: Taxonomy of Generative Models. *We show a taxonomy of generative models that either explicitly or implicitly perform maximum likelihood following [86]. The model types that are marked in grey color are discussed in greater detail in respective subsections.*

Origins and Taxonomy of Generative Models Among the first generative models that are used in computer vision are statistical models that fit (a small amount of) parameters of predefined distributions to observed data. For example, in Gaussian mixture models (GMMs), the parameters θ to define the likelihood p_θ are the mixture components and means and scales of isotropic Gaussians and they can be optimized using the Expectation-Maximization (EM) algorithm [17]. A similar example from statistical analysis for a continuous latent variable model is Probabilistic Principal Component Analysis (PPCA) [244, 282] whose parameters can be derived using EM or Singular Value Decomposition. GMMs and PPCA are examples of generative models that maximize the likelihood of the data by explicitly modeling a tractable density. This is just one of many classes of generative models that are used across the fields of machine learning and computer vision. In Figure 3.2 we show a taxonomy of popular generative model types by classifying them wrt. how they approximate the likelihood.[1] In the following, we review relevant related works for selected types of

[1]Not that we restrict our discussion to models that explicitly or implicitly perform maximum likelihood. While other model types are possible [86], this selection comprises the vast majority of approaches investigated in

generative models and discuss their relation to neural fields. For an in-depth study of the latter, we refer the interested reader to [260, 320].

3.2.1 Normalizing Flows

Similar to GMMs or PPCA, normalizing flow-based models also define a tractable density explicitly. The key idea is to connect the input space $\mathcal{X}$ with a latent space $\mathcal{Z}$ via a nonlinear, invertible mapping $G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ with network parameters θ and then make use of the change of variable formula:

$$p_\theta(\mathbf{x}) = p_0\left(G_\theta^{-1}(\mathbf{x})\right) \left| \det \left(\frac{G_\theta^{-1}(\mathbf{x})}{\mathbf{x}} \right) \right| \tag{3.1}$$

Intuitively, the normalizing flow G_θ warps its input space $\mathcal{Z}$ with a simple prior p_0 to a more complex distribution p_θ on $\mathcal{X}$ [135]. Theoretically, it is proven that any distribution on $\mathcal{X}$ can be modeled if G_θ can be arbitrarily complex [22, 297]. However, designing and parameterizing arbitrarily complex functions G_θ that are invertible and whose determinant of the Jacobian is easy to compute is challenging and has been the core area of research in the last years.

Development Among the first works that were popularized in the machine learning community is NICE [58] which uses affine coupling layers parameterized by MLPs with ReLU [191] activation. RealNVP [59] and Glow [132] are later works that use affine coupling layers in combination with more advanced masking techniques and parameterizations such as convolutional layers. Since then, many different types of flows have been explored with popular approaches being autoregressive flows like MAF [209], residual flows like iResNet [12], and ODE-based flows like FFJORD [88]. For a more in-depth discussion, we refer the reader to [135, 210].

Relation to Neural Fields Although normalizing flow models do not achieve similar image fidelity compared to state-of-the-art VAEs or GANs, their explicit and tractable density definition has several benefits, especially for other downstream applications. In the context of neural fields, they have recently been used to improve view synthesis quality for sparse inputs via an additional data term [201] or to model uncertainty [259].

3.2.2 Variational Autoencoders

Another class of generative models uses an explicit but intractable density, avoiding strict model design restrictions. Among the most popular instances of this class are Variational Autoencoders (VAEs) that optimize a lower bound to the likelihood. See Section 2.2 of our background chapter for details on how VAEs are optimized.

the computer vision literature.

Development Since its introduction [129], VAEs have been applied to a wide variety of tasks, including semi-supervised classification [167], image inpainting [220], graph modeling [133], chemical design [84], dark energy modeling in astronomy [235], and more, mainly due to its simple design and easy optimization. However, in contrast to other generative model classes, empirically it was found that generated images tend to be unrealistic and blurry which has been attributed to the MSE reconstruction loss [62], but also the limited approximation of the true posterior [346]. To overcome this limitation, incorporating adversarial losses [168], ELBO improvements [30] and reweightings [106], different regularizations [285], as well as more complex, hierarchical priors [293] are proposed. Regularized autoencoders instead use deterministic autoencoders with regularization on the latent space, and more complex priors can subsequently be fitted [82]. VQ-VAEs [207] use a discrete latent space with gradient approximation, which has been scaled to larger image resolutions via hierarchies of latent codes [236] as well as adversarial training [68].

Relation to Neural Fields Due to their simplicity, VAEs were among the first generative models used in the context of neural fields. They are used to develop generative models of neural fields representing 3D shapes [179], 3D texture [203], 4D shapes in motion [197], as well as radiance field-based scene representations [137].

3.2.3 Generative Adversarial Networks

In contrast to previously discussed approaches, Generative Adversarial Networks (GANs) do not require an explicit definition of the density. Instead, they only model it implicitly through the ability to sample from it. In contrast to generative stochastic networks [14], samples are efficiently computed via a single forward pass similar to VAEs. For details on how GANs are optimized, we refer the reader to Section 2.2 of our background chapter.

Development Soon after its introduction in [85], conditional variants of GANs such as cGAN [184], approaches to structure the latent space such as InfoGAN [39], as well as convolutional models like DCGAN [232] were proposed which significantly improved performance. The conditional generative models pix2pix [111] and CycleGAN [352] propose a UNet [243] GAN architecture and remove the need for paired training data, respectively, achieving impressive results and enabling a wide range of applications. Progressively growing the model during training [121] as well as employing larger models [29] enabled scaling to higher image resolution and fidelity for unconditional models. By improving model architectures and carefully analyzing training dynamics, StyleGAN [122] and follow-up works [123, 124, 251] achieve unprecedented photorealistic image quality at megapixel resolutions and set the state-of-the-art in several benchmarks.

Relation to Neural Fields First works use 3D supervision and combine trained neural field decoders with adversarial training in the latent space [43] or train GANs directly with processed 3D shapes [134]. With the arrival of more powerful differentiable rendering techniques for neural fields, GRAF [255] is the first GAN with a radiance field-based 3D representation that is trained from unposed 2D images only. In contrast to 2D-based GANs, *3D-aware* models like GRAF synthesize images by rendering a generated 3D scene allowing for explicit control over the camera viewpoint at test time. While a series of follow-up works like π-GAN [36] achieve higher image fidelity with improved training strategies and parameterizations, in GIRAFFE [200], which is one of the core contributions of this thesis, a compositional 3D scene representation and a combination of volume and neural rendering is proposed. Several subsequent works like StyleNeRF [93] or EG3D [37] adopt this combination of neural and volume rendering and investigate more sophisticated model architectures, bridging the gap to state-of-the-art 2D GANs in terms of image fidelity.

4 Neural Fields

After introducing important background concepts and reviewing related works, we now discuss a key concept of the scene representations we use: *neural fields*. We first introduce the concept of fields in Section 4.1 and then discuss the parameterization of neural fields in Section 4.2. Subsequently, we discuss different rendering techniques for neural fields in Section 4.3 before we study their use for various applications in the following chapters of the thesis.

4.1 Introduction

Name	Field Unit	Input Space	Output Space
2D RGB Image	RGB intensity	$[0,1]^2 \subset \mathbb{R}^2$	$[0,1]^3 \subset \mathbb{R}^3$
2D Semantic Map	class probabilities	$[0,1]^2 \subset \mathbb{R}^2$	$[0,1]^k \subset \mathbb{R}^k$
3D Signed Distance Field	signed distance	$\mathbb{R}^3$	$\mathbb{R}$
3D Occupancy Field	occupancy probability	$\mathbb{R}^3$	$[0,1] \subset \mathbb{R}$
3D Vector Field	motion vector	$\mathbb{R}^3$	$\mathbb{R}^3$
Time-Varying 3D Vector Field	time-varying motion vector	$\mathbb{R}^4$	$\mathbb{R}^3$
3D Light Field	RGB intensity	$\mathbb{R}^5$	$[0,1]^2 \subset \mathbb{R}^3$

Table 4.1: Examples of Fields in Computer Vision. *We show examples of fields that are commonly used in the computer vision literature [322]. We observe that many quantities which are of interest for computer vision applications can be expressed as fields.*

Following physics [70, 174] and recent computer vision conventions [322], we define a field as a varying physical quantity of spatial and temporal coordinates. More specifically, a field is a differentiable function

$$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m \tag{4.1}$$

where n and m are positive natural numbers. For example, a signed distance function in 3D space (see Section 2.1.5) is a *scalar field* with $n = 3$ and $m = 1$. In Table 4.1 we show examples of fields commonly used in computer vision. We observe that many quantities of interest, ranging from 2D semantic maps to 3D light fields, can be expressed as fields.

4.2 Parameterization

In classical physics, fields are often assumed to possess an underlying parametric form. The coefficients can then be derived from experiments. For example, according to Newton's law,

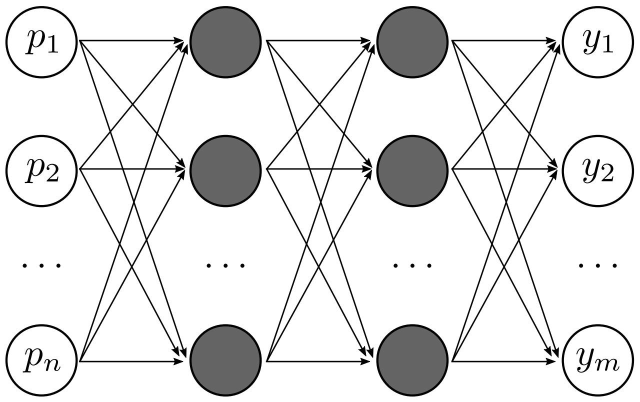


Figure 4.1: Illustration of a Neural Field. *A neural field is a multi-layer perceptron (MLP) $\mathbf{f}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ with network parameters $\theta \in \mathbb{R}^k$ that parameterizes the field $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. The neural field maps input coordinates $\mathbf{p} \in \mathbb{R}^n$ to an output quantity $\mathbf{y} \in \mathbb{R}^m$. By optimizing the network parameters θ using stochastic gradient descent, the neural field can approximate any field from data. In the illustration above, input and output layers are shown with white circles and hidden layers with dark grey circles.*

the gravitational field $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^3$ for a single particle of mass M has the form

$$\mathbf{f}(\mathbf{p}) = \frac{-GM\hat{\mathbf{p}}}{|\mathbf{p}|^2} \tag{4.2}$$

where G is a constant and $\hat{\mathbf{p}}$ is the unit vector in radial direction from the particle to $\mathbf{p}$.

In contrast, in computer vision, the field of interest is usually not assumed to have an analytic form. Instead, learning-based systems are deployed, and the fields are approximated from observations. But how should fields be parameterized such that they can be learned from data? In this thesis, we parameterize the fields of interest as *multi-layer perceptrons (MLPs)*. MLPs consist of a series of linear, fully-connected layers with non-linear activation functions, e.g., ReLU [191]. The universal approximation theorem [128] shows that MLPs can approximate any differentiable function. By optimizing the network parameters using stochastic gradient descent, the field of interest can be approximated from data.

We follow [322] and define a *neural field* as a field that is parameterized fully or in part by an MLP:

$$\mathbf{f}_\theta : \mathbb{R}^n \to \mathbb{R}^m \tag{4.3}$$

where $\theta \in \mathbb{R}^k$ are the network parameters and k the number of parameters (see Figure 4.1). MLPs have several properties which make them a promising candidate for parameterizing fields:

- Simplicity and universality: MLPs are simple conceptually and implementation-wise, and no assumption about the data modality or the input/output space needs to be made.
- Differentiability: MLPs are continuous and differentiable by construction.

- Adaptiveness: MLPs are adaptive by construction [322].
- Spectral Bias: MLPs tend to learn lower frequencies first before higher frequencies are fitted [274] due to the spectral bias [8, 233].

Adaptiveness Arguably the most common alternative to MLPs for parameterizing fields is to discretize the input space and store the quantity of interest in a regular grid. For example, 3D space can be discretized into 3D voxels and signed distances can be stored per cell. While this representation is fast to query, it requires large memory such that grid and cell sizes need to be carefully designed. Further, the same capacity is spent everywhere. In contrast, the complexity of MLPs scales with their number of network parameters. As a result, capacity can be spent adaptively where needed, and the mechanism of how to choose where to spend capacity does not need to be predefined. Further, the memory requirements are defined by the number of network parameters, not the dimension of the input space. In contrast, voxel grids are only feasible for low dimensional input spaces as the required memory grows exponentially with the dimensionality of the input space (also called “ the curse of discretization” [180]). In summary, MLPs are adaptive by design and scale independently of the field they represent.

Spectral Bias In nature, most phenomena that are perceivable to humans are continuous and smooth, e.g., temperature, motion, and gravitation. When constructing scene representations of our environments, these principles should be incorporated into the system. Indeed, MLPs have been shown to possess a spectral or frequency bias. Over the progress of optimization, they tend to first learn low frequencies before high frequencies are fitted [8, 233]. This property makes them more robust to input noise and it is hypothesized to be one reason for good generalization capabilities despite overparameterization [233]. However, depending on the application, this bias can also be a drawback, e.g., overly smooth predictions for the task of novel-view synthesis [183]. In practice, MLPs can be adapted to fit high frequencies by converting the input coordinates to Fourier features (also called “positional encoding”) before passing them to the MLP [183, 274, 350].

Discussion Parameterizing fields with MLPs also has drawbacks compared to more traditional parameterizations like voxel grids: (i) They are slower to optimize and query compared to values stored in regular grids, (ii) they cannot directly be used in standard graphics software applications compared to other 3D representations like meshes, (iii) they are harder to dissect as they represent 3D information in the weights of a neural network. To overcome some of the limitations, hybrid methods are proposed for various tasks, e.g., feature grids in combination with smaller MLPs for improved 3D reconstruction [35, 46, 115, 219] or faster view synthesis [77, 99, 337]. In summary, we believe that there is not a single parameterization that performs best in every regard, and in many cases (in particular, if efficiency is a priority), hybrid solutions potentially lead to the best results.

4.3 Rendering

4.3.1 Overview

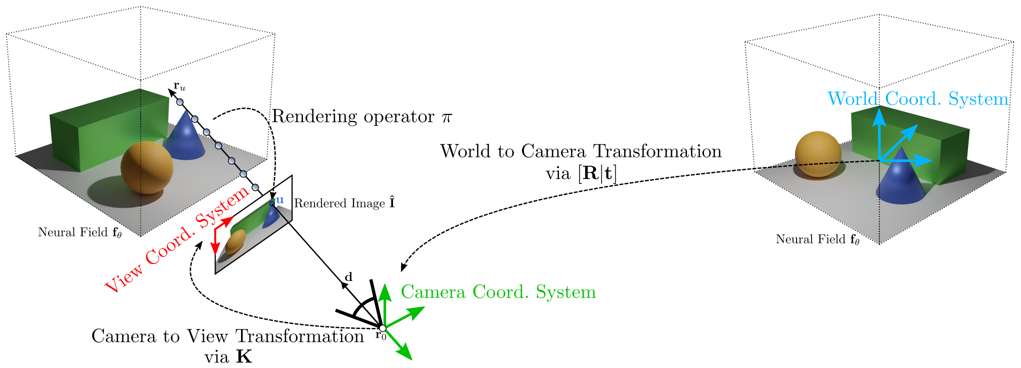

$\mathbf{r}_u$
Rendering operator π
World to Camera Transformation via $[\mathbf{R}|\mathbf{t}]$
World Coord. System
u
Rendered Image $\hat{\mathbf{I}}$
Neural Field $\mathbf{f}_\theta$
View Coord. System
Neural Field $\mathbf{f}_\theta$
d
Camera Coord. System
Camera to View Transformation via $\mathbf{K}$
$\mathbf{r}_0$

Figure 4.2: Rendering of Neural Fields. *A key concept for learning neural field-based 3D representations from 2D imagery data is the rendering process, i.e., how the neural field is rendered to the image plane. To render an image* $\hat{\mathbf{I}}$ *from the neural field* $\mathbf{f}_\theta$*, we need camera extrinsics* $\xi = [\mathbf{R}|\mathbf{t}]$ *and intrinsics* $\mathbf{K}$*, where the first describes the world (blue) to camera (green) coordinate transformation and the latter the camera (green) to view (red) coordinate transformation. The rendering operator* π *maps the ray* $\mathbf{r}_u$ *that is cast from the camera origin* $\mathbf{r}_0$ *through pixel* $\mathbf{u}$ *on the image plane to the RGB color value* $\hat{\mathbf{I}}_u$ *of that pixel* $\mathbf{u}$*.*

Our goal is to infer scene representations from sensory image data. While direct supervision, e.g., in the form of pairs of input images and the corresponding 3D geometry when performing 3D reconstruction, is acceptable for constraint settings and proofs-of-concept, our end goal is to supervise only with 2D imagery data to increase generalization and scalability. A key concept for achieving this is the rendering process of our 3D scene representations to the image plane (see Figure 4.2).

More specifically, let $\mathbf{K} \in \mathbb{R}^{3\times 3}$ be the camera intrinsics, $\xi = [\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3\times 4}$ the camera extrinsics or camera pose, $\mathbf{f}_\theta$ the neural field representing the scene, and $\mathbf{u}$ the pixel we want to render. A pinhole camera model is commonly assumed such that the intrinsic matrix $\mathbf{K}$ describes the focal length, skew parameters, and the principal point of the camera. The extrinsics define the camera rotation $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ in the scene. We cast a ray $\mathbf{r}_u$ from the camera center $\mathbf{r}_0$ through pixel $\mathbf{u}$ on the image plane with normalized direction $\mathbf{d}$ into the scene. For given $\mathbf{K}$, ξ, and $\mathbf{f}_\theta$, a rendering operator π maps ray $\mathbf{r}_u$ to the RGB color $\hat{\mathbf{I}}_u$ of pixel $\mathbf{u}$

$$\pi : \mathbf{r}_u \mapsto \hat{\mathbf{I}}_u \in [0,1]^3 \tag{4.4}$$

The predicted RGB image is obtained by repeating this for all pixels on the image plane:

$$\{\pi(\mathbf{r}_{u_i})\}_{i=1}^{H\times W} = \hat{\mathbf{I}} \in [0,1]^{H\times W\times 3} \tag{4.5}$$

where H and W indicate the height and width of the output RGB image. Note that while we discuss the rendering process only for RGB color values, other quantities of interest, e.g., surface normals, depth, or semantic classes, can be rendered similarly.

4.3.2 Rendering Techniques

In the following, we discuss three of the most important rendering operators that are used for neural fields: the ray marching operator π^{rm}, the sphere tracing operator π^{st}, and the volume rendering operator π^{vol}. The first two are surface rendering techniques where we assume that our neural field represents solid surfaces. Common choices for representing solid 3D shapes with neural fields are occupancy and signed distance fields (see Section 2.1). In contrast, for volume rendering, surfaces are not assumed to be solid and it hence can be used to render neural fields that use density-based representations. Note that in this chapter, we provide a concise overview and focus on the forward pass of techniques for rendering neural fields, not discussing their differentiability or how they can be implemented in learning-based systems. Later in Chapter 6, we will introduce and analyze our proposed differentiable ray marching algorithm for neural fields in greater detail.

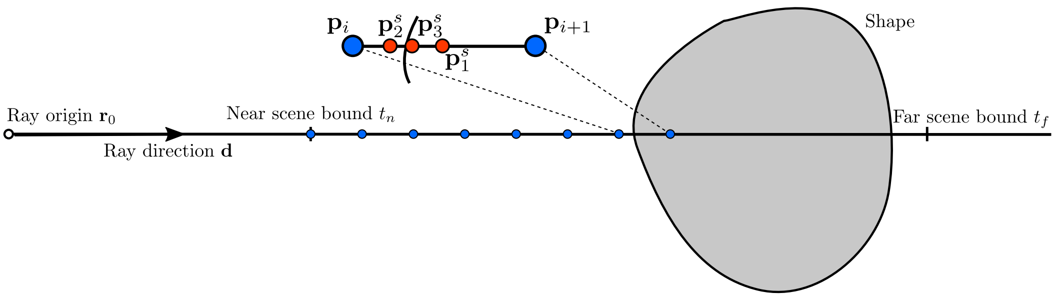


Figure 4.3: Illustration of Ray Marching. *Ray marching is a common surface rendering technique for neural fields where the ray is first sampled equidistantly between near and far scene bounds, and the neural field is evaluated at these sample locations. Once the first interval $[\mathbf{p}_i, \mathbf{p}_{i+1}]$ is found that contains the surface, the surface point prediction can be further refined using the secant method. We illustrate three secant steps $\{\mathbf{p}_i^s\}_i$ in red, where the final point $\mathbf{p}_3^s$ is the predicted surface point.*

Ray Marching For the ray marching operator π^{rm}, the ray $\mathbf{r}_u$ is sampled equidistantly within predefined near and far scene bounds $t_n, t_f \in \mathbb{R}^+$, and the neural field is evaluated at these sampling locations (see Figure 4.3). More specifically, the sampling locations are defined as

$$\left\{ \mathbf{p}_i = \mathbf{r}_0 + \mathbf{d} \left(t_n + \frac{i-1}{N_s}(t_f - t_n) \right) \mid i = 1, \dots, N_s \right\} \tag{4.6}$$

where $N_s \in \mathbb{N}^+$ is the number of samples along the ray. Once an interval $[\mathbf{p}_i, \mathbf{p}_{i+1}]$ is found in which the surface lies, the evaluation along the ray is stopped. If a surface is found, its prediction can be further refined, e.g., by applying the bisection or secant method to the found interval (see the red sample points $\mathbf{p}_i^s$ in Figure 4.3). Note that this optimization

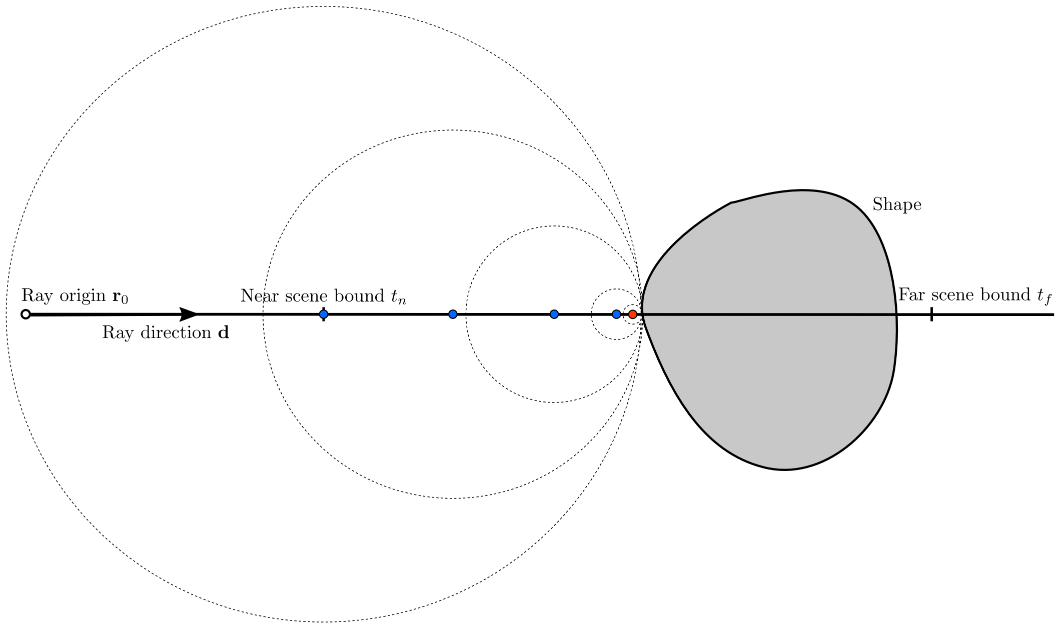


Figure 4.4: Illustration of Sphere Tracing. *Neural fields that represent signed distances can be rendered via sphere tracing. The signed distance $\hat{s}$ is evaluated for the first point along the ray that lies within the scene bounds. Next, a step of size $\gamma \cdot \hat{s}$ is taken along the ray where the hyperparameter $\gamma \in (0,1]$ accounts for prediction inaccuracies. The algorithm is converged and the surface point, here shown in red, is found once $|\hat{s}| < \varepsilon$ where $\varepsilon \in \mathbb{R}^+$ indicates the convergence threshold.*

problem is one-dimensional as the surface point is constrained to lie on the ray. Ray marching may not converge and no surface point will be found if the surface area is thinner than the sample distance. If a surface point is found, it can then be used to evaluate a quantity of interest, e.g., predicted RGB color **c** or surface normal **n**, at that 3D location given by the neural field. This evaluation is then used as the 2D rendering of the quantity of interest for the respective pixel **u**.

Sphere Tracing If the neural field predicts signed distances, this additional distance information can be used to speed up the rendering process. For the sphere tracing operator π^{st}, the first point along the ray which lies within the predefined scene bounds is evaluated and the predicted signed distance $\hat{s} \in \mathbb{R}$ is stored. Next, a step is taken along the ray to obtain the next point $\mathbf{p}_i$ for which the signed distance needs to be evaluated:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + (\gamma \cdot \hat{s}) \cdot \mathbf{d} \tag{4.7}$$

where $\gamma \cdot \hat{s}$ is the step size and $\gamma \in (0,1]$ is a hyperparameter that can be used to account for potential prediction inaccuracies. This sequence of evaluating a ray point and taking a step along the ray is repeated until $|\hat{s}| \leq \varepsilon$ where $\varepsilon \in \mathbb{R}^+$ is a small positive value indicating the convergence threshold. Note that in case of no ray-surface intersection or prediction

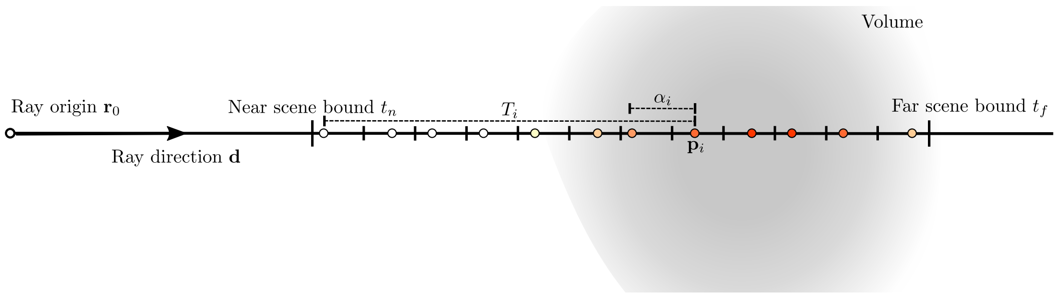


Figure 4.5: Illustration of Volume Rendering. *Volume rendering can be used to render density-based 3D representations. In classic volume rendering, a ray is cast through the scene, points are sampled within near and far scene bounds along the ray, and the neural field is evaluated at these sampling locations. These evaluations of predicted density together with estimated values of a quantity of interest, e.g., RGB color values, can be used to obtain the pixel's final rendering via a form of alpha composition with $T_i\alpha_i$ as the weights (see text for details).*

inaccuracies, the algorithm may not converge which in practice is handled by defining a maximum number of evaluations. Figure 4.4 shows an illustration of the algorithm. Similar to ray marching, the estimated surface point is then used to evaluate the quantity of interest at that 3D location which serves as the 2D rendering of the quantity of interest for the respective pixel.

Volume Rendering Volume rendering is a technique that can be used to render neural fields that represent densities instead of solid surfaces. We adhere to the volume rendering technique for neural fields proposed in [183] that builds on classical physics-inspired volume rendering [117]. For π^{vol}, the ray is divided into equally-sized bins, and a uniform sample t_i is drawn from each bin

$$t_i \sim \mathcal{U}\left(t_n + \frac{i-1}{N_s}(t_f - t_n), t_n + \frac{i}{N_s}(t_f - t_n)\right) \tag{4.8}$$

where $\mathcal{U}(\cdot,\cdot)$ indicates the uniform distribution and $N_s \in \mathbb{N}^+$ the number of samples along the ray. Figure 4.5 shows an illustration of the process. Next, we evaluate the neural field at the resulting sampling locations $\mathbf{p}_i$ and define

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \qquad \alpha_i = 1 - \exp(-\sigma_i \delta_i) \tag{4.9}$$

where σ_i is the predicted density at sample point $\mathbf{p}_i$ and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. T_i and α_i are often referred to as transmittance and alpha values, respectively. We obtain the final 2D rendering of RGB color $\hat{\mathbf{I}}_{\mathbf{u}}$ for pixel $\mathbf{u}$ as

$$\hat{\mathbf{I}}_{\mathbf{u}} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i \tag{4.10}$$

where $\mathbf{c}_i$ is the predicted RGB color in 3D space. Note that other quantities of interest like predicted normals can be similarly rendered to the image plane. Equation 4.10 illustrates that this volume rendering technique can be seen as a form of alpha composition with weights $T_i \alpha_i$.

Dicussion A key difference between the discussed volume rendering technique and the surface rendering algorithms is that the rendered quantity of interest is a sum of all evaluations along the ray (see Equation 4.10). In contrast, if ray marching or sphere tracing is performed, a single surface point is found and the quantity of interest is evaluated at that 3D location to obtain the rendered pixel value. For learning-based systems, both types of approaches have benefits and drawbacks. While surface rendering can be implemented memory-efficiently where only the evaluation at the surface point needs to be stored (see Chapter 6), it does not provide gradient information to all ray sampling locations. In contrast, volume rendering requires storing all ray evaluations for calculating the backward pass but provides a gradient signal along the entire ray. Empirically it is shown that surface rendering approaches lead to better surfaces but are prone to get stuck in local minima. Volume rendering-based systems achieve better view synthesis results and possess more stable optimization that is more robust to local minima. A recent line of works [205, 307, 334] aims at combining the two approaches, leading to improved results (see also the limitation and future work discussion in Section 6.4.1).

5 3D Reconstruction with Neural Fields

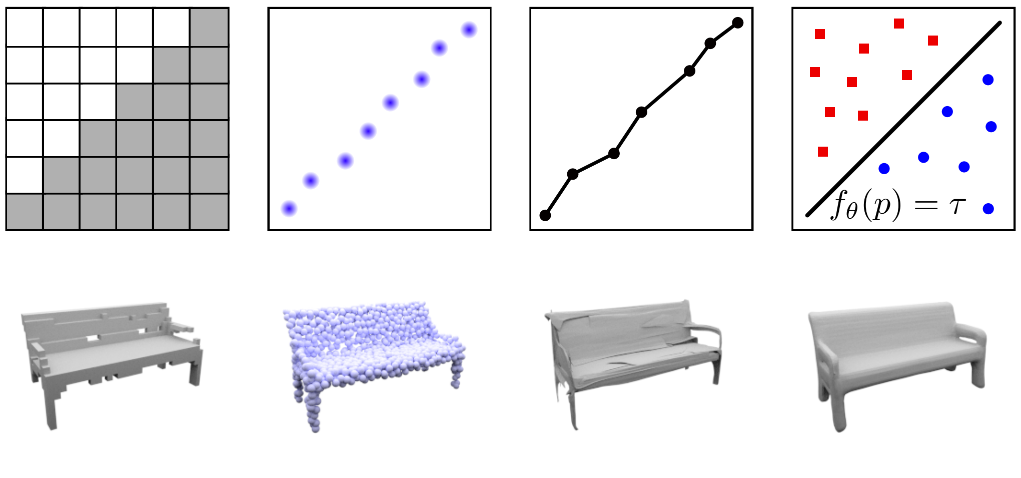
5.1 Introduction

In this chapter, we study if neural fields can be used as 3D representation in learning-based systems. More specifically, we analyze how well neural fields are suited for representing and recovering 3D geometry in discriminative as well as generative 3D reconstruction tasks. This chapter lays the foundation for developing more complex representations and learning algorithms which will be the goal of subsequent chapters of the thesis.

Prior Work Learning-based approaches for 3D reconstruction have recently gained popularity [27, 47, 83, 239, 315, 316]. In contrast to traditional multi-view stereo algorithms, learning-based models are able to encode rich prior information about the space of 3D shapes which helps to resolve ambiguities in the input. At the same time, generative models have recently achieved remarkable successes in generating realistic high-resolution images [121, 178, 304]. This success has not yet been replicated in the 3D domain as, in contrast to the 2D domain, the community has not yet agreed on a 3D output representation that is both, memory efficient and that can be efficiently inferred from data.

Existing representations in prior works can be broadly categorized into three categories: voxel-based representations [27, 74, 150, 239, 269, 292, 316] , point-based representations [2, 69], and mesh representations [119, 234, 303] (see Figure 5.1). Voxel representations are a straightforward generalization of pixels to the 3D case. Unfortunately, however, the memory footprint of voxel representations grows cubically with the resolution, hence limiting naïve implementations to 32^3 or 64^3 voxels. While it is possible to reduce the memory footprint by using data-adaptive representations such as octrees [241, 275], this approach leads to complex implementations and existing data-adaptive algorithms are still limited to relatively small 256^3 voxel grids. Point clouds [2, 69] and meshes [119, 234, 303] have been introduced as alternative representations for deep learning, using appropriate loss functions. However, point clouds lack the connectivity structure of the underlying mesh and hence require additional post-processing steps to extract 3D geometry from the model. Existing mesh representations are typically based on deforming a template mesh and hence do not allow arbitrary topologies. Moreover, both approaches are limited in the number of points/vertices which can be reliably predicted using a standard feed-forward network.

Contribution In this chapter, we propose a novel approach to 3D reconstruction based on directly learning the *continuous* 3D occupancy function (Figure 5.1d). Instead of predicting a voxelized representation at a fixed resolution, we predict the complete occupancy function with a neural network f_θ which can be evaluated at *arbitrary* resolution. This drastically



(a) *Voxel* *(b)* *Point* *(c)* *Mesh* *(d)* *Ours*

Figure 5.1: 3D Reconstruction with Neural Fields. *Existing 3D representations discretize the output space differently: (a) spatially in voxel representations, (b) in terms of predicted points, and (c) in terms of vertices for mesh representations. In contrast, (d) we propose to consider the continuous decision boundary of a classifier f_θ (e.g., a deep neural network) as a 3D surface that allows extracting 3D meshes at any resolution.*

reduces the memory footprint during training. At inference time, we extract the mesh from the learned model using a simple multi-resolution isosurface extraction algorithm which trivially parallelizes over 3D locations. We experimentally validate that our approach is able to generate high-quality meshes and demonstrate that it compares favorably to the state-of-the-art.

5.2 Method

In this section, we first introduce *Occupancy Networks* as a representation of 3D geometry. We then describe how we can learn a model that infers this representation from various forms of input such as point clouds, single images, and low-resolution voxel representations. Lastly, we describe a technique for extracting high-quality 3D meshes from our model at test time.

5.2.1 Occupancy Networks

Ideally, we would like to reason about the occupancy not only at fixed discrete 3D locations (as in voxel representations) but at *every* possible 3D point $\mathbf{p} \in \mathbb{R}^3$. We call the resulting function

$$o : \mathbb{R}^3 \to \{0, 1\} \tag{5.1}$$

the *occupancy function* of the 3D object. Our key insight is that we can approximate this 3D function with a neural network that assigns to every location $\mathbf{p} \in \mathbb{R}^3$ an occupancy probability between 0 and 1. Note that this network is equivalent to a neural network for binary classification, except that we are interested in the decision boundary which implicitly represents the object's surface.

Input Conditioning When using such a network for 3D reconstruction of an object based on observations of that object (e.g., image, point cloud, etc.), we must condition it on the input. Fortunately, we can make use of the following simple functional equivalence: a function that takes an observation $\mathbf{z} \in \mathcal{Z}$ as input and has a function from $\mathbf{p} \in \mathbb{R}^3$ to $\mathbb{R}$ as output can be equivalently described by a function that takes a pair $(\mathbf{p}, \mathbf{z}) \in \mathbb{R}^3 \times \mathcal{Z}$ as input and outputs a real number. The latter representation can be simply parameterized by a neural network f_θ that takes a pair $(\mathbf{p}, \mathbf{z})$ as input and outputs a real number that represents the probability of occupancy:

$$f_\theta : \mathbb{R}^3 \times \mathcal{Z} \to [0,1] \tag{5.2}$$

We call this network the *Occupancy Network*.

5.2.2 Training

To learn the parameters θ of the neural network $f_\theta(\mathbf{p}, \mathbf{z})$, we randomly sample points in the 3D bounding volume of the object under consideration: for the i-th sample in a training batch we sample K points $\mathbf{p}_{ij} \in \mathbb{R}^3$, $j = 1, \ldots, K$. We then evaluate the mini-batch loss $\mathcal{L}_\mathcal{B}$ at those locations:

$$\mathcal{L}_\mathcal{B}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{K} \mathcal{L}(f_\theta(\mathbf{p}_{ij}, \mathbf{z}_i), o_{ij}) \tag{5.3}$$

Here, $\mathbf{z}_i$ is the i'th observation of batch $\mathcal{B}$, $o_{ij} \equiv o(\mathbf{p}_{ij})$ denotes the true occupancy at point $\mathbf{p}_{ij}$, and $\mathcal{L}(\cdot, \cdot)$ is a cross-entropy classification loss.

Point Sampling The performance of our method depends on the sampling scheme that we employ for drawing the locations $\mathbf{p}_{ij}$ that is used for training. In Section 5.3.6 we perform a detailed ablation study comparing different sampling schemes. In practice, we found that sampling uniformly inside the bounding box of the object with additional small padding yields the best results.

Generative Modeling Our 3D representation can also be used for learning probabilistic latent variable models. Towards this goal, we introduce an encoder network $E_\phi(\cdot)$ that takes locations $\mathbf{p}_{ij}$ and occupancies o_{ij} as input and predicts mean μ_ϕ and standard deviation σ_ϕ of a Gaussian distribution $q_\phi(\mathbf{z}|(\mathbf{p}_{ij}, o_{ij})_{j=1:K})$ on latent $\mathbf{z} \in \mathbb{R}^L$ as output. We optimize a lower bound [76, 129, 238] to the negative log-likelihood of the generative model

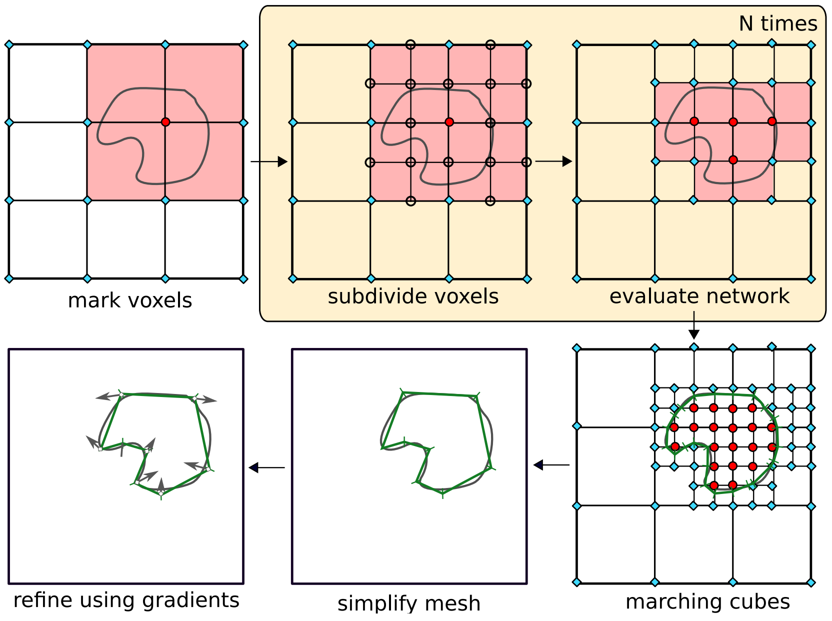


Figure 5.2: Illustration of the MISE Algorithm. *In Multiresolution IsoSurface Extraction (MISE), We first mark all points at a given resolution that have already been evaluated as either occupied (red circles) or unoccupied (cyan diamonds). We then determine all voxels that have both occupied and unoccupied corners and mark them as active (light red) and subdivide them into 8 subvoxels each. Next, we evaluate all new grid points (empty circles) that have been introduced by the subdivision. The previous two steps are repeated until the desired output resolution is reached. Finally, we extract the mesh using the marching cubes algorithm [162], simplify and refine the output mesh using first and second-order gradient information.*

$p((o_{ij})_{j=1:K}|(\mathbf{p}_{ij})_{j=1:K})$:

$$\mathcal{L}_{\mathcal{B}}^{\text{gen}}(\theta,\phi) = \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left[\sum_{j=1}^{K}\mathcal{L}(f_\theta(\mathbf{p}_{ij},\mathbf{z}_i),o_{ij}) + \text{KL}\left(q_\phi(\mathbf{z}|(\mathbf{p}_{ij},o_{ij})_{j=1:K})\,\|\,p_0(\mathbf{z})\right)\right] \tag{5.4}$$

where KL denotes the KL-divergence, $p_0(\mathbf{z})$ is a prior distribution on the latent variable $\mathbf{z}_i$ (typically Gaussian) and $\mathbf{z}_i$ is sampled according to $q_\phi(\mathbf{z}_i|(\mathbf{p}_{ij},o_{ij})_{j=1:K})$.

5.2.3 Inference

For extracting the isosurface corresponding to a new observation given a trained Occupancy Network, we introduce *Multiresolution IsoSurface Extraction (MISE)*, a hierarchical isosurface extraction algorithm (Figure 5.2). By incrementally building an octree [112, 175, 272, 314], MISE enables us to extract high-resolution meshes from the Occupancy Network without densely evaluating all points of a high-dimensional occupancy grid.

MISE We first discretize the volumetric space at an initial resolution and evaluate the Occupancy Network $f_\theta(\mathbf{p},\mathbf{z})$ for all $\mathbf{p}$ in this grid. We mark all grid points p as occupied for which $f_\theta(\mathbf{p},\mathbf{z})$ is bigger or equal to some threshold[1] τ. Next, we mark all voxels as active for which at least two adjacent grid points have differing occupancy predictions. These are the voxels that would intersect the mesh if we applied the marching cubes algorithm at the current resolution. We subdivide all active voxels into 8 subvoxels and evaluate all new grid points which are introduced to the occupancy grid through this subdivision. We repeat these steps until the desired final resolution is reached. At this final resolution, we apply the Marching Cubes algorithm [162] to extract an approximate isosurface

$$\mathcal{S}_{f_\theta} = \{\mathbf{p} \in \mathbb{R}^3 \mid f_\theta(\mathbf{p},\mathbf{z}) = \tau\} \tag{5.5}$$

Our algorithm converges to the correct mesh if the occupancy grid at the initial resolution contains points from every connected component of both the interior and the exterior of the mesh. It is hence important to take an initial resolution that is high enough to satisfy this condition. In practice, we found that an initial resolution of 32^3 was sufficient in almost all cases.

Mesh Refinement The initial mesh extracted by the Marching Cubes algorithm can be further refined. First, we simplify the mesh using the Fast-Quadric-Mesh-Simplification algorithm [78]. Next, we refine the output mesh using first and second-order (i.e., gradient) information. Towards this goal, we sample random points p_k from each face of the output mesh and minimize the loss

$$\sum_{k=1}^{K} (f_\theta(\mathbf{p}_k,\mathbf{z}) - \tau)^2 + \lambda \left\| \frac{\nabla_p f_\theta(\mathbf{p}_k,\mathbf{z})}{\|\nabla_p f_\theta(\mathbf{p}_k,\mathbf{z})\|} - \mathbf{n}(\mathbf{p}_k) \right\|^2 \tag{5.6}$$

where $\mathbf{n}(\mathbf{p}_k)$ denotes the surface normal vector of the mesh at $\mathbf{p}_k$. In practice, we set $\lambda = 0.01$. Minimization of the second term in Equation 5.6 uses second-order gradient information and can be efficiently implemented using Double-Backpropagation [64].

Note that this last step removes the discretization artifacts of the Marching Cubes approximation and would not be possible if we had directly predicted a voxel-based representation. In addition, our approach also allows for efficiently extracting normals for all vertices of our output mesh by simply backpropagating through the Occupancy Network. In total, our inference algorithm requires 3s per mesh.

5.2.4 Implementation Details

We implemented our Occupancy Network using a fully-connected neural network with 5 ResNet blocks [98] and condition it on the input using conditional batch normalization [66, 298] (see Figure 5.3). We exploit different encoder architectures depending on the type of input. For single view 3D reconstruction, we use a ResNet18 architecture [98]. For point

[1]The threshold τ is the only hyperparameter of our Occupancy Network. It determines the "thickness" of the extracted 3D surface. In our experiments, we cross-validate this threshold on a validation set.

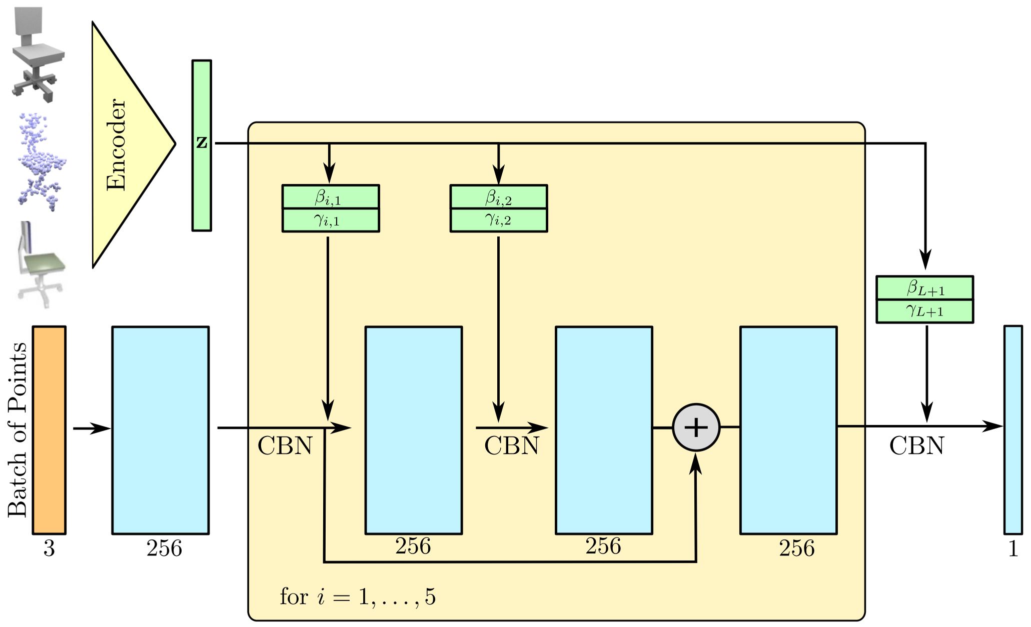


Figure 5.3: Occupancy Network Model Architecture. *We first compute an embedding* **z** *of the input. We then feed the input points through multiple fully-connected ResNet blocks. In these blocks, we use Conditional Batch-Normalization (CBN) to condition the network on* **z**. *Finally, we project the output of our network to one dimension using a fully-connected layer and apply the sigmoid function to obtain occupancy probabilities.*

clouds, we use the PointNet encoder [230]. For voxelized inputs, we use a 3D convolutional neural network [172]. For unconditional mesh generation, we use a PointNet [230] for the encoder network E_ϕ.

5.3 Experiments

We conduct three types of experiments to validate the proposed Occupancy Networks. First, we analyze the **representation power** of Occupancy Networks by examining how well the network can reconstruct complex 3D shapes from a learned latent embedding. This gives us an upper bound on the results we can achieve when conditioning our representation on additional input. Second, we **condition** our Occupancy Networks on images, noisy point clouds and low-resolution voxel representations, and compare the performance of our method to several state-of-the-art baselines. Finally, we examine the **generative** capabilities of Occupancy Networks by adding an encoder to our model and generating unconditional samples from this model.

Baselines For the single-image 3D reconstruction task, we compare our approach against several state-of-the-art baselines which leverage various 3D representations: we evaluate against 3D-R2N2 [47] as a voxel-based method, Point Set Generating Networks (PSGN) [69] as a point-based technique, and Pixel2Mesh [303] as well as AtlasNet [92] as mesh-based approaches. For point cloud inputs, we adapted 3D-R2N2 and PSGN by changing the encoder. As a mesh-based baseline, we use Deep Marching Cubes (DMC) [150] which has recently reported state-of-the-art results on this task. For the voxel super-resolution task, we assess the improvements wrt. the input.

Dataset For all of our experiments, we use the ShapeNet [34] subset of Choy et al. [47]. We also use the same voxelization, image renderings and train/test split as Choy et al. Moreover, we subdivide the training set into a training and a validation set on which we track the loss of our method and the baselines to determine when to stop training.

In order to generate watertight meshes and to determine if a point lies in the interior of a mesh (e.g., for measuring IoU) we use the code provided by Stutz et al. [269]. For a fair comparison, we sample points from the surface of the watertight mesh instead of the original model as ground truth for PSGN [69], Pixel2Mesh [303] and DMC [150]. All of our evaluations are conducted wrt. these watertight meshes.

Metrics For evaluation, we use the volumetric IoU, the Chamfer-L_1 distance and a normal consistency score. In the following, let $\mathcal{M}_{\text{pred}}$ and $\mathcal{M}_{\text{GT}}$ be the set of all points that are inside or on the predicted and ground truth mesh, respectively. The volumetric IoU is defined as the quotient of the volume of the two meshes' intersection and the volume of their union:

$$\text{IoU}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) := \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{GT}}|}{|\mathcal{M}_{\text{pred}} \cup \mathcal{M}_{\text{GT}}|}. \tag{5.7}$$

We obtain unbiased estimates of these volumes by randomly sampling 100k points from the bounding volume and determining if the points lie inside or outside $\mathcal{M}_{\text{pred}}$ and $\mathcal{M}_{\text{GT}}$, respectively.

We define the Chamfer-L_1 distance between the two meshes as

$$\begin{aligned} \text{Chamfer-}L_1(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) := & \frac{1}{2\,|\partial\mathcal{M}_{\text{pred}}|} \int_{\partial\mathcal{M}_{\text{pred}}} \min_{q \in \partial\mathcal{M}_{\text{GT}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{p} \\ & + \frac{1}{2\,|\partial\mathcal{M}_{\text{GT}}|} \int_{\partial\mathcal{M}_{\text{GT}}} \min_{p \in \partial\mathcal{M}_{\text{pred}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{q} \end{aligned} \tag{5.8}$$

where $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$ denote the surfaces of the two meshes. Moreover, we define an accuracy and completeness score of $\mathcal{M}_{\text{pred}}$ wrt. $\mathcal{M}_{\text{GT}}$:

$$\begin{aligned} \text{Accuracy}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}) &:= \frac{1}{|\partial\mathcal{M}_{\text{pred}}|} \int_{\partial\mathcal{M}_{\text{pred}}} \min_{q \in \partial\mathcal{M}_{\text{GT}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{p} \\ \text{Completeness}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}) &:= \frac{1}{|\partial\mathcal{M}_{\text{GT}}|} \int_{\partial\mathcal{M}_{\text{GT}}} \min_{p \in \partial\mathcal{M}_{\text{pred}}} \|\mathbf{p} - \mathbf{q}\|_2 \, d\mathbf{q} \end{aligned} \tag{5.9}$$

Note that this definition implies that *lower* accuracy and completeness scores are *better*.

Moreover, note that the Chamfer-L_1 distance is just the mean of the accuracy and the completeness score.

Similarly, we define the normal consistency score as

$$\begin{aligned}\text{Normal-Consistency}(\mathcal{M}_{\text{pred}},\mathcal{M}_{\text{GT}}) := &\frac{1}{2\,|\partial\mathcal{M}_{\text{pred}}|}\int_{\partial\mathcal{M}_{\text{pred}}} |\langle \mathbf{n}(\mathbf{p}), \mathbf{n}(\text{proj}_2(\mathbf{p}))\rangle|\,d\mathbf{p} \\ &+\frac{1}{2\,|\partial\mathcal{M}_{\text{GT}}|}\int_{\partial\mathcal{M}_{\text{GT}}} |\langle \mathbf{n}(\text{proj}_1(\mathbf{q})), \mathbf{n}(\mathbf{q})\rangle|\,d\mathbf{q}\end{aligned} \tag{5.10}$$

where $\langle\cdot,\cdot\rangle$ indicates the inner product, $\mathbf{n}(\mathbf{p})$ and $\mathbf{n}(\mathbf{q})$ the (unit) normal vectors on mesh surface $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$ respectively and $\text{proj}_2(\mathbf{p})$ and $\text{proj}_1(\mathbf{q})$ denote the projections of $\mathbf{p}$ and $\mathbf{q}$ onto $\partial\mathcal{M}_{\text{GT}}$ and $\partial\mathcal{M}_{\text{pred}}$ respectively. As a result, a *higher* normal consistency score is *better*.

We estimate all four quantities efficiently by sampling 100k points from the surface of both meshes and employing a KD-tree to determine the corresponding nearest neighbors from the other mesh.

5.3.1 Representation Power

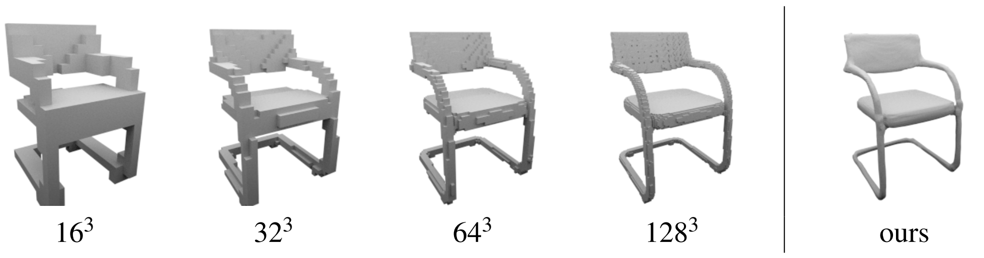


Figure 5.4: Discrete vs. Continuous. *Qualitative comparison of our continuous representation (right) to voxelizations at various resolutions (left). Note how our representation encodes details that are lost in voxel-based representations.*

In our first experiment, we investigate how well Occupancy Networks represent 3D geometry, independent of the inaccuracies of the input encoding. The question we try to answer in this experiment is whether our network can learn a memory-efficient representation of 3D shapes while at the same time preserving as many details as possible. This gives us an estimate of the representational capacity of our model and an upper bound on the performance we may expect when conditioning our model on additional input. Similarly to [275], we embed each training sample in a 512-dimensional latent space and train our neural network to reconstruct the 3D shape from this embedding.

We apply our method to the training split of the "chair" category of the ShapeNet dataset. This subset is challenging to represent as it is highly varied and many models contain high-frequency details. Since we are only interested in reconstructing the training data, we do not use separate validation and test sets for this experiment. For evaluation, we measure the volumetric IoU to the ground truth mesh.

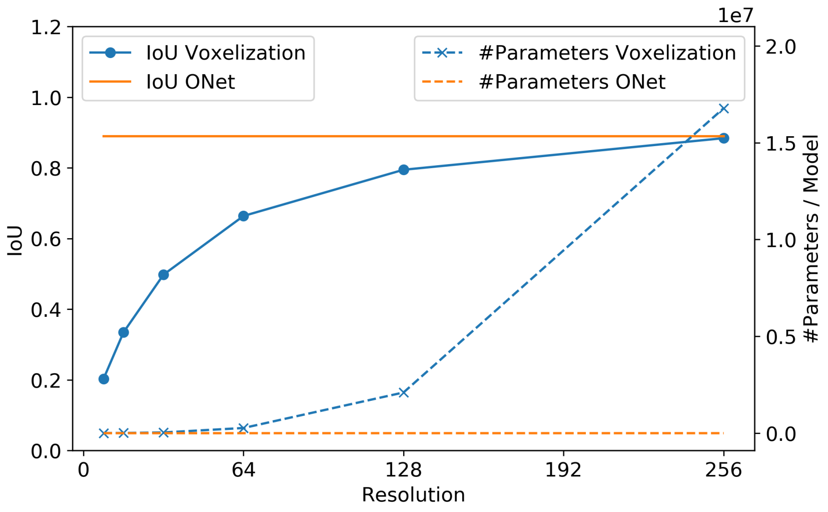


Figure 5.5: IoU vs. Resolution. *This plot shows the IoU of a voxelization to the ground truth mesh (solid blue line) in comparison to our continuous representation (solid orange line) as well as the number of parameters per model needed for the two representations (dashed lines). Note how our representation leads to larger IoU wrt. the ground truth mesh compared to a low-resolution voxel representation. At the same time, the number of parameters of a voxel representation grows cubically with the resolution, whereas the number of parameters of Occupancy Networks is independent of the resolution.*

Results Quantitative results and a comparison to voxel representations at various resolutions are shown in Figure 5.5. We see that the Occupancy Network (ONet) is able to faithfully represent the entire dataset with a high mean IoU of 0.89 while a low-resolution voxel representation is not able to represent the meshes accurately. At the same time, the Occupancy Network is able to encode all 4746 training samples with as little as 6M parameters, independently of the resolution. In contrast, the memory requirements of a voxel representation grow cubically with resolution. Qualitative results are shown in Figure 5.4. We observe that the Occupancy Network enables us to represent details of the 3D geometry which are lost in a low-resolution voxelization.

5.3.2 Single-Image 3D Reconstruction

In our second experiment, we condition the Occupancy Network on an additional view of the object from a random camera location. The goal of this experiment is to evaluate how well occupancy functions can be inferred from complex input. While we train and test our method on the ShapeNet dataset, we also present qualitative results for the KITTI [80] and the Online Products dataset [206].

ShapeNet In this experiment, we use a ResNet-18 image encoder, which was pretrained on the ImageNet dataset. For a fair comparison, we use the same image encoder for both

	IoU					Chamfer-L_1					Normal Consistency)				
category	3D-R2N2	PSGN	Pix2Mesh	AtlasNet	ONet	3D-R2N2	PSGN	Pix2Mesh	AtlasNet	ONet	3D-R2N2	PSGN	Pix2Mesh	AtlasNet	ONet
airplane	0.426	-	0.420	-	**0.571**	0.227	0.137	0.187	**0.104**	0.147	0.629	-	0.759	0.836	**0.840**
bench	0.373	-	0.323	-	**0.485**	0.194	0.181	0.201	**0.138**	0.155	0.678	-	0.732	0.779	**0.813**
cabinet	0.667	-	0.664	-	**0.733**	0.217	0.215	0.196	0.175	**0.167**	0.782	-	0.834	0.850	**0.879**
car	0.661	-	0.552	-	**0.737**	0.213	0.169	0.180	**0.141**	0.159	0.714	-	0.756	0.836	**0.852**
chair	0.439	-	0.396	-	**0.501**	0.270	0.247	0.265	**0.209**	0.228	0.663	-	0.746	0.791	**0.823**
display	0.440	-	**0.490**	-	0.471	0.314	0.284	0.239	**0.198**	0.278	0.720	-	0.830	**0.858**	0.854
lamp	0.281	-	0.323	-	**0.371**	0.778	0.314	0.308	**0.305**	0.479	0.560	-	0.666	0.694	**0.731**
loudspeaker	0.611	-	0.599	-	**0.647**	0.318	0.316	0.285	**0.245**	0.300	0.711	-	0.782	0.825	**0.832**
rifle	0.375	-	0.402	-	**0.474**	0.183	0.134	0.164	**0.115**	0.141	0.670	-	0.718	0.725	**0.766**
sofa	0.626	-	0.613	-	**0.680**	0.229	0.224	0.212	**0.177**	0.194	0.731	-	0.820	0.840	**0.863**
table	0.420	-	0.395	-	**0.506**	0.239	0.222	0.218	0.190	**0.189**	0.732	-	0.784	0.832	**0.858**
telephone	0.611	-	0.661	-	**0.720**	0.195	0.161	0.149	**0.128**	0.140	0.817	-	0.907	0.923	**0.935**
vessel	0.482	-	0.397	-	**0.530**	0.238	0.188	0.212	**0.151**	0.218	0.629	-	0.699	0.756	**0.794**
mean	0.493	-	0.480	-	**0.571**	0.278	0.215	0.216	**0.175**	0.215	0.695	-	0.772	0.811	**0.834**

Table 5.1: Single-Image 3D Reconstruction. *This table shows a numerical comparison of our approach and the baselines for single-image 3D reconstruction on the ShapeNet dataset. We measure the IoU, Chamfer-L_1 distance and Normal Consistency for various methods wrt. the ground truth mesh. Note that in contrast to prior work, we compute the IoU wrt. the high-resolution mesh and not a coarse voxel representation. All methods apart from AtlasNet [92] are evaluated on the test split by Choy et al. [47]. Since AtlasNet uses a pretrained model, we evaluate it on the intersection of the test splits from [47] and [92].*

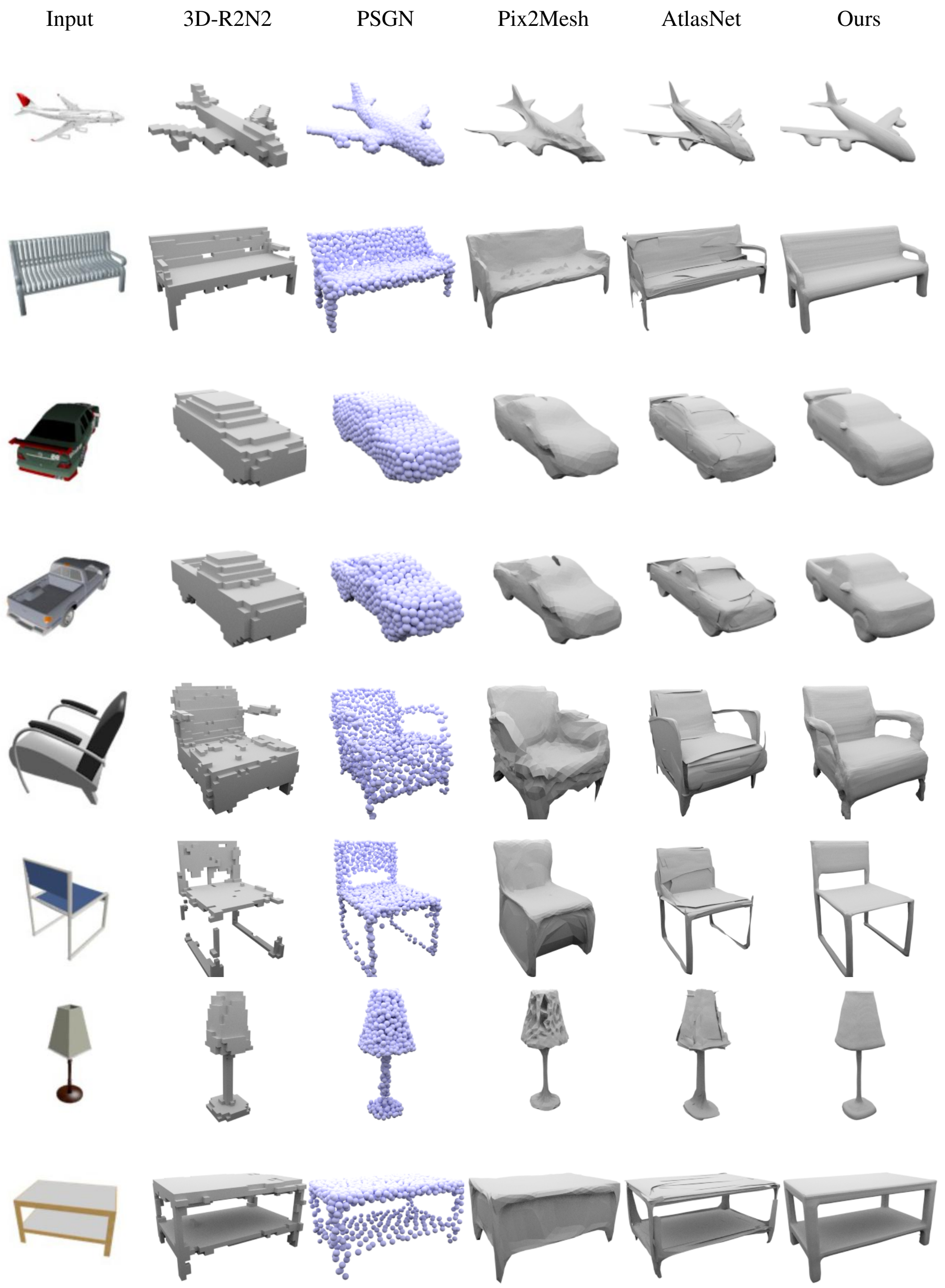


Figure 5.6: Single-View 3D Reconstruction. *The input image is shown in the first column, the other columns show the results for our method compared to various baselines.*

3D-R2N2 and PSGN. For PSGN we use a fully connected decoder with 4 layers and 512 hidden units in each layer. The last layer projects the hidden representation to a 3072-dimensional vector which we reshape into 1024 3D points. As we use only a single input view, we remove the recurrent network in 3D-R2N2. We reimplemented the method of [303] in PyTorch, closely following the Tensorflow implementation provided by the authors. For the method of [92], we use the code and pretrained model from the authors.

For all methods, we track the loss and other metrics on the validation set and stop training as soon as the target metric reaches its optimum. For 3D-R2N2 and our method, we use the IoU to the ground truth mesh as the target metric, for PSGN and Pixel2Mesh we use the Chamfer distance to the ground truth mesh as the target metric. To extract the final mesh, we use a threshold of 0.4 for 3D-R2N2 as suggested in the original publication [47]. To choose the threshold parameter τ for our method, we performed grid search on the validation set and found that $\tau = 0.2$ yields a good trade-off between accuracy and completeness.

Qualitative results from our model and the baselines are shown in Figure 5.6. We observe that all methods are able to capture the 3D geometry of the input image. However, 3D-R2N2 produces a very coarse representation and hence lacks details. In contrast, PSGN produces a high-fidelity output but lacks connectivity. As a result, PSGN requires additional lossy post-processing steps to produce a final mesh. Pixel2Mesh is able to create compelling meshes, but often misses holes in the presence of more complicated topologies. Such topologies are frequent, for example, for the “chairs“ category in the ShapeNet dataset. Similarly, AtlasNet captures the geometry well but produces artifacts in form of self-intersections and overlapping patches.

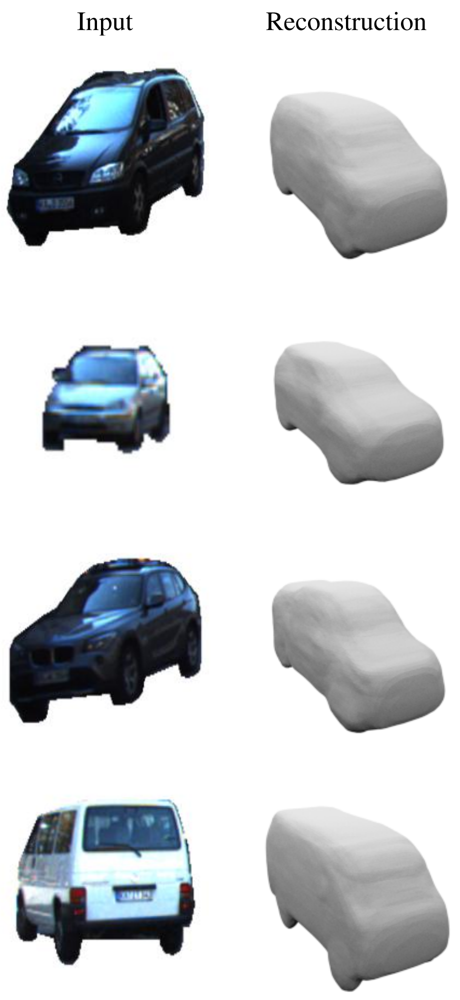
In contrast, our method captures complex topologies, produces closed meshes, and preserves most of the details.

Quantitative results are shown in Table 5.1. We observe that our method achieves the highest IoU and normal consistency to the ground truth mesh. Surprisingly, while not trained wrt. Chamfer distance as PSGN, Pixel2Mesh or AtlasNet, our method also achieves good results for this metric. Note that it is not possible to evaluate the IoU for PSGN or AtlasNet, as they do not yield watertight meshes.

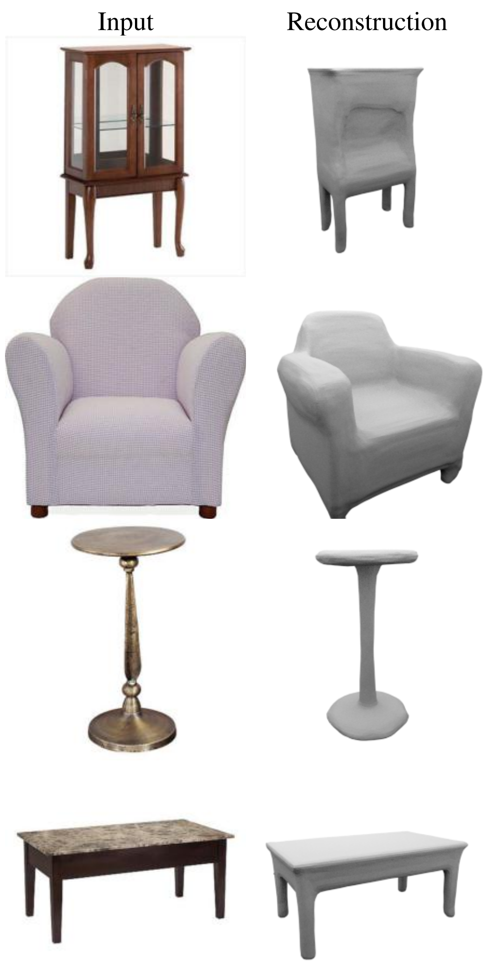
Real Data To test how well our model generalizes to real data, we apply our network to the KITTI [80] and Online Products datasets [206]. To capture the variety in viewpoints of KITTI and Online Products, we rerendered all ShapeNet objects with random camera locations and retrained our network for this task.

For the KITTI dataset, we additionally use the instance masks provided in [4] to mask and crop car regions. We then feed these images into our neural network to predict the occupancy function. Some selected qualitative results are shown in Figure 5.7a. Despite only trained on synthetic data, we observe that our method is also able to generate realistic reconstructions in this challenging setting.

For the Online Products dataset, we apply the same pretrained model. Several qualitative results are shown in Figure 5.7b. Again, we observe that our method generalizes reasonably well to real images despite being trained solely on synthetic data.



(a) KITTI



(b) Online Products

Figure 5.7: Qualitative Results for Real Data. *We applied our trained model to the KITTI and Online Products datasets. Despite only being trained on synthetic data, our model generalizes reasonably well to real data.*

	IoU	Chamfer-L_1	Normal Consistency
3D-R2N2	0.565	0.169	0.719
PSGN	-	0.144	-
DMC	0.674	0.117	0.848
ONet	**0.778**	**0.079**	**0.895**

Table 5.2: 3D Reconstruction from Point Clouds. *This table shows a numerical comparison of our approach wrt. the baselines for 3D reconstruction from point clouds on the ShapeNet dataset. We measure IoU, Chamfer-L_1 distance and Normal Consistency wrt. the ground truth mesh.*

	IoU	Chamfer-L_1	Normal Consistency
Input	0.631	0.136	0.810
ONet	**0.703**	**0.109**	**0.879**

Table 5.3: Voxel Super-Resolution. *This table shows a numerical comparison of the output of our approach in comparison to the input on the ShapeNet dataset.*

5.3.3 Point Cloud Completion

As a second conditional task, we apply our method to the problem of reconstructing the mesh from noisy point clouds. Towards this goal, we subsample 300 points from the surface of each of the (watertight) ShapeNet models and apply noise using a Gaussian distribution with zero mean and standard deviation 0.05 to the point clouds.

Again, we measure both the IoU and Chamfer-L_1 distance wrt. the ground truth mesh. The results are shown in Table 5.2. We observe that our method achieves the highest IoU and normal consistency as well as the lowest Chamfer-L_1 distance. Note that all numbers are significantly better than for the single-image 3D reconstruction task. This can be explained by the fact that this task is much easier for the recognition model, as there is less ambiguity and the model only has to fill in the gaps.

5.3.4 Voxel Super-Resolution

As a final conditional task, we apply Occupancy Networks to 3D super-resolution [262]. Here, the task is to reconstruct a high-resolution mesh from a coarse 32^3 voxelization of this mesh.

The results are shown in Table 5.3. We observe that our model considerably improves IoU, Chamfer-L_1 distance and normal consistency compared to the coarse input mesh.

5.3.5 Unconditional Mesh Generation

Finally, we apply our Occupancy Network to unconditional mesh generation, training it separately on four categories of the ShapeNet dataset in an unsupervised fashion. Our goal

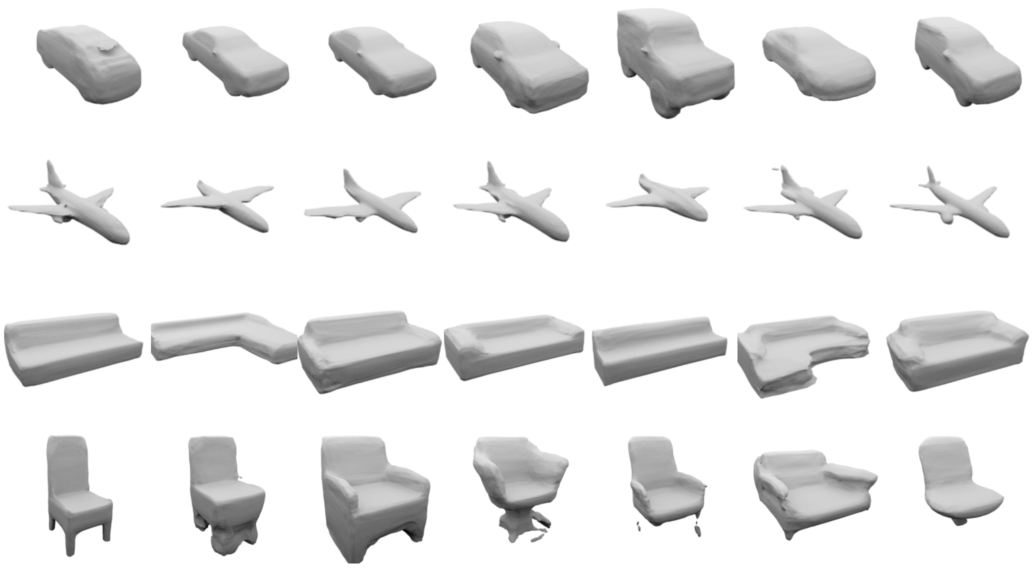


Figure 5.8: Unconditional 3D Samples. *Random samples of our unsupervised models trained on the categories "car", "airplane", "sofa" and "chair" of the ShapeNet dataset. We see that our models are able to capture the distribution of 3D objects and produce compelling new samples.*

is to explore how well our model can represent the latent space of 3D models. Some samples are shown in Figure 5.8. Indeed, we find that our model can generate compelling new models.

5.3.6 Ablation Study

In this section, we test how the various components of our model affect its performance on the single-image 3D-reconstruction task.

Effect of sampling strategy First, we examine how the sampling strategy affects the performance of our final model. We try three different sampling strategies: (i) sampling 2048 points uniformly in the bounding volume of the ground truth mesh (uniform sampling), (ii) sampling 1024 points inside and 1024 points outside mesh (equal sampling) and (iii) sampling 1024 points uniformly and 1024 points on the surface of the mesh plus some Gaussian noise with a standard deviation of 0.1 (surface sampling). We also examine the effect of the number of sampling points by decreasing this number from 2048 to 64.

The results are shown in Table 5.4a. To our surprise, we find that uniform, the simplest sampling strategy, works best. We explain this by the fact that other sampling strategies introduce bias to the model: for example, when sampling an equal number of points inside and outside the mesh, we implicitly tell the model that every object has a volume of 0.5. Indeed, when using this sampling strategy, we observe thickening artifacts in the model's

	IoU	Chamfer-L_1	Normal Consistency
Uniform	**0.571**	**0.215**	0.834
Uniform (64)	0.554	0.256	0.829
Equal	0.475	0.291	**0.835**
Surface	0.536	0.254	0.822

***(a)** Influence of Sampling Strategy*

	IoU	Chamfer-L_1	NC
Full model	**0.571**	**0.215**	**0.834**
No ResNet	0.559	0.243	0.831
No CBN	0.522	0.301	0.806

***(b)** Influence of Occupancy Network Architecture*

***Table 5.4: Ablation Study.** When we vary the sampling strategy, we observe that uniform sampling in the bounding volume performs best. Similarly, when we vary the architecture, we find that our ResNet architecture with conditional batch normalization yields the best results.*

output. Moreover, we find that reducing the number of sampling points from 2048 to 64 still leads to good performance, although the model does not perform as well as a model trained with 2048 sampling points.

Effect of architecture To test the effect of the various components of our architecture, we test two variations: (i) we remove the conditional batch normalization and replace it with a linear layer at the beginning of the network that projects the encoding of the input to the required hidden dimension and (ii) we remove all ResNet blocks in the decoder and replace them with linear blocks. The results are presented in Table 5.4b. We find that both components are helpful to achieve good performance.

5.4 Conclusion

In this chapter, we introduced Occupancy Networks, a novel neural field-based representation for 3D geometry. In contrast to existing representations, Occupancy Networks are not constrained by the discretization of the 3D space and can hence be used to represent realistic high-resolution meshes. Our experiments demonstrate that Occupancy Networks are expressive and can be used effectively for both supervised and unsupervised learning of 3D shapes.

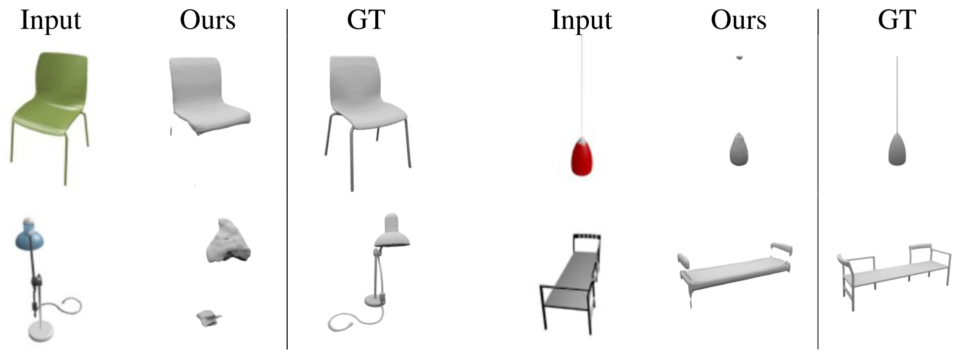


Figure 5.9: Failure Cases. *Occupancy Networks struggle with reconstructing thin details as well as objects that are very different from the training data. Positional encoding and better encoding schemes can further improve reconstruction quality (see text for details).*

5.4.1 Limitations and Future Work

Expressiveness Due to the inherent smoothness bias of multi-layer perceptrons, reconstructions are smooth. This can be an advantage when considering simple shapes where extrapolated areas, e.g., from sparse or single input views, often appear realistic. But for more complex topologies, the predictions tend to be overly smooth not containing high-frequency details (see Figure 5.9). To overcome this, the use of positional encoding in neural fields is proposed in [183] and analyzed to greater detail in [274]. The key idea is to map the three-dimensional coordinate to a higher-dimensional space before feeding it to the neural network. This allows the model to represent higher-frequency details. A related approach is investigated in [261] where ReLU activations are replaced with sinusoids. Together with a high-frequency multiplier in the first layer, it acts similarly to positional encoding and allows for higher-frequency details. Another line of research [169, 186, 273] combines multiple-level feature grids with locally-conditioned MLPs to achieve higher-quality reconstructions at the cost of larger memory requirements. We identify investigating efficient and differentiable approaches for representing low and high frequencies adaptively, i.e., depending on the shape's local structure, as promising future work.

Global Feature Encoding The Occupancy Network reconstructs a 3D shape from a single latent code representing the input image. As a result, the latent code needs to summarize the entire shape acting as a global descriptor which can lead to reconstructions that miss details (see Figure 5.9). In [247], Saito et al. use locally-pooled features for inferring reconstructions of human bodies from single views. In [324] different feature pooling strategies are compared showing that a combination of global and local features leads to the best results. Another line of work [35, 115, 219] partitions space into sub-spaces, e.g., voxel cells, and the network only needs to reconstruct the sub-space from local and global encodings. This leads to more expressive models that can reconstruct large-scale scenes like indoor rooms or houses in contrast to simple single-object scenes.

6 Differentiable Surface Rendering of Neural Fields

6.1 Introduction

We showed that neural fields are a promising representation of 3D geometry. However, we assumed that 3D ground truth data is available for training. This is a problem when scaling to more complex, real-world scenes where access to labeled data is limited. In this chapter, we analyze how neural fields that represent 3D surfaces can be differentiably rendered to the image plane. This is one key component for building models that can infer scene representations in the real world from sparse input data.

Prior Work Learning-based 3D reconstruction approaches have achieved impressive results [41, 47, 69, 92, 150, 179, 181, 212, 241, 305]. By using rich prior knowledge obtained during the training process, they are able to infer a 3D model from as little as a single image. However, most learning-based methods are restricted to synthetic data, mainly because they require accurate 3D ground truth models as supervision for training and reconstruction quality drops for a data domain shift at test time.

To overcome this barrier, several works have investigated approaches that require only 2D and 2.5D supervision in the form of multi-view images or depth maps. Most existing approaches achieve this by modifying the rendering process to make it differentiable [10, 42, 57, 81, 125, 139, 154, 155, 160, 194, 215, 223, 240, 278, 279, 286, 354]. While yielding compelling results, they are restricted to specific 3D representations (e.g., voxels or meshes) that suffer from discretization artifacts and the computational cost limits them to small resolutions or deforming a fixed template mesh. At the same time, neural field-based representations for shape [41, 179, 212] and texture [203, 247] have been proposed which do not require discretization during training and have a constant memory footprint. However, existing approaches using neural field representations require 3D ground truth for training and it remains unclear how to learn neural field representations from image data alone.

Contribution In this chapter, we introduce *Differentiable Volumetric Rendering (DVR)*. Our key insight is that we can derive analytic gradients for the predicted depth map wrt. the network parameters of the neural field-based shape and texture representation (see Figure 6.1). This insight enables us to design a differentiable renderer for neural field-based shape and texture representations and allows us to learn these representations solely from multi-view images and object masks. Since our method does not have to store volumetric data in the forward pass, its memory footprint is independent of the sampling accuracy of

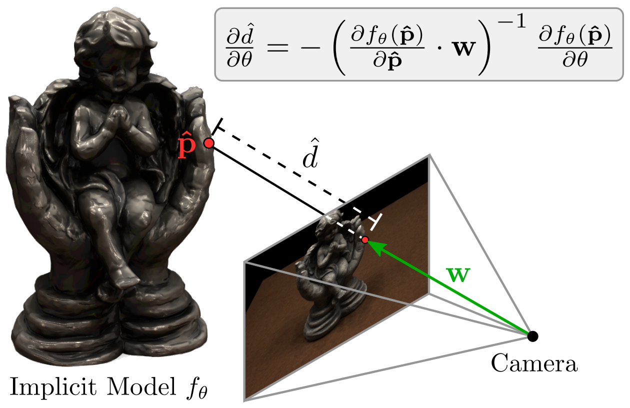

$\frac{\partial \hat{d}}{\partial \theta} = -\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w}\right)^{-1} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta}$
$\hat{\mathbf{p}}$
$\hat{d}$
$\mathbf{w}$
Camera
Implicit Model f_θ

Figure 6.1: Differentiable Surface Rendering of Neural Fields. *We show that volumetric rendering is inherently differentiable for neural field-based shape and texture representations. Using an analytic expression for the gradient of the depth $\frac{\partial \hat{d}}{\partial \theta}$ wrt. the network parameters θ, we learn neural field-based 3D representations f_θ from 2D images.*

the depth prediction step. We show that our formulation can be used for various tasks such as single- and multi-view reconstruction, and works with synthetic and real data. In contrast to [203], we do not need to condition the texture representation on the geometry but learn a *single model* with shared parameters that represents both, geometry and texture.

6.2 Method

In this section, we describe our Differentiable Volumetric Rendering (DVR) approach. We first define the implicit neural representation which we use for representing 3D shape and texture. Next, we provide a formal description of DVR and all relevant implementation details. An overview of our approach is provided in Figure 6.2.

6.2.1 Shape and Texture Representation

Shape In contrast to discrete voxel- and point-based representations, we represent the 3D shape of an object *implicitly* using the Occupancy Network introduced in the previous chapter:

$$f_\theta : \mathbb{R}^3 \times \mathcal{Z} \rightarrow [0,1] \tag{6.1}$$

The Occupancy Network $f_\theta(\mathbf{p},\mathbf{z})$ assigns a probability of occupancy to every point $\mathbf{p} \in \mathbb{R}^3$ in 3D space. For the task of single-view reconstruction, we process the input image with an encoder network $E_\theta(\cdot)$ and use the output $\mathbf{z} \in \mathcal{Z}$ to condition f_θ. The 3D surface of an object is implicitly determined by the level set $\mathcal{S}_{f_\theta} = \{\mathbf{p} \in \mathbb{R}^3 | f_\theta(\mathbf{p}) = \tau\}$ for a threshold parameter $\tau \in [0,1]$ and can be extracted at arbitrary resolution using isosurface extraction techniques (see Section 5.2.3).

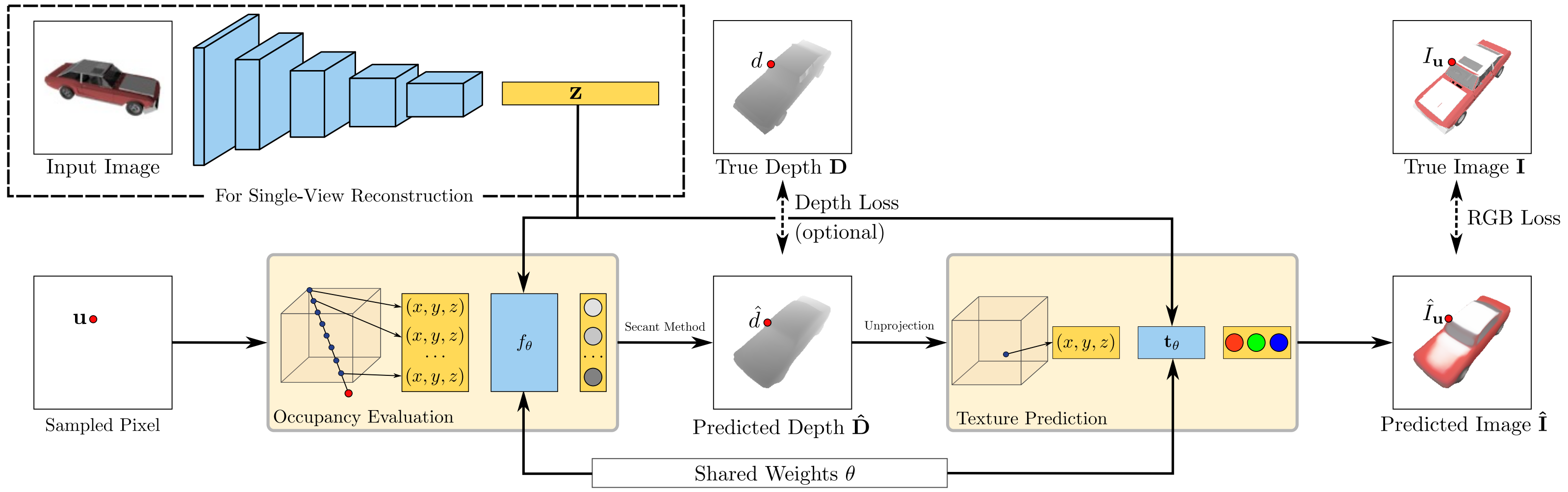


Figure 6.2: ***Differentiable Volumetric Rendering.*** *We first predict the surface depth $\hat{d}$ by performing occupancy evaluations for a given camera matrix. To this end, we project sampled pixel* $\mathbf{u}$ *to 3D and evaluate the Occupancy Network at fixed steps on the ray cast from the camera origin towards this point. We then unproject the surface depth into 3D and evaluate the Texture Field at the given 3D location. The resulting 2D rendering* $\hat{\mathbf{I}}$ *can be compared to the ground truth image. When we also have access to ground truth depth maps, we can define a loss directly on the predicted surface depth. We can make our model conditional by incorporating an additional image encoder that predicts a global descriptor* $\mathbf{z}$ *of both shape and texture.*

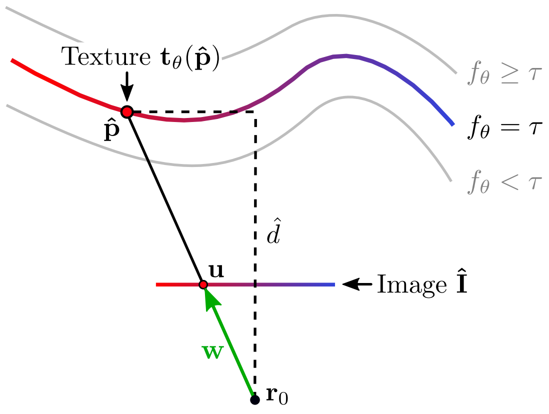


Figure 6.3: Notation. *To render an object from the Occupancy Network f_θ and Texture Field $\mathbf{t}_\theta$, we cast a ray with direction $\mathbf{d} = \frac{\mathbf{w}}{||\mathbf{w}||_2}$ through a pixel $\mathbf{u}$ and determine the intersection point $\hat{\mathbf{p}}$ with the isosurface $f_\theta(\mathbf{p}) = \tau$. Afterward, we evaluate the Texture Field $\mathbf{t}_\theta$ at $\hat{\mathbf{p}}$ to obtain the color prediction $\hat{\mathbf{I}}_\mathbf{u}$ at $\mathbf{u}$.*

Texture Similarly, we can describe the texture of a 3D object using a Texture Field [203]

$$\mathbf{t}_\theta : \mathbb{R}^3 \times \mathcal{Z} \to [0,1]^3 \tag{6.2}$$

which regresses an RGB color value $\mathbf{c} \in [0,1]^3$ for every point $\mathbf{p} \in \mathbb{R}^3$ in 3D space. Again, $\mathbf{t}_\theta$ can be conditioned on a latent embedding $\mathbf{z}$ of the object. The texture of an object is given by the values of $\mathbf{t}_\theta$ on the object's surface ($f_\theta = \tau$). In this work, we implement f_θ and $\mathbf{t}_\theta$ as a single neural network with two shallow heads.

Supervision Recent works [41, 179, 203, 212, 247] have shown that it is possible to learn f_θ and $\mathbf{t}_\theta$ with 3D supervision (i.e., ground truth 3D models). However, ground truth 3D data is often very expensive or even impossible to obtain for real-world datasets. In the next section, we introduce DVR, an alternative approach that enables us to learn both f_θ and $\mathbf{t}_\theta$ from 2D images alone. For clarity, we drop the condition variable $\mathbf{z}$ in the following.

6.2.2 Differentiable Volumetric Rendering

Our goal is to learn f_θ and $\mathbf{t}_\theta$ from 2D image observations. Consider a single image observation. We define a photometric reconstruction loss

$$\mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}) = \sum_{\mathbf{u}} ||\hat{\mathbf{I}}_\mathbf{u} - \mathbf{I}_\mathbf{u}|| \tag{6.3}$$

which we aim to optimize. Here, $\mathbf{I}$ denotes the observed image and $\hat{\mathbf{I}}$ is the image rendered by our implicit model.[1] Moreover, $\mathbf{I}_\mathbf{u}$ denotes the RGB value of the observation $\mathbf{I}$ at pixel

[1]Note that the rendered image $\hat{\mathbf{I}}$ depends on θ through f_θ and $\mathbf{t}_\theta$. We have dropped this dependency here to avoid clutter in the notation.

$\mathbf{u}$ and $||\cdot||$ is a (robust) photo-consistency measure such as the ℓ_1-norm. To minimize the reconstruction loss $\mathcal{L}$ wrt. the network parameters θ using gradient-based optimization techniques, we must be able to (i) **render** $\hat{\mathbf{I}}$ given f_θ and $\mathbf{t}_\theta$ and (ii) compute **gradients** of $\mathcal{L}$ wrt. the network parameters θ. Our core contribution is to provide solutions to both problems, leading to an efficient algorithm for learning implicit 3D representations from 2D images.

Rendering We adopt the ray marching rendering operator π^{rm} in DVR (see Section 4.3). More specifically, for a camera located at $\mathbf{r}_0$ we can predict the color $\hat{\mathbf{I}}_\mathbf{u}$ at pixel $\mathbf{u}$ by casting a ray from $\mathbf{r}_0$ through $\mathbf{u}$ and determining the first point of intersection $\hat{\mathbf{p}}$ with the isosurface $\mathcal{S}_{f_\theta} = \{\mathbf{p} \in \mathbb{R}^3 | f_\theta(\mathbf{p}) = \tau\}$ as illustrated in Figure 6.3. The color value $\hat{\mathbf{I}}_\mathbf{u}$ is then given by $\hat{\mathbf{I}}_\mathbf{u} = \mathbf{t}_\theta(\hat{\mathbf{p}})$.

Gradients To obtain gradients of $\mathcal{L}$ with respect to θ, we first use the multivariate chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{\mathbf{u}} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{I}}_\mathbf{u}} \cdot \frac{\partial \hat{\mathbf{I}}_\mathbf{u}}{\partial \theta} \tag{6.4}$$

Here, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ denotes the Jacobian matrix for a vector-valued function $\mathbf{f}$ with vector-valued argument $\mathbf{x}$ and $\cdot$ indicates matrix multiplication. By exploiting $\hat{\mathbf{I}}_\mathbf{u} = \mathbf{t}_\theta(\hat{\mathbf{p}})$, we obtain

$$\frac{\partial \hat{\mathbf{I}}_\mathbf{u}}{\partial \theta} = \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} \tag{6.5}$$

since both $\mathbf{t}_\theta$ as well as $\hat{\mathbf{p}}$ depend on θ. Because $\hat{\mathbf{p}}$ is defined implicitly, calculating $\frac{\partial \hat{\mathbf{p}}}{\partial \theta}$ is non-trivial. We first exploit that $\hat{\mathbf{p}}$ lies on the ray from $\mathbf{r}_0$ through $\mathbf{u}$. For any pixel $\mathbf{u}$, this ray can be described by $\mathbf{r}(d) = \mathbf{r}_0 + d\mathbf{w}$ where $\mathbf{w}$ is the vector connecting $\mathbf{r}_0$ and $\mathbf{u}$ (see Figure 6.3).[2] Since $\hat{\mathbf{p}}$ must lie on $\mathbf{r}$, there exists a depth value $\hat{d}$, such that $\hat{\mathbf{p}} = \mathbf{r}(\hat{d})$. We call $\hat{d}$ the *surface depth*. This enables us to rewrite $\frac{\partial \hat{\mathbf{p}}}{\partial \theta}$ as

$$\frac{\partial \hat{\mathbf{p}}}{\partial \theta} = \frac{\partial \mathbf{r}(\hat{d})}{\partial \theta} = \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} \tag{6.6}$$

For computing the gradient of the surface depth $\hat{d}$ with respect to θ we exploit *implicit differentiation* [7, 245]. Differentiating $f_\theta(\hat{\mathbf{p}}) = \tau$ on both sides wrt. θ, we obtain:

$$\begin{aligned} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} &= 0 \\ \Leftrightarrow \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} &= 0 \end{aligned} \tag{6.7}$$

[2]Note that to ensure generality, we do not assume $\mathbf{w}$ to be normalized, and the relation to the in Section 4.3 introduced normalized ray direction is $\mathbf{d} = \frac{\mathbf{w}}{||\mathbf{w}||_2}$. It follows that $\mathbf{w} = \mathbf{d}$ if $||\mathbf{w}||_2 = 1$.

Rearranging Equation 6.7, we arrive at the following closed-form expression for the gradient of the surface depth $\hat{d}$:

$$\frac{\partial \hat{d}}{\partial \theta} = -\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w}\right)^{-1} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \tag{6.8}$$

We remark that calculating the gradient of the surface depth $\hat{d}$ wrt. the network parameters θ only involves calculating the gradient of f_θ at $\hat{\mathbf{p}}$ wrt. the network parameters θ and the surface point $\hat{\mathbf{p}}$. Thus, in contrast to voxel-based approaches [215, 286], we do not have to store intermediate results (e.g., volumetric data) for computing the gradient of the loss wrt. the parameters, resulting in a memory-efficient algorithm. In the next section, we describe our implementation of DVR which makes use of reverse-mode automatic differentiation to compute the full gradient Equation 6.4.

6.2.3 Implementation

To use automatic differentiation, we have to implement the forward and backward pass for the surface depth prediction step $\theta \rightarrow \hat{d}$. In the following, we describe how both passes are implemented.

Forward Pass We implement the forward pass as ray marching. As visualized in Figure 6.3, we can determine $\hat{d}$ by finding the first occupancy change on the ray $\mathbf{r}$. To detect an occupancy change, we evaluate the Occupancy Network $f_\theta(\cdot)$ at n equally-spaced samples on the ray $\{\mathbf{p}_j^{\text{ray}}\}_{j=1}^n$. Using a step size of Δs, we can express the coordinates of these points in world-coordinates as

$$\mathbf{p}_j^{\text{ray}} = \mathbf{r}(j\Delta s + s_0) \tag{6.9}$$

where s_0 determines the closest possible surface point. We first find the smallest j for which f_θ changes from free space ($f_\theta < \tau$) to occupied space ($f_\theta \geq \tau$):

$$j = \underset{j'}{\operatorname{argmin}} \left(f_\theta(\mathbf{p}_{j'+1}^{\text{ray}}) \geq \tau > f_\theta(\mathbf{p}_{j'}^{\text{ray}}) \right) \tag{6.10}$$

We obtain an approximation to the surface depth $\hat{d}$ by applying the iterative secant method to the interval $[j\Delta s + s_0, (j+1)\Delta s + s_0]$. In practice, we compute the surface depth for a batch of N_p points in parallel. It is important to note that we do not need to unroll the forward pass or store any intermediate results as we exploit implicit differentiation to directly obtain the gradient of $\hat{d}$ wrt. θ.

Backward Pass The input to the backward pass is the gradient $\lambda = \frac{\partial \mathcal{L}}{\partial \hat{d}}$ of the loss wrt. a single surface depth prediction. The output of the backward pass is $\lambda \frac{\partial \hat{d}}{\partial \theta}$, which can be computed using Equation 6.8. In practice, however, we would like to implement the backward pass not only for a single surface depth $\hat{d}$ but for a whole batch of depth values.

We can implement this efficiently by rewriting $\lambda \frac{\partial \hat{d}}{\partial \theta}$ as

$$\mu \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \quad \text{with} \quad \mu = -\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \lambda \tag{6.11}$$

Importantly, the left term in Equation 6.11 corresponds to a normal backward operation applied to the neural network f_θ and the right term in Equation 6.11 is just an (element-wise) scalar multiplication for all elements in the batch. We can hence conveniently compute the backward pass of the operator $\theta \to \hat{d}$ by first multiplying the incoming gradient λ element-wise with a factor and then backpropagating the result through the operator $\theta \to f_\theta(\hat{\mathbf{p}})$. Both operations can be efficiently parallelized in common deep-learning frameworks.

6.2.4 Training

During training, we assume that we are given N images $\{\mathbf{I}_k\}_{k=1}^N$ together with corresponding camera intrinsics, extrinsics, and object masks $\{\mathbf{M}_k\}_{k=1}^N$. As our experiments show, our method works with as little as one image per object. In addition, our method can also incorporate depth information $\{\mathbf{D}_k\}_{k=1}^N$, if available.

For training f_θ and $\mathbf{t}_\theta$, we randomly sample an image $\mathbf{I}_k$ and N_p points $\mathbf{u}$ on the image plane. We distinguish the following three cases: First, let $\mathcal{P}_0$ denote the set of points $\mathbf{u}$ that lie inside the object mask $\mathbf{M}_k$ and for which the Occupancy Network predicts a finite surface depth $\hat{d}$ as described in Section 6.2.3. For these points, we can define a loss $\mathcal{L}_{\text{rgb}}(\theta)$ directly on the predicted image $\hat{\mathbf{I}}_k$. Moreover, let $\mathcal{P}_1$ denote the points $\mathbf{u}$ which lie outside the object mask $\mathbf{M}_k$. While we cannot define a photometric loss for these points, we can define a loss $\mathcal{L}_{\text{freespace}}(\theta)$ that encourages the network to remove spurious geometry on the corresponding rays. Finally, let $\mathcal{P}_2$ denote the set of points $\mathbf{u}$ which lie inside the object mask $\mathbf{M}_k$, but for which the Occupancy Network does not predict a finite surface depth $\hat{d}$. Again, we cannot use a photometric loss for these points, but we can define a loss $\mathcal{L}_{\text{occupancy}}(\theta)$ that encourages the network to produce a finite surface depth.

RGB Loss For each point in $\mathcal{P}_0$, we detect the predicted surface depth $\hat{d}$ as described in Section 6.2.3. We define a photo-consistency loss for the points as

$$\mathcal{L}_{\text{rgb}}(\theta) = \sum_{u \in \mathcal{P}_0} \|\xi(\mathbf{I})_u - \xi(\hat{\mathbf{I}})_u\| \tag{6.12}$$

where $\xi(\cdot)$ computes image features and $\|\cdot\|$ defines a robust error metric. In practice, we use RGB values and (optionally) image gradients as features and an ℓ_1-loss for $\|\cdot\|$.

Depth Loss When the depth is also given, we can directly incorporate an ℓ_1 loss on the predicted surface depth as

$$\mathcal{L}_{\text{depth}}(\theta) = \sum_{u \in \mathcal{P}_0} |d - \hat{d}|_1 \tag{6.13}$$

where d indicates the ground truth depth value of the sampled image point $\mathbf{u}$ and $\hat{d}$ denotes the predicted surface depth for pixel $\mathbf{u}$.

Freespace Loss If a point $\mathbf{u}$ lies outside the object mask but the predicted surface depth $\hat{d}$ is finite, the network falsely predicts surface point $\hat{\mathbf{p}} = \mathbf{r}(\hat{d})$. Therefore, we penalize this occupancy with

$$\mathcal{L}_{\text{freespace}}(\theta) = \sum_{u \in \mathcal{P}_1} \text{BCE}(f_\theta(\hat{\mathbf{p}}_u), 0) \tag{6.14}$$

where BCE is the binary cross entropy. When no surface depth is predicted, we apply the freespace loss to a randomly sampled point on the ray.

Occupancy Loss If a point $\mathbf{u}$ lies inside the object mask but the predicted surface depth $\hat{d}$ is infinite, the network falsely predicts no surface points on ray $\mathbf{r}$. To encourage predicting occupied space on this ray, we uniformly sample depth values d_{random} and define

$$\mathcal{L}_{\text{occupancy}}(\theta) = \sum_{u \in \mathcal{P}_2} \text{BCE}(f_\theta(\mathbf{r}_u(d_{\text{random}})), 1) \tag{6.15}$$

In the single-view reconstruction experiments, we instead use the first point on the ray which lies inside all object masks (depth of the visual hull). If we have additional depth supervision, we use the ground truth depth for the occupancy loss. Intuitively, $\mathcal{L}_{\text{occupancy}}$ encourages the network to occupy space along the respective rays which can then be used by $\mathcal{L}_{\text{rgb}}$ in Equation 6.12 and $\mathcal{L}_{\text{depth}}$ in Equation 6.13 to refine the initial occupancy.

Normal Loss Optionally, our representation allows us to incorporate a smoothness prior by regularizing surface normals. This is useful, especially for real-world data as training with 2D or 2.5D supervision includes unconstrained areas where this prior enforces more natural shapes. We define this loss as

$$\mathcal{L}_{\text{normal}}(\theta) = \sum_{u \in \mathcal{P}_0} ||\mathbf{n}(\hat{\mathbf{p}}_u) - \mathbf{n}(\mathbf{q}_u)||_2 \tag{6.16}$$

where $\mathbf{n}(\cdot)$ denotes the normal vector, $\hat{\mathbf{p}}_u$ the predicted surface point and $\mathbf{q}_u$ a randomly sampled neighbor of $\hat{\mathbf{p}}_u$.

6.2.5 Implementation Details

We implement the combined network with 5 fully-connected ResNet [98] blocks and ReLU activation. The output dimension of the last layer is 4, with one dimension for the occupancy probability and three dimensions for the texture. For the single-view reconstruction experiments, we encode the input image with a ResNet-18 [98] encoder network E_ϕ which outputs a 256-dimensional latent code z. To facilitate training, we start with a ray sampling accuracy of $n = 16$ which we iteratively increase to $n = 128$ by doubling n after 50, 150, and 250 thousand iterations. We choose the sampling interval $[s_0, n\Delta s + s_0]$ such that it covers the volume of interest for each object. We set $\tau = 0.5$ for all experiments. We train on a single

NVIDIA V100 GPU with a batch size of 64 images with 1024 random pixels each. We use the Adam optimizer [130] with a learning rate of 10^{-4} which we decrease by a factor of 5 after 750 and 1000 epochs, respectively.

6.3 Experiments

We conduct two different types of experiments to validate our approach. First, we investigate how well our approach reconstructs 3D shape and texture from a **single RGB image** when trained on a large collection of RGB or RGB-D images. Here, we consider both the case where we have access to multi-view supervision and the case where we use only a single RGB-D image per object during training. Next, we apply our approach to the challenging task of **multi-view reconstruction**, where the goal is to reconstruct complex 3D objects from real-world multi-view imagery.

6.3.1 Single-View Reconstruction

First, we investigate to which degree our method can infer a 3D shape and texture representation from single views. We train a single model jointly on all categories.

Datasets To adhere to community standards [47, 179, 305], we use the Choy et al. [47] subset (13 classes) of the ShapeNet dataset [34] for 2.5D and 3D supervised methods with training, validation, and test split from [179]. While we use the renderings from Choy et al. [47] as input, we additionally render 24 images of resolution 256^2 with depth maps and object masks per object which we use for supervision. We randomly sample the viewpoint on the northern hemisphere as well as the distance of the camera to the object to get diverse supervision data. For 2D supervised methods, we adhere to community standards [125, 155, 327] and use the renderings and splits from [125]. Similar to [47, 125, 179], we train with objects in the canonical pose.

Baselines We compare against the following methods which all produce watertight meshes as output: 3D-R2N2 [47] (voxel-based), Pixel2Mesh [305] (mesh-based), and ONet [179] (implicit representation). We further compare against both the 2D and the 2.5D supervised version of Differentiable Ray Consistency (DRC) [286] (voxel-based) and the 2D supervised Soft Rasterizer (SoftRas) [155] (mesh-bapresed). For 3D-R2N2, we use the pretrained model from [179] which was shown to produce better results than the original model from [47]. For the other baselines, we use the pretrained models[3] from the authors.

Multi-View Supervision We first consider the case where we have access to multi-view supervision with $N = 24$ images and corresponding object masks. In addition, we also investigate the case when ground truth depth maps are given.

[3]Unfortunately, we cannot show texture results for DRC and SoftRas as texture prediction is not part of the official code repositories.

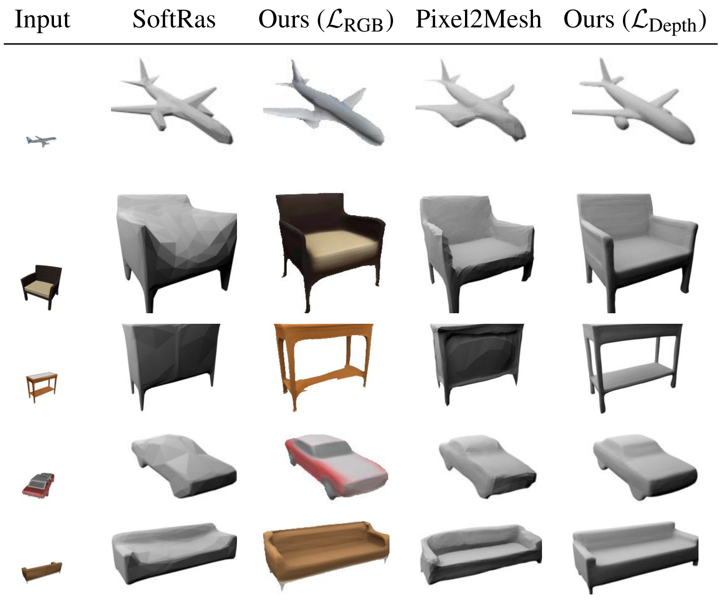

Input
SoftRas
Ours ($\mathcal{L}_{\text{RGB}}$)
Pixel2Mesh
Ours ($\mathcal{L}_{\text{Depth}}$)

Figure 6.4: Single-View Reconstruction. *We show the input renderings from [47] and the output of our 2D supervised ($\mathcal{L}_{RGB}$) and 2.5D supervised ($\mathcal{L}_{Depth}$) model, Soft Rasterizer [155] and Pixel2Mesh [305]. For 2D supervised methods, we use a corresponding view from [125] as input.*

We evaluate the results using the Chamfer-L_1 distance from [179]. In contrast to previous works [47, 155, 179, 286], we compare directly wrt. to the ground truth shape models, not the voxelized or watertight versions.

In Table 6.1 and Figure 6.4 we show quantitative and qualitative results for our method and various baselines. We can see that our method is able to infer accurate 3D shape and texture representations from single-view images when only trained on multi-view images and object masks as the supervision signal. Quantitatively (Table 6.1), our method performs best among the approaches with 2D supervision and rivals the quality of methods with full 3D supervision. When trained with depth, our method performs comparably to the methods which use full 3D information. Qualitatively (Figure 6.4), we see that in contrast to the mesh-based approaches, our method is not restricted to certain topologies. When trained with the photo-consistency loss $\mathcal{L}_{\text{RGB}}$, we see that our approach is able to predict accurate texture information in addition to the 3D shape.

Single-View Supervision The previous experiment indicates that our model is able to infer accurate shape and texture information without 3D supervision. A natural question to ask is how many images are required during training. To this end, we investigate the case when only *a single image* with depth and camera information is available. Since we represent the 3D shape in a canonical object coordinate system, the hypothesis is that the model can aggregate the information over multiple training instances, although it sees every object only from one perspective. As the same image is used both as input and supervision

	2D Supervision			2.5D Supervision		3D Supervision		
category	DRC (Mask) [286]	SoftRas [155]	Ours ($\mathcal{L}_{\text{RGB}}$)	DRC (Depth) [286]	Ours ($\mathcal{L}_{\text{Depth}}$)	3D R2N2 [47]	ONet [179]	Pixel2Mesh [305]
airplane	0.659	**0.149**	0.190	0.377	**0.143**	0.215	**0.151**	0.183
bench	-	0.241	**0.210**	-	**0.165**	0.210	**0.171**	0.191
cabinet	-	0.231	**0.220**	-	**0.183**	0.246	**0.189**	0.194
car	0.340	0.221	**0.196**	0.316	**0.179**	0.250	0.181	**0.154**
chair	0.660	0.338	**0.264**	0.510	**0.226**	0.282	**0.224**	0.259
display	-	0.284	**0.255**	-	**0.246**	0.323	0.275	**0.231**
lamp	-	**0.381**	0.413	-	**0.362**	0.566	0.380	**0.309**
loudspeaker	-	0.320	**0.289**	-	**0.295**	0.333	0.290	**0.284**
rifle	-	**0.155**	0.175	-	**0.143**	0.199	0.160	**0.151**
sofa	-	0.407	**0.224**	-	**0.221**	0.264	0.217	**0.211**
table	-	0.374	**0.280**	-	**0.180**	0.247	**0.185**	0.215
telephone	-	**0.131**	0.148	-	**0.130**	0.221	0.155	**0.145**
vessel	-	**0.233**	0.245	-	**0.206**	0.248	0.220	**0.201**
mean	0.553	0.266	**0.239**	0.401	**0.206**	0.277	0.215	**0.210**

Table 6.1: Single-View Reconstruction. *We report Chamfer-L_1 distances wrt. the ground truth meshes for the single-view experiment. We compare against Differentiable Ray Consistency (DRC) [286] (2D and 2.5D supervision), Soft Rasterizer [155] (2D supervision), 3D-R2N2 [47], Occupancy Networks (ONet) [179], and Pixel2Mesh [305] (all 3D supervision).*

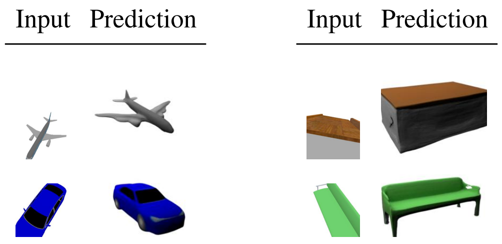
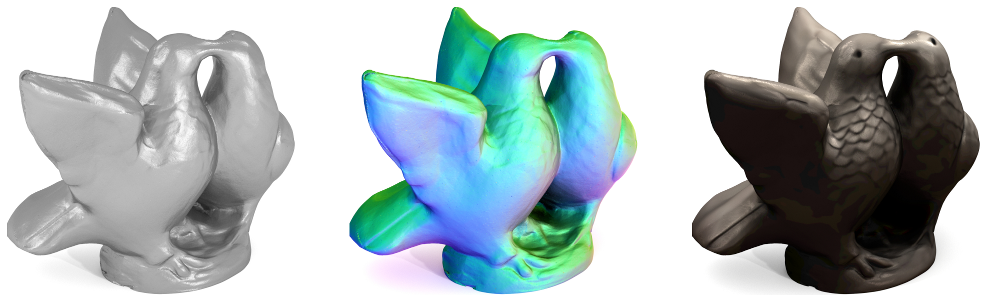


Figure 6.5: Single-View Reconstruction with Single-View Supervision. *While only trained with a single view per object, our model predicts accurate 3D geometry and texture.*



***(a)** Shape* ***(b)** Normals* ***(c)** Texture*

Figure 6.6: Multi-View Stereo. *We show the shape, normals, and the textured shape for our method trained with 2D images and sparse depth maps for scan* 106 *of the DTU dataset [1].*

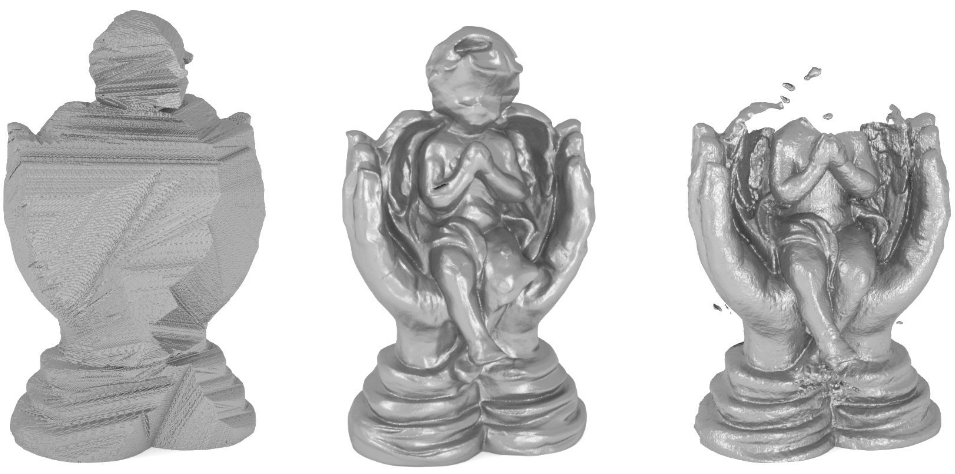
signal, we now condition on our renderings instead of the ones provided by Choy et al. [47].

Surprisingly, Figure 6.5 shows that our method can infer appropriate 3D shape and texture when only a single view is available per object, confirming our hypothesis. Quantitatively, the Chamfer distance of the model trained with $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{Depth}}$ with only a single view (0.410) is comparable to the model trained with $\mathcal{L}_{\text{Depth}}$ with 24 views (0.383). The reason for the numbers being worse than in Section 6.3.1 is that for our renderings, we do not only sample the viewpoint but also the distance to the object resulting in a much harder task (see Figure 6.5).

6.3.2 Multi-View Reconstruction

Finally, we investigate if our method is also applicable to multi-view reconstruction in real-world scenarios. We investigate two cases: First, when multi-view images and object masks are given. Second, when additional sparse depth maps are given which can be obtained from classic multi-view stereo algorithms [254]. For this experiment, we do not condition our model and train one model per object.

Dataset We conduct this experiment on scans 65, 106, and 118 from the challenging real-world DTU dataset [1]. The dataset contains 49 or 65 images with camera information for each object and baseline and structured light ground truth data. The presented objects



*(**a**) Visual Hull [141]* *(**b**) Ours ($\mathcal{L}_{RGB}$)* *(**c**) Ground Truth*

Figure 6.7: Comparison against Visual Hull. *We show the visual hull, the shape prediction of our model trained with $\mathcal{L}_{RGB}$, and the ground truth for scan* 118 *of the DTU dataset. Our method uses RGB cues to improve over the visual hull and predicts parts that are missing in the ground truth.*

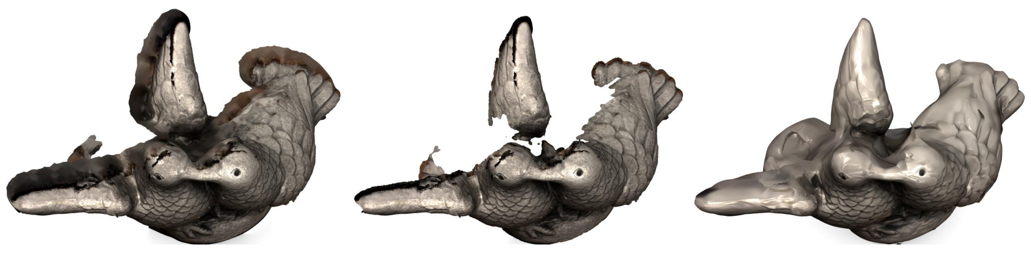
are challenging as their appearance changes in different viewpoints due to specularities. Our sampling-based approach allows us to train on the full image resolution of 1200×1600. We label the object masks ourselves and always remove the same images with profound changes in lighting conditions, e.g., caused by the appearance of scanner parts in the background.

Baselines We compare against classical approaches that have 3D meshes as output. To this end, we run screened Poisson surface reconstruction (sPSR) [127] on the output of the classical MVS algorithms Campbell et al. [33], Furukawa et al. [72], Tola et al. [284], and Colmap [254]. We find that the results on the DTU benchmark for the baselines are highly sensitive to the trim parameter of sPSR and therefore report results for the trim parameters 0 (watertight output), 5 (good qualitative results) and 7 (good quantitative results). For a fair comparison, we use the object masks to remove all points which lie outside the visual hull from the predictions of the baselines before running sPSR. We use the official DTU evaluation script in "surface mode".

Results We show qualitative and quantitative results in Figure 6.6 and Table 6.2. Qualitatively, we find that our method can be used for multi-view 3D reconstruction, directly resulting in watertight meshes. The ability to accurately model cavities of the objects shows that our model uses texture information to improve over the visual hull (Figure 6.7). Quantitatively, Table 6.2 shows that our approach rivals the results from highly tuned MVS algorithms. We note that the DTU ground truth is itself sparse (Figure 6.7c) and methods are therefore rewarded for trading off completeness for accuracy, which explains the better quantitative performance of the baselines for higher trim parameters (Figure 6.8).

	Trim Param.	Accuracy	Completeness	Chamfer-L_1
Tola [284] + sPSR	0	2.409	1.242	1.826
Furu [72] + sPSR	0	2.146	0.888	1.517
Colmap [254] + sPSR	0	**1.881**	0.726	**1.303**
Camp [33] + sPSR	0	2.213	**0.670**	1.441
Tola [284] + sPSR	5	1.531	1.267	1.399
Furu [72] + sPSR	5	1.733	0.888	1.311
Colmap [254] + sPSR	5	**1.400**	0.782	**1.091**
Camp [33] + sPSR	5	1.991	**0.670**	1.331
Tola [284] + sPSR	7	**0.396**	1.424	0.910
Furu [72] + sPSR	7	0.723	0.955	0.839
Colmap [254] + sPSR	7	0.446	1.020	**0.733**
Camp [33] + sPSR	7	1.466	**0.719**	1.092
Ours ($\mathcal{L}_{\text{RGB}}$)	-	1.054	**0.760**	0.907
Ours ($\mathcal{L}_{\text{RGB}}$ + $\mathcal{L}_{\text{Depth}}$)	-	**0.789**	0.775	**0.782**

Table 6.2: Multi-View Stereo. *We show quantitative results for scans* 65, 106, *and* 118 *on the DTU dataset. For the baselines, we perform screened Poisson surface reconstruction (sPSR) [127] with trim parameters* 0, 5, *and* 7 *to obtain the final output. It shows that our generic method achieves results comparable to the highly optimized MVS methods.*



***(a)** Colmap* 5 ***(b)** Colmap* 7 ***(c)** Ours*

Figure 6.8: Effect of Trim Parameter. *We show screened Poisson surface reconstructions [127] with trim parameters* 5 *and* 7 *for Colmap [254] and the prediction of our model trained with* $\mathcal{L}_{RGB} + \mathcal{L}_{Depth}$ *for scan* 106 *of the DTU dataset.*

6.4 Conclusion

This chapter presented Differentiable Volumetric Rendering (DVR). Observing that volumetric surface rendering is inherently differentiable for neural fields allows us to formulate an analytic expression for the gradients of the depth with respect to the network parameters. Our experiments showed that DVR enables us to learn neural field-based 3D shape and texture representations from multi-view imagery without 3D supervision, rivaling models that are learned with full 3D supervision. Moreover, we found that our model can also be used for multi-view 3D reconstruction.

6.4.1 Limitations and Future Work

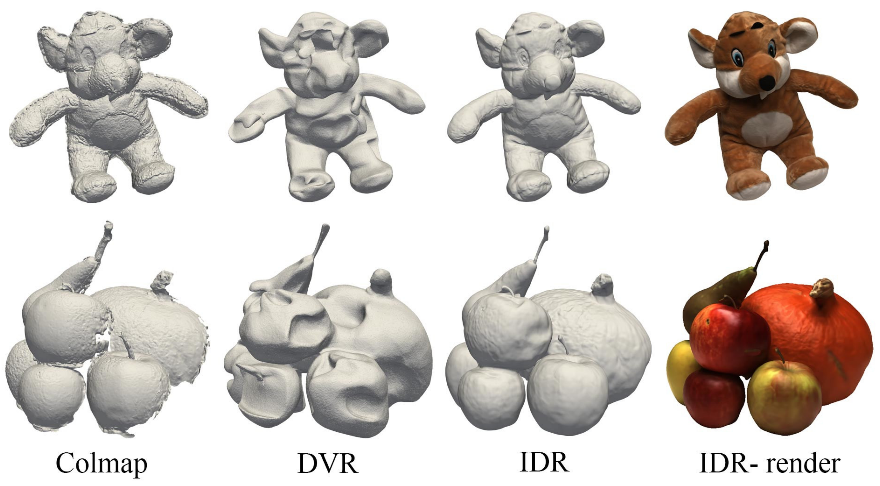


Figure 6.9: Failure Cases. *Our proposed DVR system cannot model view-dependent lighting effects which results in cavity-like artifacts in the reconstructions of scenes with specularities. This has been addressed in follow-up works like IDR [333] where the color prediction is conditioned on the viewing direction leading to better results.*

View Dependency The proposed method does not model view-dependent effects like specular highlights or reflections. This leads to inconsistent 3D geometry predictions like small cavities allowing the model to predict specular highlights (see Figure 6.9). In the follow-up work IDR [333], Yariv et al. addressed this by adding the viewing direction as input to the model for the color prediction. This leads to improved results and more accurate geometry predictions.

Mask Supervision We derived analytic formulations for the gradient of the depth wrt. the network parameters. For this formula to hold, we assume that a surface point is found along each ray, and we add additional losses for supervising which ray should be occupied or empty (see Section 6.2.4). In contrast, Mildenhall et al. [183] replace the single-surface representation with a density-based volume representation which is differentiable everywhere and hence does not require mask supervision. Recent works [205, 307, 334] combine both approaches to achieve 3D reconstruction methods which can be trained from multi-view images without masks.

Camera Poses We train our DVR model from either multi-view or single-view supervision with the respective camera poses. In most works, the camera poses are assumed to be given as either synthetic data is used or well-engineered SfM algorithms like COLMAP [254] can be employed as a preprocessing step to obtain camera poses. However, for more realistic scenarios like single images of different real-world scenes or sparse coverages of individual scenes, the feature matching of classical SfM methods breaks down leading to non-reliable pose estimation. To overcome this limitation, works are proposed that optimize a 3D scene representation and camera poses simultaneously from multi-view images [113, 152, 308]. While these approaches perform well for a restricted pose range (e.g., forward-facing cameras) or if initial poses are well selected, they are prone to get stuck in local minima due to the joint optimization via gradient descent. In contrast, GAN-based approaches have been used to learn 3D representations from unposed images of one [177] or many scenes [36, 200, 255], including scenes with 360° pose ranges. The system that we will discuss in Chapter 8 is a GAN-based generative model that does not require poses as input and that can be trained from raw, unposed image collections.

7 4D Reconstruction with Neural Fields

7.1 Introduction

In Chapter 5, we established that neural fields are well suited as 3D representation in learning-based systems and developed a method for inferring them from 2D supervision via differentiable surface rendering in Chapter 6. The models we discussed so far are only able to recover a static scene. Our goal is, however, to develop systems that can be employed in the real world where objects are not static, but in motion. As a result, in this chapter, we develop a neural field-based 3D shape and motion representation that allows for 4D reconstruction, 4D sequence interpolation, and 4D generative tasks.

Prior Work Most traditional works in the area of 4D reconstruction are restricted to a fixed domain by utilizing a template model [6, 15, 52, 109, 120, 287, 349], requiring a multi-view setup [145, 188, 189, 193, 208, 267, 290], or making strong assumptions about the motion ,e.g., rigidity or linearity [11, 176, 218, 291, 300]. For example, Mustafa et al. [188, 189] perform 4D reconstruction of dynamic scenes by utilizing multiple views. However, the method requires a sufficient number of wide-baseline views to cover the scene and is limited by ambiguities in these views. Wand et al. [300] present a carefully engineered technique to reconstruct deforming 3D geometry from point clouds. While producing compelling results, their method is restricted to spatiotemporal smooth and small movements, assumes temporally dense sampling of the point clouds, and is computationally costly. Another successful line of work utilizes template models to guide the reconstruction process [6, 60, 109, 120, 287, 349]. While providing a valuable framework for classical and learning-based models, by definition those results are restricted by the quality and availability of a template model and are extremely domain-specific. In addition, obtaining an adequate template is very costly, so most existing efforts focus on particular shape categories such as human bodies, hands, or faces [19, 161, 216, 224, 242].

More recently, learning-based approaches for recovering the 3D geometry from various forms of input have shown promising results [43, 47, 69, 92, 150, 179, 212, 241, 303]. In contrast to traditional methods, they leverage prior knowledge obtained during the training process to resolve ambiguities. In particular, recent neural field-based representations [43, 179, 212] achieve impressive results at limited memory costs. However, it remains unclear how to extend these approaches to the task of 4D reconstruction, i.e., reconstructing 3D shapes over time. Naïvely discretizing the temporal domain would lead to high memory cost and slow inference. Furthermore, it would neither provide implicit correspondences nor a physical description of the temporal evolution. While not only unsatisfactory from a scientific viewpoint, these problems also limit the use of existing 4D reconstruction techniques in applications where fast inference and reliable correspondences are desirable.

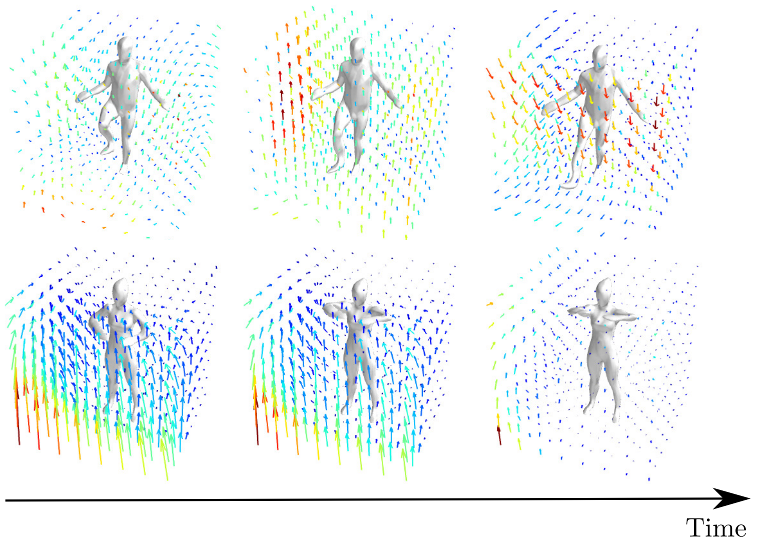
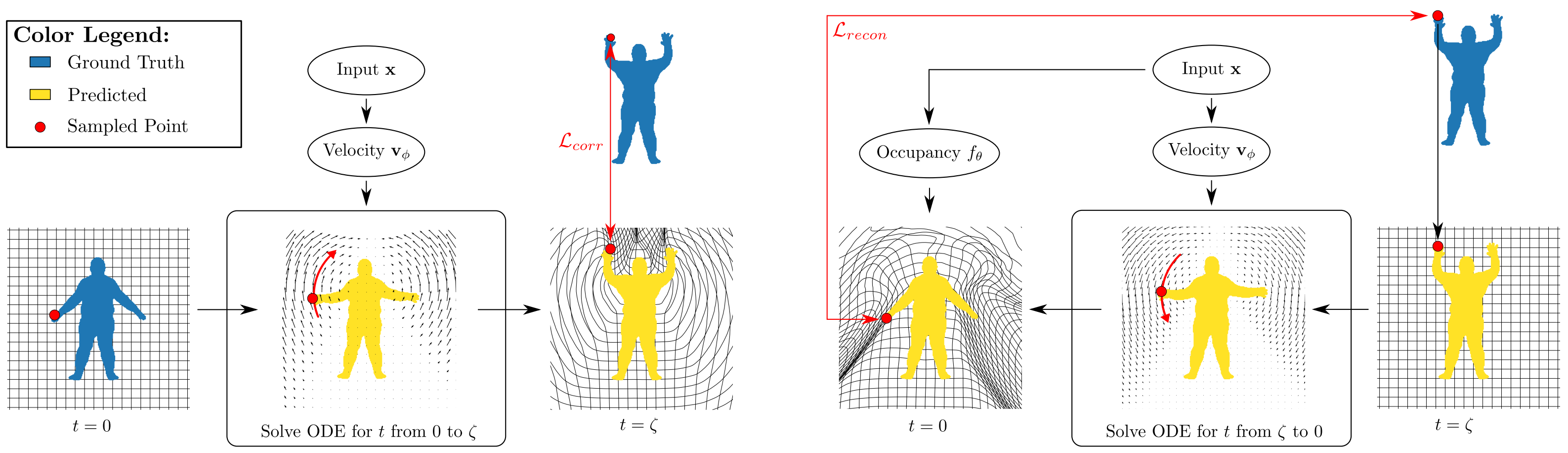


Figure 7.1: 4D Reconstruction with Neural Fields. *In Occupancy Flow, we represent time-varying 3D geometry by a temporally and spatially continuous vector field which assigns a motion vector to every point in space and time, thus implicitly capturing correspondences. We demonstrate that our representation can be used for 4D reconstruction from point cloud and image sequences as well as interpolation, shape matching, and generative tasks.*

Contribution In this chapter, we propose a novel continuous 4D representation (Figure 7.1) that implicitly models correspondences. More specifically, we parameterize a vector field with a neural network that assigns a 3D vector of motion to every 4D point in space and time. We combine this model with the previously discussed Occupancy Network (ONet) that represents shape continuously as the decision boundary of a binary classifier in 3D space. As every point in space is assigned an occupancy value as well as a continuous trajectory over time, we term our new representation *Occupancy Flow* (OFlow). Our representation is not only spatially and temporally continuous, but also *implicitly* provides correspondences at *every* point in space so that OFlow can be seen as the continuous generalization of scene flow [294, 295]. As a result, OFlow is not only suitable for reconstruction tasks, but also for a broader range of applications such as learning shape interpolations, finding correspondences between shapes, or learning probabilistic latent variable models. Furthermore, by modeling the temporal evolution of 3D shapes using continuum mechanics, our representation has a principled physical interpretation.

7.2 Method

In this section, we introduce our novel time-varying representation of 3D geometry which we term *Occupancy Flow* (OFlow). We start by formally introducing our model. Next, we explain how this representation can be learned from various types of input such as sequences



***(a)** Forward flow Φ_ϕ* ***(b)** Backward flow Ψ_ϕ*

Figure 7.2: Model Overview. *(a) During inference and to compute the correspondence loss $\mathcal{L}_{corr}$ defined in Equation 7.9, we propagate points on the ground truth mesh at $t = 0$ forward in time by integrating an input-dependent vector field $\mathbf{v}_\phi$. We obtain the correspondence loss $\mathcal{L}_{corr}$ by taking the ℓ_2-distance between the propagated points and the ground truth points on the mesh at $t = \zeta$. (b) To compute the reconstruction loss $\mathcal{L}_{recon}$, we go backward in time to transform a random point $\mathbf{p}$ into the coordinate system at $t = 0$. This allows us to compute the predicted occupancy probability $\hat{o}_{\theta,\phi}(\mathbf{p}, \zeta, \mathbf{z})$ by evaluating the Occupancy Network f_θ at $t = 0$ using Equation 7.7. The reconstruction loss is now given by taking the binary cross-entropy wrt. the ground truth occupancy at $t = \zeta$.*

of point clouds or images. Finally, the inference procedure, as well as implementation details, are provided. Figure 7.2 contains an overview of our method.

7.2.1 Occupancy Flow

We consider the challenging problem of estimating non-rigid 3D geometry jointly over space and time. More specifically, we are interested in inferring the evolution of a continuous 3D shape representation that implicitly and densely captures correspondences across time.

Velocity Field Let $\mathbf{s} : [0, T_{\text{end}}] \rightarrow \mathbb{R}^3$ define the continuous *3D trajectory* of a point over the time interval $[0, T_{\text{end}}]$ such that $\mathbf{s}(0) \in \mathbb{R}^3$ and $\mathbf{s}(T_{\text{end}}) \in \mathbb{R}^3$ denote the start and end locations of the trajectory. Let further $\mathbf{v} : \mathbb{R}^3 \times [0, T_{\text{end}}] \rightarrow \mathbb{R}^3$ denote the continuous *velocity field* which describes the 3D velocity at every point in space and time. The relationship between $\mathbf{s}(\cdot)$ and $\mathbf{v}(\cdot, \cdot)$ is governed by the following differential equation

$$\frac{\partial \mathbf{s}(t)}{\partial t} = \mathbf{v}(\mathbf{s}(t), t) \tag{7.1}$$

with $t \in [0, T_{\text{end}}]$.

Forward Flow When solving this ordinary differential equation (ODE) [277] for every initial condition $\mathbf{s}(0) = \mathbf{p}$ with $\mathbf{p} \in \mathbb{R}^3$ we obtain the *forward flow* $\Phi : \mathbb{R}^3 \times [0, T_{\text{end}}] \rightarrow \mathbb{R}^3$ (Figure 7.2a) satisfying:

$$\frac{\partial \Phi}{\partial t}(\mathbf{p}, t) = \mathbf{v}(\Phi(\mathbf{p}, t), t) \quad \text{s.t.} \quad \Phi(\mathbf{p}, 0) = \mathbf{p} \tag{7.2}$$

Intuitively, the flow $\Phi(\mathbf{p}, t)$ describes the location of initial point $\mathbf{p}$ at time t when following the vector field $\mathbf{v}(\cdot, \cdot)$. In order to propagate spatial information (e.g., volumetric occupancy or mesh vertices) forward in time, we can reformulate Equation 7.2 as follows

$$\Phi(\mathbf{p}, \zeta) = \mathbf{p} + \int_0^{\zeta} \mathbf{v}(\Phi(\mathbf{p}, t), t) \mathrm{d}t \tag{7.3}$$

where $\zeta \in [0, T_{\text{end}}]$ denotes an arbitrary point in time and $\mathbf{p}$ a spatial location in $\mathbb{R}^3$. This equation can be solved with standard numerical solvers such as Runge-Kutta [277]. We can also regard $\Phi(\cdot, \zeta)$ as a coordinate transformation that transforms a coordinate system at time $t = 0$ to a coordinate system at time $t = \zeta$. In the field of continuum mechanics [9], these coordinate systems are often referred to as "material coordinate system" and "spatial coordinate system", respectively.

Backward Flow We define the *backward flow* $\Psi : \mathbb{R}^3 \times [0, T_{\text{end}}] \rightarrow \mathbb{R}^3$ (Figure 7.2b) as the inverse transformation of Φ. This inverse transformation can be computed by solving the reverse ODE

$$\frac{\partial \mathbf{r}(t)}{\partial t} = -\mathbf{v}(\mathbf{r}(t), t) \quad \text{s.t.} \quad \mathbf{r}(\zeta) = \mathbf{p} \tag{7.4}$$

for every $(\mathbf{p}, \zeta) \in \mathbb{R}^3 \times [0, T_{\text{end}}]$ and setting $\Psi(\mathbf{p}, \zeta) = \mathbf{r}(0)$. As correspondences across time are implicitly captured, it is sufficient to represent the 3D shape in the coordinate system at time $t = 0$. The 3D shape at other points in time can then be obtained by propagation using Equation 7.3.

Shape Representation For representing the 3D shape at time $t = 0$ we choose the previously discussed occupancy function $f : \mathbb{R}^3 \to \{0, 1\}$ representation which assigns an occupancy value to every 3D point. In contrast to mesh- or point-based representations, occupancy functions allow for representing smooth shapes at arbitrary resolution and with arbitrary topology.

Parameterization We parameterize both the occupancy function $f(\cdot)$ as well as the velocity field $\mathbf{v}(\cdot, \cdot)$ using neural networks

$$f_\theta \quad : \quad \mathbb{R}^3 \times \mathcal{Z} \to [0, 1] \tag{7.5}$$

$$\mathbf{v}_\phi \quad : \quad \mathbb{R}^3 \times [0, T_{\text{end}}] \times \mathcal{Z} \to \mathbb{R}^3 \tag{7.6}$$

where θ and ϕ denote the network parameters, and $\mathcal{Z}$ the latent space for conditioning the networks on some latent code. In the following, we will refer to $f_\theta(\cdot, \cdot)$ as the *Occupancy Network* and $\mathbf{v}_\phi(\cdot, \cdot, \cdot)$ as the *Velocity Network*. We will now describe how the parameters of Equation 7.5 and Equation 7.6 can be learned from data.

7.2.2 Training

Our goal is to learn the parameters θ and ϕ of $f_\theta(\cdot, \cdot)$ and $\mathbf{v}_\phi(\cdot, \cdot, \cdot)$ using samples drawn from the 4D occupancy space-time volume, i.e., each sample represents the occupancy state at a particular point in space and time. Since we have chosen $t = 0$ as the reference coordinate system for representing the shape, each sample with $t > 0$ must be mapped back to its location at $t = 0$ in order to train the Occupancy and the Velocity Network. Towards this goal we use the backward flow $\Psi : \mathbb{R}^3 \times [0, T_{\text{end}}] \to \mathbb{R}^3$ described above (Figure 7.2b). The predicted occupancy $\hat{o}_{\theta,\phi}(\mathbf{p}, t, \mathbf{z})$ of 3D point $\mathbf{p}$ at time t for some latent code $\mathbf{z}$ is given by

$$\hat{o}_{\theta,\phi}(\mathbf{p}, t, \mathbf{z}) := f_\theta\left(\Psi_\phi(\mathbf{p}, t, \mathbf{z}), \mathbf{z}\right) \tag{7.7}$$

where we used the notation Ψ_ϕ to indicate that the inverse transformation depends on the parameters of the Velocity Network $\mathbf{v}_\phi(\cdot, \cdot, \cdot)$.

Reconstruction Loss The model can be trained by minimizing the binary cross-entropy error (BCE) between the predicted occupancy $\hat{o}$ and the observed occupancy o of 3D point $\mathbf{p}$ at time ζ:

$$\mathcal{L}_{recon}(\theta, \phi) = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{p}, \zeta, \mathbf{z}, o) \in \mathcal{B}} \text{BCE}(\hat{o}_{\theta,\phi}(\mathbf{p}, \zeta, \mathbf{z}), o) \tag{7.8}$$

Here, $\mathcal{B}$ denotes a mini-batch that comprises samples from multiple sequences and at multiple time instances ζ.

Correspondence Loss It is important to note that training our model does *not* require any correspondences across time. However, if available, additional correspondence information can be incorporated (Figure 7.2a) by propagating 3D points $\mathbf{p}$ from time $t = 0$ to time $t = \zeta$ using the forward flow $\Phi(\mathbf{p}, \zeta)$ in Equation 7.3. The correspondence loss function minimizes the ℓ_2 distance between the predicted location $\Phi_\phi(\mathbf{s}(0), \zeta, \mathbf{z})$ and the observed location $\mathbf{s}(\zeta)$ at time ζ for latent code $\mathbf{z}$ as follows

$$\mathcal{L}_{corr}(\phi) = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{s},\zeta,\mathbf{z})\in\mathcal{B}} \|\Phi_\phi(\mathbf{s}(0), \zeta, \mathbf{z}) - \mathbf{s}(\zeta)\|_2 \tag{7.9}$$

where $\mathbf{s}$ denotes the ground truth trajectory of a 3D point.

Gradient Computation The gradients of Equation 7.8 and Equation 7.9 can be efficiently obtained using the adjoint sensitivity method [40, 226] by solving a second augmented ODE backward in time. This way, the memory footprint can be kept constant with the tradeoff of longer computing time. For adaptive ODE solvers, relative and absolute error tolerances can be chosen to balance time and accuracy. For details, we refer the reader to [40].

7.2.3 Inference

Mesh Extraction For a new observation $\mathbf{z}$, we predict the time-varying 3D shape by first reconstructing the shape in the reference coordinate system at $t = 0$, followed by propagating the reconstruction into the future $t \in (0, T_{\text{end}}]$. While various shape representations could be employed with our method, we utilize the previously discussed Multiresolution IsoSurface Extraction (MISE) to extract a mesh $\mathcal{M}_0 = (\mathcal{P}_0, \mathcal{F}_0)$ from the prediction of the Occupancy Network f_θ at time $t = 0$. Here, $\mathcal{P}_0$ and $\mathcal{F}_0$ denote the vertices and the faces of mesh $\mathcal{M}_0$, respectively.

Mesh Propagation For later time steps t, we use the trained Velocity Network $\mathbf{v}_\phi$ in order to obtain the forward transformation $\Phi_\phi(\mathbf{p}_i, t, \mathbf{z})$ for all vertices $\mathbf{p}_i$ in $\mathcal{P}_0$ by solving Equation 7.3. The mesh at time t is given as:

$$\mathcal{M}_t = \left(\{\Phi_\phi(\mathbf{p}_i, t, \mathbf{z}) \,|\, \mathbf{p}_i \in \mathcal{P}_0\}, \mathcal{F}_0\right) \tag{7.10}$$

Note that the mesh has to be extracted only once during inference. Therefore, inference for a large number of time steps is significantly faster compared to the naïve solution which extracts a mesh independently at every time step. Moreover, we implicitly obtain temporal correspondences (i.e., the mesh vertices correspond across time) even when using only the reconstruction loss Equation 7.8 during training.

7.2.4 Implementation Details

For both the Occupancy Network and the Velocity Network we use a fully-connected ResNet-based [98] architecture shown in Figure 7.3. For conditioning the Occupancy Network f_θ

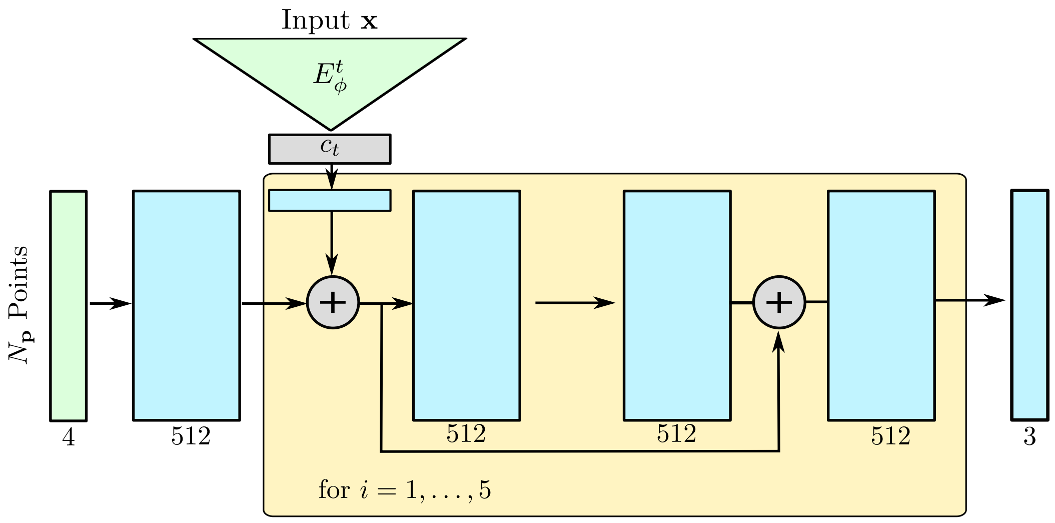


Figure 7.3: Velocity Network Architecture. *Green color indicates input, cyan fully connected layers, and gray other operations. The Occupancy Network architecture is similar except that the input point dimension is* 3 *(no temporal axis), the outputs are occupancy probabilities of dimension* 1*, and conditional batch normalization [66, 298] is used instead of the adding operation for conditioning on input* $\mathbf{z}$*.*

and the Velocity Network $\mathbf{v}_\phi$ on a sequence of observations $\mathbf{z} = (\mathbf{z}_i)_{i=1,\ldots,N_s}$ with length N_s, we use two separate encoder networks $E^s_\theta(\mathbf{z}_1)$ and $E^t_\phi(\mathbf{z})$, where the *spatial encoder* $E^s_\theta(\mathbf{z}_1)$ is only applied to the first observation $\mathbf{z}_1$ and the *temporal encoder* $E^t_\phi(\mathbf{z})$ is applied to the whole sequence of N_s observations $\mathbf{z}$. The input $\mathbf{z}$ could for example be a sequence of images where $\mathbf{z}_i$ indicates the i-th image of this sequence. While we use the output of the spatial encoder to condition the Occupancy Network f_θ on $\mathbf{z}$, we use the output of the temporal encoder to condition the Velocity Network $\mathbf{v}_\phi$ on $\mathbf{z}$. Depending on whether we use a sequence of point clouds or a sequence of images as input, we use a PointNet [230] or a Resnet-18 [98] for the spatial encoder E^s_θ. For the temporal encoder E^t_ϕ, we use an adjusted PointNet architecture with input dimension $3 \times L$ and a 3D convolutional network for point cloud and image input, respectively. We use the Adam optimizer [129] with a learning rate of 10^{-4} and train with batch size 16.

7.3 Experiments

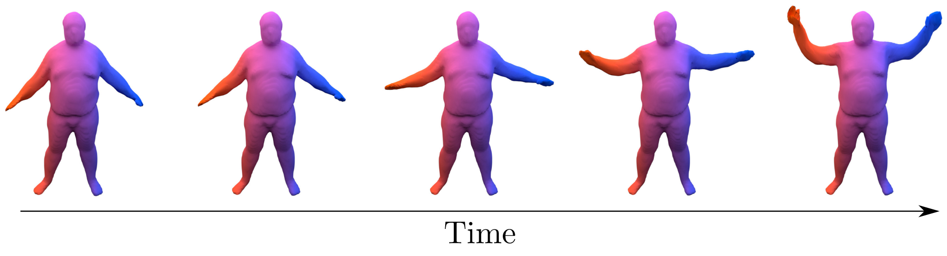
We conduct four different types of experiments to investigate the effectiveness of our approach. First, we evaluate the **representation power** of our vector field-based representation by training it to reproduce complex 3D motions. We further investigate the **reconstruction capacity** of our representation by conditioning the network on a sequence of images or noisy point clouds. We then investigate the quality of the learned **interpolations and correspondences** between two meshes or point clouds, respectively. Finally, we examine its **generative capabilities** by training a variational autoencoder [129] and investigating the quality of the latent representation.

Baselines A natural baseline for 4D reconstruction from image sequences or point clouds is to extend Occupancy Networks (ONet) to the temporal domain by sampling points in 4D space. Similar to our method, this ONet 4D is continuous in time and space and can hence represent complex motions of 3D objects with arbitrary topology. However, in contrast to our representation, extracting meshes from this ONet 4D is time-consuming (as mesh extraction is done at every frame) and does not yield correspondences across time. As an additional baseline, we implement a 4D extension of Point Set Generation Network (PSGN) [69] by predicting a set of trajectories instead of single points. For a fair comparison, we train this PSGN 4D both with and without temporal correspondences. For the former case, we evaluate the Chamfer-loss independently per time step. For the latter case, we introduce a generalization of the Chamfer-loss which considers entire trajectories of points instead of independent 3D locations at each point in time. In the shape matching and interpolation experiment, we compare against nearest neighbor matching, Coherent Point Drift (CPD) [190], and 3D-Coded [91], a state-of-the-art method for finding correspondences between human shapes.

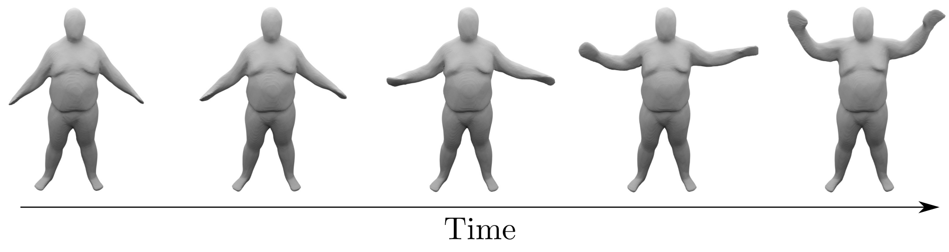
Datasets We use the Dynamic FAUST (D-FAUST) [24] dataset which contains scans and meshes for 129 sequences of 10 real humans performing various motions such as “punching”, “chicken wings”, or “jumping jacks”. D-FAUST is very challenging not only due to the fine structure of the human body, but also its non-rigid complex movements which include soft-tissue motion. As each sequence is relatively long (up to 1.2k steps) and to increase the size of the dataset, we subsample each sequence into smaller clips of 17 to 50 time steps, depending on the experiment. We randomly divide all sequences into training (105), validation (6), and test (9) sequences so that the models are evaluated on combinations of individuals and motions not seen during training. In addition, we withhold one individual (12 sequences) to test generalization capabilities across individuals.

Due to the lack of publicly available datasets of time-varying non-rigid 3D geometry, we further introduce *Warping Cars*, a synthetic dataset of large-scale deformations of cars. It allows examining how well our method performs on other types of deforming objects than humans. To this end, we utilize the ShapeNet [34] “car” category and apply random displacement fields to obtain a continuous warping motion.

Metrics We use volumetric IoU and Chamfer distance for evaluating the reconstruction at each time step. We refer to [179] for an in-depth description of these metrics. For evaluating the quality of the estimated correspondences, we introduce a correspondence distance as follows: The K points $\mathbf{p}^{(k)}(0)$, $k \in \{1, \dots, K\}$, of the output at $t = 0$ are assigned to the nearest neighbor $\mathbf{p}_{GT}^{(k)}(0)$ on the ground truth mesh. We then find the point $\mathbf{p}_{GT}^{(k)}(\zeta)$ corresponding to $\mathbf{p}_{GT}^{(k)}(0)$ on the ground truth mesh at $t = \zeta$. Similarly, we find the point $\mathbf{p}^{(i)}(\zeta)$ corresponding to $\mathbf{p}^{(k)}(0)$ in the output of the method. The correspondence ℓ_2-distance at time $t = \zeta$ is then defined as the average ℓ_2-distance between the points $\mathbf{p}^{(k)}(\zeta)$ and $\mathbf{p}_{GT}^{(k)}(\zeta)$. Note that this distance can only be computed for methods like ours that yield



(a) Occupancy Flow



(b) 4D Occupancy Network

	IoU	Chamfer	Time (s)	Time w/o MC (s)
ONet 4D	**94.6 %**	**0.028**	15.509	5.802
OFlow	93.4 %	0.031	**0.716**	**0.520**

(c) Reconstruction Accuracy and Runtime

Figure 7.4: Representation Power. *Correspondences are shown with the same color. While both, ONet 4D and OFlow, successfully learn to represent the complex 3D motion, only OFlow yields correspondences over time which also results in faster inference. We show inference times for all* 50 *time steps with and without marching cubes (MC).*

correspondences across time, but not ONet 4D. Similar to [69, 179] we use $1/10$ times the length of the maximal edge length of the object's bounding box as unit 1.

7.3.1 Representation Power

In this experiment, we investigate how well our Occupancy Flow model can represent 3D shapes in motion. In particular, we would like to disentangle the influence of the spatial and temporal encoders E_θ^s and E_ϕ^t from the representation power of the Occupancy Flow model. Towards this goal, we train our networks to reconstruct complex 3D motions without any external input $\mathbf{x}$. For training, we select 3 sequences of length 50 from the training split of the D-FAUST dataset on which we (separately) train our networks only using the $\mathcal{L}_{recon}$ loss in Equation 7.8. We compare against ONet 4D.

Results The results of this experiment are shown in Figure 7.4. We see that our method learns an accurate representation of the deforming 3D geometry, yielding similar IOU and Chamfer values as ONet 4D. However, in contrast to ONet 4D, we only have to extract

	IoU	Chamfer	Correspond.
PSGN 4D	-	0.108	3.234
PSGN 4D (w/ cor.)	-	0.101	0.102
ONet 4D	77.9 %	0.084	-
OFlow	79.9 %	0.073	0.122
OFlow (w/ cor.)	**81.5 %**	**0.065**	**0.094**

***(a)** Seen individuals*

	IoU	Chamfer	Correspond.
PSGN 4D	-	0.127	3.041
PSGN 4D (w/ cor.)	-	0.119	0.131
ONet 4D	66.6 %	0.140	-
OFlow	69.6 %	0.095	0.149
OFlow (w/ cor.)	**72.3 %**	**0.084**	**0.117**

***(b)** Unseen individual*

***Table 7.1: 4D Point Cloud Completion (D-FAUST).** These tables show quantitative results for the 4D point cloud completion experiment on the D-FAUST dataset. We report volumetric IoU (higher is better), Chamfer distance (lower is better) and the correspondence ℓ_2-distance (lower is better) for both individuals seen during training and the unseen individual.*

	IoU	Chamfer	Correspond.
PSGN 4D	-	**0.157**	3.886
ONet 4D	69.7 %	0.190	-
OFlow	**70.7 %**	0.169	**0.283**

***Table 7.2: 4D Point Cloud Completion (Warping Cars).** This table shows quantitative results for the 4D point cloud completion experiment on the warping cars dataset.*

a mesh once for $t = 0$ whose vertices we then propagate forward in time by solving a time-dependent ODE, leading to much faster inference. Moreover, while both ONet 4D and our approach successfully learn to represent the complex 3D motion, only our approach yields correspondences over time.

7.3.2 4D Point Cloud Completion

In this first reconstruction experiment, the input for the network is 300 discrete point trajectories, each consisting of $N_s = 17$ time steps. We perturb the point clouds with Gaussian noise with a standard deviation of 0.01. A real-world scenario for this would for example be (noisy) motion capture data from a set of markers.

We train our method using the reconstruction loss $\mathcal{L}_{recon}$ in Equation 7.8, which does not

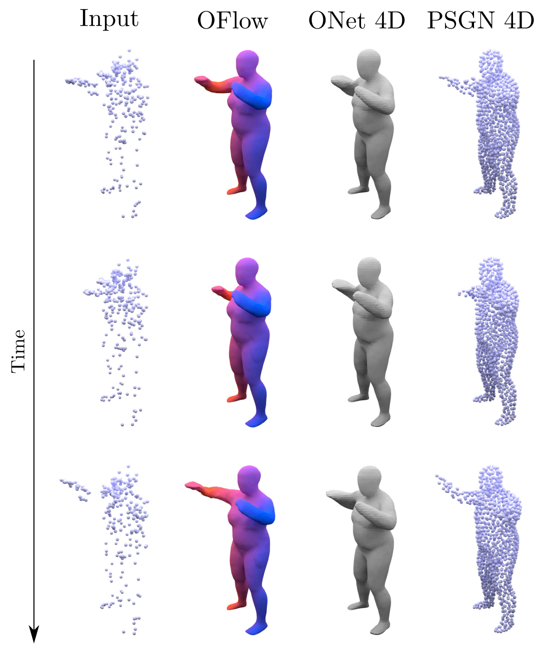


Figure 7.5: 4D Point Cloud Completion. *We show three equally spaced time steps between 0 and 1 for the input and the output of OFlow (w/ correspond.), ONet 4D, and PSGN 4D (w/ correspond.). The color coding for the first method illustrates correspondences across time.*

utilize any correspondences. Moreover, we also investigate the performance of our method when trained with both the reconstruction loss $\mathcal{L}_{recon}$ and the correspondence-based loss $\mathcal{L}_{corr}$ in Equation 7.9.

We compare against ONet 4D and PSGN 4D. For a fair comparison, we train all methods with the same ResNet-based [98] PointNet [230] temporal encoder from Section 7.2.4. We do not use an additional spatial encoder for ONet 4D and PSGN 4D as both methods do not represent shape and motion disentangled.

Results The quantitative and qualitative results for the D-FAUST dataset are summarized in Table 7.1 and Figure 7.5. We observe that OFlow outperforms ONet 4D in terms of IOU and achieves the lowest Chamfer distance compared to both PSGN variants and ONet 4D. This is surprising, as PSGN was explicitly trained on the Chamfer distance whereas OFlow was not. OFlow trained with both losses achieves the lowest correspondence ℓ_2-distance. Interestingly, OFlow trained only with the reconstruction loss achieves an only slightly worse correspondence loss even though it did not use any correspondences during training. In contrast, the PSGN variant that does not use any correspondences during training does not learn meaningful correspondences. This shows that our vector field representation is helpful for learning correspondences over time. Qualitatively (Figure 7.5), we observe that OFlow learns a realistic 3D motion while ONet 4D does not. PSGN is also able to reconstruct the 3D motion but lacks spatial connectivity. Quantitative results for the Warping Cars dataset are shown in Table 7.2. We see that OFlow also works well in a very different domain and

	IoU	Chamfer	Correspond.
PSGN 4D	-	0.258	2.576
PSGN 4D (w/ cor.)	-	0.265	2.580
ONet 4D	44.0 %	0.348	-
OFlow	56.6 %	0.193	0.292
OFlow (w/ cor.)	**59.6 %**	**0.166**	**0.226**

***(a)** D-FAUST*

	IoU	Chamfer	Correspond.
PSGN 4D	-	**0.251**	3.949
ONet 4D	55.6 %	0.319	-
OFlow	**58.2 %**	0.277	0.491
OFlow (w/ cor.)	58.0 %	0.263	**0.487**

***(b)** Warping cars*

Table 7.3: Image-based 4D Reconstruction. *The two tables summarize the quantitative results for 4D reconstruction from image sequences.*

achieves the best IoU and correspondence ℓ_2-distance.

7.3.3 Reconstruction from Image Sequences

In this experiment, we consider 4D reconstruction from a sequence of single-view images as observation $\mathbf{x}$. For all methods, we use the temporal encoder architecture described in Section 7.2.4.

Results In Table 7.3 and Figure 7.6 we provide a summary of the quantitative and qualitative results. Similar to [179] and others, we observe that reconstruction from single-view image sequences is a harder task than 4D point cloud completion. We suspect the global image encoding as well as occlusions to be the main challenge as the viewpoint is sampled randomly for the clips which sometimes causes the motion to be invisible in the images. The quantitative performance differences are similar to the point cloud experiment. The qualitative results in Figure 7.6 show that while OFlow can reconstruct the complicated 3D motion from the provided sequence reasonably well, the other methods struggle to do so. It suggests that the disentangled shape and motion representation of OFlow results in better reconstructions and biases the network towards a physically plausible motion.

7.3.4 Interpolation and Mesh Correspondence

The goal of the next two experiments is to investigate to which degree our method can be used for shape matching and interpolation. In both experiments, the task is to find a continuous transformation between the underlying surfaces of two randomly sampled point

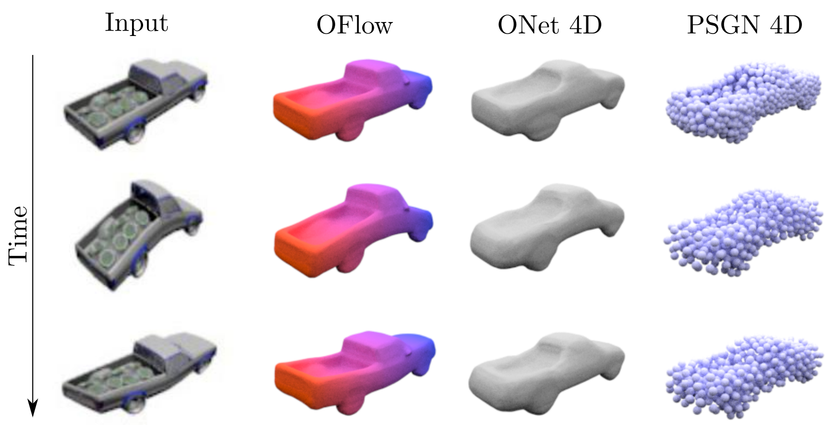


Figure 7.6: Image-based 4D Reconstruction. *We show three time steps between 0 and 1 for input as well as the output of OFlow, ONet 4D and PSGN 4D. Similar to Figure 7.5, the color coding illustrates the correspondences.*

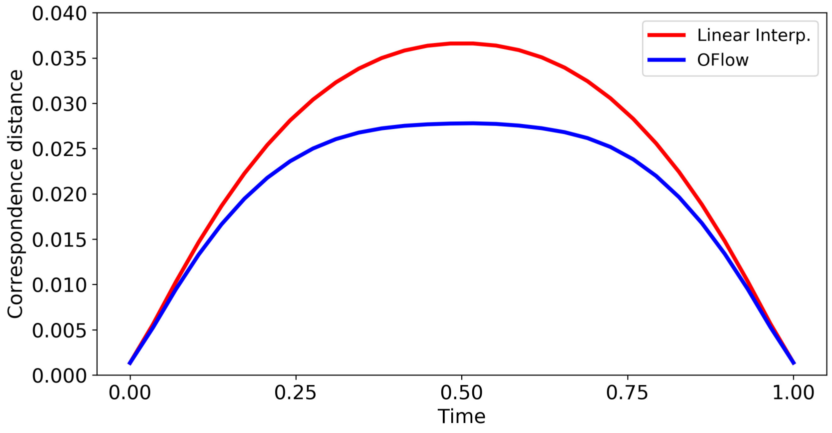
	Correspond	Time (s)
Baseline NN	0.374	**0.004**
Coherent Point Drift [190]	0.189	343.621
OFlow	0.167	0.608
3D-Coded [91]	**0.096**	199.368

Table 7.4: Shape Matching. *This table shows results for shape matching from point clouds on the D-FAUST dataset.*

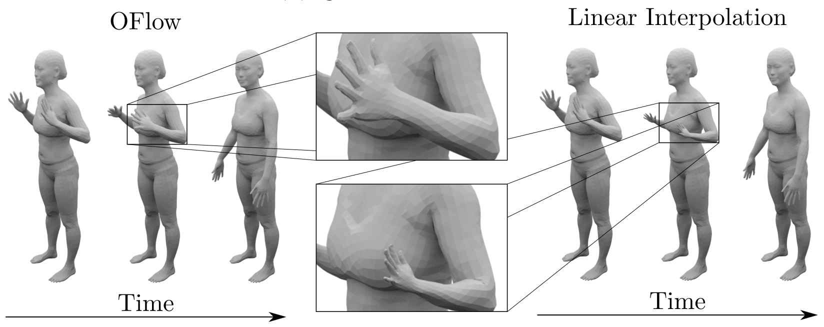
clouds. We train our model only using the correspondence loss Equation 7.9 as recovering the 3D shape is not required in this setting.

We first evaluate the quality of the correspondences learned by our method. We use the same splits on the D-FAUST dataset as before. We compare against nearest neighbor matching, non-rigid Coherent Point Drift [190] (CPD), and the specialized state-of-the-art learning-based method 3D-Coded [91]. While the first two find nearest neighbors or an optimal fit of GMM centroids in the second point cloud, the latter learns mappings to a human template model. For nearest neighbor matching, OFlow and 3D-Coded [91], we use two randomly sampled point clouds of size $10,000$ as input. As Coherent Point Drift [190] directly matches the point sets, we did not obtain competitive results for this method by using random point clouds so that we used the full set of vertices in this case. To adhere to community standards [23] we project predicted points that do not lie on the surface onto the final mesh for evaluation.

Results Our results are shown in Table 7.4. Even though our method is primarily concerned with 4D reconstruction, we find that it also estimates high-quality correspondences,



***(a)** Quantitative Results.*



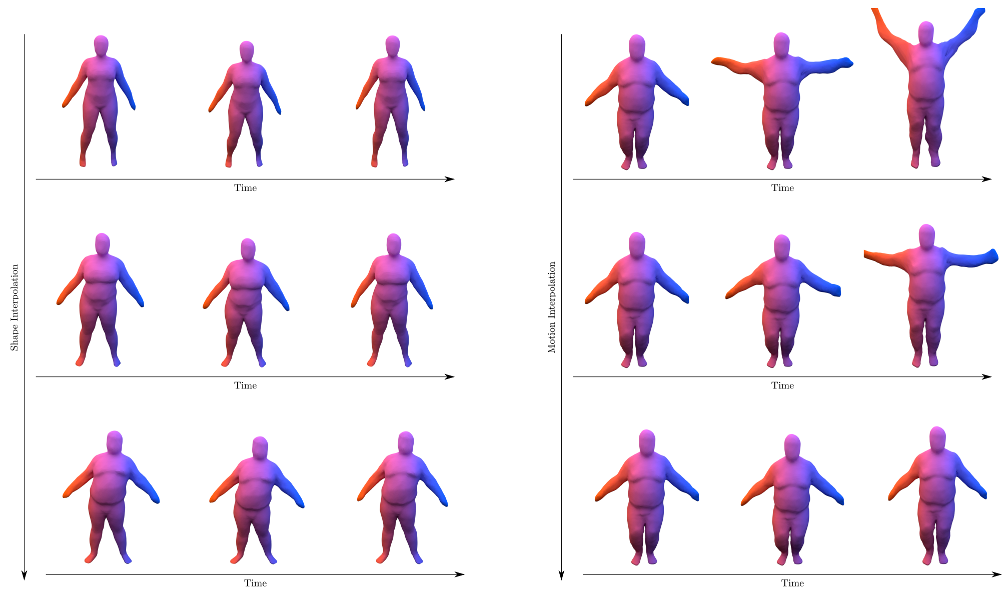
***(b)** Qualitative Results.*

***Figure 7.7: Interpolation.** The figure shows a quantitative and qualitative comparison of Occupancy Flow and the linear interpolation baseline. Occupancy Flow is able to better capture the non-linear motion of non-rigid 3D shapes.*

outperforming both the nearest neighbor as well as the CPD baselines. While it performs worse than 3D-Coded, OFlow requires only a fraction of its inference time. Moreover, we remark that 3D-Coded is a highly specialized matching method including costly fine-tuning for every registration whereas our approach is a general-purpose 4D reconstruction method that estimates correspondences implicitly.

To evaluate the interpolation capabilities of OFlow, we increase the sequence length L from 17 to 30 and compare against the linear interpolation baseline. For OFlow, we predict the forward and backward motion and average the results. For both methods, we evaluate the correspondence ℓ_2-distance for all 30 time steps.

Quantitative and qualitative results are shown in Figure 7.7. We observe that OFlow improves over the linear interpolation baseline as it is able to capture non-linear motion.



***(a)** Shape Interpolation with Fixed Motion.* ***(b)** Motion Interpolation with Fixed Shape.*

***Figure 7.8: Latent Space Interpolations.** In Figure 7.8a we show three equally spaced steps of a latent shape interpolation with fixed motion. Similarly, in Figure 7.8b we show three equally spaced steps of a latent motion interpolation with a fixed shape. The figures show that OFlow is able to learn a meaningful latent representation of both the shape and the motion.*

7.3.5 Generative Modeling

We further conducted experiments investigating the generative capabilities of our representation. For this, we adjust the spatial and temporal encoders E_θ^s and E_ϕ^t for 4D point cloud completion to predict means and log standard deviations $(\mu_s, \log\sigma_s)$ and $(\mu_t, \log\sigma_t)$ of Gaussian distributions $q_\theta^s(\mathbf{z}|\mathbf{x})$ and $q_\phi^t(\mathbf{z}|\mathbf{x})$ instead of latent codes $\mathbf{z}_s$ and $\mathbf{z}_t$, similar to Chapter 5. We then obtain the spatial and temporal latent codes for conditioning the Occupancy and Velocity Networks by sampling from the resulting distributions q_θ^s and q_ϕ^t. In contrast to having only one latent space, this decoupling of shape and motion allows us to sample spatial and temporal latent codes individually. As discussed in Section 2.2, we combine the reconstruction loss with the KL-divergence between the predicted and prior distributions to optimize the ELBO.

Results In Figure 7.8 and 7.9, we show results for two example tasks of our generative model: latent space interpolations and motion transfer. For the latter, we encode the shape and motion of a sequence into latent codes, and can then apply the encoded motion to a given shape. The results show that OFlow is able to learn a smooth and meaningful latent space representation. The decoupling of shape and motion allows us to encode and sample the two components individually. This flexibility enables OFlow to be used in various tasks

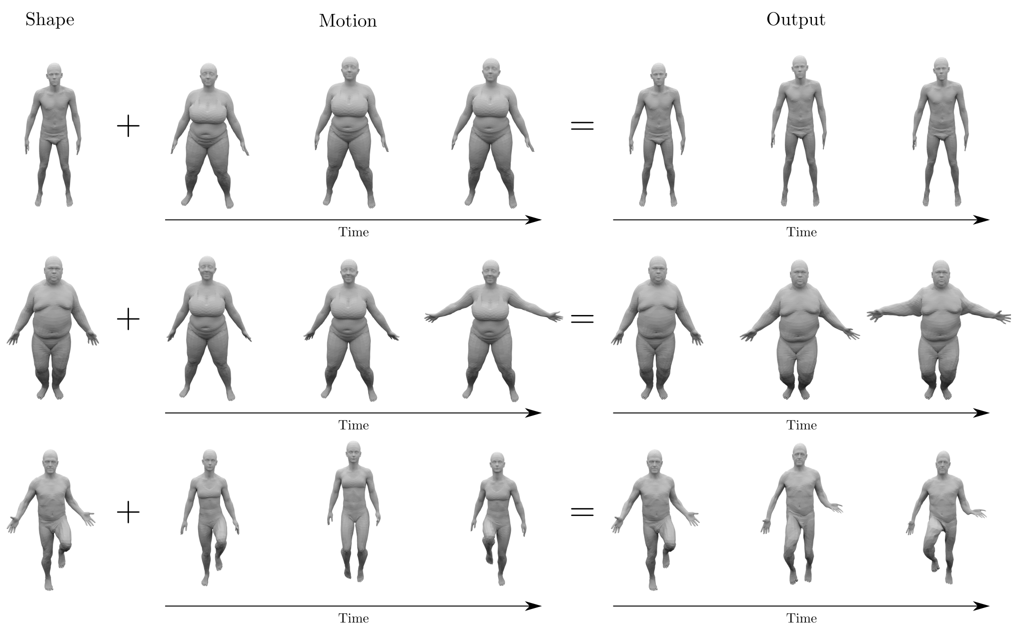


Figure 7.9: Motion Transfer. *We show three examples of the motion transfer experiment. We take a start shape (first column) and encode the motion from another sequence (second column) to transfer this motion to the shape (third column). We see that OFlow is able to transfer the motion to another shape reasonably well despite changes in topology and pose.*

ranging from shape or motion interpolation to motion transfer.

7.4 Conclusion

In this chapter, we introduced Occupancy Flow, a novel neural field-based 4D representation of time-changing 3D geometry. In contrast to existing 4D representations, it does not utilize a template model, is continuous in space and time, and yields implicit temporal correspondences. Our experiments validate that it can be used effectively for shape matching and interpolation, 4D reconstruction, and generative tasks.

7.4.1 Limitations and Future Work

Supervision Data Similar to how we identified requiring 3D supervision data as a key limitation of Occupancy Networks in Chapter 5 and developed a differentiable surface rendering technique to overcome this in Chapter 6, the same limitations exist for Occupancy Flow. As a result, follow-up works proposed methods that allow for inferring neural field-based 4D representations from multi-view images [65, 75, 149, 213, 214, 220, 228] where most methods adapt our approach to split the task into 3D scene reconstruction and motion reconstruction. However, many works fit network weights to single dynamic scenes and require long optimization times. As a result, we identify developing systems that can infer 4D representations from sparse inputs with acceptable memory, computing, and time complexity as a promising research direction.

Disentangled Representations We represent dynamic scenes as static scenes together with their motion. However, most real-world scenes consist of static and dynamic elements, and the natural decomposition of a scene would be to divide it into its individual elements. This has been addressed in [342], where a self-supervised method is proposed to discover static and dynamic components of the scene automatically. While the model proposed in [342] can only handle rigid motion, NeRFPlayer [265] automatically decomposes the scene into static, dynamic, and new content leading to improved view synthesis and composition results.

Modeling Articulated Objects While our proposed OFlow system is a generic 4D representation that does not make strong assumptions about the represented shape or motion type, improvements in controllability and reconstruction quality can be expected if the shape or motion space is restricted to specific classes. In particular representing articulated objects, i.e., objects that are composed of multiple rigid parts connected by joints that allow for rotation or translation [288], yields more controllability. As a result, models are proposed that combine neural field-based shape representations and articulated human [114, 182, 202, 248, 270, 283, 309], animal [330], or general shape models [185]. For articulated human shape modeling, in particular works that represent motion using linear blend skinning weights [45, 61, 182, 248] based on the SMPL body model [161] achieve high-quality results.

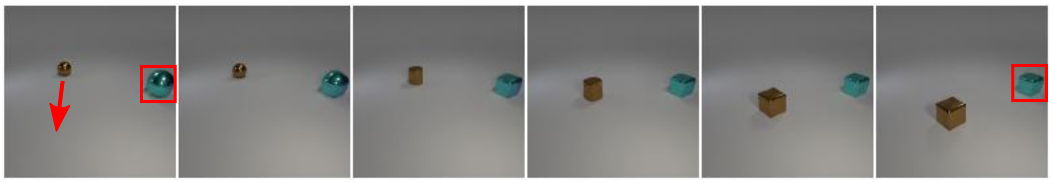
8 Compositional 3D-Aware Generative Modeling with Neural Fields

8.1 Introduction

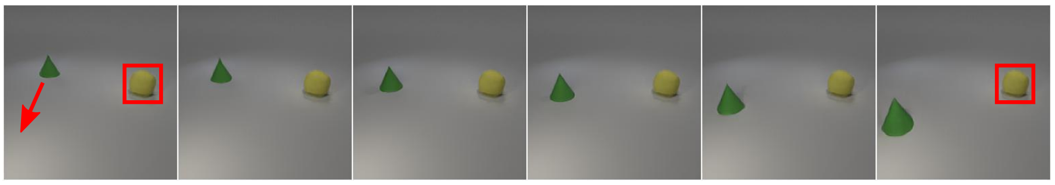
In previous chapters, we investigated how neural fields can be used to represent 3D geometry, 3D texture, and 4D shapes in motion. For training the developed systems, we used either 3D supervision or posed 2D images. In this chapter, we go one step further to truly unsupervised learning: training from raw, unposed image collections only. More specifically, we develop a generative system of 3D scenes that can be trained from collections of images without any annotation such as camera pose, object masks, etc. Further, we incorporate compositionality into the model such that at test time, not only new 3D scenes can be generated and rendered from different viewpoints, but also individual objects can be controlled wrt. their shape, position, and appearance.

Prior Work The computer vision community has made great strides towards highly-realistic image generation. In particular, Generative Adversarial Networks (GANs) [85] emerged as a powerful class of generative models. They are able to synthesize photorealistic images at resolutions of 1024^2 pixels and beyond [29, 48, 49, 122, 123]. Despite these successes, synthesizing realistic 2D images is not the only aspect required in applications of generative models. The generation process should also be controllable in a simple and consistent manner. To this end, many works [39, 87, 122, 140, 144, 157, 159, 217, 237, 348, 351] investigate how disentangled representations can be learned from data without explicit supervision. Definitions of disentanglement vary [13, 158], but commonly refer to being able to control an attribute of interest, e.g., object shape, size, or pose, without changing other attributes. Most approaches, however, do not consider the compositional nature of scenes and operate in the 2D domain, ignoring that our world is three-dimensional. This often leads to entangled representations (see Figure 8.1) and control mechanisms are not built-in, but need to be discovered in the latent space a posteriori. These properties, however, are crucial for successful applications, e.g., a movie production where complex object trajectories need to be generated in a consistent manner.

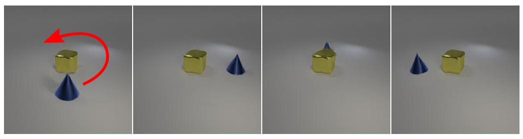
As a result, several works investigate how 3D representations can be incorporated as inductive bias into generative models [74, 101, 102, 103, 104, 151, 164, 195, 196, 239, 255]. While many approaches use additional supervision [5, 44, 301, 316, 353], we focus on works that are trained on raw image collections like our approach. Henzler et al. [102] learn voxel-based representations using differentiable rendering. The results are 3D controllable but show artifacts due to the limited voxel resolutions caused by their cubic memory growth. Nguyen-Phuoc et al. [195, 196] propose voxelized feature-grid representations which are



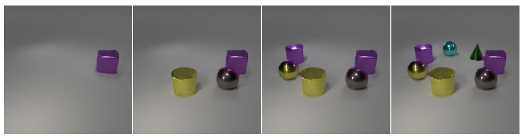
(a) *Translation of Left Object (2D-based Method [217])*



(b) *Translation of Left Object (Ours)*



(c) *Circular Translation (Ours)*



(d) *Add Objects (Ours)*

Figure 8.1: Controllable Image Generation. *While most generative models operate in 2D, we incorporate a compositional 3D scene representation into the generative model. This leads to more consistent image synthesis results, e.g., note how, in contrast to our method, translating one object might change the other when operating in 2D (Fig. 8.1a and 8.1b). It further allows us to perform complex operations like circular translations (Figure 8.1c) or adding more objects at test time (Figure 8.1d). Both methods are trained unsupervised on raw unposed image collections of two-object scenes.*

rendered to 2D via a reshaping operation. While achieving impressive results, training becomes less stable and results less consistent for higher resolutions. Liao et al. [151] use abstract features in combination with primitives and differentiable rendering. While handling multi-object scenes, they require additional supervision in the form of pure background images which are hard to obtain for real-world scenes. Schwarz et al. [255] propose Generative Neural Radiances Fields (GRAF). While achieving controllable image synthesis at high resolutions, this representation is restricted to single-object scenes and results degrade on more complex, real-world imagery.

Contribution In this chapter, we introduce *GIRAFFE*, a novel method for generating scenes in a controllable and photorealistic manner while training from raw unstructured image collections. Our key insight is twofold: First, incorporating a compositional 3D scene representation directly into the generative model leads to more controllable image synthesis. Second, combining this explicit 3D representation with a neural rendering pipeline results in faster inference and more realistic images. To this end, we represent scenes as Compositional Generative Neural Feature Fields (Figure 8.2). We volume render the scene to a feature image of relatively low resolution to save time and computation. A neural renderer processes

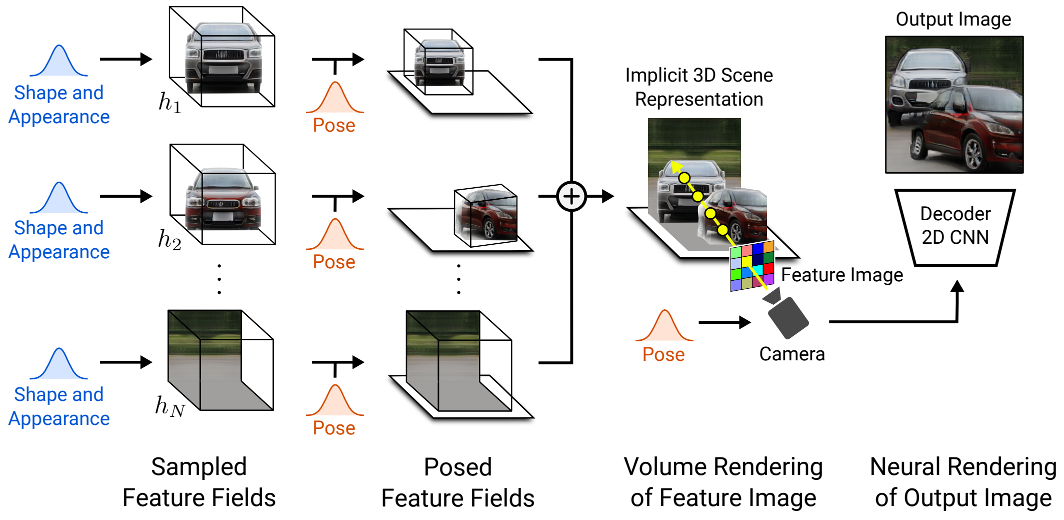


Figure 8.2: Compositional and 3D-Aware Generative Modeling with Neural Fields. *We represent scenes as Compositional Generative Neural Feature Fields (GIRAFFE). For a randomly sampled camera, we volume render a feature image of the scene based on individual Feature Fields. A 2D neural rendering network converts the feature image into an RGB image. While training only on raw image collections, at test time we are able to control the image formation process wrt. camera pose, object poses, as well as the objects' shapes and appearances. Further, our model generalizes beyond the training data, e.g., we can synthesize scenes with more objects than were present in the training images. Note that for clarity we visualize volumes in color instead of features.*

these feature images and outputs the final renderings. This way, our approach achieves high-quality images and scales to real-world scenes. We find that our method allows for controllable image synthesis of single-object as well as multi-object scenes when trained on raw unstructured image collections.

8.2 Method

Our goal is a controllable image synthesis pipeline that can be trained from raw image collections without additional supervision. In the following, we discuss the main components of our method. First, we model individual objects as neural Feature Fields (Section 8.2.1). Next, we exploit the additive property of Feature Fields to composite scenes from multiple individual objects (Section 8.2.2). For rendering, we explore an efficient combination of volume and neural rendering techniques (Section 8.2.3). Finally, we discuss how we train our model from raw image collections (Section 8.2.4). Figure 8.3 contains an overview of our method.

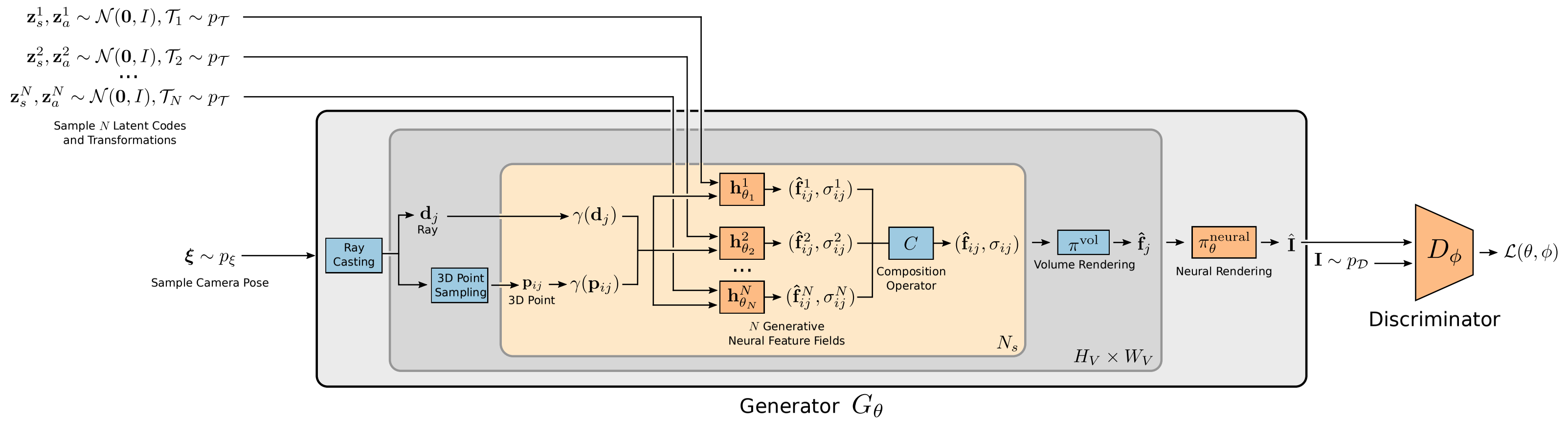

$\mathbf{z}_s^1, \mathbf{z}_a^1 \sim \mathcal{N}(\mathbf{0}, I), \mathcal{T}_1 \sim p_{\mathcal{T}}$
$\mathbf{z}_s^2, \mathbf{z}_a^2 \sim \mathcal{N}(\mathbf{0}, I), \mathcal{T}_2 \sim p_{\mathcal{T}}$
...
$\mathbf{z}_s^N, \mathbf{z}_a^N \sim \mathcal{N}(\mathbf{0}, I), \mathcal{T}_N \sim p_{\mathcal{T}}$
Sample N Latent Codes and Transformations
$\boldsymbol{\xi} \sim p_{\xi}$
Sample Camera Pose
Ray Casting
$\mathbf{d}_j$
Ray
3D Point Sampling
$\mathbf{p}_{ij}$
3D Point
$\gamma(\mathbf{d}_j)$
$\gamma(\mathbf{p}_{ij})$
$\mathbf{h}_{\theta_1}^1$
$\mathbf{h}_{\theta_2}^2$
$\mathbf{h}_{\theta_N}^N$
$(\hat{\mathbf{f}}_{ij}^1, \sigma_{ij}^1)$
$(\hat{\mathbf{f}}_{ij}^2, \sigma_{ij}^2)$
$(\hat{\mathbf{f}}_{ij}^N, \sigma_{ij}^N)$
N Generative Neural Feature Fields
C
Composition Operator
$(\hat{\mathbf{f}}_{ij}, \sigma_{ij})$
N_s
π^{vol}
Volume Rendering
$\hat{\mathbf{f}}_j$
$H_V \times W_V$
$\pi_\theta^{\text{neural}}$
Neural Rendering
$\hat{\mathbf{I}}$
$\mathbf{I} \sim p_{\mathcal{D}}$
D_ϕ
$\mathcal{L}(\theta, \phi)$
Discriminator
Generator G_θ

Figure 8.3:* *GIRAFFE. *Our generator G_θ takes a camera pose ξ and N shape and appearance codes $\mathbf{z}_s^i, \mathbf{z}_a^i$ and affine transformations $\mathcal{T}_i$ as input and synthesizes an image of the generated scene which consists of $N-1$ objects and a background. The discriminator D_ϕ takes the generated image $\hat{\mathbf{I}}$ and the real image $\mathbf{I}$ as input and our full model is trained with an adversarial loss. At test time, we can control the camera pose, the shape and appearance codes of the objects, and the objects' poses in the scene. Orange indicates learnable and blue non-learnable operations.*

8.2.1 Objects as Neural Feature Fields

Neural Radiance Fields A radiance field is a continuous function which maps a 3D point $\mathbf{p} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{S}^2$ to a volume density $\sigma \in [0, \infty)$ and an RGB color value $\mathbf{c} \in [0,1]^3$. A key observation in [183, 274] is that the low dimensional input $\mathbf{p}$ and $\mathbf{d}$ need to be mapped to higher-dimensional features to be able to represent complex signals when f is parameterized with a neural network. More specifically, a predefined positional encoding is applied element-wise to each component of $\mathbf{p}$ and $\mathbf{d}$:

$$\gamma(t,L) = (\sin(2^0 t\pi), \cos(2^0 t\pi), \ldots, \sin(2^L t\pi), \cos(2^L t\pi)) \tag{8.1}$$

where t is a scalar input, e.g., a component of $\mathbf{p}$ or $\mathbf{d}$, and L the number of frequency octaves. In the context of generative models, we observe an additional benefit of this representation: It introduces an inductive bias to learn 3D shape representations in canonical orientations which otherwise would be arbitrary (see Figure 8.11).

Mildenhall et al. [183] propose to learn Neural Radiance Fields (NeRFs) by parameterizing them with multi-layer perceptrons (MLPs):

$$\begin{aligned} \mathbf{f}_\theta : \mathbb{R}^{L_\mathbf{p}} \times \mathbb{R}^{L_\mathbf{d}} &\to [0,\infty] \times [0,1]^3 \\ (\gamma(\mathbf{p}), \gamma(\mathbf{d})) &\mapsto (\sigma, \mathbf{c}) \end{aligned} \tag{8.2}$$

where θ indicates the network parameters and $L_\mathbf{p}, L_\mathbf{d}$ the output dimensionalities of the positional encodings.

Generative Neural Feature Fields While [183] fits θ to multiple posed images of a single scene, Schwarz et al. [255] propose a generative model for Neural Radiance Fields (GRAF) that is trained from unposed image collections. To learn a latent space of NeRFs, they condition the MLP on shape and appearance codes $\mathbf{z}_s, \mathbf{z}_a \sim \mathcal{N}(\mathbf{0}, I)$:

$$\begin{aligned} \mathbf{g}_\theta : \mathbb{R}^{L_\mathbf{p}} \times \mathbb{R}^{L_\mathbf{d}} \times \mathbb{R}^{M_s} \times \mathbb{R}^{M_a} &\to [0,\infty] \times [0,1]^3 \\ (\gamma(\mathbf{p}), \gamma(\mathbf{d}), \mathbf{z}_s, \mathbf{z}_a) &\mapsto (\sigma, \mathbf{c}) \end{aligned} \tag{8.3}$$

where M_s, M_a are the dimensionalities of the latent codes.

In this work, we explore a more efficient combination of volume and neural rendering. We replace GRAF's formulation for the three-dimensional color output $\mathbf{c}$ with a more generic M_f-dimensional feature $\hat{\mathbf{f}}$[1] and represent objects as Generative Neural Feature Fields:

$$\begin{aligned} \mathbf{h}_\theta : \mathbb{R}^{L_\mathbf{p}} \times \mathbb{R}^{L_\mathbf{d}} \times \mathbb{R}^{M_s} \times \mathbb{R}^{M_a} &\to [0,\infty] \times \mathbb{R}^{M_f} \\ (\gamma(\mathbf{p}), \gamma(\mathbf{d}), \mathbf{z}_s, \mathbf{z}_a) &\mapsto (\sigma, \hat{\mathbf{f}}) \end{aligned} \tag{8.4}$$

Object Representation A key limitation of NeRF and GRAF is that the entire scene is represented by a single model. As we are interested in disentangling different entities in the scene, we need control over the pose, shape and appearance of *individual* objects (we

[1]Note that $\hat{\mathbf{f}}$ indicates a feature vector, not having any relation to the in Chapter 4 introduced generic field $\mathbf{f}$.

consider the background as an object as well). We, therefore, represent each object using a separate Feature Field in combination with an affine transformation

$$\mathcal{T} = \{\mathbf{s}, \mathbf{t}, \mathbf{R}\} \tag{8.5}$$

where $\mathbf{s}, \mathbf{t} \in \mathbb{R}^3$ indicate scale and translation parameters, and $\mathbf{R} \in SO(3)$ a rotation matrix. Using this representation, we transform points from object to scene space as follows:

$$k(\mathbf{p}) = \mathbf{R} \cdot \begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \end{bmatrix} \cdot \mathbf{p} + \mathbf{t} \tag{8.6}$$

In practice, we volume render in scene space and evaluate the Feature Field in its canonical object space (see Figure 8.2):

$$(\sigma, \hat{\mathbf{f}}) = \mathbf{h}_\theta(\gamma(k^{-1}(\mathbf{p})), \gamma(k^{-1}(\mathbf{d})), \mathbf{z}_s, \mathbf{z}_a) \tag{8.7}$$

This allows us to arrange multiple objects in a scene. All object Feature Fields share their weights and $\mathcal{T}$ is sampled from a dataset-dependent distribution (see Section 8.2.4).

8.2.2 Scene Compositions

As discussed above, we describe scenes as compositions of N entities where the first $N-1$ are the objects in the scene and the last represents the background. We consider two cases: First, N is fixed across the dataset such that the images always contain $N-1$ objects plus the background. Second, N is varied across the dataset. In practice, we use the same representation for the background as for objects except that we fix the scale and translation parameters $\mathbf{s}_N, \mathbf{t}_N$ to span the entire scene, and to be centered at the scene space origin.

Composition Operator To define the composition operator C, let's recall that a Feature Field of a single entity $\mathbf{h}^i_{\theta_i}$ predicts a density $\sigma_i \in \mathbb{R}^+$ and a feature vector $\hat{\mathbf{f}}_i \in \mathbb{R}^{M_f}$ for a given point $\mathbf{p}$ and viewing direction $\mathbf{d}$. When combining non-solid objects, a natural choice [63] for the overall density at $\mathbf{p}$ is to sum up the individual densities and to use the density-weighted mean to combine all features at $(\mathbf{p}, \mathbf{d})$:

$$C(\mathbf{p}, \mathbf{d}) = \left(\sigma, \frac{1}{\sigma}\sum_{i=1}^{N} \sigma_i \hat{\mathbf{f}}_i\right), \text{ where } \quad \sigma = \sum_{i=1}^{N} \sigma_i \tag{8.8}$$

While being simple and intuitive, this choice for C has an additional benefit: We ensure gradient flow to all entities with a density greater than 0.

8.2.3 Scene Rendering

3D Volume Rendering While previous works [156, 170, 183, 255] volume render an RGB color value, we extend this formulation to rendering an M_f-dimensional feature vector $\hat{\mathbf{f}}$.

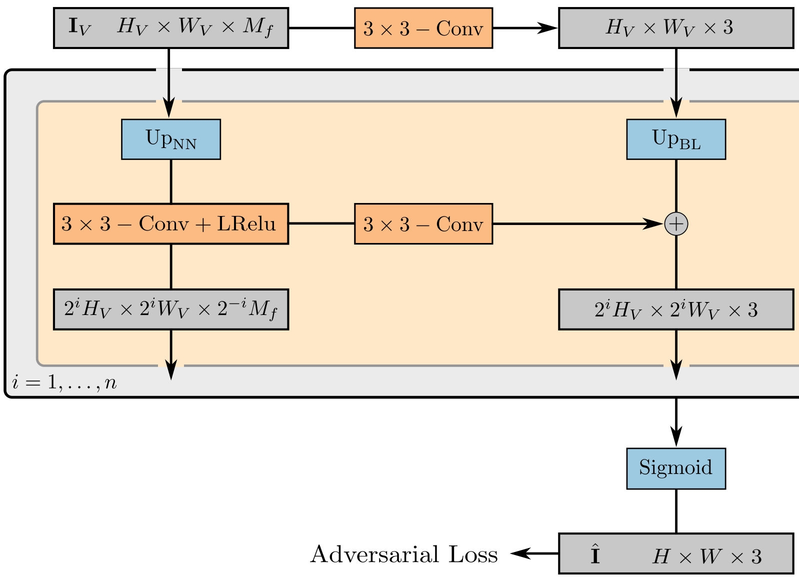

$\mathbf{I}_V$ $H_V \times W_V \times M_f$
$3 \times 3 -$ Conv
$H_V \times W_V \times 3$
Up$_{\text{NN}}$
Up$_{\text{BL}}$
$3 \times 3 -$ Conv $+$ LRelu
$3 \times 3 -$ Conv
$2^i H_V \times 2^i W_V \times 2^{-i} M_f$
$2^i H_V \times 2^i W_V \times 3$
$i = 1, \ldots, n$
Sigmoid
Adversarial Loss
$\hat{\mathbf{I}}$ $H \times W \times 3$

Figure 8.4: Neural Rendering Operator. *The feature image $\mathbf{I}_V$ is processed by n blocks of nearest neighbor upsampling and 3×3 convolutions with leaky ReLU activations. At every resolution, we map the feature image to an RGB image with a 3×3 convolution and add it to the previous output via bilinear upsampling. We apply a sigmoid activation to obtain the final image $\hat{\mathbf{I}}$. Gray color indicates outputs, orange learnable, and blue non-learnable operations.*

For given camera extrinsics ξ, let $\{\mathbf{p}_j\}_{j=1}^{N_s}$ be sample points along the camera ray with direction $\mathbf{d}$ for a given pixel, and $(\sigma_j, \hat{\mathbf{f}}_j) = C(\mathbf{p}_j, \mathbf{d})$ the corresponding densities and feature vectors of the field. The volume rendering operator π^{vol} [117] maps these evaluations to the pixel's final feature vector $\hat{\mathbf{f}}$ (see Section 4.3 for details). Using numerical integration as in [183], $\hat{\mathbf{f}}$ is obtained as

$$\hat{\mathbf{f}} = \sum_{j=1}^{N_s} T_j \alpha_j \hat{\mathbf{f}}_j \quad T_j = \prod_{k=1}^{j-1} (1 - \alpha_k) \quad \alpha_j = 1 - e^{-\sigma_j \delta_j} \tag{8.9}$$

where T_j is the transmittance, α_j the alpha value for $\mathbf{p}_j$, and $\delta_j = \left|\left|\mathbf{p}_{j+1} - \mathbf{p}_j\right|\right|_2$ the distance between neighboring sample points. The entire feature image is obtained by evaluating π^{vol} at every pixel. For efficiency, we render feature images at a resolution of 16^2 which is lower than the output resolution of 64^2 or 256^2 pixels. We then upsample the low-resolution feature maps to higher-resolution RGB images using 2D neural rendering. As evidenced by our experiments, this has two advantages: increased rendering speed and improved image quality.

2D Neural Rendering The neural rendering operator

$$\pi_\theta^{\text{neural}} : \mathbb{R}^{H_V \times W_V \times M_f} \to [0,1]^{H \times W \times 3} \tag{8.10}$$

with weights θ maps the feature image $\mathbf{I}_V \in \mathbb{R}^{H_V \times W_V \times M_f}$ to the final synthesized image $\hat{\mathbf{I}} \in [0,1]^{H \times W \times 3}$. We parameterize $\pi_\theta^{\text{neural}}$ as a 2D convolutional neural network (CNN) with leaky ReLU [166, 323] activation (Figure 8.4) and combine nearest neighbor upsampling with 3×3 convolutions to increase the spatial resolution. We choose small kernel sizes and no intermediate layers to only allow for spatially small refinements to avoid entangling global scene properties during image synthesis while at the same time allowing for increased output resolutions. Inspired by [123], we map the feature image to an RGB image at every spatial resolution and add the previous output to the next via bilinear upsampling. These skip connections ensure a strong gradient flow to the Feature Fields. We obtain our final image prediction $\hat{\mathbf{I}}$ by applying a sigmoid activation to the last RGB layer. We validate our design choices in an ablation study (Table 8.4).

8.2.4 Training

Generator We denote the full generative process formally as

$$G_\theta(\{\mathbf{z}_s^i, \mathbf{z}_a^i, T_i\}_{i=1}^N, \xi) = \pi_\theta^{\text{neural}}(\mathbf{I}_V) \quad \text{where} \quad \mathbf{I}_V = \{\pi^{\text{vol}}(\mathbf{r}_k)\}_{k=1}^{H_V \times W_V} \tag{8.11}$$

and N is the number of entities in the scene, and $\mathbf{r}_k$ is the k-th ray of the feature image $\mathbf{I}_V$.

Discriminator We parameterize the discriminator D_ϕ as a CNN [232] with leaky ReLU activation.

Name	Number of Images	Object Rotation	Camera Elevation	Horizontal Translation	Depth Translation	Object Scale
Chairs [211]	152,680	360°	90°	-	-	-
Cats [345]	9407	70°	10°	-	-	-
CelebA [153]	202,599	70°	10°	-	-	-
CompCars [326]	136,726	360°	10°	$[-0.12, 0.12]$	$[-0.22, 0.22]$	$[0.8, 1]$
Churches [336]	126,227	360°	0°	$[-0.15, 0.15]$	$[-0.15, 0.15]$	$[0.8, 1]$
CelebA-HQ [121]	30,000	90°	10°	-	-	-
FFHQ [122]	70,000	70°	10°	-	-	-
Clevr-2 [116]	54,336	0°	0°	$[-0.7, 0.7]$	$[-0.7, 0.7]$	-

Table 8.1: Dataset Parameters. *We report relevant camera and object transformation parameters for all datasets. We use the same dataset-specific parameters for experiments at* 64^2 *and* 256^2 *pixels.*

Training During training, we sample the number of entities in the scene $N \sim p_N$, the latent codes $\mathbf{z}_s^i, \mathbf{z}_a^i \sim \mathcal{N}(\mathbf{0}, I)$, as well as a camera pose $\xi \sim p_\xi$ and object-level transformations

$\mathcal{T}_i \sim p_{\mathcal{T}}$. In practice, we define p_ξ and $p_{\mathcal{T}}$ as uniform distributions over dataset-dependent camera elevation angles and valid object transformations, respectively (see Table 8.1). The motivation for this choice is that in most real-world scenes, objects are arbitrarily rotated, but not tilted due to gravity. The observer (the camera in our case), in contrast, can freely change its elevation angle wrt. the scene.

We train our model with the non-saturating GAN objective [85] and R_1 gradient penalty [178]

$$\begin{aligned}\mathcal{V}(\theta,\phi) = {} & \mathbb{E}_{\mathbf{z}_s^i,\mathbf{z}_a^i \sim \mathcal{N},\, \xi\sim p_\xi,\, c\mathcal{T}_i \sim p_{\mathcal{T}}} \left[f(D_\phi(G_\theta(\{\mathbf{z}_s^i,\mathbf{z}_a^i,\mathcal{T}_i\}_i,\xi))) \right] \\ & + \mathbb{E}_{\mathbf{I}\sim p_{\mathcal{D}}} \left[f(-D_\phi(\mathbf{I})) - \lambda \|\nabla D_\phi(\mathbf{I})\|^2 \right]\end{aligned} \tag{8.12}$$

where $f(t) = -\log(1+\exp(-t))$, $\lambda = 10$, and $p_{\mathcal{D}}$ indicates the data distribution.

8.2.5 Implementation Details

All object Feature Fields $\{\mathbf{h}^i_{\theta_i}\}_{i=1}^{N-1}$ share their weights and we parametrize them as MLPs with ReLU activations. We use 8 layers with a hidden dimension of 128 and a density and a feature head of dimensionality 1 and $M_f = 128$, respectively. For the background Feature Field $\mathbf{h}^N_{\theta_N}$, we use half the layers and hidden dimension. We use $L_{\mathbf{p}} = 2\cdot 3\cdot 10$ and $L_{\mathbf{d}} = 2\cdot 3\cdot 4$ for the positional encodings. We sample $M_s = 64$ points along each ray and render the feature image $\mathbf{I}_V$ at 16^2 pixels. We use an exponential moving average [335] with decay 0.999 for the weights of the generator. We use the RMSprop optimizer [281] with a batch size of 32 and learning rates of 1×10^{-4} and 5×10^{-4} for the discriminator and generator, respectively. For experiments at 256^2 pixels, we set $M_f = 256$ and halve the generator learning rate to 2.5×10^{-4}.

8.3 Experiments

Datasets We report results on commonly-used single-object datasets *Chairs* [204], *Cats* [344], *CelebA* [153], and *CelebA-HQ* [121]. The first consists of synthetic renderings of Photoshape chairs [211], and the others are image collections of cat and human faces, respectively. The data complexity is limited as the background is purely white or only takes up a small part of the image. We further report results on the more challenging single-object, real-world datasets *CompCars* [326], *LSUN Churches* [336], and *FFHQ* [122]. For *CompCars*, we randomly crop the images to achieve more variety of the object's position in the image.[2] For these datasets, disentangling objects is more complex as the object is not always in the center and the background is more cluttered and takes up a larger part of the image. To test our model on multi-object scenes, we use the script from [116] to render scenes with 2, 3, 4, or 5 random primitives (*Clevr-N*). To test our model on scenes with a varying number of objects, we also run our model on the union of them (*Clevr-2345*).

Baselines We compare against voxel-based PlatonicGAN [102], BlockGAN [196], and HoloGAN [195], and radiance field-based GRAF [255]. We further compare against Holo-

[2]We do not apply random cropping for [102, 255] as they can only handle scenes with centered objects.

GAN w/o 3D Conv, a variant of [195] proposed in [255] for higher resolutions. We additionally report a ResNet-based [98] 2D GAN [178] for reference.

Metrics We report the Frechet Inception Distance (FID) score [105] to quantify image quality. We use 20,000 real and fake samples to calculate the FID score.

8.3.1 Controllable Scene Generation

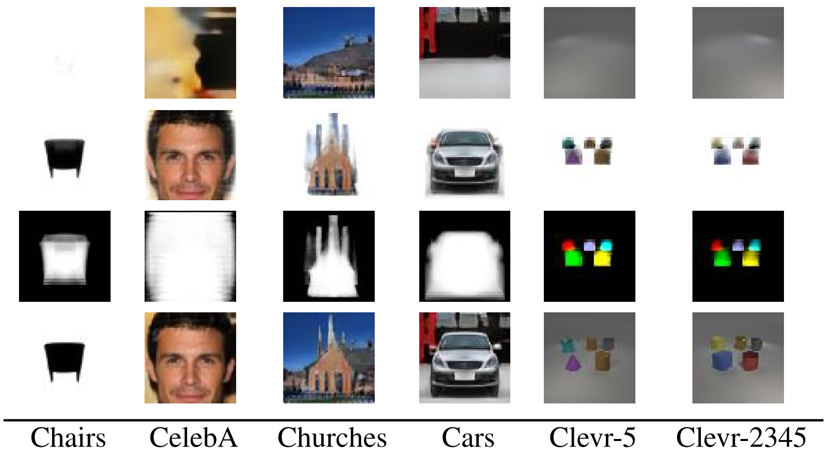


Figure 8.5: Scene Disentanglement. *From top to bottom, we show only backgrounds, only objects, color-coded object alpha maps, and the final synthesized images at* 64^2 *pixel resolution. Disentanglement emerges without supervision, and the model learns to generate plausible backgrounds although the training data only contains images with objects.*

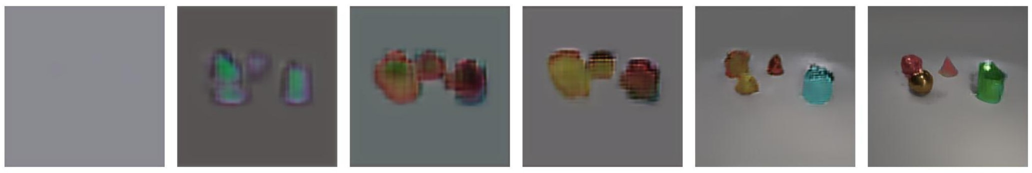
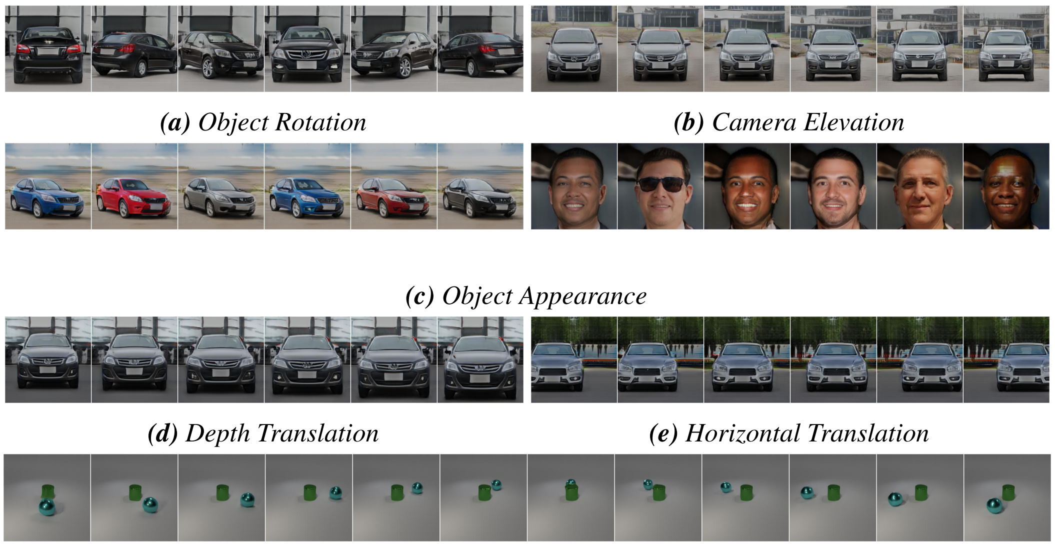


Figure 8.6: Training Progression. *We show renderings of our model on Clevr-2345 at* 256^2 *pixels after* 0, 1, 2, 3, 10, *and* 100-*thousand iterations. Unsupervised disentanglement emerges already at the very beginning of training.*

Disentangled Scene Generation We first analyze to which degree our model learns to generate disentangled scene representations. In particular, we are interested if objects are disentangled from the background. Towards this goal, we exploit the fact that our composition operator is a simple addition operation (Eq. 8.8) and render individual components



(a) Object Rotation *(b) Camera Elevation*

(c) Object Appearance

(d) Depth Translation *(e) Horizontal Translation*

(f) Circular Translation of One Object Around Another Object

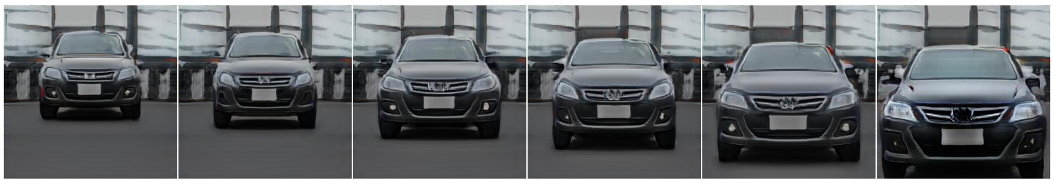
Figure 8.7: Controllable Scene Generation at 256^2 ***Pixel Resolution.*** *Controlling the generated scenes during image synthesis: Here we rotate or translate objects, change their appearances, and perform complex operations like circular translations.*

and object alpha maps (Eq. 8.9). Note that while we always render the feature image at 16^2 during training, we can choose arbitrary resolutions at test time.

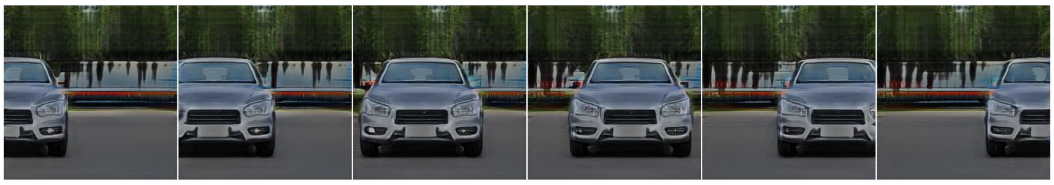
Figure 8.5 suggests that our method disentangles objects from the background. Note that this disentanglement emerges without any supervision, and the model learns to generate plausible backgrounds without ever having seen a pure background image, implicitly solving an inpainting task. We further observe that our model correctly disentangles individual objects when trained on multi-object scenes with fixed or a varying number of objects. We further find that unsupervised disentanglement is a property of our model which emerges already at the very beginning of training (Figure 8.6). Note how our model synthesizes individual objects before spending capacity on representing the background.

Controllable Scene Generation As individual components of the scene are correctly disentangled, we analyze how well they can be controlled. More specifically, we are interested if individual objects can be rotated and translated, but also how well shape and appearance can be controlled. In Figure 8.7, we show examples in which we control the scene during image synthesis. We rotate individual objects, translate them in 3D space, or change the camera elevation. By modeling the shape and appearance of each entity with a different latent code, we are further able to change the objects' appearances without altering their shape.

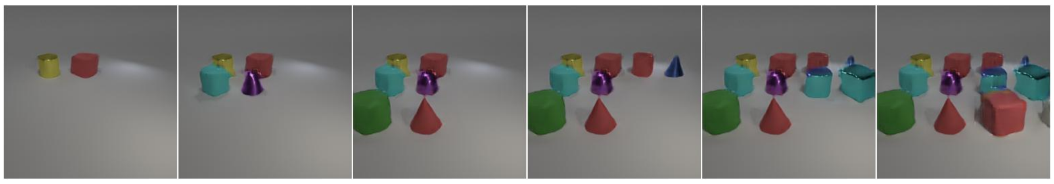
Generalization Beyond Training Data The learned compositional scene representations allow us to generalize outside the training distribution. For example, we can increase the



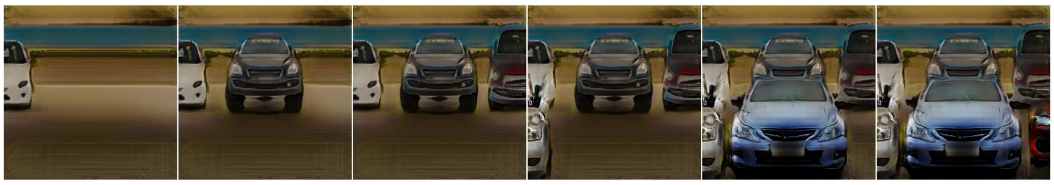
(a) *Increase Depth Translation*



(b) *Increase Horizontal Translation*



(c) *Add Additional Objects (Trained on Two-Object Scenes)*

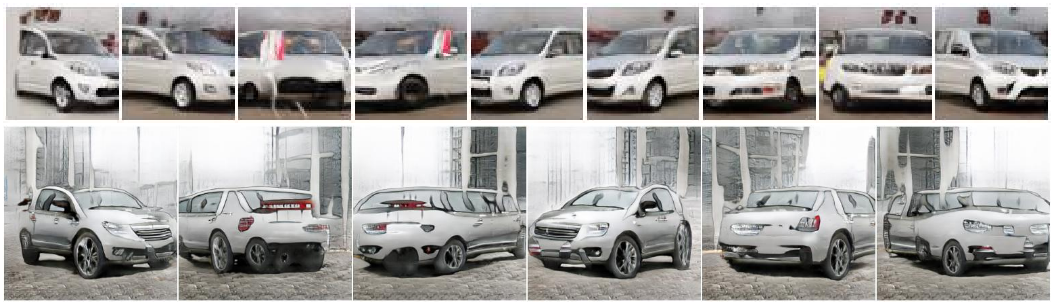


(d) *Add Additional Objects (Trained on Single-Object Scenes)*

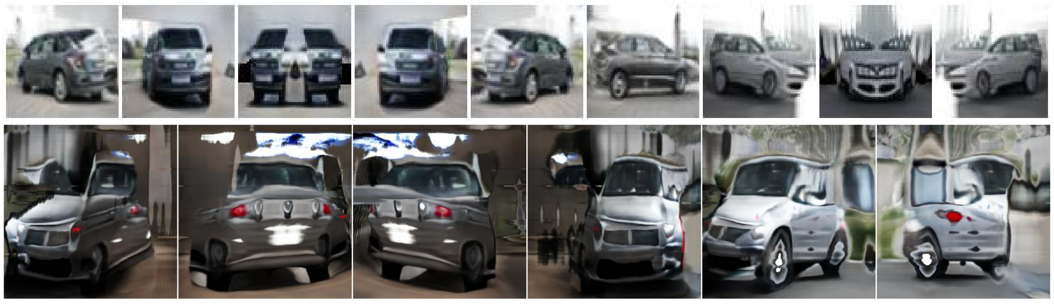
Figure 8.8: Generalization Beyond Training Data. *As individual objects are correctly disentangled, our model allows for generating out-of-distribution samples at test time. For example, we can increase the translation ranges or add more objects than there were present in the training data.*

translation ranges of objects or add more objects than there were present in the training data (see Figure 8.8).

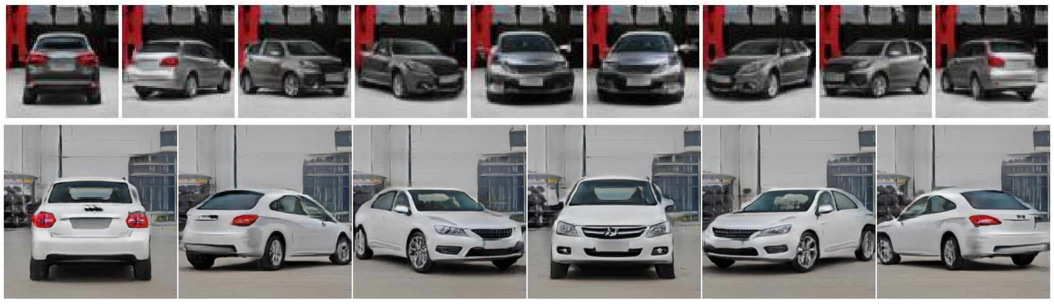
8.3.2 Comparison to Baseline Methods



(a) 360° *Object Rotation for HoloGAN [195]*



(b) 360° *Object Rotation for GRAF [255]*



(c) 360° *Object Rotation for Our Method*

Figure 8.9: Qualitative Comparison. *Compared to baseline methods, we achieve more consistent image synthesis for complex scenes with cluttered backgrounds at* 64^2 *(top rows) and* 256^2 *(bottom rows) pixel resolutions. Note that we disentangle the object from the background and are able to rotate only the object while keeping the background fixed.*

Compared to baseline methods, our method achieves similar or better FID scores at both 64^2 and 256^2 pixel resolutions (see Table 8.2). Qualitatively, we observe that while all approaches allow for controllable image synthesis on datasets of limited complexity, results are less consistent for the baseline methods on more complex scenes with cluttered backgrounds. Further, our model disentangles the object from the background, such that we are able to control the object independent of the background (Figure 8.9).

We further note that our model achieves similar or better FID scores than the ResNet-

	Cats	CelebA	Cars	Chairs	Churches
2D GAN [178]	18	15	**16**	59	19
Plat. GAN [102]	318	321	299	199	242
BlockGAN [196]	47	69	41	41	28
HoloGAN [195]	27	25	17	59	31
GRAF [255]	26	25	39	34	38
Ours	**8**	**6**	**16**	**20**	**17**

(a) *64 × 64 Pixels*

	CelebA-HQ	FFHQ	Cars	Churches	Clevr-2
HoloGAN [195]	61	192	34	58	241
w/o 3D Conv	33	70	49	66	273
GRAF [255]	49	59	95	87	106
Ours	**21**	**32**	**26**	**30**	**31**

(b) *256 × 256 Pixels*

Table 8.2: Quantitative Comparison. *We report FID (↓) for baselines and our method.*

2D GAN	Plat. GAN	BlockGAN	HoloGAN	GRAF	Ours
1.69	381.56	4.44	7.80	0.68	0.41

Table 8.3: Network Parameter Comparison. *We report the number of generator network parameters in million.*

based 2D GAN [178] despite fewer network parameters (0.41m compared to 1.69m). This confirms our initial hypothesis that using a 3D representation as inductive bias results in better outputs. Note that for a fair comparison, we only report methods that are similar wrt. network size and training time (see Table 8.3).

8.3.3 Ablation Studies

Full	-Skip	-Act.	+NN. RGB Up.	+Bi. Feat. Up.
16.16	16.66	21.61	17.28	20.68

Table 8.4: Ablation Study. *We report FID (↓) on CompCars without RGB skip connections (-Skip), without final activation (-Act.), with nearest neighbor instead of bilinear image upsampling (+ NN. RGB Up.), and with bilinear instead of nearest neighbor feature upsampling (+ Bi. Feat. Up.).*

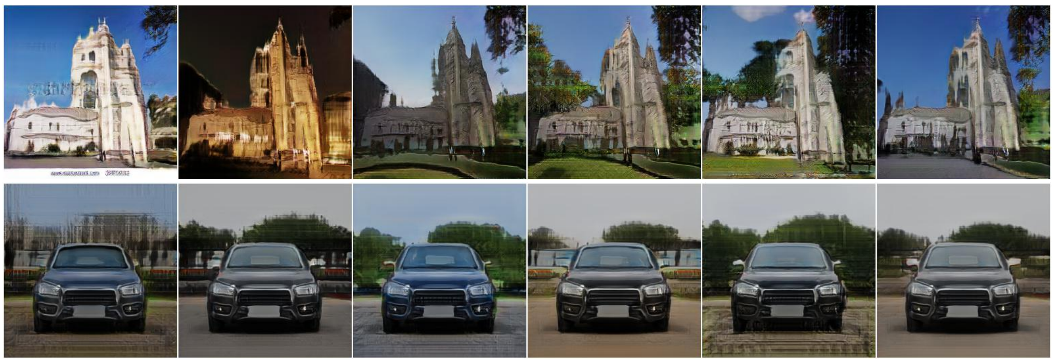
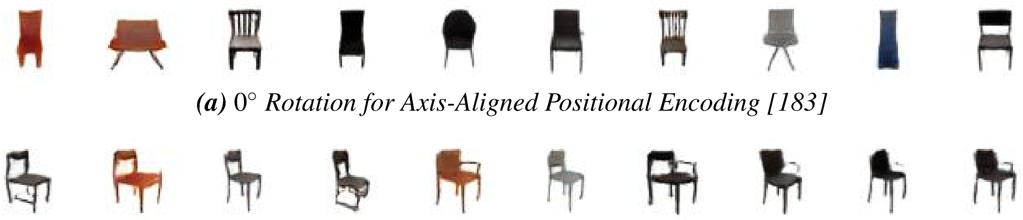


Figure 8.10: ***Neural Renderer.*** *We change the background while keeping the foreground object fixed for our method at* 256^2 *pixel resolution. Note how the neural renderer realistically adapts the objects' appearances to the background.*



(a) 0° *Rotation for Axis-Aligned Positional Encoding [183]*

(b) 0° *Rotation for Random Fourier Features [274]*

Figure 8.11: Canonical Pose. *In contrast to random Fourier features [274], axis-aligned positional encoding Equation 8.1 encourages the model to learn objects in a canonical pose.*

Importance of Individual Components The ablation study in Table 8.4 shows that our design choices of RGB skip connections, final activation function, and selected upsampling types improve results and lead to higher FID scores.

Effect of Neural Renderer A key difference to [255] is that we combine volume with neural rendering. The quantitative (Table 8.2) and qualitative comparisons (Figure 8.9) indicate that our approach leads to better results, in particular for complex, real-world data. Our model is more expressive and can better handle the complexity of real scenes, e.g., note how the neural renderer realistically adapts object appearances to the background (Figure 8.10). Further, we observe a rendering speed up: compared to [255], total rendering time is reduced from 110.1ms to 4.8ms, and from 1595.0ms to 5.9ms for 64^2 and 256^2 pixels, respectively.

Positional Encoding We use axis-aligned positional encoding for the input point and viewing direction (Eq. 8.1). Surprisingly, this encourages the model to learn canonical representations as it introduces a bias to align the object axes with highest symmetry with the canonical axes which allows the model to exploit object symmetry (Figure 8.11).

8.4 Conclusion

In this chapter, we presented *GIRAFFE*, a novel method for controllable image synthesis. Our key idea is to incorporate a compositional 3D scene representation into the generative model. By representing scenes as Compositional Generative Neural Feature Fields, we disentangle individual objects from the background as well as their shape and appearance without explicit supervision. Combining this with a neural renderer yields fast and controllable image synthesis.

8.4.1 Limitations and Future Work

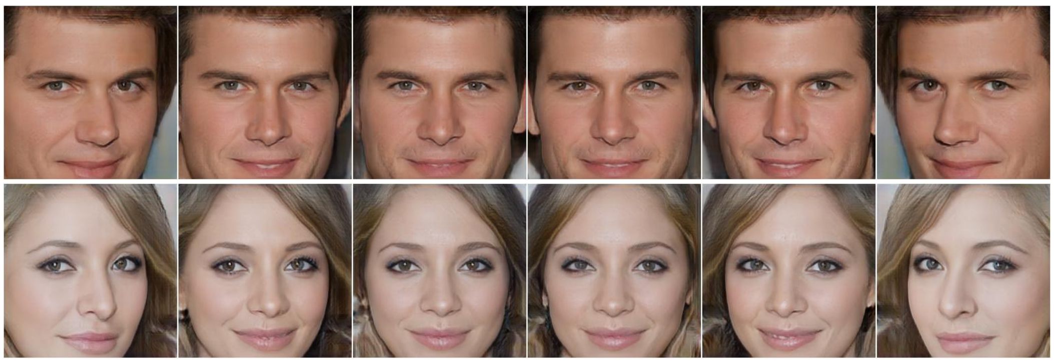


Figure 8.12: Dataset Bias. *Eye and hair rotation are examples of dataset biases: They primarily face the camera, and our model tends to entangle them with the object rotation.*

Dataset Bias Our method struggles to disentangle factors of variation if there is an inherent bias in the data. We show an example in Figure 8.12: In the celebA-HQ dataset, the eye and hair orientation is predominantly pointing towards the camera, regardless of the face rotation. When rotating the object, the eyes and hair in our generated images do not stay fixed but are adjusted to meet the dataset bias. Recent works try to address this with different generator pose conditioning strategies [37, 256], dual discrimination [37], or path length regularization [93]. From a broader perspective, adding flexibility to the model to overcome dataset biases while ensuring 3D consistency is an active research area.

Object Transformation Distributions We sometimes observe disentanglement failures, e.g. for *Churches* where the background contains a church, or for *CompCars* where the foreground contains background elements. A recent follow-up work [325] addresses this by additional regularization, e.g., in the form of bounding box, mask coverage, and mask binarization losses, achieving better disentanglement results.

Image Fidelity and Resolution While outperforming relevant baselines, the image fidelity, e.g., measured in FID, still lacks behind state-of-the-art 2D GAN methods like StyleGAN 2 [123]. The same holds for the final image output resolution: While our method allows for renderings up to 256×256 pixels, recent 2D-based generative models scale to megapixel resolutions and beyond. Several follow-up works are proposed that improve both of these aspects. Notably, StyleNeRF [93] combines bigger models, larger feature images, and an improved neural renderer to achieve 3D-aware image synthesis at megapixel resolutions. Similarly, EG3D [37] improves in terms of image fidelity by improving the latent code conditioning using multi-plane feature grids. Very recently, GRAM HD [321], EpiGRAF [321], and VoxGRAF [256] achieve an increase in image fidelity and output resolution (first two) without the use of a neural renderer via a manifold reparameterization, well-engineered patch-wise training, and sparse voxel grids, respectively.

Scene Complexity Our method achieves promising results on single-object scenes as well as multi-object scenes consisting of geometric primitives. However, the complexity of these scenes is still limited considering the number of objects in the scene, the fore- and background disentanglement, as well as camera pose distributions. The latter is addressed in a recent follow-up work [199] that shows that more complex, non-uniform camera distributions can be learned jointly with the 3D-aware generator. In [25], incorporating additional supervision in the form of segmentation masks is investigated in the context of learning compositional 3D representations of human faces. We identify scaling our model to scenes with more objects in the scene, more cluttered background, and more complex camera pose distributions, potentially by incorporating more supervision data, as a promising future research direction.

9 Conclusion

In this thesis, we have first investigated how 3D geometry should be represented in learning-based systems. We have focused in particular on how voxel, point cloud, and mesh-based representations require discretizing 3D space to achieve acceptable memory and computational complexity. We have found that the proposed neural field-based 3D representation Occupancy Networks (i) can represent 3D geometry without discretization while having a constant memory footprint, (ii) can be used in learning-based systems for end-to-end single-view 3D reconstruction, (iii) allows for learning a meaningful latent space of 3D shapes facilitating generative tasks. Next, we have observed that the requirement of 3D supervision during training is a key limitation for scaling to real-world scenes. To overcome this limitation, we have investigated how neural fields can be inferred from 2D supervision only. We have found that the proposed differentiable surface rendering technique Differentiable Volumetric Rendering (i) allows for inferring neural field-based shape and texture representations from posed multi- or single-view image supervision, (ii) exhibits a low memory footprint due to the use of implicit differentiation at the surface point, (iii) enables neural-field based 3D reconstruction for real-world data. Subsequently, we have discussed how applications in real-world environments require representations of not only 3D geometry but also motion. As a result, we have investigated how 3D shapes in motion can be represented in learning-based systems. We have observed that the proposed representation Occupancy Flow (i) combines Occupancy Networks with a neural field-based velocity field to represent continuous and smooth 4D data, (ii) enables 4D reconstruction with implicitly-captured correspondences, (iii) achieves a disentangled representation of shape and motion enabling a variety of discriminative and generative tasks ranging from 4D reconstruction to latent interpolations, to motion transfer. Next, we have found that while the proposed systems allow for efficiently representing 3D and 4D data, a key concept of natural scenes is not yet incorporated into the models: compositionality. As a consequence, we have developed GIRAFFE, a 3D-aware generative model that (i) incorporates the concept of compositionality into the generator architecture enabling controllability of the size, position, shape, and appearance of individual objects in the 3D scene, (ii) allows for efficient and high-quality image synthesis with full control over the camera viewpoint, (iii) scales to real-world scenarios like Internet photo collections as it can be trained from raw, unposed image collections.

9.1 Limitations and Future Perspectives

In the following, we provide a broader view and longer-term perspectives for research on neural scene representations for 3D reconstruction and generative modeling. For more specific discussions on limitations and future directions wrt. the individual contributions, we refer the reader to the conclusion sections of the respective chapters.

9.1.1 Compositional Scene Representations

Learning object-centric representations from low-level image sensory data is a promising direction to solve tasks in the real world that involve higher-level reasoning or planning. In fact, the object-based compositional understanding of natural environments is considered a cornerstone of human understanding and reasoning [249, 266]. As a result, a series of unsupervised object discovery methods are proposed. While first works operate in the 2D domain [31, 89, 159], more recent approaches perform the reasoning in 3D [200, 249, 268, 339] including our proposed GIRAFFE model (see Chapter 8), achieving better decomposition and allowing to scale to visually more complex scenes. However, all of these works are still limited to (i) comparably simple object categories, (ii) synthetic or well-curated image data, (iii) comparably simple object compositions and simple backgrounds. Further, they often require large amounts of training data. We identify developing robust systems that can infer or generate object-centric compositional representations of real-world scenes from sparse and potentially noisy input data as a longer-term research direction. Promising directions might include:

Building Bottom-Up Hierarchies The key idea is to build and train robust representations for individual object categories. In a subsequent step, they can be used to infer more complex scene representations that are composed of these individual object-level representations [95, 328]. Key challenges for this approach include how to handle out-of-distribution scenarios (e.g., an unseen object class) and how to model inter-object relationships (e.g., light reflections or shadows cast onto other objects).

Combining Different Forms of Supervision While obtaining accurate supervision data is not feasible for real-world applications, fully-unsupervised methods might not directly lead to the desired scene representations. A combination in the form of large-scale unsupervised pretraining and a shorter, smaller-scale fine-tuning stage with access to fully or weakly labeled data could combine the benefits of both training strategies and utilize all available data [250]. Key challenges for this approach are estimating the required coverage and amount of the labeled part of the data and obtaining high-quality, potentially 3D, labels.

Exploring Pseudo Ground Truth and Multi-Modal Data In contrast to state-of-the-art 3D reconstruction methods and 3D-aware generative models, 2D-based vision models, as well as deep language models, have been scaled to massive amounts of data already, achieving unprecedented prediction and generation capabilities. In 3D modeling, not only predictions but also intermediate representations of such large-scale models can be used as an additional source of supervision or guidance to obtain high-quality 3D scene representations [227].

9.1.2 Generalization

Similar to other fields of machine learning, our proposed systems are also still limited wrt. their generalization capabilities. For example, our 3D reconstruction system might perform poorly for an unseen object category or the quality of inferred 3D shape and texture representations from multiple views drops when the input images are sparse or only cover a small part of the scene. However, being able to generalize is one key property needed for robust systems that can be employed in the real world. To improve generalization capabilities, we identify the following research directions as interesting:

Incorporating Powerful Inductive Biases While incorporating inductive biases, e.g., modeling scenes in 3D and not in 2D, has already been shown to improve the systems' generalization capabilities, more design choices can be explored. We identify in particular physically-inspired inductive biases such as modeling the light transport more accurately [296] or incorporating the laws of physics as constraints in dynamical systems [50] as promising future research.

Finding a Balance of Global and Local Information Flow While breaking one global problem into multiple local problems can make the system more generalizable in the presence of high-quality input data, the system can also become less robust to noise and sparse inputs [341]. We identify developing a good balance of global and local information flow as a promising research direction. Key challenges for this approach can include how to avoid task and input data dependence, i.e., how to find a good balance independent of the task to which the model is applied and independent of the input data.

Gaining a Better Understanding of Neural Networks While a series of works try to gain a better theoretical understanding of neural networks, it is still limited and most architectural improvements, e.g., the use of positional encoding in the context of neural fields, are discovered empirically [183, 274]. A better understanding of these models, in particular, their generalization capabilities despite overparameterization could enable architectural as well as optimization strategy improvements. Key challenges include finding a good balance between a well-founded analysis and an experimental setup that is not prohibitively restrictive and that is close to real-world setups.

9.1.3 Gap to Robotics

Our end goal is to develop systems that are employed in the real world. Despite this, models developed in the computer vision community including our proposed systems are often analyzed mainly wrt. computer vision and graphics interests, e.g., image quality, reconstruction quality, or rendering times. For employment in the real world, other, more robotics-focused aspects such as eligibility of feature representations for downstream tasks, robustness to noise, or efficiency of the proposed system might be similar or even more important, depending on the application. We identify the following approaches as promising to bring the computer vision and robotics community more closely together:

Developing Well-Engineered Implementations While large machine learning software packages reduced the entry barrier and enabled faster experimental cycles, resulting implementations can often be made more efficient with more appropriate data structures or custom implementations [186]. This could enable more widespread use of respective algorithms in robotics applications. Further, reducing the required hardware resources could also improve the imbalance of competition between small and large research labs.

Building Interdisciplinary Benchmarks While benchmarks are a common way of comparing different models, they often only focus on individual aspects such as classification, detection, or semantic labeling scores. Developing a benchmark that measures different qualities of a model that are related to both computer vision and robotics applications might help to bridge the two disciplines.

Incentivising Ideas over Performance in the Review Process In many cases, not top performance, but other properties such as simplicity, efficiency, or beauty are reasons why some approaches are more widely adopted than others. However, the current review system still tends to overweight performance. We identify adopting this, e.g., by respective tutorials or classes for students and other reviewers, as a promising direction to change this imbalance.

A Credits

In the following, we discuss the contributions and credits of individual authors for the four research projects that are part of this thesis. Overall, results presented in this thesis are often outcomes of collaborative projects [179, 197, 198, 200] such that assigning contributions to individuals can only be done approximately. Further, all authors of the projects contributed significantly to the individual publications which would not have been possible otherwise.

3D Reconstruction with Neural Fields (Chapter 5) The initial idea to represent 3D geometry as the decision boundary of a binary classifier was initially suggested to Lars Mescheder (LM) by Michael Oechsle (MO) and Andreas Geiger (AG) (see also [180]). LM was the project lead being involved in every part of the project. AG was further mainly involved in manuscript writing and proofreading, figure creation, and project direction discussions. The author of this thesis Michael Niemeyer (MN) and MO were further involved in parts of the experimental evaluation of the approach, baseline implementations and evaluations, figure creation, proofreading of the manuscript, and project direction discussions.

Differentiable Surface Rendering of Neural Fields (Chapter 6) The initial idea of the implicit differentiation and its implementation for implicit surfaces was brainstormed by MN, LM, MO, and AG, while specific design choices were mostly developed by MN and LM. MN was further involved in every part of the project as the project lead. LM was further involved in the baseline implementation, figure creation, manuscript writing and proofreading, and project direction discussions. AG was further involved in manuscript writing and proofreading, figure creation, and project direction discussions. MO was further involved in figure creation, baseline implementation and evaluations, rendering, and project direction discussions.

4D Reconstruction with Neural Fields (Chapter 7) The initial idea to represent 4D shapes in motion as an Occupancy Network and a parameterized vector field was mainly brainstormed by MN, LM, MO, and AG. The structure of the method section (see Section 7.2) was developed jointly by MN, LM, and AG. MN was further involved in every part of the project as the project lead. LM was further involved in specific code design and baseline selection choices, manuscript writing and proofreading, figure creation, and project direction discussions. MO was further involved in manuscript proofreading and project direction discussions. AG was further involved in manuscript writing and proofreading, figure creation, and project direction discussions.

Compositional 3D-Aware Generative Modeling with Neural Fields (Chapter 8) Next to AG, Katja Schwarz and Yiyi Liao were part of early discussions and brainstorming meetings providing valuable comments on early results, potential baselines, and interesting project directions. MN was further involved in every part of the project as the project lead. AG was further involved in manuscript writing and proofreading, figure creation, and project direction discussions.

B Publications

In the following, we list all publications that the author of this thesis Michael Niemeyer (MN) was involved in during his Ph.D. and shortly discuss MN's contributions to the non-first-author projects.

Publications part of this thesis

- L. Mescheder, M. Oechsle, **M. Niemeyer**, S. Nowozin, and A. Geiger. "Occupancy Networks: Learning 3D Reconstruction in Function Space". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).* 2019
  - Contribution: MN was involved in the code development and evaluation of baselines, evaluation of the approach, figure creation, manuscript proofreading, and project direction discussions.
- **M. Niemeyer**, L. Mescheder, M. Oechsle, and A. Geiger. "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).* 2020
- **M. Niemeyer**, L. Mescheder, M. Oechsle, and A. Geiger. "Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV).* 2019
- **M. Niemeyer** and A. Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).* 2021

Publications not part of this thesis

- M. Oechsle, L. Mescheder, **M. Niemeyer**, T. Strauss, and A. Geiger. "Texture Fields: Learning Texture Representations in Function Space". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV).* 2019
  - Contribution: MN was involved in the main idea development, manuscript proofreading, and project direction discussions.
- S. Peng, **M. Niemeyer**, L. Mescheder, M. Pollefeys, and A. Geiger. "Convolutional Occupancy Networks". In: *Proc. of the European Conf. on Computer Vision (ECCV).* 2020
  - Contribution: MN was involved in the main idea development, dataset generation, specific code design choices, figure creation, and manuscript proofreading.

- M. Oechsle, **M. Niemeyer**, C. Reiser, L. Mescheder, T. Strauss, and A. Geiger. "Learning Implicit Surface Light Fields". In: *Proc. of the International Conf. on 3D Vision (3DV).* 2020
  - Contribution: MN was involved in the main idea development, figure creation, dataset preparation, execution of experiments, manuscript proofreading, and project direction discussions.
- K. Schwarz, Y. Liao, **M. Niemeyer**, and A. Geiger. "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2020
  - Contribution: MN was involved in the main idea development, specific model design choices, code development, evaluation of baselines, manuscript proofreading, and project direction discussions.
- **M. Niemeyer** and A. Geiger. "CAMPARI: Camera-Aware Decomposed Generative Neural Radiance Fields". In: *Proc. of the International Conf. on 3D Vision (3DV).* 2021
- S. Peng, C. M. Jiang, Y. Liao, **M. Niemeyer**, M. Pollefeys, and A. Geiger. "Shape As Points: A Differentiable Poisson Solver". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2021
  - Contribution: MN was involved in the dataset preparation, manuscript proofreading, and project direction discussions.
- **M. Niemeyer**, J. T. Barron, B. Mildenhall, M. S. M. Sajjadi, A. Geiger, and N. Radwan. "RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).* 2022
- Z. Yu, S. Peng, **M. Niemeyer**, T. Sattler, and A. Geiger. "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2022
  - Contribution: MN was involved in the project organization and management, project direction choices, and manuscript text and figure creation.
- K. Schwarz, A. Sauer, **M. Niemeyer**, Y. Liao, and A. Geiger. "VoxGRAF: Fast 3D-Aware Image Synthesis with Sparse Voxel Grids". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2022
  - Contribution: MN was involved in the project organization and management, project direction choices, and manuscript text and figure creation.

Bibliography

[1] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. “Large-Scale Data for Multiple-View Stereopsis”. In: *International Journal of Computer Vision (IJCV)* 120 (2016).

[2] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. “Learning Representations and Generative Models for 3D Point Clouds”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. “Building Rome in a Day”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2009.

[4] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. “Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes”. In: *Proc. of the British Machine Vision Conf. (BMVC)*. 2017.

[5] H. Alhaija, S. Mustikovela, A. Geiger, and C. Rother. “Geometric Image Synthesis”. In: *Proc. of the Asian Conf. on Computer Vision (ACCV)*. 2018.

[6] T. Alldieck, M. A. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll. “Video Based Reconstruction of 3D People Models”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[7] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman. “Controlling Neural Level Sets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[8] R. Basri, M. Galun, A. Geifman, D. W. Jacobs, Y. Kasten, and S. Kritchman. “Frequency Bias in Neural Networks for Input of Non-Uniform Density”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2020.

[9] R. C. Batra. *Elements of continuum mechanics*. Aiaa, 2006.

[10] B. G. Baumgart. *Geometric Modeling for Computer Vision.* Stanford University, 1974.

[11] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger. “Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?” In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[12] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J. Jacobsen. “Invertible Residual Networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[13] Y. Bengio, A. C. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2013).

[14] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski. "Deep Generative Stochastic Networks Trainable by Backprop". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2014.

[15] A. Bermano, T. Beeler, Y. Kozlov, D. Bradley, B. Bickel, and M. H. Gross. "Detailed spatio-temporal reconstruction of eyelids". In: *ACM Trans. on Graphics* (2015).

[16] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. "The ball-pivoting algorithm for surface reconstruction". In: *IEEE Trans. on Visualization and Computer Graphics (VCG)* (1999).

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] A. Blake, A. Zisserman, and G. Knowles. *Surface descriptions from stereo and shading*. 1985.

[19] V. Blanz and T. Vetter. "A Morphable Model for the Synthesis of 3D Faces". In: *ACM Trans. on Graphics*. 1999.

[20] J. F. Blinn. "Light reflection functions for simulation of clouds and dusty surfaces". In: *ACM Trans. on Graphics*. 1982.

[21] M. A. Boden. *Mind as machine: A history of cognitive science*. Oxford University Press, 2008.

[22] V. I. Bogachev, A. V. Kolesnikov, and K. V. Medvedev. "Triangular transformations of measures". In: *Sbornik: Mathematics* 196 (2005).

[23] F. Bogo, J. Romero, M. Loper, and M. J. Black. "FAUST: Dataset and Evaluation for 3D Mesh Registration". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[24] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. "Dynamic FAUST: Registering Human Bodies in Motion". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[25] M. BR, A. Tewari, X. Pan, M. Elgharib, and C. Theobalt. "gCoRF: Generative Compositional Radiance Fields". In: *arXiv.org* 2210.17344 (2022).

[26] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. *JAX: composable transformations of Python+NumPy programs*. 2018.

[27] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. "Generative and Discriminative Voxel Modeling with Convolutional Neural Networks". In: *arXiv.org* 1608.04236 (2016).

[28] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *Signal Processing Magazine* 34 (2017).

[29] A. Brock, J. Donahue, and K. Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[30] Y. Burda, R. B. Grosse, and R. Salakhutdinov. “Importance Weighted Autoencoders”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2016.

[31] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. M. Botvinick, and A. Lerchner. “MONet: Unsupervised Scene Decomposition and Representation”. In: *arXiv.org* 1901.11390 (2019).

[32] F. Calakli and G. Taubin. “SSD: Smooth Signed Distance Surface Reconstruction”. In: *Computer Graphics Forum* 30 (2011).

[33] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. “Using multiple hypotheses to improve depth-maps for multi-view stereo”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2008.

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

[35] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. “Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[36] E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. “pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[37] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. “EG3D: Efficient Geometry-aware 3D Generative Adversarial Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[38] S. Chandrasekhar. “Radiative transfer”. In: *Journal of the Royal Meteorological Society* (1950).

[39] X. Chen, X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[40] L. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. “Searching for Efficient Multi-Scale Architectures for Dense Image Prediction”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[41] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[42] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[43] Z. Chen and H. Zhang. "Learning Implicit Fields for Generative Shape Modeling". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[44] K. Chen, R. Oldja, N. Smolyanskiy, S. Birchfield, A. Popov, D. Wehr, I. Eden, and J. Pehserl. "MVLidarNet: Real-Time Multi-Class Scene Understanding for Autonomous Driving Using Multiple Views". In: *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[45] X. Chen, Y. Zheng, M. Black, O. Hilliges, and A. Geiger. "SNARF: Differentiable Forward Skinning for Animating Non-Rigid Neural Implicit Shapes". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[46] J. Chibane, T. Alldieck, and G. Pons-Moll. "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[47] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[48] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[49] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. "Stargan v2: Diverse image synthesis for multiple domains". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[50] M. Chu, L. Liu, Q. Zheng, E. Franz, H. Seidel, C. Theobalt, and R. Zayer. "Physics informed neural fields for smoke reconstruction with sparse data". In: *ACM Trans. on Graphics* (2022).

[51] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018.

[52] H. Coskun, F. Achilles, R. S. DiPietro, N. Navab, and F. Tombari. "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[53] B. Curless and M. Levoy. "A Volumetric Method for Building Complex Models from Range Images". In: *ACM Trans. on Graphics*. 1996.

[54] B. Dai, Y. Wang, J. Aston, G. Hua, and D. P. Wipf. "Connections with Robust PCA and the Role of Emergent Sparsity in Variational Autoencoder Models". In: *Journal of Machine Learning Research (JMLR)* (2018).

[55] B. Dai and D. P. Wipf. "Diagnosing and Enhancing VAE Models". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[56] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei. “Imagenet: A large-scale hierarchical image database”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[57] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau. “Single-image SVBRDF capture with a rendering-aware deep network”. In: *ACM Trans. on Graphics*. 2018.

[58] L. Dinh, D. Krueger, and Y. Bengio. “NICE: Non-linear Independent Components Estimation”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[59] L. Dinh, J. Sohl-Dickstein, and S. Bengio. “Density estimation using Real NVP”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[60] J. Dong, J. G. Burnham, B. Boots, G. C. Rains, and F. Dellaert. “4D crop monitoring: Spatio-temporal reconstruction for agriculture”. In: *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*. 2017.

[61] Z. Dong, C. Guo, J. Song, X. Chen, A. Geiger, and O. Hilliges. “PINA: Learning a Personalized Implicit Neural Avatar from a Single RGB-D Video Sequence”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[62] A. Dosovitskiy and T. Brox. “Generating images with perceptual similarity metrics based on deep networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[63] R. A. Drebin, L. C. Carpenter, and P. Hanrahan. “Volume rendering”. In: *ACM Trans. on Graphics*. 1988.

[64] H. Drucker and Y. Le Cun. “Improving generalization performance using double backpropagation”. In: *IEEE Trans. on Neural Networks* (1992).

[65] Y. Du, Y. Zhang, H. Yu, J. B. Tenenbaum, and J. Wu. “Neural Radiance Flow for 4D View Synthesis and Video Processing”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[66] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. “Adversarially Learned Inference”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[67] S. M. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. C. Rabinowitz, H. King, C. Hillier, M. M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis. “Neural scene representation and rendering”. In: *Science* 360 (2018).

[68] P. Esser, R. Rombach, and B. Ommer. “Taming Transformers for High-Resolution Image Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[69] H. Fan, H. Su, and L. J. Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).

[70] R. P. Feynman, R. B. Leighton, and M. Sands. “The feynman lectures on physics; vol. i”. In: *American Journal of Physics* 33 (1965).

[71] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. “Building Rome on a Cloudless Day”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2010.

[72] Y. Furukawa and J. Ponce. “Accurate, Dense, and Robust Multiview Stereopsis”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2010).

[73] Y. Furukawa and C. Hernandez. “Multi-View Stereo: A Tutorial”. In: *Foundations and Trends in Computer Graphics and Vision* (2013).

[74] M. Gadelha, S. Maji, and R. Wang. “3D Shape Induction from 2D Views of Multiple Objects”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[75] C. Gao, A. Saraf, J. Kopf, and J.-B. Huang. “Dynamic View Synthesis from Dynamic Monocular Video”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[76] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. “Neural Processes”. In: *arXiv.org* 1807.01622 (2018).

[77] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. P. C. Valentin. “FastNeRF: High-Fidelity Neural Rendering at 200FPS”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[78] M. Garland and P. S. Heckbert. “Simplifying surfaces with color and texture using quadric error metrics”. In: *Visualization’98. Proceedings*. IEEE. 1998.

[79] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[80] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* 32 (2013).

[81] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. “Unsupervised Training for 3D Morphable Model Regression”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[82] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. J. Black, and B. Schölkopf. “From Variational to Deterministic Autoencoders”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2020.

[83] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. “Learning a Predictable and Generative Vector Representation for Objects”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[84] R. Gómez-Bombarelli, D. Duvenaud, J. M. Hernández-Lobato, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Central Science* (2018).

[85] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[86] I. J. Goodfellow. "NIPS 2016 Tutorial: Adversarial Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[87] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. "Recurrent Independent Mechanisms". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[88] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[89] K. Greff, R. L. Kaufmann, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. "Multi-Object Representation Learning with Iterative Variational Inference". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[90] F. Groh, P. Wieschollek, and H. P. A. Lensch. "Flex-Convolution (Million-Scale Point-Cloud Learning Beyond Grid-Worlds)". In: *Proc. of the Asian Conf. on Computer Vision (ACCV)*. 2018.

[91] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. "3D-CODED: 3D Correspondences by Deep Deformation". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[92] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[93] J. Gu, L. Liu, P. Wang, and C. Theobalt. "StyleNeRF: A Style-based 3D Aware Generator for High-resolution Image Synthesis". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2022.

[94] K. Guo, D. Zou, and X. Chen. "3D Mesh Labeling via Deep Convolutional Neural Networks". In: *ACM Trans. on Graphics*. 2015.

[95] M. Guo, A. Fathi, J. Wu, and T. Funkhouser. "Object-centric neural scene rendering". In: *arXiv.org* 2012.08503 (2020).

[96] C. Häne, S. Tulsiani, and J. Malik. "Hierarchical Surface Prediction for 3D Object Reconstruction". In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[97] A. Hanson. *Computer vision systems*. Elsevier, 1978.

[98] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[99] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. E. Debevec. “Baking Neural Radiance Fields for Real-Time View Synthesis”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[100] J. Heinly, J. L. Schönberger, E. Dunn, and J. Frahm. “Reconstructing the world in six days”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[101] P. Henderson and V. Ferrari. “Learning single-image 3D reconstruction by generative modelling of shape, pose and shading”. In: *International Journal of Computer Vision (IJCV)* (2019).

[102] P. Henzler, N. J. Mitra, and T. Ritschel. “Escaping Plato’s Cave: 3D Shape From Adversarial Rendering”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[103] P. Henderson and C. H. Lampert. “Unsupervised object-centric video generation and decomposition in 3D”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[104] P. Henderson, V. Tsiminaki, and C. H. Lampert. “Leveraging 2D Data to Learn Textured 3D Mesh Generation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[105] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[106] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[107] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (1997).

[108] P. Huang, K. Matzen, J. Kopf, N. Ahuja, and J. Huang. “DeepMVS: Learning Multi-View Stereopsis”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018.

[109] Y. Huang. “Towards Accurate Marker-Less Human Shape and Pose Estimation over Time”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2017.

[110] K. Hui, R. Li, J. Hu, and C. Fu. “Neural Template: Topology-aware Reconstruction and Disentangled Generation of 3D Meshes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[111] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[112] C. L. Jackins and S. L. Tanimoto. “Oct-trees and their use in representing three-dimensional objects”. In: *Computer Graphics and Image Processing* (1980).

[113] Y. Jeong, S. Ahn, C. B. Choy, A. Anandkumar, M. Cho, and J. Park. “Self-Calibrating Neural Radiance Fields”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[114] T. Jeruzalski, B. Deng, M. Norouzi, J. P. Lewis, G. E. Hinton, and A. Tagliasacchi. “NASA: Neural Articulated Shape Approximation”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[115] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. “Local Implicit Grid Representations for 3D Scenes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[116] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[117] J. T. Kajiya and B. V. Herzen. “Ray tracing volume densities”. In: *ACM Trans. on Graphics*. 1984.

[118] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. “End-to-end Recovery of Human Shape and Pose”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[119] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. “Learning Category-Specific Mesh Reconstruction from Image Collections”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[120] A. Kanazawa, J. Zhang, P. Felsen, and J. Malik. “Learning 3D Human Dynamics from Video”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[121] T. Karras, T. Aila, S. Laine, and J. Lehtinen. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[122] T. Karras, S. Laine, and T. Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[123] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. “Analyzing and Improving the Image Quality of StyleGAN”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[124] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. “Alias-Free Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[125] H. Kato, Y. Ushiku, and T. Harada. “Neural 3D Mesh Renderer”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[126] M. M. Kazhdan, M. Bolitho, and H. Hoppe. “Poisson surface reconstruction”. In: *Eurographics Symposium on Geometry Processing (SGP)*. 2006.

[127] M. M. Kazhdan and H. Hoppe. “Screened poisson surface reconstruction”. In: *ACM Trans. on Graphics* 32 (2013).

[128] T. Kim and T. Adali. “Approximation by Fully Complex Multilayer Perceptrons”. In: *Neural Computation* (2003).

[129] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2014.

[130] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[131] D. P. Kingma, T. Salimans, and M. Welling. “Improving Variational Inference with Inverse Autoregressive Flow”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[132] D. P. Kingma and P. Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[133] T. N. Kipf and M. Welling. “Variational Graph Auto-Encoders”. In: *arXiv.org* 1611.07308 (2016).

[134] M. Kleineberg, M. Fey, and F. Weichert. “Adversarial Generation of Continuous Implicit Shape Representations”. In: *EUROGRAPHICS*. 2020.

[135] I. Kobyzev, S. J. D. Prince, and M. A. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2021).

[136] C. Kong, C.-H. Lin, and S. Lucey. “Using Locally Corresponding CAD Models for Dense 3D Reconstructions From a Single Image”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[137] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende. “NeRF-VAE: A Geometry Aware 3D Scene Generative Model”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2021.

[138] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[139] A. Kundu, Y. Li, and J. M. Rehg. “3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[140] H. Kwak and B. Zhang. “Generating Images Part by Part with Composite Generative Adversarial Networks”. In: *arXiv.org* 1607.05387 (2016).

[141] A. Laurentini. “The Visual Hull Concept for Silhouette-Based Image Understanding”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 16 (1994).

[142] Y. LeCun, C. Cortes, and C. Burges. “MNIST handwritten digit database”. In: *ATT Labs* (2010).

[143] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* (2015).

[144] W. Lee, D. Kim, S. Hong, and H. Lee. “High-Fidelity Synthesis with Disentangled Representation”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[145] V. Leroy, J. Franco, and E. Boyer. “Multi-view Dynamic Shape Refinement Using Local Temporal Integration”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[146] V. Leroy, J. Franco, and E. Boyer. “Shape Reconstruction Using Volume Sweeping and Learned Photoconsistency”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[147] M. Levoy. “Display of surfaces from volume data”. In: *IEEE Computer Graphics and Applications (CGA)* (1988).

[148] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. “PointCNN: Convolution On X-Transformed Points”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[149] T. Li, M. Slavcheva, M. Zollhöfer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, and Z. Lv. “Neural 3D Video Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[150] Y. Liao, S. Donne, and A. Geiger. “Deep Marching Cubes: Learning Explicit Surface Representations”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[151] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger. “Towards Unsupervised Learning of Generative Models for 3D Controllable Image Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[152] C. Lin, W. Ma, A. Torralba, and S. Lucey. “BARF: Bundle-Adjusting Neural Radiance Fields”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[153] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. “Semantic Image Segmentation via Deep Parsing Network”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.

[154] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J. Lien. “Material Editing Using a Physically Based Rendering Network”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[155] S. Liu, W. Chen, T. Li, and H. Li. “Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[156] L. Liu, J. Gu, K. Z. Lin, T. Chua, and C. Theobalt. “Neural Sparse Voxel Fields”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[157] M. Liu, X. Huang, J. Yu, T. Wang, and A. Mallya. “Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications”. In: *Proc. of the IEEE*. 2021.

[158] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. “Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[159] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. “Object-Centric Learning with Slot Attention”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[160] M. M. Loper and M. J. Black. “OpenDR: An Approximate Differentiable Renderer”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.

[161] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. “SMPL: A Skinned Multi-Person Linear Model”. In: *ACM Trans. on Graphics* (2015).

[162] W. E. Lorensen and H. E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *ACM Trans. on Graphics*. 1987.

[163] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision (IJCV)* 60 (2004).

[164] S. Lunz, Y. Li, A. W. Fitzgibbon, and N. Kushman. “Inverse Graphics GAN: Learning to Generate 3D Shapes from Unstructured 2D Data”. In: *arXiv.org* 2002.12674 (2020).

[165] W. Luo, A. Schwing, and R. Urtasun. “Efficient Deep Learning for Stereo Matching”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[166] A. L. Maas, A. Y. Hannun, and A. Y. Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. of the International Conf. on Machine learning (ICML) Workshops*. 2013.

[167] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. “Auxiliary Deep Generative Models”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2016.

[168] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow. “Adversarial Autoencoders”. In: *arXiv.org* 1511.05644 (2015).

[169] J. N. P. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. “Acorn: adaptive coordinate networks for neural scene representation”. In: *ACM Trans. on Graphics* (2021).

[170] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. “NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[171] D. Marr and T. Poggio. “Cooperative Computation of Stereo Disparity: A cooperative algorithm is derived for extracting disparity information from stereo image pairs.” In: *Science* (1976).

[172] D. Maturana and S. Scherer. “VoxNet: A 3D Convolutional Neural Network for real-time object recognition”. In: *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*. 2015.

[173] N. Mayer, E. Ilg, P. Haeusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[174] E. McMullin. “The origins of the field concept in physics”. In: *Physics in Perspective* 4 (2002).

[175] D. Meagher. “Geometric modeling using octree encoding”. In: *Computer Graphics and Image Processing (CGIP)* 19 (1982).

[176] M. Menze and A. Geiger. “Object Scene Flow for Autonomous Vehicles”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[177] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu. “GNeRF: GAN-based Neural Radiance Field without Posed Camera”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[178] L. Mescheder, A. Geiger, and S. Nowozin. “Which Training Methods for GANs do actually Converge?” In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[179] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[180] L. Mescheder. “Stability and Expressiveness of Deep Generative Models”. PhD thesis. University of Tübingen, Germany, 2020.

[181] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. “Implicit Surface Representations as Layers in Neural Networks”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[182] M. Mihajlovic, Y. Zhang, M. J. Black, and S. Tang. “LEAP: Learning Articulated Occupancy of People”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[183] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “NeRF: Representing scenes as neural radiance fields for view synthesis”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[184] M. Mirza and S. Osindero. “Conditional Generative Adversarial Nets”. In: *arXiv.org* 1411.1784 (2014).

[185] J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang. “A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation”. In: *ICCV*. 2021.

[186] T. Müller, A. Evans, C. Schied, and A. Keller. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. on Graphics* (2022).

[187] J. Munkberg, W. Chen, J. Hasselgren, A. Evans, T. Shen, T. Müller, J. Gao, and S. Fidler. “Extracting Triangular 3D Models, Materials, and Lighting From Images”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[188] A. Mustafa, H. Kim, J. Guillemaut, and A. Hilton. “General Dynamic Scene Reconstruction from Multiple View Video”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.

[189] A. Mustafa, H. Kim, J. Guillemaut, and A. Hilton. “Temporally Coherent 4D Reconstruction of Complex Dynamic Scenes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[190] A. Myronenko and X. B. Song. “Point Set Registration: Coherent Point Drift”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 32 (2010).

[191] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2010.

[192] S. K. Nayar, M. Watanabe, and M. Noguchi. “Real-Time Focus Range Sensor”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 1995.

[193] J. Neumann and Y. Aloimonos. “Spatio-Temporal Stereo Using Multi-Resolution Subdivision Surfaces”. In: *International Journal of Computer Vision (IJCV)* (2002).

[194] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. “RenderNet: A deep convolutional network for differentiable rendering from 3D shapes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[195] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y. Yang. “HoloGAN: Unsupervised Learning of 3D Representations From Natural Images”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[196] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y. Yang, and N. Mitra. “BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[197] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. “Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[198] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. “Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[199] M. Niemeyer and A. Geiger. “CAMPARI: Camera-Aware Decomposed Generative Neural Radiance Fields”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2021.

[200] M. Niemeyer and A. Geiger. “GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[201] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. M. Sajjadi, A. Geiger, and N. Radwan. “RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[202] A. Noguchi, X. Sun, S. Lin, and T. Harada. “Neural Articulated Radiance Field”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[203] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. “Texture Fields: Learning Texture Representations in Function Space”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[204] M. Oechsle, M. Niemeyer, C. Reiser, L. Mescheder, T. Strauss, and A. Geiger. “Learning Implicit Surface Light Fields”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2020.

[205] M. Oechsle, S. Peng, and A. Geiger. “UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[206] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. “Deep metric learning via lifted structured feature embedding”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[207] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[208] M. R. Oswald, J. Stühmer, and D. Cremers. “Generalized Connectivity Constraints for Spatio-temporal 3D Reconstruction”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.

[209] G. Papamakarios, I. Murray, and T. Pavlakou. “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[210] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. “Normalizing Flows for Probabilistic Modeling and Inference”. In: *Journal of Machine Learning Research (JMLR)* (2021).

[211] K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. “PhotoShape: Photorealistic Materials for Large-Scale Shape Collections”. In: *Communications of the ACM* (2018).

[212] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[213] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. "Nerfies: Deformable Neural Radiance Fields". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[214] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz. "HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields". In: *ACM Trans. on Graphics* (2021).

[215] D. Paschalidou, A. O. Ulusoy, C. Schmitt, L. van Gool, and A. Geiger. "RayNet: Learning Volumetric 3D Reconstruction with Ray Potentials". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[216] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. "A 3D Face Model for Pose and Illumination Invariant Face Recognition". In: *Proc. of International Conf. on Advanced Video and Signal Based Surveillance (AVSS)*. 2009.

[217] W. S. Peebles, J. Peebles, J. Zhu, A. A. Efros, and A. Torralba. "The Hessian Penalty: A Weak Prior for Unsupervised Disentanglement". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[218] Y. Pekelny and C. Gotsman. "Articulated Object Reconstruction and Markerless Motion Capture from Depth Video". In: *Computer Graphics Forum* (2008).

[219] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. "Convolutional Occupancy Networks". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[220] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou. "Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[221] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger. "Shape As Points: A Differentiable Poisson Solver". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[222] A. Pentland. "Local Shading Analysis". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (1984).

[223] F. Petersen, A. H. Bermano, O. Deussen, and D. Cohen-Or. "Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer". In: *arXiv.org* 1903.11149 (2019).

[224] L. Pishchulin, S. Wuhrer, T. Helten, C. Theobalt, and B. Schiele. "Building statistical shape spaces for 3D human modeling". In: *Pattern Recognition* (2017).

[225] M. Pollefeys. "Detailed Real-Time Urban 3D Reconstruction from Video". In: *International Journal of Computer Vision (IJCV)* 78 (2008).

[226] L. S. Pontryagin, V. G. Boltyanshii, R. V. Gamkrelidze, and E. F. Mishenko. *The Mathematical Theory of Optimal Processes*. John Wiley and Sons, 1962.

[227] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. “Dreamfusion: Text-to-3d using 2d diffusion”. In: *arXiv.org* 2209.14988 (2022).

[228] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. “D-NeRF: Neural Radiance Fields for Dynamic Scenes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[229] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. “Volumetric and Multi-View CNNs for Object Classification on 3D Data”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[230] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[231] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[232] A. Radford, L. Metz, and S. Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2016.

[233] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. C. Courville. “On the Spectral Bias of Neural Networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[234] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. “Generating 3D faces using Convolutional Mesh Autoencoders”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[235] S. Ravanbakhsh, F. Lanusse, R. Mandelbaum, J. G. Schneider, and B. Póczos. “Enabling Dark Energy Science with Deep Generative Models of Galaxy Images”. In: *Proc. of the Conf. on Artificial Intelligence (AAAI)*. 2017.

[236] A. Razavi, A. van den Oord, and O. Vinyals. “Generating Diverse High-Fidelity Images with VQ-VAE-2”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[237] S. Reed, K. Sohn, Y. Zhang, and H. Lee. “Learning to disentangle factors of variation with manifold interaction”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2014.

[238] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2014.

[239] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. “Unsupervised Learning of 3D Structure from Images”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[240] E. Richardson, M. Sela, R. Or-El, and R. Kimmel. “Learning Detailed Face Reconstruction from a Single Image”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[241] G. Riegler, A. O. Ulusoy, and A. Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[242] J. Romero, D. Tzionas, and M. J. Black. “Embodied hands: modeling and capturing hands and bodies together”. In: *ACM Trans. on Graphics* (2017).

[243] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015.

[244] S. Roweis. “EM Algorithms for PCA and SPCA”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1997.

[245] W. Rudin et al. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York, 1964.

[246] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[247] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. “PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[248] S. Saito, J. Yang, Q. Ma, and M. J. Black. “SCANimate: Weakly Supervised Learning of Skinned Clothed Avatar Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[249] M. S. M. Sajjadi, D. Duckworth, A. Mahendran, S. van Steenkiste, F. Pavetić, M. Lučić, L. J. Guibas, K. Greff, and T. Kipf. “Object Scene Representation Transformer”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[250] M. S. M. Sajjadi, H. Meyer, E. Pot, U. Bergmann, K. Greff, N. Radwan, S. Vora, M. Lucic, D. Duckworth, A. Dosovitskiy, J. Uszkoreit, T. A. Funkhouser, and A. Tagliasacchi. “Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[251] A. Sauer, K. Schwarz, and A. Geiger. “StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets”. In: *ACM Trans. on Graphics*. 2022.

[252] F. Schaffalitzky and A. Zisserman. “Multi-view Matching for Unordered Image Sets, or ”How Do I Organize My Holiday Snaps?””. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2002.

[253] J. L. Schönberger, F. Radenovic, O. Chum, and J. Frahm. “From single image query to detailed 3D reconstruction”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[254] J. L. Schönberger and J.-M. Frahm. “Structure-from-Motion Revisited”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[255] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. “GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[256] K. Schwarz, A. Sauer, M. Niemeyer, Y. Liao, and A. Geiger. “VoxGRAF: Fast 3D-Aware Image Synthesis with Sparse Voxel Grids”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[257] S. Seitz and C. Dyer. “Photorealistic scene reconstruction by voxel coloring”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 1997.

[258] T. Shen, J. Gao, K. Yin, M. Liu, and S. Fidler. “Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[259] J. Shen, A. Agudo, F. Moreno-Noguer, and A. Ruiz. “Conditional-Flow NeRF: Accurate 3D Modelling with Reliable Uncertainty Quantification”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2022.

[260] Z. Shi, S. Peng, Y. Xu, Y. Liao, and Y. Shen. “Deep Generative Models on 3D Representations: A Survey”. In: *arXiv.org* 2210.15663 (2022).

[261] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[262] E. Smith, S. Fujimoto, and D. Meger. “Multi-View Silhouette and Depth Decomposition for High Resolution 3D Object Representation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[263] N. Snavely, S. M. Seitz, and R. Szeliski. “Photo Tourism: Exploring Photo Collections in 3D”. In: *ACM Trans. on Graphics*. 2006.

[264] S. Song and J. Xiao. “Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[265] L. Song, A. Chen, Z. Li, Z. Chen, L. Chen, J. Yuan, Y. Xu, and A. Geiger. “NeRF-Player: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields”. In: *arXiv.org* 2210.15947 (2022).

[266] E. S. Spelke and K. D. Kinzler. “Core knowledge”. In: *Developmental science* (2007).

[267] J. Starck and A. Hilton. “Surface Capture for Performance-Based Animation”. In: *IEEE Computer Graphics and Applications (CGA)* (2007).

[268] K. Stelzner, K. Kersting, and A. R. Kosiorek. “Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation”. In: *arXiv.org* 2104.01148 (2021).

[269] D. Stutz and A. Geiger. “Learning 3D Shape Completion from Laser Scan Data with Weak Supervision”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[270] S.-Y. Su, F. Yu, M. Zollhöfer, and H. Rhodin. “A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[271] R. Szeliski. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011.

[272] R. Szeliski. “Rapid octree construction from image sequences”. In: *CVGIP: Image understanding* (1993).

[273] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. T. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. “Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[274] M. Tancik, 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.

[275] M. Tatarchenko, A. Dosovitskiy, and T. Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[276] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. “What Do Single-View 3D Reconstruction Networks Learn?” In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[277] G. Teschl. *Ordinary differential equations and dynamical systems*. American Mathematical Soc., 2012.

[278] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt. “MoFA: Model-Based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[279] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. “Self-Supervised Multi-Level Face Model Learning for Monocular Reconstruction at Over 250 Hz”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[280] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. “KPConv: Flexible and Deformable Convolution for Point Clouds”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[281] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[282] M. E. Tipping and C. M. Bishop. “Probabilistic Principal Component Analysis”. In: *Journal of the Royal Statistical Society (JRSS)* 61 (1999).

[283] G. Tiwari, N. Sarafianos, T. Tung, and G. Pons-Moll. “Neural-GIF: Neural Generalized Implicit Functions for Animating People in Clothing”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[284] E. Tola, C. Strecha, and P. Fua. “Efficient large-scale multi-view stereo for ultra high-resolution image sets”. In: *Machine Vision and Applications (MVA)* (2012).

[285] I. O. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. “Wasserstein Auto-Encoders”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[286] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. “Multi-view supervision for single-view reconstruction via differentiable ray consistency”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[287] H. Tung, H. Tung, E. Yumer, and K. Fragkiadaki. “Self-supervised Learning of Motion Capture”. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

[288] S. Uchida. “Text Localization and Recognition in Images and Video”. In: *Handbook of Document Image Processing and Recognition*. 2014.

[289] S. Ullman. *The Interpretation of Structure from Motion*. Vol. 203. The Royal Society, 1979.

[290] A. O. Ulusoy, O. Biris, and J. L. Mundy. “Dynamic Probabilistic Volumetric Models”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2013.

[291] A. O. Ulusoy and J. L. Mundy. “Image-based 4-d Reconstruction Using 3-d Change Detection”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.

[292] A. O. Ulusoy, A. Geiger, and M. J. Black. “Towards Probabilistic Volumetric Reconstruction using Ray Potentials”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2015.

[293] A. Vahdat and J. Kautz. “NVAE: A Deep Hierarchical Variational Autoencoder”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[294] S. Vedula, P. Rander, R. T. Collins, and T. Kanade. “Three-dimensional scene flow”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2005).

[295] S. Vedula, S. Baker, P. Rander, R. T. Collins, and T. Kanade. “Three-dimensional scene flow”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 1999.

[296] D. Verbin, P. Hedman, B. Mildenhall, T. E. Zickler, J. T. Barron, and P. P. Srinivasan. “Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[297] C. Villani. *Topics in optimal transportation*. Vol. 58. American Mathematical Society, 2003.

[298] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. “Modulating early visual processing by language”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[299] H. Wallach and D. O’Connell. “The kinetic depth effect”. In: *Journal of Experimental Psychology* (1953).

[300] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. J. Guibas, and A. Schilling. “Reconstruction of deforming geometry from time-varying point clouds”. In: *Eurographics Symposium on Geometry Processing (SGP)*. 2007.

[301] X. Wang and A. Gupta. “Generative Image Modeling Using Style and Structure Adversarial Networks”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[302] P. Wang, Y. Gan, Y. Zhang, and P. Shui. “3D Shape Segmentation via Shape Fully Convolutional Networks”. In: *Computers & Graphics* (2017).

[303] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. “Pixel2mesh: Generating 3d mesh models from single rgb images”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[304] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. “High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[305] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu. “Transferring gans: generating images from limited data”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[306] W. Wang, X. Qiangeng, D. Ceylan, R. Mech, and U. Neumann. “DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[307] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. *NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction*. 2021.

[308] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. “NeRF–: Neural Radiance Fields Without Known Camera Parameters”. In: *arXiv.org* 2102.07064 (2021).

[309] C. Weng, B. Curless, P. P. Srinivasan, J. T. Barron, and I. Kemelmacher-Shlizerman. “HumanNeRF: Free-viewpoint Rendering of Moving People from Monocular Video”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[310] C. S. Wheatstone. “XVIII. Contributions to the physiology of vision. —Part the first. On some remarkable, and hitherto unobserved, phenomena of binocular vision”. In: *Philosophical Transactions of the Royal Society of London* (1838).

[311] F. Williams, Z. Gojcic, S. Khamis, D. Zorin, J. Bruna, S. Fidler, and O. Litany. “Neural Fields as Learnable Kernels for 3D Reconstruction”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[312] P. H. Winston, B. Horn, M. Minsky, Y. Shirai, and D. Waltz. *The psychology of computer vision*. McGraw-Hill Companies, 1975.

[313] P. H. Winston. *Heterarchy in the M.I.T. Robot*. Tech. rep. MIT Artificial Intelligence Laboratory, 1971.

[314] K.-Y. Wong and R. Cipolla. “Structure and motion from silhouettes”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2001.

[315] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. “3D ShapeNets: A deep representation for volumetric shapes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[316] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[317] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. “MarrNet: 3D shape reconstruction via 2.5D sketches”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[318] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum. “Learning Shape Priors for Single-View 3D Completion and Reconstruction”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[319] C. Wu. “Towards Linear-time Incremental Structure from Motion”. In: *Proc. of the International Conf. on 3D Vision (3DV)*. 2013.

[320] W. Xia and J.-H. Xue. “A Survey on 3D-aware Image Synthesis”. In: *arXiv.org* 2210.14267 (2022).

[321] J. Xiang, J. Yang, Y. Deng, and X. Tong. “GRAM-HD: 3D-Consistent Image Generation at High Resolution with Generative Radiance Manifolds”. In: *arXiv.org* 2206.07255 (2022).

[322] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. “Neural Fields in Visual Computing and Beyond”. In: *Computer Graphics Forum* (2022).

[323] B. Xu, N. Wang, T. Chen, and M. Li. “Empirical Evaluation of Rectified Activations in Convolutional Network”. In: *arXiv.org* 1505.00853 (2015).

[324] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. “DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[325] Y. Xue, Y. Li, K. K. Singh, and Y. J. Lee. “GIRAFFE HD: A High-Resolution 3D-aware Generative Model”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[326] J. Yang and H. Li. “Dense, Accurate Optical Flow Estimation with Piecewise Parametric Model”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[327] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. “Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[328] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui. "Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[329] B. Yang, C. Bao, J. Zeng, H. Bao, Y. Zhang, Z. Cui, and G. Zhang. "NeuMesh: Learning Disentangled Neural Mesh-Based Implicit Field for Geometry and Texture Editing". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2022.

[330] G. Yang, M. Vo, N. Natalia, D. Ramanan, V. Andrea, and J. Hanbyul. "BANMo: Building Animatable 3D Neural Models from Many Casual Videos". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[331] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. "MVSNet: Depth Inference for Unstructured Multi-view Stereo". In: *Proc. of the European Conf. on Computer Vision (ECCV)* (2018).

[332] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. "Recurrent MVSNet for High-resolution Multi-view Stereo Depth Inference". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019).

[333] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman. "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[334] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. "Volume rendering of neural implicit surfaces". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[335] Y. Yazici, C. Foo, S. Winkler, K. Yap, G. Piliouras, and V. Chandrasekhar. "The Unusual Effectiveness of Averaging in GAN Training". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[336] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop". In: *arXiv.org* 1506.03365 (2015).

[337] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. "PlenOctrees for Real-time Rendering of Neural Radiance Fields". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[338] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou. "PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[339] H. Yu, L. J. Guibas, and J. Wu. "Unsupervised Discovery of Object Radiance Fields". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2022.

[340] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu. "Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[341] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger. “MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[342] W. Yuan, Z. Lv, T. Schmidt, and S. Lovegrove. “STaR: Self-Supervised Tracking and Reconstruction of Rigid Objects in Motion With Neural Rendering”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[343] J. Žbontar and Y. LeCun. “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”. In: *Journal of Machine Learning Research (JMLR)* 17 (2016).

[344] L. Zhang, B. Curless, A. Hertzmann, and S. M. Seitz. “Shape and Motion under Varying Illumination: Unifying Structure from Motion, Photometric Stereo, and Multi-view Stereo”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2003.

[345] W. Zhang, J. Sun, and X. Tang. “Cat Head Detection - How to Effectively Exploit Shape and Texture Features”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2008.

[346] S. Zhao, J. Song, and S. Ermon. “Towards Deeper Understanding of Variational Autoencoding Models”. In: *arXiv.org* 1702.08658 (2017).

[347] X. Zhang, Z. Zhang, C. Zhang, J. B. Tenenbaum, W. T. Freeman, and J. Wu. “Learning to Reconstruct Shapes from Unseen Classes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[348] B. Zhao, B. Chang, Z. Jie, and L. Sigal. “Modular Generative Adversarial Networks”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[349] Q. Zheng, X. Fan, M. Gong, A. Sharf, O. Deussen, and H. Huang. “4D Reconstruction of Blooming Flowers”. In: *Computer Graphics Forum* (2017).

[350] E. D. Zhong, T. Bepler, J. H. Davis, and B. Berger. “Reconstructing continuous distributions of 3D protein structure from cryo-EM images”. In: *Proc. of the International Conf. on Learning Representations (ICLR)* (2020).

[351] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. “Learning a Discriminative Model for the Perception of Realism in Composite Images”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.

[352] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[353] J. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman. “Visual Object Networks: Image Generation with Disentangled 3D Representations”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[354] J. Zienkiewicz, A. J. Davison, and S. Leutenegger. “Real-time height map fusion using differentiable rendering”. In: *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*. 2016.