SELF-SUPERVISED SCENE REPRESENTATION LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Vincent Sitzmann
May 2020

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Gordon Wetzstein) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Leonidas Guibas)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Brian Wandell)

Approved for the Stanford University Committee on Graduate Studies

Preface

In this thesis, *Self-supervised Scene Representation Learning*, we propose novel approaches to enable artificial intelligence models to infer representations of 3D environments conditioned exclusively on posed images.

- We propose to exploit 3D-structured feature spaces in the form of voxelgrids of features, combined with a principled image formation model that enables occlusion reasoning. This method enables high-quality novel-view synthesis of real-world environments, and recovers geometry in an unsupervised manner.
- We propose a continuous, 3D-structure aware neural scene representation, Scene Representation Networks (SRNs). This enables the learning of priors over 3D environments and thereby, full 3D reconstruction of simple scenes from few observations.
- We demonstrate that the features discovered by SRNs in a self-supervised manner contain semantic information that can be leveraged for dense 3D semantic segmentation with very little labeled data in a semis-supervised learning regime.

This dissertation is based on the following publications, all of which are collaborative works. For each publication, the contributions of each author are listed.

Sitzmann, V., Thies, J., Heide, F., Niessner, M., Wetzstein, G., and Zollhöfer, M., Deepvoxels: Learning persistent 3d feature embeddings, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, ©IEEE, 2019 [1]. This work is covered in Chapter 2. The initial idea was conceived by Dr. Zollhöfer, Dr. Heide, and the author. The scene representation and rendering algorithm were developed by the author. All the authors contributed equally to the writing of the final published article.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G., Scene representation networks: Continuous 3D-structure-aware neural scene representations, Advances in Neural Information Processing Systems, 2019 [2]. This work is covered in Chapter 3. The initial idea was conceived by Prof. Wetzstein, Dr. Zollhöfer, and the author. The scene representation, generalization scheme, and rendering algorithm

were developed by the author. The author wrote a significant portion of the published article that all authors took part in writing.

Kohli, A., Sitzmann, V., and Wetzstein, G., Inferring Semantic Information with 3D Neural Scene Representations. arXiv, 2020 [3]. This work is covered in Chapter 4. The initial idea was conceived by the author. Amit Kohli and the author jointly developed the semi-supervised training scheme for semantic segmentation with scene representation networks. The published manuscript was written by all authors.

Acknowledgments

I would like to thank a number of people who have supported me during my Ph.D.

First and foremost, I would like to thank my adviser, Prof. Gordon Wetzstein, who offered me a position in his research group at Stanford. I am most grateful for Gordon's support of my own research interest, which slowly moved away from computational imaging and displays towards 3D computer vision, artificial intelligence, and neural scene representations. Though notably different from the core research direction of his lab, Gordon nevertheless was the most effective and supportive adviser one could hope for, with an amazing capability of identifying key research questions and their applications across fields, as well as the mental capacity to quickly pick up knowledge that allowed him to provide effective guidance. He not only advised me effectively with respect to my research, but also my professional development, project management, and student supervision. I greatly admire Gordon's leadership and his unwavering commitment to provide his students with the best research environment possible.

I would also like to extend special thanks to Dr. Michael Zollhöfer, who served as an amazing mentor and collaborator during my first steps in 3D computer vision, computer graphics, and machine learning. Michael is not only an amazing researcher, but also one of the most humble, supportive and knowledgeable collaborators I had the pleasure to interact with.

During my time at Stanford, I interacted most with Dr. Robert Konrad, Dr. Nitish Padmanaban, Dr. Julien Martel, Dr. Kevin Boyle, and Gabe Buckmaster. I am grateful not only for many fruitful and thought-provoking discussions, but also for their continuous support as friends.

I would like to thank Prof. Noah Snavely and Richard Tucker for hosting me at Google AI in New York City during the summer of 2019. Noah's work was a key inspiration for my passion for 3D computer vision, and it was a privilege collaborating on questions in generalization and compositionality in implicit neural representations.

I'd like to thank my parents and my sister for their continuing and loving support, for being amazing role models of great human beings, and for the best upbringing one could wish for.

Lastly, I would like to thank all my collaborators and colleagues that share my passion for this line of work, and I am looking forward to many more illuminating discussions in the future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

Humans have an extraordinary understanding of their physical environment. This is a cornerstone of intelligent behavior, and fundamental to our everyday lives. Central to this skill is *scene representation*. Scene representation is the process of converting observations of an environment—visual, haptic, auditory, or otherwise—into a compact model of the environment [7].

The field of *artificial intelligence* has long sought to reproduce the process of scene representation. Besides its role in general intelligence, it is also a key building block of applications of computer science that require interaction or interpretation of observations captured in our world, such as autonomous navigation and robotic grasping. Its applications further extend to fields such as computer graphics, where we are interested in discovering compact representations of 3D scenes and image formation models to improve speed and accuracy of rendering algorithms. The purpose of this thesis is to advance the field of machine learning of scene representations, towards artificial intelligence that can infer accurate models of 3D environments.

1.1 Self-supervised Scene Representation Learning

We are interested in algorithms that ingest image observations of an environment and convert them into a compact representation that encodes the environment's properties such that they may be leveraged for other downstream tasks. To this end, we leverage Deep Neural Networks (DNNs), a powerful new class of algorithms. From skin cancer detection [8] to image classification [9] to scene segmentation, object detection, and tracking in autonomous vehicles—DNNs often outperform conventional algorithms by a large margin.

In all of these examples, a DNN must infer some property of an environment, given a set of incomplete observations. DNNs solve such problems by transforming observations into a feature representation of the scene [10]. Such a neural scene representation [11] allows the DNN to reason

about the environment, including previously unobserved information. We note that the word "neural" in this context does *not* imply any connections to the nervous system, but is rather alluding to feature-based representations learned by neural networks.

Today, DNNs with seemingly unlimited capacity to learn almost anything excel in *supervised* settings: tasks where massive, human-labeled datasets are readily available for training. However, collecting and annotating large-scale datasets for many different tasks is costly, error-prone, time-consuming, and in many cases entirely infeasible. Moreover, the resulting representations are biased by human decision-making and are specialized to a specific task.

In contrast, humans and other intelligent organisms require little to no supervision to learn to interact with their environment. Yet, the compact representations they learn generalize to a variety of tasks, and they can easily integrate observations from a variety of senses such as touch, vision, and sound. It is thus desirable to build models that learn rich neural scene representations only from observations of scenes, without human-labeled datasets.

In this work, we will discuss models that perform such self-supervised learning of neural scene representations, by virtue of being generative models. Generative models are DNNs that are capable of generating output in the same domain as the observations—such as images. They can be trained given only a dataset of raw observations: the model is fed all but one observations and trained to predict the held-out observation. If a generative model succeeds at generating accurate observations of an environment unseen at training time, this is strong evidence that it has learned a neural scene representation that captures the latent structure of the data.

1.1.1 Formulation

The models we will discuss in this work will consist of three parts: An observation encoder, a neural scene representation, and a neural renderer. Please see Figure 1.1 for a visualization of the algorithm.

Training data. We will often collect a dataset $\mathcal{C}$ of observations of a single scene, and then consider a meta-dataset containing several such datasets in Chapter 3. Presently, the complexity of 3D scenes that are tractable in this framework is limited. In this work, we will consider 3D scenes that range in complexity from single objects to simple, synthetic room environments. Generally, we assume visual observations, where each observation is a tuple $(\mathcal{I}, \mathbf{E}, \mathbf{K})$ of an image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$ along with its respective extrinsic $\mathbf{E} = \big[\mathbf{R}|\mathbf{t}\big] \in \mathbb{R}^{3\times 4}$ and intrinsic $\mathbf{K} \in \mathbb{R}^{3\times 3}$ camera matrices [12]. The dataset $\mathcal{C}$ with N observations is thus of the form:

$$\mathcal{C} = \{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^{N} \tag{1.1}$$

In this work, we assume that camera extrinsic and intrinsic parameters are available. For real captures, these can be obtained using sparse bundle adjustment [13].

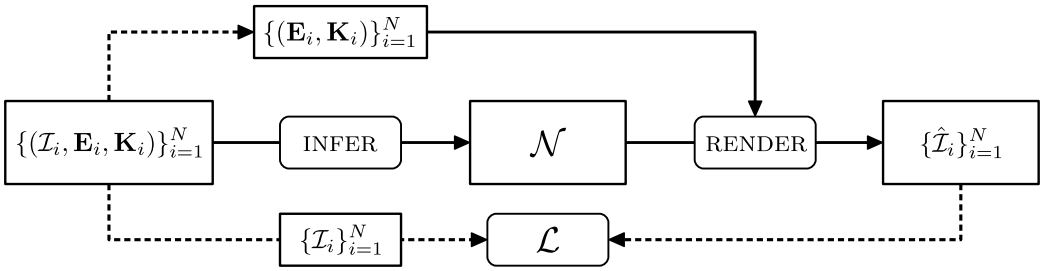

Figure 1.1: Visualization of proposed self-supervised scene representation learning framework. At training time (continuous and dotted lines), we infer a neural scene representation $\mathcal{N}$ from a set of images $\mathcal{I}_i$ and their camera parameters $\mathbf{E}_i$ and $\mathbf{K}_i$. Using a differentiable renderer RENDER, we then re-render images in the dataset $\hat{\mathcal{I}}_i$, allowing us to enforce a 2D re-rendering loss $\mathcal{L}$ and train the full model end-to-end using only 2D posed images. At test time (continuous lines only), we may render arbitrary camera perspectives (see Chapter 2 and Chapter 3) or leverage inferred neural scene representations $\mathcal{N}$ for other downstream tasks, such as semantic segmentation (see Chapter 4).

Neural Scene Representation. We first formalize the neural scene representation itself. In this work, we investigate *distributed* representations [10], i.e., representations learned as activations of neural networks in the process of optimization via backpropagation [14]. We thus formalize a neural scene representation $\mathcal{N}$ simply as a k-dimensional vector of real numbers, $\mathcal{N} \in \mathbb{R}^k$. We note that the word "neural" in this context does *not* imply any connections to the nervous system, but is rather intended to indicate that these representations are learned by neural networks.

Observation encoder. The observation encoder is an algorithm that maps a set of M observations to a representation $\mathcal{N}$ of the underlying scene. In its most general form, it is described by the function:

$$\text{INFER} : \{\mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{3\times 4} \times \mathbb{R}^{3\times 3}\}_M \mapsto \mathbb{R}^k, \quad \text{INFER}(\{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^M) = \mathcal{N}. \tag{1.2}$$

Neural Renderer. The neural renderer is an algorithm that maps a neural scene representation $\mathcal{N}$ as well as intrinsic and extrinsic camera parameters to an estimate of the respective perspective image $\hat{\mathcal{I}}$:

$$\text{RENDER} : \mathbb{R}^k \times \mathbb{R}^{3\times 4} \times \mathbb{R}^{3\times 3} \mapsto \mathbb{R}^{H\times W\times 3}, \quad \text{RENDER}(\mathcal{N}, \mathbf{E}, \mathbf{K}) = \hat{\mathcal{I}}. \tag{1.3}$$

Re-Rendering Loss. Finally, these three building blocks allow us to "close the loop" and obtain a training signal as the error $\mathcal{L}$ defined between a ground-truth image observation in our dataset, $\mathcal{I}$, and its estimate $\hat{\mathcal{I}} = \text{RENDER}(\mathcal{N}, \mathbf{E}, \mathbf{K})$

$$\mathcal{L}(\hat{\mathcal{I}}, \mathcal{I}) = \mathcal{L}(\text{RENDER}(\mathcal{N}, \mathbf{E}, \mathbf{K}), \mathcal{I}) \tag{1.4}$$

1.1.2 Example Scene Representation Learner

A simple example of a scene representation learner is a classic convolutional auto-encoder, extended by control over the target camera pose, as previously proposed by Tatarchenko et al. [4]. The observation encoder is implemented as a convolutional encoder that ingests a single image and outputs a latent vector $\mathbf{z} \in \mathbb{R}^{4096}$, the neural scene representation. The inferred latent vector is then concatenated with a neural representation inferred from the extrinsic camera parameters $\mathbf{E}$ of the perspective we aim to render. Finally, this vector is decoded into an image via a convolutional decoder with transpose convolutions. The rendering function can therefore be expressed as the concatenation of the neural scene representation with the camera parameter embedding and subsequent decoding via the convolutional decoder.

We note that this model makes very limited assumptions on the structure of INFER, RENDER, and the space of neural scene representations itself. This is because, as convolutional neural networks, both the encoder and decoder may in principle parameterize *any* function of their inputs and outputs. A core question of this work is the investigation whether such unconstrained scene representation learners can successfully infer the 3D structure of the scene underlying the input observations, or whether it is beneficial to equip the learner with additional assumptions about the training data.

1.1.3 The Role of Inductive Biases

An inductive bias is defined as the set of assumptions that a machine learning algorithm makes about the data it encounters [15]. In this work, we will see that inductive biases on the 3D structure of a scene are essential to scene representation learners. This becomes apparent when considering the general formulation of RENDER, and INFER. For a given dataset $\mathcal{C}$, there may exist several different (potentially infinitely many) combinations of the functions RENDER and INFER, with a space of neural scene representations that is only loosely constrained, that perfectly explain the training set. However, the image formation process of cameras is well understood, and it is known that a model that successfully renders observations of 3D-dimensional environments has to observe multi-view and perspective geometry [12].

Yet, as a powerful universal function approximator, an unconstrained learner may overfit on the training data, rote-memorizing observations instead of discovering fundamental structure of the underlying environments. Optimization may also fall into local optima, where the generative model has found a suboptimal heuristic to explain training data. In this work, we demonstrate that such unconstrained models fail to generate observations with camera intrinsic and extrinsic parameters unseen at training time.

To alleviate this, we propose models that incorporate inductive biases about multi-view and perspective geometry. Formally, this is equivalent to making assumptions about the structure of RENDER, INFER, and the space of neural scene representations itself.

1.2 Related Work

Computer vision and artificial intelligence researchers have proposed a variety of scene representations towards accurate reconstruction of 3D environments and their properties. Here, we review prior work.

Model-based reconstruction. Classic reconstruction approaches such as structure-from-motion exploit multi-view geometry [12, 16] to build dense 3D point clouds [17–21], voxel grids [22], or height-map based representations [23, 24] of the imaged environment. However, these approaches are constrained by multi-view geometry only, and do not leverage learned priors over scene representation parameters. As a result, they require densely sampled observations of a scene to infer its 3D structure. Furthermore, the features encoded in the final representations are hand-crafted by experts, and are thus constrained in which information they may represent. This motivates scene representation learning as a potential path towards reconstruction of scene properties from few observations, including such properties that may be difficult to model analytically, such as material properties, affordance, or semantic information.

2D representation learning. Representation learning—the learning of feature representations that generalize across tasks without labels, given only observations—is a long-standing challenge in machine learning. A large body of recent work explores self-supervised representation learning on images. Here, the weights of a convolutional neural network (CNN) are sought such that the CNN extracts either a single or a per-pixel feature representation of the input image. The model is trained for a surrogate objective in an unsupervised manner on a large dataset of images. Such surrogate objectives may be 2D generative modeling [25–27], predicting representations of unseen pixels given a set of context pixels [28, 29], enforcing of invariance of image class to data augmentation [30], predicting image rotation [31], predicting image color from grayscale [32, 33], wide-baseline matching [34], predicting of or equivariance to ego-motion [35, 36], or predicting the spatial layout of an image [37]. The resulting representations can then be leveraged in downstream tasks, such as 2D bounding box detection, 2D image segmentation, or image classification, and have achieved impressive results in these domains. However, the representations regressed by these models are fundamentally 2D-structured, in that they regress either image- or pixel-wise representations. This lack of 3D inductive bias makes these approaches incapable of reasoning about multi-view consistency or scene parts occluded in the input image. Current 2D representation learning techniques are therefore incapable of supporting 3D tasks.

Geometric deep learning. Geometric deep learning has explored various representations to infer scene geometry from sparse observations. Discretization-based techniques leverage machine learning to parameterize classic computer graphics representations such as voxel grids [38–45], octree

hierarchies [46–48], point clouds [49–51], multiplane images [52], patches [53], or meshes [44, 54–56]. Methods based on function spaces continuously represent space as the decision boundary of a learned binary classifier [57] or a continuous signed distance field [58–60]. While these techniques are successful at modeling geometry, they often require 3D supervision, and it is unclear how to efficiently infer and represent appearance and other scene properties. In this work, we explore neural scene representations that learn to infer appearance, geometry, and semantic information, given only posed 2D images.

Deep-learning scene representations. The artificial intelligence community has long sought to build models that reproduce human reasoning about 3D environments, given only 2D image supervision, and has proposed a variety of feature-based representations to this end. Following seminal work by Eslami et al. [11], we use the term "neural scene representation" to refer to such feature-based scene representations where features are learned by a neural network, but note that the word "neural" in this context does not indicate a relationship to the nervous system. Latent codes of autoencoders may be interpreted as a feature representation of the underlying scene. Such models first leverage a convolutional image encoder to regress a latent code. Novel views may then be rendered by concatenating target pose and latent code [4] or performing view transformations directly in the latent space [5], and decoding the resulting representation using a convolutional decoder. Generative Query Networks [11, 61] aggregate latent codes from several image observations into a single scene representation via averaging, and add a probabilistic reasoning framework that models uncertainty due to incomplete observations. Such convolutional encoder-decoder based models, however, are not equipped with explicit knowledge of the 3D structure of the underlying scenes, or the multi-view and projective geometry governing the image formation process, and are instead expected to discover 3D structure from data. Graphs may be leveraged for scene representation, by attaching a 3D coordinate as well as a learned feature to a graph node and leveraging graph neural networks for processing [62]. While this approach equips the representation with a notion of locality, 3D structure remains coarse, and encoder and decoder remain oblivious to the image formation model. Compositional and repetitive structure may be modeled by representing scenes as programs [63] that are inferred from images by a convolutional encoder and a sequence-to-sequence Long short-term memory [64]. This approach, however, reasons about assemblies of monolithic, predefined objects, and does not learn feature representations of objects, their shape or appearance. In this work, we investigate models that learn feature representations of scene properties such as geometry, shape, and semantic class given only 2D posed images. We show that inductive biases on multi-view geometry and perspective image formation are critical for correctly inferring the 3D structure of the underlying scene, and investigate approaches to equipping learners with such inductive biases.

Neural image synthesis. Deep models for 2D image and video synthesis have recently shown promising results in generating photorealistic images. Some of these approaches are based on (variational) auto-encoders [65, 66], generative flows [67, 68], or autoregressive per-pixel models [69, 70]. In particular, generative adversarial networks [71–75] and their conditional variants [6, 76, 77] have recently achieved photo-realistic single-image generation. Compositional Pattern Producing Networks [78, 79] learn multi-layer perceptrons that directly map 2D image coordinates to color. Some approaches build on explicit spatial or perspective transformations in the networks [80–83]. Recently, following the spirit of "vision as inverse graphics" [84, 85], deep neural networks have been applied to the task of inverting graphics engines [86–90]. However, these 2D generative models only learn to parameterize the manifold of 2D natural images, and struggle to generate images that are multi-view consistent, since the underlying 3D scene structure cannot be exploited.

1.3 Research Questions and Contributions

We formulate the research questions and contributions as follows:

Research Question 1: *How can we enable artificial intelligence models to recover 3D structure of environments when supervised only with posed images?*

In Chapter 2, we demonstrate that neural network models without an inductive bias on 3D structure fail to infer the true 3D structure of an environment given only posed 2D images. We propose to combine a 3D-structured latent space, a voxelgrid of features—DeepVoxels—with an image formation model that explicitly models occlusions. We demonstrate that DeepVoxels succeeds in inferring 3D appearance and geometry given only 2D observations of an environment, and demonstrate applications in novel view synthesis.

While the proposed 3D-structured latent space successfully enables inference of 3D structure, voxel grids do not scale to large scenes due to a cubical growth in memory requirements with spatial resolution. They further require discretization of 3D space, and thus do not naturally admit the representation of smooth surfaces and coherent scene parts. Most importantly, it has proven difficult to learn priors for few-shot reconstruction over scenes represented by voxelgrids. This motivates the following research question:

Research Question 2: *Can we find a representation whose memory does not scale with spatial resolution, and that allows the learning of priors over its parameters, enabling reconstruction from few observations?*

In Chapter 3, we propose Scene Representation Networks (SRNs). Instead of a voxelgrid of features, we propose to model a 3D environment as a function that maps world coordinates to a feature representation of local scene properties. This function is directly approximated as a fully connected neural network, effectively encoding the scene in the parameters of that network. We

demonstrate the learning of a distribution over the weights of such SRNs, thereby enabling full 3D reconstruction of shape and geometry given only a single posed image.

Research Question 3: *Can we leverage the features learned by the proposed SRNs to learn downstream tasks from limited supervision?*

The goal of representation learning is the learning of representations that may support a variety of applications. In Chapter 4, we leverage the prior over 3D environments learned by SRNs to perform dense 3D semantic segmentation of objects. We demonstrate that this is feasible in a semi-supervised framework, given only extremely limited per-pixel semantic labels. At test time, this subsequently enables joint 3D reconstruction of objects and dense 3D semantic segmentation, given only a single posed image.

Chapter 2

DeepVoxels

Recent years have seen significant progress in applying generative machine learning methods to the creation of synthetic imagery. Many deep neural networks, for example based on (variational) autoencoders, are able to inpaint, refine, or even generate complete images from scratch [65, 66]. A very prominent direction is generative adversarial networks [71] which achieve impressive results for image generation, even at high resolutions [91] or conditional generative tasks [6]. These developments allow us to perform highly-realistic image synthesis in a variety of settings; e.g., purely generative, conditional, etc.

However, while each generated image is of high quality, a major challenge is to generate a series of coherent views of the same environment. Such consistent view generation would require the network to have a neural representation that fundamentally understands the 3D layout of the scene; e.g., how would the same chair look from a different viewpoint? Unfortunately, this is challenging to learn for existing generative neural network architectures that are based on a series of 2D convolution kernels. Here, spatial layout and transformations of a real, 3D environment would require a tedious learning process which maps 3D operations into 2D convolution kernels [92]. In addition, the generator network in these approaches is commonly based on a U-Net architecture with skip connections [93]. Although skip connections enable efficient propagation of low-level features, the learned 2D-to-2D mappings typically struggle to generalize to large 3D transformations, due to the fact that the skip connections bypass higher-level reasoning.

To tackle similar challenges in the context of learning-based 3D reconstruction and semantic scene understanding, the field of 3D deep learning has seen large and rapid progress over the last few years. Existing approaches are able to predict surface geometry with high accuracy. Many of these techniques are based on explicit 3D representations in the form of occupancy grids [42, 94], signed distance fields [46], point clouds [49, 83], or meshes [54]. While these approaches handle the geometric reconstruction task well, they are not directly applicable to the synthesis of realistic imagery, since it is unclear how to represent color information at a sufficiently high resolution.

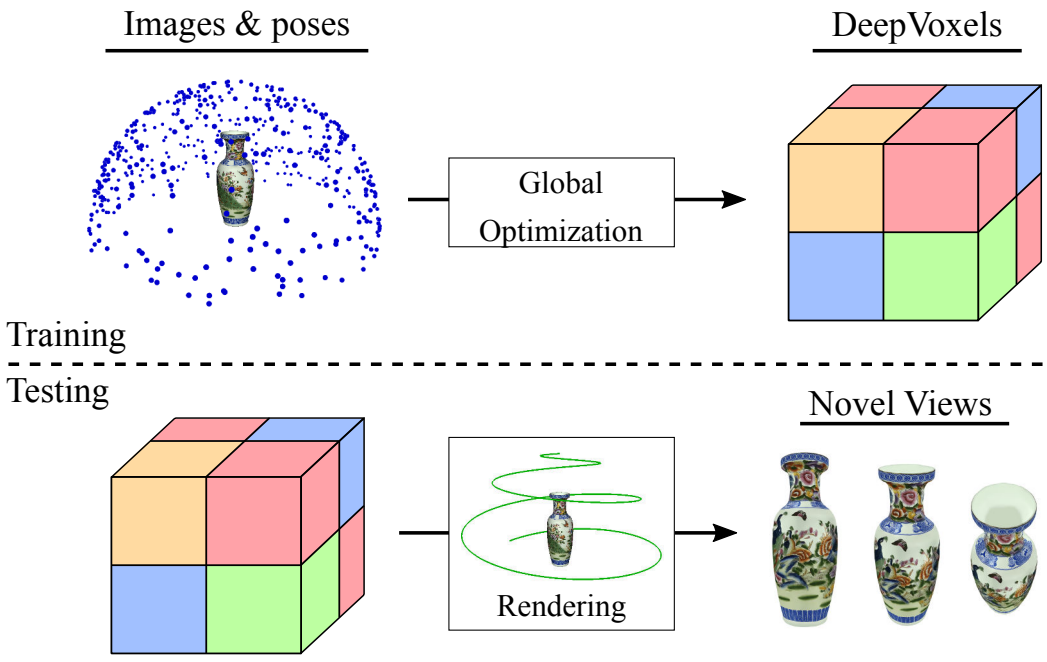

Figure 2.1: During training, we learn a persistent DeepVoxels representation that encodes the view-dependent appearance of a 3D scene from a dataset of posed multi-view images (top). At test time, DeepVoxels enable novel view synthesis (bottom).

There also exists a large body of work on learning low-dimensional embeddings of images that can be decoded to novel views [4, 5, 11, 95–97]. Some of these techniques make use of the object's 3D rotation by explicitly rotating the latent space feature vector [5, 97]. While such 3D techniques are promising, they have thus far not been successful in achieving sufficiently high fidelity for the task of photorealistic image synthesis.

In this chapter, we aim at overcoming the fundamental limitations of existing 2D generative models, demonstrating that a 3D inductive bias is necessary in order to enable scene representation learners to discover 3D shape and appearance of an environment conditioned only on posed images. To this end, we propose *DeepVoxels*, a learned representation that encodes the view-dependent appearance of a 3D scene without having to explicitly model its geometry. DeepVoxels leverages a Cartesian 3D grid of learned features as its core neural scene representation, reflecting the 3D structure of the environments of interest. This allows condensing posed input images of an environment into a persistent latent representation without explicitly having to model its geometry (see Fig. 2.1) We further combine insights from 3D geometric computer vision to equip INFER and RENDER with an inductive bias of 3D structure. Specifically, we implement INFER as an iterative algorithm, where each iteration extracts 2D features, lifts them to 3D via an unprojection operation, and finally updates the neural scene representation with these 3D lifted features. We implement RENDER as a

differentiable projection operation with a learned z-buffer, forcing the learner to explicitly reason about occlusions. Finally, we leverage advances in image-to-image mappings by formulating an adversarial loss function. We apply DeepVoxels to the problem of novel view synthesis, demonstrating high-quality results for a variety of challenging scenes.

In summary, our approach makes the following technical contributions:

- A novel neural scene representation for image synthesis that makes use of the underlying 3D scene information.
- Explicit occlusion reasoning based on learned soft visibility that leads to higher-quality results and better generalization to novel viewpoints.
- Differentiable image formation to enforce perspective and multi-view geometry in a principled and interpretable manner during training.
- Training without requiring 3D supervision.

Scope In this chapter, we present first steps towards 3D-structured neural scene representations. To this end, we limit the scope of our investigation to allow an in-depth discussion of the challenges fundamental to this approach. We assume Lambertian scenes, without specular highlights or other view-dependent effects. While the proposed approach can deal with light specularities, these are not modeled explicitly. Classical approaches will achieve impressive results on the presented scenes. However, these approaches rely on the explicit reconstruction of geometry. As we will demonstrate in the following chapter, neural scene representation are essential to develop generative models that can generalize across scenes to solve reconstruction problems where only few observations are available. We thus compare to such baselines exclusively.

2.1 Method

The core of our approach is a novel 3D-structured scene representation called DeepVoxels. DeepVoxels is a viewpoint-invariant, persistent and uniform 3D voxel grid of features. The underlying 3D grid enforces spatial structure on the learned per-voxel code vectors. The final output image is formed based on a 2D network that receives the perspective re-sampled version of this 3D volume, i.e., the canonical view volume of the target view, as input. The 3D part of our approach takes care of spatial reasoning, while the 2D part enables fine-scale feature synthesis. In the following, we first introduce the training corpus and then present our end-to-end approach for finding the scene-specific DeepVoxels representation from a set of multi-view images without explicit 3D supervision.

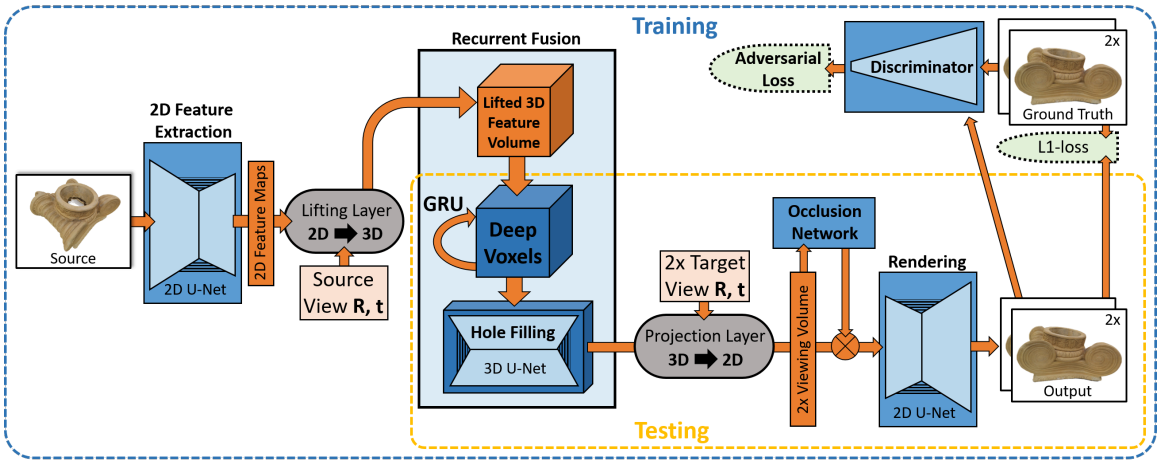

Figure 2.2: Overview of all model components. At the heart of our encoder-decoder based architecture is a novel viewpoint-invariant and persistent 3D volumetric scene representation called DeepVoxels that enforces spatial structure on the learned per-voxel code vectors.

2.1.1 Dataset

We assume a scene-specific dataset $\mathcal{C} = \{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^{M}$ as discussed in Section 1.1.1, with images $\mathcal{I}$ along with their respective extrinsic $\mathbf{E}$ and intrinsic $\mathbf{K}$ camera matrices [12]. In the following, we will refer to a single tuple $(\mathcal{I}, \mathbf{E}, \mathbf{K})$ as a *view*. In each iteration of the optimizer, we will sample from this dataset three times. First, we sample two target views $\mathcal{T}^0$ and $\mathcal{T}^1$ from the whole dataset at random. For each pair of target views $\mathcal{T}^0, \mathcal{T}^1$ we then randomly select a single source view $\mathcal{S}$ from the top-5 nearest neighbors in terms of view direction angle to target view $\mathcal{T}^0$. This sampling heuristic makes it highly likely that points in the source view are visible in the target view $\mathcal{T}^0$. While not essential to training, this ensures meaningful gradient flow for every optimization step, while encouraging multi-view consistency to the random target view $\mathcal{T}^1$. We sample the training corpus $\mathcal{C}$ dynamically during training.

2.1.2 Architecture Overview

Our network architecture is summarized in Fig. 2.2. On a high level, it can be seen as an encoder-decoder based architecture with the persistent 3D DeepVoxels representation as its latent space. During training, we feed a source view $\mathcal{S}$ to the encoder and try to predict the target view $\mathcal{T}$. We first extract a set of 2D feature maps from the source view using a 2D feature extraction network. To learn a view-independent 3D feature representation, we explicitly lift image features to 3D based on a differentiable lifting layer. The lifted 3D feature volume is fused with our persistent DeepVoxels scene representation using a gated recurrent network architecture. Specifically, the persistent 3D feature volume is the hidden state of a gated recurrent unit (GRU) [98]. After feature fusion, the volume is processed by a 3D fully convolutional network. The volume is then mapped to the camera

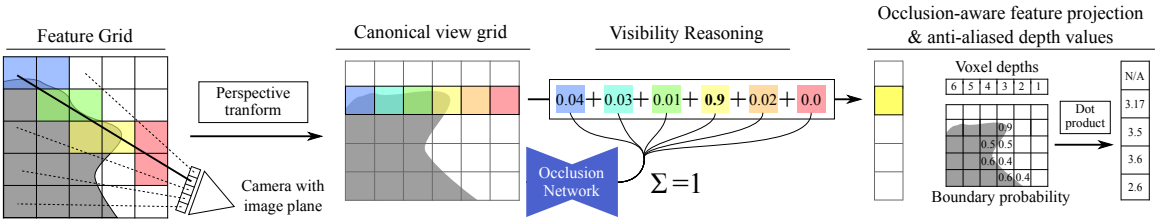

Figure 2.3: Illustration of the occlusion-aware projection operation. The feature volume (represented by feature grid) is first resampled into the canonical view volume via a projection transformation and trilinear interpolation. The occlusion network then predicts per-pixel softmax weights along each depth ray. The canonical view volume is then collapsed along the depth dimension via a softmax-weighted sum of voxels to yield the final, occlusion-aware feature map. The per-voxel visibility weights can be used to compute a depth map.

coordinate systems of the two target views via a differentiable reprojection layer, resulting in the canonical view volume. A dedicated, structured occlusion network operates on the canonical view volume to reason about voxel visibility and flattens the view volume to a 2D view feature map (see Fig. 2.3). Finally, a learned 2D rendering network forms the two final output images. Our network is trained end-to-end, without the need of supervision in the 3D domain, by a 2D re-rendering loss that enforces that the predictions match the target views. In the following, we provide more details.

Camera Model We follow a perspective pinhole camera model that is fully specified by its extrinsic $\mathbf{E} = [\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3\times4}$ and intrinsic $\mathbf{K} \in \mathbb{R}^{3\times3}$ camera matrices [12]. Here, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the global camera rotation and $\mathbf{t} \in \mathbb{R}^3$ its translation. Assume we are given a position $\mathbf{x} \in \mathbb{R}^3$ in 3D coordinates, then the mapping from world space to the canonical camera volume is given as:

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ d \end{pmatrix} = \mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{t}) \ . \tag{2.1}$$

Here, u and v specify the position of the voxel center on the screen and d is its depth from the camera. Given a pixel and its depth, we can invert this mapping to compute the corresponding 3D point $\mathbf{x} = \mathbf{R}^T(\mathbf{K}^{-1}\mathbf{u} - \mathbf{t})$.

Feature Extraction We extract 2D feature maps from the source view based on a fully convolutional feature extraction network. The image is first downsampled by a series of stride-2 convolutions until a resolution of 64×64 is reached. A 2D U-Net architecture [93] then extracts a 64×64 feature map that is the input to the subsequent volume lifting.

Lifting 2D Features to 3D Observations The lifting layer lifts 2D features into a temporary 3D volume, representing a single 3D observation, which is then integrated into the persistent DeepVoxels

representation. We position the 3D feature volume in world space such that its center roughly aligns with the scene's center of gravity, which can be obtained cheaply from the keypoint point cloud obtained from sparse bundle adjustment. The spatial extent is set such that the complete scene is inside the volume. We try to bound the scene as tightly as possible to not lose spatial resolution. Lifting is implemented by a gathering operation. For each voxel, the world space position of its center is projected to the source view's image space following Eq. 2.1. We extract a feature vector from the feature map using bilinear sampling and store the result in the code vector associated with the voxel. Note, our approach is based only on a set of registered multi-view images and we do not have access to the scene geometry or depth maps, rather our approach learns automatically to resolve the depth ambiguity based on a gated recurrent network in 3D.

Integrating Lifted Features into DeepVoxels Lifted observations are integrated into the DeepVoxels representation via an integration network that is based on gated recurrent units (GRUs) [98]. In contrast to the standard application of GRUs, the integration network operates on the same volume across the full training procedure, i.e., the hidden state is *persistent* across all training steps and never reset, leading to a geometrically consistent representation of the whole training corpus. We use a uniform volumetric grid of size $w \times h \times d$ voxels, where each voxel has f feature channels, i.e., the stored code vector has size f. We employ one gated recurrent unit for each voxel, such that at each time step, all the features in a voxel have to be updated jointly. The goal of the gated recurrent units is to incrementally fuse the lifted features and the hidden state during training, such that the best persistent 3D volumetric feature representation is discovered. The gated recurrent units implement the mapping

$$\mathbf{Z}_t = \sigma(\mathbf{W}_z \mathbf{X}_t + \mathbf{U}_z \mathbf{H}_{t-1} + \mathbf{B}_z) \ , \tag{2.2}$$

$$\mathbf{R}_t = \sigma(\mathbf{W}_r \mathbf{X}_t + \mathbf{U}_r \mathbf{H}_{t-1} + \mathbf{B}_r) \ , \tag{2.3}$$

$$\mathbf{S}_t = \mathrm{ReLU}(\mathbf{W}_s \mathbf{X}_t + \mathbf{U}_s (\mathbf{R}_t \circ \mathbf{H}_{t-1}) + \mathbf{B}_s) \ , \tag{2.4}$$

$$\mathbf{H}_t = (1 - \mathbf{Z}_t) \circ \mathbf{H}_{t-1} + \mathbf{Z}_t \circ \mathbf{S}_t \ . \tag{2.5}$$

Here, $\mathbf{X}_t$ is the lifted 3D feature volume of the current timestep t, the $\mathbf{W}_\bullet$ and $\mathbf{U}_\bullet$ are trainable 3D convolution weights, and the $\mathbf{B}_\bullet$ are trainable tensors of biases. We follow Cho et al. [98] and employ a sigmoid activation σ to compute the response of the tensor of update gates $\mathbf{Z}_t$ and reset gates $\mathbf{R}_t$. Based on the previous hidden state $\mathbf{H}_{t-1}$, the per-voxel reset values $\mathbf{R}_t$, and the lifted 3D feature volume $\mathbf{X}_t$, the tensor of new feature proposals $\mathbf{S}_t$ for the current time step t is computed. $\mathbf{U}_s$ and $\mathbf{W}_s$ are single 3D convolutional layers. The new hidden state $\mathbf{H}_t$, the DeepVoxels representation for the current time step, is computed as a per-voxel linear combination of the old state $\mathbf{H}_{t-1}$ and the new DeepVoxel proposal $\mathbf{S}_t$. The GRU performs one update step per lifted observation. We note that generally, GRUs are leveraged as sequential processes, and their weights are therefore

usually optimized via backpropagation in time to optimally incorporate information obtained over a sequence of several timesteps. While this is also, in principle, possible in the proposed framework, in this work, we do not perform backpropagation in time. This means that the weights of the GRU are optimized to integrate new observations optimally for *only the current* iteration, and no information flows backwards in time to previous integration steps. In this mode of operation, the GRU can be interpreted as a learned update rule for the DeepVoxels that is conditioned only on the information of the current update step. Finally, we apply a 3D inpainting U-Net that learns to fill holes in this feature representation. At test time, only the optimally learned persistent 3D volumetric features, the DeepVoxels, are used to form the image corresponding to a novel target view. The 2D feature extraction, lifting layer and GRU gates are discarded and are not required for inference, see Fig. 2.2.

Projection Layer The projection layer implements the inverse of the lifting layer, i.e., it maps the 3D code vectors to the canonical coordinate system of the target view, see Fig. 2.3 (left). Projection is also implemented based on a gathering operation. For each voxel of the canonical view volume, its corresponding position in the persistent world space voxel grid is computed. An interpolated code vector is then extracted via a trilinear interpolation and stored in the feature channels of the canonical view volume.

Occlusion Module Occlusion reasoning is essential for correct image formation and generalization to novel viewpoints. To this end, we propose a dedicated occlusion network that computes soft visibility for each voxel. Each pixel in the target view is represented by one column of voxels in the canonical view volume, see Fig. 2.3 (left). First, this column is concatenated with a feature column encoding the distance of each voxel to the camera, similar as in [99]. This allows the occlusion network to reason about voxel order. The feature vector of each voxel in this canonical view volume is then compressed to a low-dimensional feature vector of dimension 4 by a single 3D convolutional layer. This compressed volume is input to a 3D U-Net for occlusion reasoning. For each ray, represented by a single-pixel column, this network predicts a scalar per-voxel visibility weight based on a softmax activation, see Fig. 2.3 (middle). The canonical view volume is then flattened along the depth dimension with a weighted average, using the predicted visibility values. The softmax weights can further be used to compute a depth map, which provides insight into the occlusion reasoning of the network, see Fig. 2.3 (right).

Rendering and Loss The rendering network is a mirrored version of the feature extraction network with higher capacity. A 2D U-Net architecture takes as input the flattened canonical view volume from the occlusion network and provides reasoning across the full image, before a number of transposed convolutions directly regress the pixel values of the novel view. We train our persistent DeepVoxels representation based on a combined ℓ_1-loss and adversarial cross entropy loss [71]. We found that an adversarial loss accelerates the generation of high-frequency detail earlier on in

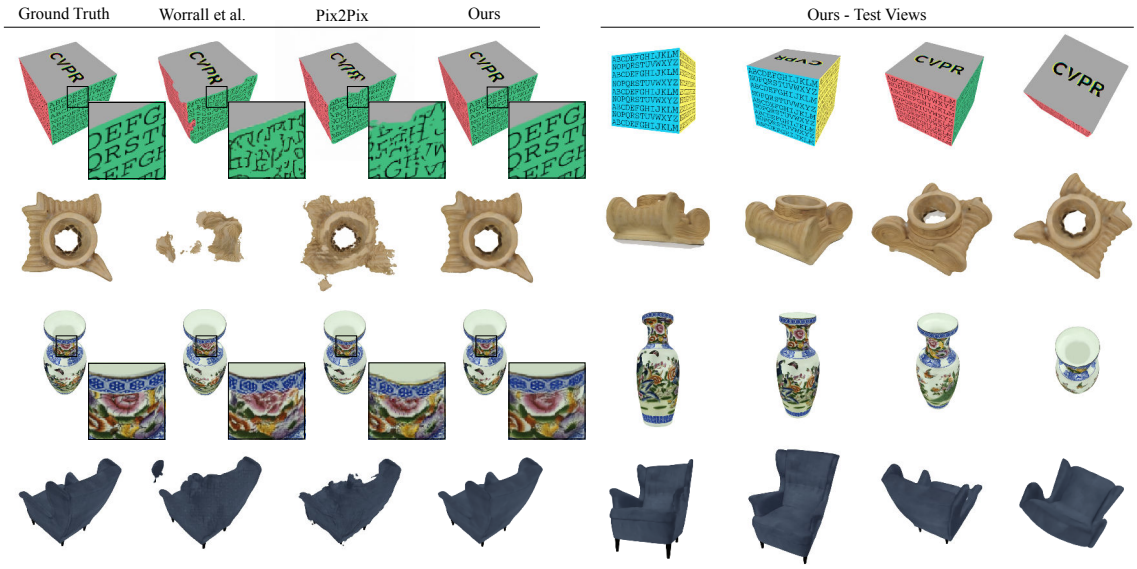

Figure 2.4: Left: Comparison of the best three performing models to ground truth. From Left to right: Ground truth, Worrall et al. [5], Isola et al. [6] (Pix2Pix), and ours. Our outputs are closest to the ground truth, performing well even in challenging cases such as the strongly foreshortened letters on the cube or the high-frequency detail of the vase. Right: Other samples of novel views generated by our model.

training. Our adversarial discriminator is a fully convolutional patch-based discriminator [100]. We solve the resulting minimax optimization problem using ADAM [101].

2.2 Analysis

In this section, we demonstrate that DeepVoxels is a rich and semantically meaningful 3D scene representation that allows high-quality re-rendering from novel views. First, we present qualitative and quantitative results on synthetic renderings of high-quality 3D scans of real-world objects, and compare the performance to strong machine-learning baselines with increasing reliance on geometrically structured latent spaces. Next, we demonstrate that DeepVoxels can also be used to generate novel views on a variety of real captures, even if these scenes may violate the Lambertian assumption. Finally, we demonstrate quantitative and qualitative benefits of explicitly reasoning about voxel visibility via the occlusion module, as well as improved model interpretability. Please see the supplement for further studies on the sensitivity to the number of training images, the size of the voxel volume, as well as noisy camera poses.

Dataset and Metrics We evaluate model performance on synthetic data obtained from rendering 4 high-quality 3D scans (see Fig. 2.4). We center each scan at the origin and scale it to lie within

	Vase	Pedestal	Chair	Cube	Mean
	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Near. Neighb.	23.26 / 0.92	21.49 / 0.87	20.69 / 0.94	18.32 / 0.83	20.94 / 0.89
TCO [4]	22.28 / 0.91	23.25 / 0.89	20.22 / 0.95	19.12 / 0.84	21.22 / 0.90
WRL [5]	23.41 / 0.92	22.70 / 0.89	19.52 / 0.94	19.23 / 0.85	21.22 / 0.90
Pix2Pix [6]	26.36 / 0.95	25.41 / 0.91	23.04 / 0.96	19.69 / 0.86	23.63 / 0.92
Ours	**27.99 / 0.96**	**32.35 / 0.97**	**33.45 / 0.99**	**28.42 / 0.97**	**30.55 / 0.97**

Table 2.1: Quantitative comparison to four baselines. Our approach obtains the best results in terms of PSNR and SSIM on all objects.

the unit cube. For the training set, we render the object from 479 poses uniformly distributed on the northern hemisphere. For the test set, we render 1000 views on an Archimedean spiral on the northern hemisphere. All images are rendered in a resolution of 1024×1024 and then resized using area averaging to 512×512 to minimize aliasing. We evaluate reconstruction error in terms of PSNR and SSIM [102].

Implementation All models are implemented in PyTorch [103]. Unless specified otherwise, we use a cube volume with 32^3 voxels. We average the ℓ_1 loss over all pixels in the image. The ℓ_1 and adversarial loss are weighted $200:1$. Models are trained until convergence using ADAM with a learning rate of $4 \cdot 10^{-4}$. One model is trained per scene. The proposed architecture has 170 million parameters. At test time, rendering a single frame takes 71ms.

Baselines We compare to three strong baselines with increasing reliance on geometry-aware latent spaces. The first baseline is a Pix2Pix architecture [6] that receives as input the per-pixel view direction, i.e., the normalized, world-space vector from camera origin to each pixel, and is trained to translate these images into the corresponding color image. This baseline is representative of recent achievements in 2D image-to-image translation. The second baseline is a deep autoencoder that receives as input one of the top-5 nearest neighbors of the target view, and the pose of both the target and the input view are concatenated in the deep latent space, as proposed by Tatarchenko et al. [4]. The inputs of this model at training time are thus identical to those of our model. The third baseline learns an interpretable, rotation-equivariant latent space via the method proposed in [5, 104] and used previously in [97], by being fed one of the top-5 nearest neighbor views and then rotating the latent embedding with the rotation matrix that transforms the input to the output pose. At test time, the previous two baselines receive the top-1 nearest neighbor to supply the model with the most relevant information. We approximately match the number of parameters of each network, with all baselines having equally or slightly more parameters than our model. We train all baselines to convergence with the same loss function. For the exact baseline architectures and number of parameters, please see the supplement.

Object-specific Novel View Synthesis We train our network and all baselines on synthetic renders of four high-quality 3D scans. Table 2.1 compares PSNR and SSIM of the proposed architecture and the baselines. The best-performing baseline is Pix2Pix [6]. This is surprising, since no geometrical constraints are enforced, as opposed to the approach by Worrall et al. [5]. The proposed architecture with strongly structured latent space outperforms all baselines by a wide margin of an average 7dB. Fig. 2.4 shows a qualitative comparison as well as further novel views sampled from the proposed model. The proposed model displays robust 3D reasoning that does not break down even in challenging cases. Notably, other models have a tendency to "snap" onto views seen in the training set, while the proposed model smoothly follows the test trajectory. Please see the supplemental video for a demonstration of this behavior. We hypothesize that this improved generalization to unseen views is due to the explicit multi-view constraints enforced by the proposed latent space. The baseline models are not explicitly enforcing projective and epipolar geometry, which may allow them to parameterize latent spaces that are not properly representing the low-dimensional manifold of rotations. Although the resolution of the proposed voxel grid is 16 times smaller than the image resolution, our model succeeds in capturing fine detail much smaller than the size of a single voxel, such as the letters on the sides of the cube or the detail on the vase. This may be due to the use of trilinear interpolation in the lifting and projection steps, which allow for a fine-grained representation to be learned. Please see the video for full sequences, and the supplemental material for two additional synthetic scenes.

Voxel Embedding vs. Rotation-Equivariant Embedding As reflected in Tab. 2.1, we outperform [5] by a wide margin both qualitatively and quantitatively. The proposed model is constrained through multi-view geometry, while [5] has more degrees of freedom. Lacking occlusion reasoning, depth maps are not made explicit. The model may thus parameterize latent spaces that do not respect multi-view geometry. This increases the risk of overfitting, which we observe empirically, as the baseline snaps to nearest neighbors seen during training. While the proposed voxel embedding is memory hungry, it is very parameter efficient. The use of 3D convolutions means that the parameter count is independent of the voxel grid size. Giving up spatial structure means Worrell et al. [5] abandon convolutions and use fully connected layers. However, to achieve the same latent space size of $32^3 \times 64$ features would necessitate more than $4.4 \cdot 10^{12}$ parameters between just the fully connected layers before and after the feature transformation layer, which is infeasible. In contrast, the proposed 3D inpainting network only has $1.7 \cdot 10^7$ parameters, five orders of magnitude less. To address memory inefficiency, the dense grid may be replaced by a sparse alternative in the future.

Occlusion Reasoning and Interpretability An essential part of the rendering pipeline is the depth test. Similarly, the rendering network ought to be able to reason about occlusions when regressing the output view. A naive approach might flatten the depth dimension of the canonical camera volume and subsequently reduce the number of features using a series of 2D convolutions.

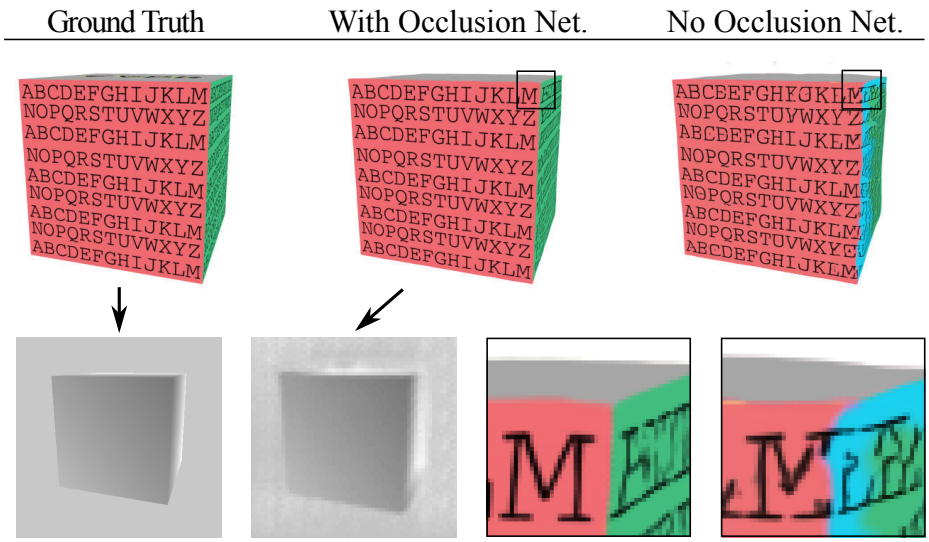

Figure 2.5: The occlusion module is critical to model performance. It boosts performance from 23.26dB to 28.42dB (cube), and from 30.02dB to 32.35dB (pedestal). Left: ground truth view and depth map. Center: view generated with the occlusion module and learned depth map (64×64 pixels). Note that the object background is unconstrained in the depth map and may differ from ground truth. Right: without the occlusion module, the occluded, blue side of the cube (see Fig. 2.4) "shines through", and severe artifacts appear (see inset). In addition to decreasing parameter count and boosting performance, the occlusion module generates depth maps fully unsupervised, demonstrating 3D reasoning.

This leads to a drastic increase in the number of network parameters. At training time, this further allows the network to combine features from several depths equally to regress on pixel colors in the target view. At inference time, this results in severe artifacts and occluded parts of the object "shining through" (see Fig. 2.5). Our occlusion network forces learning to use a softmax-weighted sum of voxels along each ray, which penalizes combining voxels from several depths. As a result, novel views generated by the network with the occlusion module perform much more favorably at test time, as demonstrated in Fig. 2.5, than networks without the occlusion module. The depth map generated by the occlusion model further demonstrates that the proposed model indeed learns the 3D structure of the scene. We note that the depth map is learned in a fully unsupervised manner and arises out of the pure necessity of picking the most relevant voxel. Please see the supplement for more examples of learned depth maps.

Novel View Synthesis for Real Captures We train our network on real captures obtained with a DSLR camera. Camera poses, intrinsic camera parameters and keypoint point clouds are obtained via sparse bundle adjustment. The voxel grid origin is set to the respective point cloud's center of gravity. Voxel grid resolution is set to 64. Each voxel stores 8 feature channels. Test trajectories are obtained by linearly interpolating two randomly chosen training poses. Scenes depict

Figure 2.6: Novel views of real captures.

a drinking fountain, two busts, a globe, and a bag of coffee. See Fig. 2.6 for example model outputs. The drinking fountain and the globe have noticeable specularities, which are handled gracefully. While the coffee bag is generally represented faithfully, inconsistencies appear on its highly specular surface. Generally, results are of high quality, and only details that are significantly smaller than a single voxel, such as the tiles in the sink of the fountain, show artifacts. Please refer to `https://vsitzmann.github.io/deepvoxels` for detailed results as well as a nearest-neighbor baseline.

2.3 Limitations

Although we have demonstrated high-quality view synthesis results for a variety of challenging scenes, the proposed approach has a number of limitations. By construction, the employed 3D volume is memory inefficient, thus we have to trade local resolution for spatial extent. The proposed model can be trained with a voxel resolution of 64^3 with 8 feature channels, filling a GPU with 12GB of memory. Please note, compelling results can already be achieved with quite small volume resolutions. Synthesizing images from viewpoints that are significantly different from the training set, i.e., extrapolation to unseen camera parameters, is challenging for all learning-based approaches. While this is also true for DeepVoxels and detail is lost when viewing scenes from poses far away from training poses, DeepVoxels generally deteriorates gracefully and the 3D structure of the scene is preserved. Please refer to the appendix for failure cases as well as examples of pose extrapolation.

2.4 Conclusion

In this chapter, we have proposed a novel 3D-structure-aware neural scene representation, DeepVoxels, that encodes the view-dependent appearance of a 3D scene using only supervision through posed images. We have demonstrated that inductive biases based in multi-view and perspective geometry are necessary to achieve this feat, significantly outperforming approaches without such inductive biases. Our approach is a first step towards 3D-structured neural scene representations and the goal of overcoming the fundamental limitations of existing 2D generative models by introducing native 3D operations into the network.

Chapter 3

Scene Representation Networks

In the previous chapter, we proposed DeepVoxels, a 3D-structured neural scene representation in the form of a voxelgrid of features. However, the DeepVoxels representation is discrete, limiting achievable spatial resolution, and does not enable the learning of a prior over the parameters of the voxelgrid. As a result, DeepVoxels does not allow the reconstruction of scenes when only few observations are available.

In this chapter, we introduce *Scene Representation Networks* (SRNs), a continuous neural scene representation, along with a differentiable rendering algorithm, that model both 3D scene geometry and appearance, enforce 3D structure in a multi-view consistent manner, and naturally allow generalization of shape and appearance priors across scenes. The key idea of SRNs is to represent a scene implicitly as a continuous, differentiable function that maps a 3D world coordinate to a feature-based representation of the scene properties at that coordinate. This allows SRNs to naturally interface with established techniques of multi-view and projective geometry while operating at high spatial resolution in a memory-efficient manner. Similar to DeepVoxels, INFER and RENDER are factorized into sub-routines that respect the 3D structure of the underlying environment. Specifically, RENDER is implemented as a differentiable ray-marching algorithm that first recovers the intersection of a camera ray with scene geometry and subsequently renders the color of only this intersection point, thereby respecting occlusions. As in Chapter 2, INFER is implemented as an iterative algorithm. However, different from DeepVoxels, INFER is implicitly modeled as the backpropagation of the loss through the RENDER algorithm. As DeepVoxels before, SRNs can be trained end-to-end, supervised only by a set of posed 2D images of a scene. SRNs generate high-quality images *without any 2D convolutions*, exclusively operating on individual pixels, which enables image generation at arbitrary resolutions. They generalize naturally to camera transformations and intrinsic parameters that were completely unseen at training time. For instance, SRNs that have only ever seen objects from a constant distance are capable of rendering close-ups of said objects flawlessly. We evaluate SRNs on a variety of challenging 3D computer vision problems, including novel view synthesis, few-shot scene

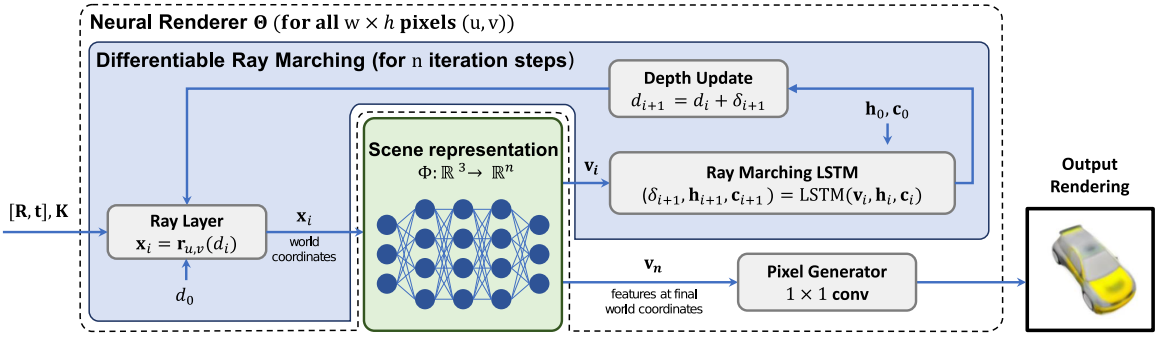

Figure 3.1: Overview: at the heart of SRNs lies a continuous, 3D-aware neural scene representation, Φ, which represents a scene as a function that maps (x, y, z) world coordinates to a feature representation of the scene at those coordinates (see Sec. 3.1.1). A neural renderer Θ, consisting of a learned ray marcher and a pixel generator, can render the scene from arbitrary novel view points (see Sec. 3.1.2).

reconstruction, joint shape and appearance interpolation, and unsupervised discovery of a non-rigid face model.

To summarize, our approach makes the following key contributions:

- A *continuous*, 3D-structure-aware neural scene representation and renderer, SRNs, that efficiently encapsulate both scene geometry and appearance.
- End-to-end training of SRNs without explicit supervision in 3D space, purely from a set of posed 2D images.
- We demonstrate novel view synthesis, shape and appearance interpolation, and few-shot reconstruction, as well as unsupervised discovery of a non-rigid face model, and significantly outperform baselines from recent literature.

3.1 Formulation

Given a training set $\mathcal{C} = \{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^N$ of N tuples of images $\mathcal{I}_i \in \mathbb{R}^{H \times W \times 3}$ along with their respective extrinsic $\mathbf{E}_i = \left[\mathbf{R}|\mathbf{t}\right] \in \mathbb{R}^{3\times4}$ and intrinsic $\mathbf{K}_i \in \mathbb{R}^{3\times3}$ camera matrices [12], our goal is to distill this dataset of observations into a neural scene representation Φ that strictly enforces 3D structure and allows to generalize shape and appearance priors across scenes. In addition, we are interested in a rendering function Θ that allows us to render the scene represented by Φ from arbitrary viewpoints. In the following, we first formalize Φ and Θ and then discuss a framework for optimizing Φ, Θ for a single scene given only posed 2D images. Note that this approach does *not* require information about scene geometry. Additionally, we show how to learn a family of scene representations for an entire class of scenes, discovering powerful shape and appearance priors.

3.1.1 Representing Scenes as Functions

Our key idea is to represent a scene as a function Φ that maps a spatial location $\mathbf{x}$ to a feature representation $\mathbf{v}$ of learned scene properties at that spatial location:

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^n, \quad \mathbf{x} \mapsto \Phi(\mathbf{x}) = \mathbf{v}. \tag{3.1}$$

The feature vector $\mathbf{v}$ may encode visual information such as surface color or reflectance, but it may also encode higher-order information, such as the signed distance of $\mathbf{x}$ to the closest scene surface. This continuous formulation can be interpreted as a generalization of discrete neural scene representations. Voxel grids, for instance, discretize $\mathbb{R}^3$ and store features in the resulting 3D grid [1, 38, 94, 105–107]. Point clouds [83, 108, 109] may contain points at any position in $\mathbb{R}^3$, but only sparsely sample surface properties of a scene. In contrast, Φ densely models scene properties and can in theory model arbitrary spatial resolutions, as it is continuous over $\mathbb{R}^3$ and can be sampled with arbitrary resolution. In practice, we represent Φ as a multi-layer perceptron (MLP), and spatial resolution is thus limited by the capacity of the MLP.

In contrast to recent work on representing scenes as unstructured or weakly structured feature embeddings [4, 5, 11], Φ is explicitly aware of the 3D structure of scenes, as the input to Φ are world coordinates $(x, y, z) \in \mathbb{R}^3$. This allows interacting with Φ via the toolbox of multi-view and perspective geometry that the physical world obeys, only using learning to approximate the unknown properties of the scene itself. In Sec. 3.2, we show that this formulation leads to multi-view consistent novel view synthesis, data-efficient training, and a significant gain in model interpretability.

3.1.2 Neural Rendering

Given a scene representation Φ, we introduce a neural rendering algorithm Θ, that maps a scene representation Φ as well as the intrinsic $\mathbf{K}$ and extrinsic $\mathbf{E}$ camera parameters to an image $\mathcal{I}$:

$$\Theta : \mathcal{X} \times \mathbb{R}^{3\times4} \times \mathbb{R}^{3\times3} \rightarrow \mathbb{R}^{H\times W\times3}, \quad (\Phi, \mathbf{E}, \mathbf{K}) \mapsto \Theta(\Phi, \mathbf{E}, \mathbf{K}) = \mathcal{I}, \tag{3.2}$$

where $\mathcal{X}$ is the space of all functions Φ.

The key complication in rendering a scene represented by Φ is that geometry is represented implicitly. The surface of a wooden table top, for instance, is defined by the subspace of $\mathbb{R}^3$ where Φ undergoes a change from a feature vector representing free space to one representing wood. To render a single pixel in the image observed by a virtual camera, we thus have to solve two sub-problems: (i) finding the world coordinates of the intersections of the respective camera rays with scene geometry, and (ii) mapping the feature vector $\mathbf{v}$ at that spatial coordinate to a color. We first propose a neural ray marching algorithm with learned, adaptive step size to find ray intersections with scene geometry, and subsequently discuss the pixel generator network that learns the feature-to-color mapping.

Differentiable Ray Marching Algorithm

Algorithm 1 Differentiable Ray-Marching

function FINDINTERSECTION($\Phi, \mathbf{K}, \mathbf{E}, (u, v)$)
  $d_0 \leftarrow 0.05$ ▷ Near plane
  $(\mathbf{h}_0, \mathbf{c}_0) \leftarrow (\mathbf{0}, \mathbf{0})$ ▷ Initial state of LSTM
  for $i \leftarrow 0$ to max_iter **do**
    $\mathbf{x}_i \leftarrow \mathbf{r}_{u,v}(d_i)$ ▷ Calculate world coordinates
    $\mathbf{v}_i \leftarrow \Phi(\mathbf{x}_i)$ ▷ Extract feature vector
    $(\delta, \mathbf{h}_{i+1}, \mathbf{c}_{i+1}) \leftarrow LSTM(\mathbf{v}, \mathbf{h}_i, \mathbf{c}_i)$ ▷ Predict steplength using ray marching LSTM
    $d_{i+1} \leftarrow d_i + \delta$ ▷ Update d
  return $\mathbf{r}_{u,v}(d_{max\_iter})$

Intersection testing intuitively amounts to solving an optimization problem, where the point along each camera ray is sought that minimizes the distance to the surface of the scene. To model this problem, we parameterize the points along each ray, identified with the coordinates (u, v) of the respective pixel, with their distance d to the camera ($d > 0$ represents points in front of the camera):

$$\mathbf{r}_{u,v}(d) = \mathbf{R}^T(\mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ d \end{pmatrix} - \mathbf{t}), \quad d > 0, \tag{3.3}$$

with world coordinates $\mathbf{r}_{u,v}(d)$ of a point along the ray with distance d to the camera, camera intrinsics $\mathbf{K}$, and camera rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. For each ray, we aim to solve

$$\begin{aligned} \arg\min \quad & d \\ \text{s.t.} \quad & \mathbf{r}_{u,v}(d) \in \Omega, \quad d > 0 \end{aligned} \tag{3.4}$$

where we define the set of all points that lie on the surface of the scene as Ω.

Here, we take inspiration from the classic sphere tracing algorithm [110]. Sphere tracing belongs to the class of ray marching algorithms, which solve Eq. 3.4 by starting at a distance d_{init} close to the camera and stepping along the ray until scene geometry is intersected. Sphere tracing is defined by a special choice of the step length: each step has a length equal to the signed distance to the closest surface point of the scene. Since this distance is only 0 on the surface of the scene, the algorithm takes non-zero steps until it has arrived at the surface, at which point no further steps are taken. Extensions of this algorithm propose heuristics to modifying the step length to speed up convergence [111]. We instead propose to *learn* the length of each step.

Specifically, we introduce a *ray marching long short-term memory (RM-LSTM)* [112], that maps the feature vector $\Phi(\mathbf{x}_i) = \mathbf{v}_i$ at the current estimate of the ray intersection $\mathbf{x}_i$ to the length of the next ray marching step. The algorithm is formalized in Alg. 1.

Given our current estimate d_i, we compute world coordinates $\mathbf{x}_i = \mathbf{r}_{u,v}(d_i)$ via Eq. 3.3. We then compute $\Phi(\mathbf{x}_i)$ to obtain a feature vector $\mathbf{v}_i$, which we expect to encode information about nearby scene surfaces. We then compute the step length δ via the RM-LSTM as $(\delta, \mathbf{h}_{i+1}, \mathbf{c}_{i+1}) = LSTM(\mathbf{v}_i, \mathbf{h}_i, \mathbf{c}_i)$, where $\mathbf{h}$ and $\mathbf{c}$ are the output and cell states, and increment d_i accordingly. We iterate this process for a constant number of steps. This is critical, because a dynamic termination criterion would have no guarantee for convergence in the beginning of the training, where both Φ and the ray marching LSTM are initialized at random. The final step yields our estimate of the world coordinates of the intersection of the ray with scene geometry. The z-coordinates of running and final estimates of intersections in camera coordinates yield depth maps, which we denote as $\mathbf{d}_i$, which visualize every step of the ray marcher. This makes the ray marcher interpretable, as failures in geometry estimation show as inconsistencies in the depth map. Note that depth maps are differentiable with respect to all model parameters, but are not required for training Φ. Please see the supplement for a contextualization of the proposed rendering approach with classical rendering algorithms.

Pixel Generator Architecture

The pixel generator takes as input the 2D feature map sampled from Φ at world coordinates of ray-surface intersections and maps it to an estimate of the observed image. As a generator architecture, we choose a per-pixel MLP that maps a single feature vector $\mathbf{v}$ to a single RGB vector. This is equivalent to a convolutional neural network (CNN) with only 1×1 convolutions. Formulating the generator without 2D convolutions has several benefits. First, the generator will always map the same (x, y, z) coordinate to the same color value. Assuming that the ray-marching algorithm finds the correct intersection, the rendering is thus trivially multi-view consistent. This is in contrast to 2D convolutions, where the value of a single pixel depends on a neighborhood of features in the input feature map. When transforming the camera in 3D, e.g. by moving it closer to a surface, the 2D neighborhood of a feature may change. As a result, 2D convolutions come with no guarantee on multi-view consistency. With our per-pixel formulation, the rendering function Θ operates independently on all pixels, allowing images to be generated with arbitrary resolutions and poses. On the flip side, we cannot exploit recent architectural progress in CNNs, and a per-pixel formulation requires the ray marching, the SRN and the pixel generator to operate on the same (potentially high) resolution, requiring a significant memory budget. Please see the supplement for a discussion of this trade-off.

3.1.3 Generalizing Across Scenes

We now generalize SRN from learning to represent a single scene to learning shape and appearance priors over several instances of a single class. Formally, we assume that we are given a set of M instance datasets $\mathcal{D} = \{\mathcal{C}_j\}_{j=1}^{M}$, where each $\mathcal{C}_j$ consists of tuples $\{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^{N}$ as discussed in Sec. 3.1.1.

We reason about the set of functions $\{\Phi_j\}_{j=1}^M$ that represent instances of objects belonging to the same class. By parameterizing a specific Φ_j as an MLP, we can represent it with its vector of parameters $\phi_j \in \mathbb{R}^l$. We assume scenes of the same class have common shape and appearance properties that can be fully characterized by a set of latent variables $\mathbf{z} \in \mathbb{R}^k$, $k < l$. Equivalently, this assumes that all parameters ϕ_j live in a k-dimensional subspace of $\mathbb{R}^l$. Finally, we define a mapping

$$\Psi : \mathbb{R}^k \to \mathbb{R}^l, \quad \mathbf{z}_j \mapsto \Psi(\mathbf{z}_j) = \phi_j \tag{3.5}$$

that maps a latent vector $\mathbf{z}_j$ to the parameters ϕ_j of the corresponding Φ_j. We propose to parameterize Ψ as an MLP, with parameters ψ. This architecture was previously introduced as a Hypernetwork [113], a neural network that regresses the parameters of another neural network. We share the parameters of the rendering function Θ across scenes. We note that assuming a low-dimensional embedding manifold has so far mainly been empirically demonstrated for classes of single objects. Here, we similarly only demonstrate generalization over classes of single objects.

Finding latent codes $\mathbf{z}_j$. To find the latent code vectors $\mathbf{z}_j$, we follow an auto-decoder framework [58]. For this purpose, each object instance $\mathcal{C}_j$ is represented by its own latent code $\mathbf{z}_j$. The $\mathbf{z}_j$ are free variables and are optimized jointly with the parameters of the hypernetwork Ψ and the neural renderer Θ. We assume that the prior distribution over the $\mathbf{z}_j$ is a zero-mean multivariate Gaussian with a diagonal covariance matrix. Please refer to [58] for additional details.

3.1.4 Joint Optimization

To summarize, given a dataset $\mathcal{D} = \{\mathcal{C}_j\}_{j=1}^M$ of instance datasets $\mathcal{C} = \{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^N$, we aim to find the parameters ψ of Ψ that maps latent vectors $\mathbf{z}_j$ to the parameters of the respective scene representation ϕ_j, the parameters θ of the neural rendering function Θ, as well as the latent codes $\mathbf{z}_j$ themselves. We formulate this as an optimization problem with the following objective:

$$\underset{\{\theta,\psi,\{\mathbf{z}_j\}_{j=1}^M\}}{\arg\min} \sum_{j=1}^{M}\sum_{i=1}^{N} \underbrace{\|\Theta_\theta(\Phi_{\Psi(\mathbf{z}_j)}, \mathbf{E}_i^j, \mathbf{K}_i^j) - \mathcal{I}_i^j\|_2^2}_{\mathcal{L}_{\text{img}}} + \underbrace{\lambda_{dep}\|\min(\mathbf{d}_{i,final}^j, \mathbf{0})\|_2^2}_{\mathcal{L}_{\text{depth}}} + \underbrace{\lambda_{lat}\|\mathbf{z}_j\|_2^2}_{\mathcal{L}_{\text{latent}}}. \tag{3.6}$$

Where $\mathcal{L}_{\text{img}}$ is an ℓ_2-loss enforcing closeness of the rendered image to ground-truth, $\mathcal{L}_{\text{depth}}$ is a regularization term that accounts for the positivity constraint in Eq. 3.4, and $\mathcal{L}_{\text{latent}}$ enforces a Gaussian prior on the $\mathbf{z}_j$. In the case of a single scene, this objective simplifies to solving for the parameters ϕ of the MLP parameterization of Φ instead of the parameters ψ and latent codes $\mathbf{z}_j$. We solve Eq. 3.6 with stochastic gradient descent. Note that the whole pipeline can be trained end-to-end, without requiring any (pre-)training of individual parts. In Sec. 3.2, we demonstrate that SRNs discover both geometry and appearance, initialized at random, without requiring prior knowledge

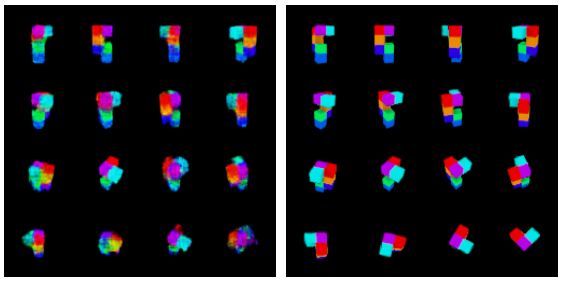

Figure 3.2: Shepard-Metzler object from 1k-object training set, 15 observations each. SRNs (right) outperform dGQN (left) on this small dataset.

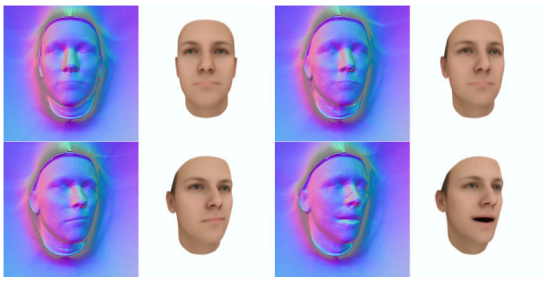

Figure 3.3: Non-rigid animation of a face. Note that mouth movement is directly reflected in the normal maps.

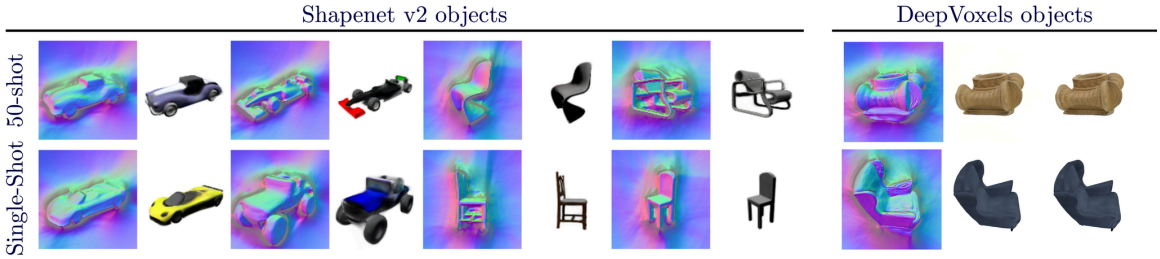

Figure 3.4: Normal maps for a selection of objects. We note that geometry is learned fully unsupervised and arises purely out of the perspective and multi-view geometry constraints on the image formation.

of either scene geometry or scene scale, enabling multi-view consistent novel view synthesis.

Few-shot reconstruction. After finding model parameters by solving Eq. 3.6, we may use the trained model for few-shot reconstruction of a new object instance, represented by a dataset $\mathcal{C} = \{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^N$. We fix θ as well as ψ, and estimate a new latent code $\hat{\mathbf{z}}$ by minimizing

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \sum_{i=1}^{N} \|\Theta_\theta(\Phi_{\Psi(\mathbf{z})}, \mathbf{E}_i, \mathbf{K}_i) - \mathcal{I}_i\|_2^2 + \lambda_{dep} \|\min(\mathbf{d}_{i,final}, \mathbf{0})\|_2^2 + \lambda_{lat} \|\mathbf{z}\|_2^2 \tag{3.7}$$

3.2 Experiments

We train SRNs on several object classes and evaluate them for novel view synthesis and few-shot reconstruction. We further demonstrate the discovery of a non-rigid face model. Please see the supplement for a comparison on single-scene novel view synthesis performance with DeepVoxels [1].

Implementation Details. Hyperparameters, computational complexity, and full network architectures for SRNs and all baselines are in the supplement. Training of the presented models takes

Figure 3.5: Interpolating latent code vectors of cars and chairs in the Shapenet dataset while rotating the camera around the model. Features smoothly transition from one model to another.

on the order of 6 days. A single forward pass takes around 120 ms and 3 GB of GPU memory per batch item. Code and datasets are available.

Shepard-Metzler objects. We evaluate our approach on 7-element Shepard-Metzler objects in a limited-data setting. We render 15 observations of 1k objects at a resolution of 64×64. We train both SRNs and a deterministic variant of the Generative Query Network [11] (dGQN, please see supplement for an extended discussion). Note that the dGQN is solving a harder problem, as it is inferring the scene representation in each forward pass, while our formulation requires solving an optimization problem to find latent codes for unseen objects. We benchmark novel view reconstruction accuracy on (1) the training set and (2) few-shot reconstruction of 100 objects from a held-out test set. On the training objects, SRNs achieve almost pixel-perfect results with a PSNR of 30.41 dB. The dGQN fails to learn object shape and multi-view geometry on this limited dataset, achieving 20.85 dB. See Fig. 3.2 for a qualitative comparison. In a two-shot setting (see Fig. 3.7 for reference views), we succeed in reconstructing any part of the object that has been observed, achieving 24.36 dB, while the dGQN achieves 18.56 dB. In a one-shot setting, SRNs reconstruct an object consistent with the observed view. As expected, due to the current non-probabilistic implementation, both the dGQN and SRNs reconstruct an object resembling the mean of the hundreds of feasible objects that may have generated the observation, achieving 17.51 dB and 18.11 dB respectively.

Shapenet v2. We consider the "chair" and "car" classes of Shapenet v.2 [114] with 4.5k and 2.5k model instances respectively. We disable transparencies and specularities, and train on 50 observations of each instance at a resolution of 128×128 pixels. Camera poses are randomly generated on a sphere with the object at the origin. We evaluate performance on (1) novel-view synthesis of objects in the training set and (2) novel-view synthesis on objects in the held-out, official Shapenet v2 test sets, reconstructed from one or two observations, as discussed in Sec. 3.1.4. Fig. 3.7 shows the sampled poses for the few-shot case. In all settings, we assemble ground-truth novel views by sampling 250 views in an Archimedean spiral around each object instance. We compare SRNs to three baselines from recent literature. Table 3.1 and Fig. 3.6 report quantitative and qualitative results respectively. In all settings, we outperform all baselines by a wide margin. On the training

Figure 3.7: Single- (left) and two-shot (both) reference views.

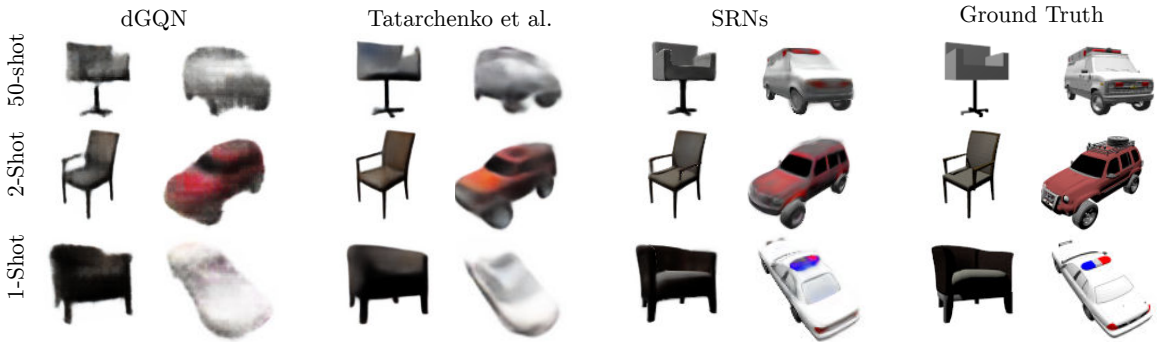

Figure 3.6: Qualitative comparison with [4] and the deterministic variant of the GQN [11], for novel view synthesis on the Shapenet v2 "cars" and "chairs" classes. We compare novel views for objects reconstructed from 50 observations in the training set (top row), two observations and a single observation (second and third row) from a test set. SRNs consistently outperforms these baselines with multi-view consistent novel views, while also reconstructing geometry. Please see the supplemental video for more comparisons, smooth camera trajectories, and reconstructed geometry.

	50 images (training set)		2 images		Single image	
	Chairs	Cars	Chairs	Cars	Chairs	Cars
TCO [4]	24.31 / 0.92	20.38 / 0.83	21.33 / 0.88	18.41 / 0.80	21.27 / 0.88	18.15 / 0.79
WRL [5]	24.57 / 0.93	19.16 / 0.82	22.28 / 0.90	17.20 / 0.78	22.11 / 0.90	16.89 / 0.77
dGQN [11]	22.72 / 0.90	19.61 / 0.81	22.36 / 0.89	18.79 / 0.79	21.59 / 0.87	18.19 / 0.78
SRNs	**26.23 / 0.95**	**26.32 / 0.94**	**24.48 / 0.92**	**22.94 / 0.88**	**22.89 / 0.91**	**20.72 / 0.85**

Table 3.1: PSNR (in dB) and SSIM of images reconstructed with our method, the deterministic variant of the GQN [11] (dGQN), the model proposed by [4] (TCO), and the method proposed by [5] (WRL). We compare novel-view synthesis performance on objects in the training set (containing 50 images of each object), as well as reconstruction from 1 or 2 images on the held-out test set.

set, we achieve very high visual fidelity. Generally, views are perfectly multi-view consistent, the only exception being objects with distinct, usually fine geometric detail, such as the windscreen of convertibles. None of the baselines succeed in generating multi-view consistent views. Several views per object are usually entirely degenerate. In the two-shot case, where most of the object has been seen, SRNs still reconstruct both object appearance and geometry robustly. In the single-shot case, SRNs complete unseen parts of the object in a plausible manner, demonstrating that the learned priors have truthfully captured the underlying distributions.

Supervising parameters for non-rigid deformation. If latent parameters of the scene are known, we can condition on these parameters instead of jointly solving for latent variables $\mathbf{z}_j$. We generate 50 renderings each from 1000 faces sampled at random from the Basel face model [115]. Camera poses are sampled from a hemisphere in front of the face. Each face is fully defined by a 224-dimensional parameter vector, where the first 160 parameterize identity, and the last 64

dimensions control facial expression. We use a constant ambient illumination to render all faces. Conditioned on this disentangled latent space, SRNs succeed in reconstructing face geometry and appearance. After training, we animate facial expression by varying the 64 expression parameters while keeping the identity fixed, even though this specific combination of identity and expression has not been observed before. Fig. 3.3 shows qualitative results of this non-rigid deformation. Expressions smoothly transition from one to the other, and the reconstructed normal maps, which are directly computed from the depth maps (not shown), demonstrate that the model has learned the underlying geometry.

Geometry reconstruction. SRNs reconstruct geometry in a fully unsupervised manner, purely out of necessity to explain observations in 3D. Fig. 3.4 visualizes geometry for 50-shot, single-shot, and single-scene reconstructions.

Latent space interpolation. Our learned latent space allows meaningful interpolation of object instances. Fig. 3.5 shows latent space interpolation.

Pose extrapolation. Due to the explicit 3D-aware and per-pixel formulation, SRNs naturally generalize to 3D transformations that have never been seen during training, such as camera close-ups or camera roll, even when trained only on up-right camera poses distributed on a sphere around the objects. Please see the supplemental video for examples of pose extrapolation.

Failure cases. The ray marcher may "get stuck" in holes of surfaces or on rays that closely pass by occluders, such as commonly occur in chairs. SRNs generates a continuous surface in these cases, or will sometimes step through the surface. If objects are far away from the training distribution, SRNs may fail to reconstruct geometry and instead only match texture. In both cases, the reconstructed geometry allows us to analyze the failure, which is impossible with black-box alternatives. See Fig. 3.8 and the supplemental video.

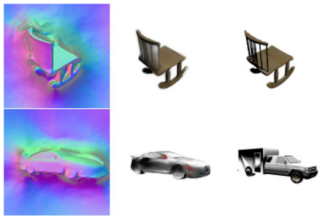

Figure 3.8: Failure cases.

Towards representing room-scale scenes. We demonstrate reconstruction of a room-scale scene with SRNs. We train a single SRN on 500 observations of a minecraft room. The room contains multiple objects as well as four columns, such that parts of the scene are occluded in most observations. After training, the SRN enables novel view synthesis of the room. Though generated images are blurry, they are largely multi-view consistent, with artifacts due to ray marching failures only at object boundaries and thin structures. The SRN succeeds in inferring geometry and appearance of the room, reconstructing occluding columns and objects correctly, failing only on low-texture areas (where geometry is only weakly constrained) and thin tubes placed between columns.

Please see `https://vsitzmann.github.io/srns` for qualitative results.

3.3 Conclusion

In this chapter, we introduced SRNs, a 3D-structured neural scene representation that implicitly represents a scene as a continuous, differentiable function. This function maps 3D coordinates to a feature-based representation of the scene and can be trained end-to-end with a differentiable ray marcher to render the feature-based representation into a set of 2D images. SRNs do not require shape supervision and can be trained only with a set of posed 2D images. We demonstrate results for novel view synthesis, shape and appearance interpolation, and few-shot reconstruction.

There are several exciting avenues for future work. SRNs could be explored in a probabilistic framework [11, 61], enabling sampling of feasible scenes given a set of observations. SRNs could be extended to model view- and lighting-dependent effects, translucency, and participating media. They could also be extended to other image formation models, such as computed tomography or magnetic resonance imaging. Currently, SRNs require camera intrinsic and extrinsic parameters, which can be obtained robustly via bundle-adjustment. However, as SRNs are differentiable with respect to camera parameters; future work may alternatively integrate them with learned algorithms for camera pose estimation [116]. SRNs also have exciting applications outside of vision and graphics, and future work may explore SRNs in robotic manipulation or as the world model of an independent agent.

Complex 3D scenes and compositionality. Here, we outline a particularly relevant direction for future work. While SRNs can represent simple room-scale scenes, few-shot reconstruction of complex, cluttered 3D environments remains an open problem. While this problem has many facets, we identify compositionality as a key challenge to achieving this goal. We formalize a simple notion of compositionality, that directly motivates a toy experiment that may test for this property. Given a room containing several objects, we identify the number of objects as a latent variable, n. For a model to be compositional, then, it needs to be capable of out-of-distribution generalization with respect to this latent variable. For instance, assume a training set of cubical rooms, with two chairs placed in each room — thus, $n = 2$. A scene representation learner, such as SRNs, is now trained for reconstruction of neural scene representations based on this dataset. For this model to be compositional, it should subsequently be able to reconstruct an unseen scene exactly identical to a scene in the training set, except there are *three* chairs placed in the room, i.e., $n = 3$. The present formulation of SRNs does *not* have this property. While it can be trained to represent and reconstruct room-scale scenes containing several objects each, it will fail to reconstruct a scene with a previously unseen number of objects. This is expected considering the previous discussion of inductive biases (see Section 1.1.3). Currently, SRNs do not have an inductive bias for compositionality. Similar

to the failure of models to discover 3D structure without an appropriate inductive bias, SRNs thus fail to recover compositional structure. Some prior work has attempted to equip computer vision models with inductive biases for compositionality. Scenes may be represented as a *set* of latent vectors, as opposed to a single latent vector, as in SRNs. In this paradigm, contributions of latent vectors to the global scene are usually controlled via softmax weighting, enabling the model to construct a scene of a variable number of components [117, 118]. Yet, these approaches are focused on image-space decomposition, effectively assigning each pixel to a component, and do *not* infer a global, scene-scale and 3D-structured representation. Furthermore, these approaches require an assumption on the cardinality of the set of components represented by latent codes. Closely related is the notion of exploiting shift invariance for compositionality. By design, a single filter in the first layer of a convolutional neural network (CNN) is oblivious of its location in the image. Thus, we can train a CNN on a dataset of images where each image contains, for instance, a single MNIST digit. Then, we may assemble an image that contains *two* non-overlapping MNIST digits. Shift invariance will guarantee that convolutional filters will generate the same local features for both MNIST digits at their respective locations. We note that this is a simplification, as not all popular CNN architectures are indeed strictly shift invariant and this is dependent on multiple factors such as receptive field, padding, etc., but this simplified discussion serves to illustrate the underlying principle. This property also applies to 3D convolutional neural networks. Recent work has leveraged this property by representing a scene via a 3D voxelgrid, where voxels are locally decoded into a continuous 3D scene representation, ambivalent to the number of objects in the scene [119, 120]. However, these models arguably side-step the issue of compositionality by virtue of not generating a single, global representation of the scene, but rather, learning *local primitives* of the scene, and encoding these primitives in a localized data structure such as a voxelgrid. As a result, these representations so far do not support few-shot reconstruction, instead requiring fairly dense point clouds. Another promising approach are recently proposed capsule networks [121, 122]. Capsule networks explicitly model object-part relationships by representing a scene as a hierarchy of capsules, small neural networks that each represent a particular feature in the target signal and reason about the pose of this feature. This succeeds in disentangling highly overlapping MNIST digits, but still requires the number of components to match between training and test time. To summarize, while several attempts at equipping scene representations with an inductive bias on compositionality exist, the author is unaware of an approach that generates a global, compact and 3D-structured scene representation that solves the proposed toy experiment.

Chapter 4

Inferring Semantic Information with SRNs

Representations of 3D objects learned by humans are multi-modal and support learning of new information with extremely limited supervision. For instance, a person does not need to be told that a car wheel is a car wheel thousands of times, but only a few tens of times. Subsequently, this newly learned semantic label can be directly associated with the person's mental image of a car. In addition, representations learned by humans enable 3D semantic reasoning: Observing a single side-view picture of a car, humans can easily imagine what the other side of the car will look like, including the different semantic classes of the parts involved, such as car door, wheel, or car window.

A similar level of semi-supervised learning and 3D scene understanding is also crucial for many tasks in computer vision, robotics, and autonomous driving. In these applications, algorithms must reason about a 3D scene given only partial information, such as a single image. In robotic grasping, for instance, a robot has to simultaneously reason about the 3D geometry, appearance, and semantic structure of an object in order to choose the optimal grasping point. In addition, human labeling is expensive, and these applications would thus greatly benefit from label-efficient learning approaches.

Recent progress in representation learning has enabled competitive performance on 2D tasks when only a limited amount of training data is available [25, 28, 123–125]. Here, 2D feature extractors are trained with massive amounts of unlabeled data on a surrogate task. Once the representation is learned, a limited amount of training data can be sufficient to train a simple classifier on the pre-trained feature representation [28]. While these approaches are applicable to 2D, image-based problems, they do not build a 3D-aware representation. Given a single image observation, they are incapable of making predictions about unseen perspectives of the scene or occluded parts, a task that is critical to 3D scene understanding and interaction.

In Chapter 3, we have introduced a 3D-structure-aware representation, Scene Representation

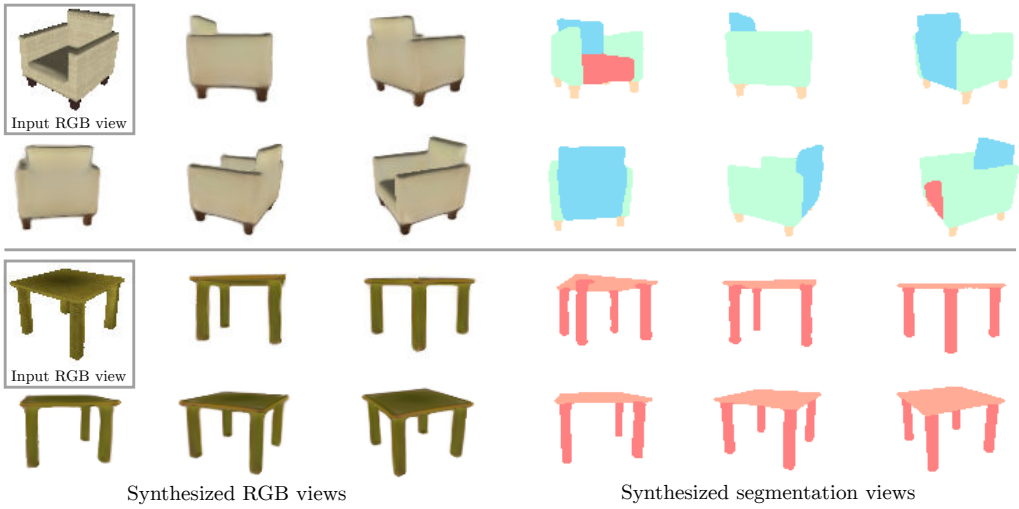

Figure 4.1: We leverage 3D-structure-aware representation learning for 3D reconstruction and semantic segmentation of objects, jointly reasoning about shape, appearance, and semantics. In this particular application, a single RGB image of an unseen object (upper left) is fed into the network, which is then capable of synthesizing perspectively consistent 3D RGB views of the object (left) as well as part-level semantic segmentation labels (right).

Networks, that enables prior-based predictions about occluded parts in the context of view synthesis. A naïve approach to extend SRNs to semantic segmentation would be to generate arbitrary perspectives of a scene from limited observations and then apply standard 2D methods for semantic segmentation or other tasks. Such an image-based approach, however, fails to infer a compact, multi-modal representation that would allow for joint reasoning about *all* aspects of the scene.

In this chapter, we propose to leverage SRNs not for view synthesis, but as a representation learning backbone, enabling downstream tasks by inferring a multi-modal, compact 3D representation of objects from 2D images. This approach enables, for the first time, dense 3D semantic segmentation given only 2D observations. We then embed the latent 3D feature representation, learned in an unsupervised manner given only posed 2D RGB images, in a standard semi-supervised learning strategy for semantic segmentation. This enables dense 3D semantic segmentation given extremely limited labeled training data of just a few tens of semantic segmentation labels. We demonstrate that this unique combination of unsupervised, 3D-structure-aware pre-training and supervised fine-tuning enables multi-view consistent view synthesis and semantic segmentation (see Fig. 4.1). Our approach further enables several other novel applications, including interpolation of 3D segmentation labels as well as 3D view and semantic label synthesis from just a single observed image or semantic mask. To summarize, we make the following key contributions:

- We extend Scene Representation Networks to perform semantic segmentation, leading to a semantically and 3D-structure-aware neural scene representation.

- In a semi-supervised learning framework, we demonstrate that the resulting representation can be leveraged to perform dense 3D semantic segmentation from only 2D observations, given as few as 30 semantic segmentation masks. We demonstrate that features learned by the 3D neural scene representation far outperform a neural scene representation without 3D structure.
- We demonstrate multi-view consistent rendering of semantic segmentation masks, including parts of the object that are occluded in the observation.
- We demonstrate joint interpolation of geometry, appearance, and semantic labels, and demonstrate how a neural scene representation can be inferred from either a color image or a semantic segmentation mask.

4.1 Prior work in semantic segmentation

The advent of deep learning has had a transformative impact on the field of semantic segmentation. Seminal work by Long et al. [126] introduced fully convolutional neural networks for pixel-level semantic labeling. Numerous CNN-based approaches further refined this initial idea of semantic segmentation on images [93, 127–129]. Closely related are CNNs for semantic segmentation of RGB-D data. Here, depth is treated as an additional input channel fed directly to the CNN [130–133]. In either case, semantic segmentation is performed on a per-pixel basis, i.e., each pixel is assigned a semantic class. Importantly, these models do not infer a global, 3D-structure-aware representation of the underlying scene. As a result, they are fundamentally incapable of predicting semantic labels of an *unseen* perspective of the scene. Recent work in this area has increasingly incorporated ideas from 3D computer vision. Semantic segmentation has thus been formulated in cases where both geometry and color information are available [134–136]. However, these methods operate on point clouds or voxel grids and therefore rely on explicit geometry representations. To the best of our knowledge, no semantic segmentation approach infers 3D semantic labels given a posed 2D image, including occluded parts. This is enabled by our approach, which may generate semantic label masks for novel camera perspectives, accurately generating, for instance, the 3D semantic label of the leg of a chair that was entirely occluded in the image observation. Note that we do *not* claim performance gains on 2D semantic segmentation. Our goal is to learn a single representation that jointly encodes information about 3D geometry, appearance and semantic segmentation. While we do rely on comparisons in image space, as this is the only data we have access to, we stress that this is merely a surrogate to demonstrate that the 3D representation contains semantic information, and not an attempt at an incremental improvement on 2D semantic segmentation.

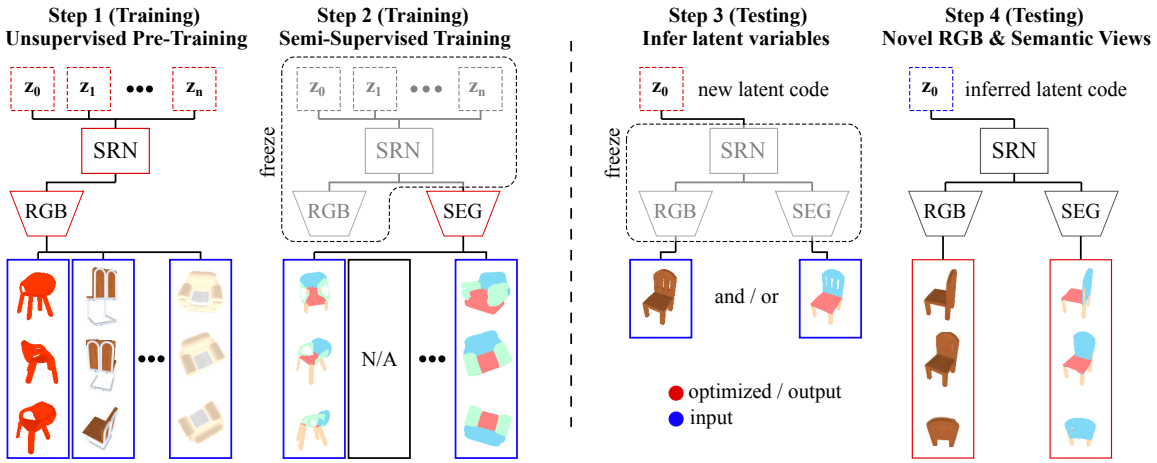

Figure 4.2: Overview of the proposed semi-supervised method. From left to right: (1) We train a scene representation network (SRN) for novel view synthesis of RGB images using a large dataset of 2D posed images in an autodecoder-framework [58], where each object instance is represented by its own code vector $\mathbf{z}_i$. (2) We then freeze code vectors and weights of the SRN and train a linear segmentation classifier on SRN features, using human-annotated semantic labels of a *very small* (30 images) subset of object instances in the training set. (3) At test time, given a *single* posed RGB image and/or label mask of an instance unseen at training time, we infer the latent code of the novel object. (4) Subsequently, we may render multi-view consistent novel RGB and semantic segmentation views of the object instance.

4.2 Method

In the following section, we demonstrate how we can extend SRNs to perform 3D semantic segmentation by adding a *Segmentation Renderer* in parallel to the existing *RGB Renderer*. We then view SRNs through the lense of representation learning and apply a semi-supervised learning strategy. This yields 3D semantic segmentation from 2D RGB observations and their camera parameters, given an extremely limited number of semantic segmentation masks.

4.2.1 Semantically-aware Scene Representation Networks

We extend the SRN framework to perform joint 3D reconstruction and semantic segmentation. We formalize semantic segmentation as a function that maps a world coordinate $\mathbf{x}$ to a distribution over semantic labels $\mathbf{y}$. This can be seen as a generalization of point cloud- and voxel-grid-based semantic segmentation approaches [49, 134, 137], which label a discrete set of world coordinates, sparsely sampling an underlying, continuous function. We recall that the key idea of SRNs it to encode a scene in the weights $\mathbf{w} \in \mathbb{R}^l$ of a fully connected neural network, the SRN itself. To this end, a scene is modeled as a function that maps world coordinates $\mathbf{x}$ to a feature representation of

local scene properties $\mathbf{v}$:

$$\text{SRN} : \mathbb{R}^3 \to \mathbb{R}^n, \quad \mathbf{x} \mapsto \text{SRN}(\mathbf{x}) = \mathbf{v}. \tag{4.1}$$

To leverage SRNs for semantic segmentation, we represent the semantic label function as a composition of the scene representation network (eq. 4.1) and an additional *Segmentation Renderer* SEG that maps a feature vector $\mathbf{v}$ to a distribution over class labels $\mathbf{y}$:

$$\text{SEG} : \mathbb{R}^n \to \mathbb{R}^m, \quad \mathbf{v} \mapsto \text{SEG}(\mathbf{v}) = \mathbf{y}. \tag{4.2}$$

In other words, this amounts to adding a *Segmentation Renderer* in parallel to the existing *RGB Renderer*. We may now enforce a per-pixel cross-entropy loss on the output of SEG at any world coordinate $\mathbf{x}$:

$$\mathcal{L}_{\text{co}} = \sum_{j=1}^{c} \hat{y}_j \log \sigma(\text{SEG}(\text{SRN}(\mathbf{x})) \tag{4.3}$$

where $\hat{y}_j$ is a one-hot ground-truth class label with c number of classes, and σ is the softmax function. The segmentation renderer can be trained end-to-end with the classic SRN architecture. In effect, this supervises the features $\mathbf{v}$ encoded in SRN to carry information about geometry via the ray-marcher, RGB color via the *RGB Renderer*, and semantic information via the *Segmentation Renderer* SEG. At test time, this formulation allows us to infer a code vector $\mathbf{z}$ from either RGB information, semantic segmentation information, or both. In any of these cases, a new code vector is inferred by freezing all network weights, initializing a new code vector $\mathbf{z}$, and optimizing $\mathbf{z}$ to minimize image reconstruction and/or cross entropy losses, see Fig. 4.2, Step 3.

4.2.2 Semi-Supervised Learning of Semantically-aware SRNs

While training an SRN end-to-end with a segmentation renderer on a large dataset of human-labeled images is intuitive, it has a significant weakness: it relies on a massive amount of labeled semantic data. Such labeled data may be difficult to obtain for a variety of different computer vision tasks. Moreover, it is desirable for an independent agent to infer an understanding of the different modes of an object it has not encountered. Such an unsupervised exploration cannot rely on a pipeline that requires thousands or millions of interactions with each object class to infer semantic properties.

For computer vision models that operate on per-pixel features, such as image recognition, object bounding box detection, or 2D semantic segmentation, the emerging field of *representation learning* aims to address this problem [25, 28, 124, 125]. However, none of these approaches infer a 3D-aware representation that would support predictions about parts of an object that are occluded in the input image.

Inspired by these approaches, we interpret SRNs as a representation learning technique. The

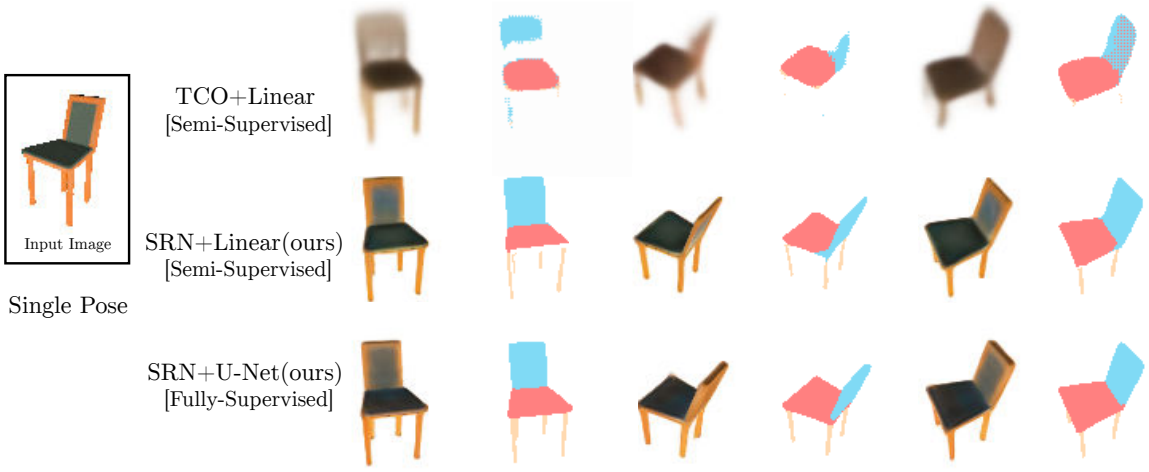

Figure 4.3: Comparison of the *single view* models, which can synthesize arbitrary RGB and segmentation views from a single posed RGB image. The proposed semi-supervised *SRN+linear* qualitatively outperforms the baseline semi-supervised approach by Tatarchenko et al. [4] (TCO) and is comparable to the fully-supervised *SRN+U-Net* approach in terms of 3D consistency and semantic segmentation. Note that all other models, including the *Oracle RGB+U-Net*, cannot perform such a task and require all views of ground truth RGB images at test time in order to perform segmentation.

multi-view re-rendering loss at training time can be seen as enforcing that the underlying representation, SRN, encodes information about appearance and geometry. We hypothesize that features that encode appearance and geometry will also be useful for the downstream task of dense 3D semantic segmentation.

We assume that for a small subset of our training corpus of RGB images and their camera parameters, we are given a few human-labeled per-pixel semantic labels. We now embed SRNs in a standard semi-supervised training framework. Fig. 4.2 summarizes the proposed semi-supervised approach. In the first step, we pre-train the weights of the hypernetwork HN, the latent embeddings $\mathbf{z}_i$ of the object instances in the training set, as well as the weights of the differentiable rendering purely for image reconstruction requiring only posed RGB images as well as their extrinsic and intrinsic camera parameters. Subsequently, we freeze $\mathbf{z}_i$ as well as the weights of HN and the differentiable renderer and train the proposed *Segmentation Renderer* SEG on the learned feature vectors $\mathbf{v}$, supervised with human-labeled semantic segmentation masks of a small subset of the training images. In this case of limited training data, we choose to parameterize the segmentation renderer SEG as a simple linear classifier.

4.3 Analysis

In this section, we demonstrate that the proposed representation learning approach using 3D-structure-aware neural scene representations succeeds in dense 3D semantic segmentation given extremely few labels.

Note that we do *not* claim performance gains on existing 2D semantic segmentation approaches. Instead, our goal is to learn a single, compact representation that jointly encodes information about 3D geometry, appearance and semantic segmentation. To do so, we have to rely on comparisons in image space, as this is the only data we have access to. We stress that this is merely a surrogate to demonstrate that the 3D representation contains semantic information, and not an attempt at an incremental improvement on 2D semantic segmentation. Note that existing 2D representation learning techniques are *not* applicable to the problem at hand, as they do not infer a 3D-aware representation and therefore rely on image input that shows all the parts of an object for which features are to be extracted. While it is possible to achieve similar input-output behavior with 2D approaches by building a pipeline that first leverages SRNs for novel view synthesis and subsequently feeds the image to a 2D model, this does *not* demonstrate a multi-modal 3D representation, but rather encodes 3D information in the SRNs representation and semantic information in the 2D architecture. This doesn't support simultaneous reasoning about multiple modalities in 3D, which is critical to many real-world computer vision tasks. We thus refrain from comparisons to such baselines.

Implementation. We implement all models in PyTorch. We train SRN-based models on Nvidia RTX8000 GPUs, and other models on Pascal TitanX GPUs. The SRN as well as the *RGB Renderer* are implemented as 4-layer MLPs with 256 units each, ReLU nonlinearities, and LayerNorm before each nonlinearity. The raymarcher is implemented as an LSTM [112] with 256 units. We ray march for 10 steps. We train our models using ADAM with a learning rate of 4e−4. SRN-based models are trained for 20k steps at a resolution of 64 with a batch size of 92, and trained for another 85k steps at a resolution of 128 with a batch size of 16. Image reconstruction loss and cross-entropy loss are weighted $200 : 8$, such that their magnitudes are approximately equal.

Dataset. For all experiments, we use the PartNet [138] and ShapeNet [114] datasets, which contains 3D meshes as well as their human-labeled semantic segmentation for a variety of object classes. We conduct experiments using the chairs and tables classes, with 4489 and 5660 object instances in the training set, 617 and 839 in the validation set, and 1214 and 1656 in the test set respectively. Partnet contains labels at several resolutions. We conduct all experiments at the coarsest level of segmentation, leading to 6 and 11 semantic classes respectively. We render observations using the Blender internal rasterizer. For training and validation sets, we render 50 camera perspectives sampled at random on a sphere around each object instance. For the test set, we render 251 camera

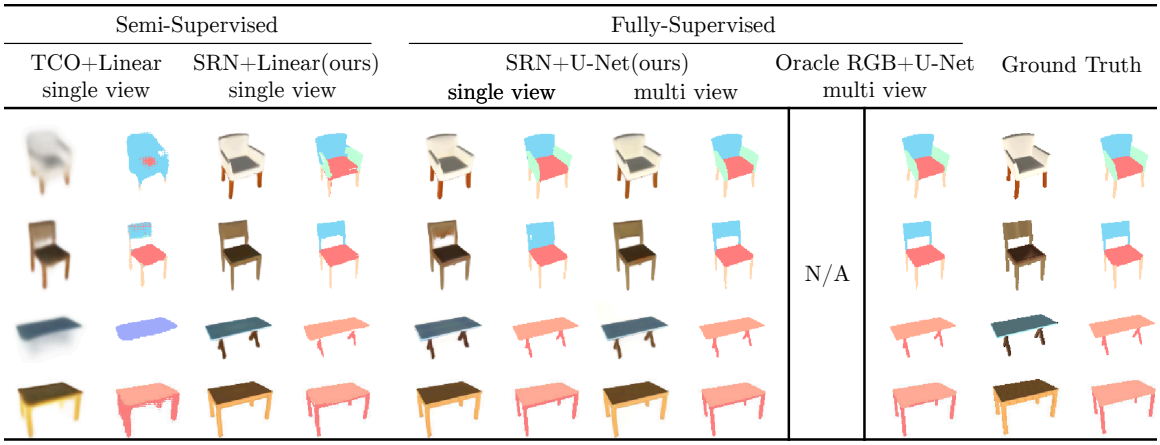

Figure 4.4: Qualitative comparison of semi-supervised and fully supervised approaches. *Semi-supervised approaches* (left column) are first pre-trained in an unsupervised manner on a large dataset of posed RGB images. Subsequently, a linear segmentation classifier is fit to a per-pixel feature representation on only *30 pairs of RGB images and their per-pixel semantic labels*. At test time, these methods receive a single posed RGB image. The proposed semi-supervised *SRN+linear* approach succeeds in reconstructing occluded geometry, semantic labels, and appearance, given only a single observation, and far outperforms the baseline 3D representation learning approach by Tatarchenko et al. [4]. *Fully supervised approaches* (center column) are trained on the full training corpus of RGB images and their per-pixel semantic class. At test time, *Oracle RGB+U-Net* receives novel RGB views of the object from an oracle, representing the upper bound of achievable segmentation accuracy. The *SRN+U-Net* baseline first leverages the 3D representation inferred by SRNs for novel view synthesis and segments the resulting image using a 2D U-Net. Here, the SRN representation is inferred from either a *single view* or *multiple views*. This serves as an upper bound for segmentation accuracy if only limited 2D observations are available. Please note that neither of these methods demonstrate a multi-modal 3D representation that encodes information about 3D appearance, geometry, and semantic information, instead performing 2D semantic segmentation in image space. Please see the supplement for semi-supervised *SRN+Linear* multi-shot results.

perspectives sampled from an Archimedean spiral around each object instance.

Evaluation. For quantitative evaluation of segmentation accuracy in image space, we adopt the mean pixel intersection over union (mIOU) and shape mIOU metric. We compute mIOU over all classes including the background class. For a single image, mIOU averages intersection over union over all classes. We subsequently compute the mean of mIOUs over all images and instances. In contrast, shape mIOU averages intersection over union scores across all images and instances separately for each semantic class. Note that the shape mIOU score is generally much lower than the mIOU score. This is due to the fact that the chosen objects have rare semantic classes that appear only in a small subset of all instances and are thus very difficult to score well on, lowering the per class average of shape mIOU.

	Semi-supervised, small dataset		Supervised, small dataset	Supervised, full dataset		
	TCO+linear single view	SRN+linear (ours) single view	Oracle RGB + U-Net multi view	SRN+U-Net single view	SRN+U-Net multi view	Oracle RGB + U-Net multi view
Chairs	28.4 / 23.3	48.7 / 42.3	42.2 / 38.0	60.9 / 51.8	74.2 / 63.7	77.3 / 66.0
Tables	32.8 / 11.4	58.7 / 18.3	50.3 / 17.9	70.8 / 26.5	78.9 / 40.5	81.0 / 44.7

Table 4.1: Quantitative comparison of semi-supervised and supervised approaches. We benchmark methods on mIOU as well as shape-mIOU. *Semi-supervised* approaches (left column) as well as the *Supervised, small-dataset* baseline are trained on 10 randomly sampled instances, 3 observations each. *Supervised, full dataset* (center column) baselines are trained on all training examples.

4.3.1 Representation learning for semi-supervised semantic segmentation.

We experimentally evaluate the proposed multi-modal, 3D-aware neural scene representation. We show how this enables dense 3D semantic segmentation from extremely few labels, given only *a single 2D observation of an object*, supporting subsequent multi-view consistent rendering of semantic information.

As discussed in 4.2.2, we first pre-train one scene representation network per object class to obtain a 3D-structure-aware neural scene representation. We then pseudo-randomly sample 10 object instances from the training set such that all semantic classes are present. For each of these instances, we randomly sample 3 posed images. Following the proposed semi-supervised approach, we now freeze the weights of all neural networks and latent codes. We train a simple linear classifier to map features at the intersection points of camera rays with scene geometry to semantic labels.

We benchmark the proposed method with a semi-supervised approach that uses an auto-encoder-based neural scene representation backend, the novel-view synthesis architecture of Tatarchenko et al. [4]. We pre-train this architecture for novel-view synthesis on the full training set to convergence of the validation error and then retrieve features before the last transpose convolutional layer. We then train a single linear transpose convolutional layer on these features on the same subset of labeled examples as the proposed semi-supervised approach for direct comparison.

As a 3D-structure aware reference model, we train the proposed model end-to-end with a U-Net segmentation classifier (see Sec. 2.1) on the full training dataset. This yields an upper bound of segmentation accuracy of an SRN-based approach in a fully supervised regime of abundant labeled training data. Note that this reference model does *not* infer a compact, multi-modal 3D-aware representation. Instead, this model may perform semantic segmentation in image space, and thus does not come with guarantees that the 3D-aware intermediate representation encodes all information necessary for 3D semantic reasoning.

We first demonstrate that the proposed method enables single-shot reconstruction of a representation that jointly encodes color, geometry, and semantic information. Fig. 4.3 shows the output of the auto-encoder style baseline, the proposed semi-supervised approach, and the end-to-end trained

Figure 4.5: Interpolating latent code vectors while tracking the camera around the model. Both semantic labels and color features transition smoothly from object to object, demonstrating a tight coupling of semantic labels, geometry and texture of the objects.

fully supervised reference model. The proposed semi-supervised approach succeeds in generating a multi-view consistent, dense 3D semantic segmentation, and performs comparable to the end-to-end supervised reference model. Lacking the 3D-structure-aware representation that the proposed model utilizes, the auto-encoder based neural scene representation baseline fails to perform multi-view consistent semantic segmentation. The first four columns of Fig. 4.4 show further qualitative results for dense 3D semantic segmentation given *single* and *multiple* input views. Finally, Table 4.1 shows quantitative results for the discussed methods. Consistent with qualitative results, the proposed semi-supervised approach far outperforms the auto-encoder based neural scene representation and even approaches the performance of the single view, fully-supervised SRN reference model (see Table 4.1, column 4 and Fig. 4.3). While the proposed model's linear classifier sometimes struggles with parts of objects with higher inter-instance variance, it performs similarly to the reference models on common parts of objects, such as backrests, legs or the seat in the case of chairs. Thus, the proposed method is the best model in the most difficult regime of single view reconstruction with semi-supervision and is comparable to the performance of the SRN reference model trained in a fully-supervised regime.

4.3.2 2D reference models with novel-view oracle.

As a reference for how well 2D semantic segmentation algorithms perform on this task, we train a modern U-Net architecture on all pairs of images and their per-pixel semantic labels in the training dataset. This 2D approach does not support predictions about parts of the object that are occluded in the input view. For this reason and to obtain an upper bound for the semantic segmentation performance, at test time, we feed this architecture with a ground-truth RGB rendering of each test view. We note that this is a *significantly* easier task, as these models do not have to perform any 3D reconstruction or, in fact, any 3D reasoning at all, and can instead infer a per-pixel semantic label from 2D pixel neighborhoods with perfect information.

Parameters of the U-Net are approximately matched with the proposed SRN-based approach. Each downsampling layer consists of one stride-one convolutional layer, followed by one stride-two

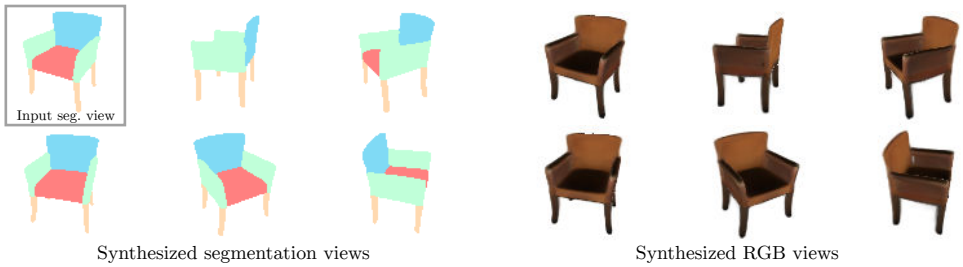

Figure 4.6: The proposed representation learning method is bi-directional: it may infer a neural scene representation either from RGB images or semantic segmentation masks, or both. Here, we show renderings of a chair, reconstructed from a single semantic segmentation mask, using the proposed fully supervised model.

convolutional layer. Each upsampling layer consists of one stride-two transpose convolutional layer, followed by one stride-one convolutional layer. We use BatchNorm and LeakyReLU activations after each convolutional block and dropout with a rate of 0.1. We train this model using the Adam optimizer with a learning rate of 4e−4 and a batch size of 64 until convergence of validation error after about 80k iterations or 20 epochs.

As expected, this oracle model (Table 4.1, column 6) outperforms the SRN reference models as well as the proposed semi-supervised method. However, it exists in the easiest regime of all the models, having access to the full dataset of segmentation maps for training and all the oracle RGB views at test time. Qualitatively, for more common objects in the test set, the single-view SRN reference model and the proposed single-view, semi-supervised model actually perform comparably to the oracle model, despite receiving only a small subset of the total information at both train and test time. Furthermore, the proposed models are able to perform the task of generating novel appearance and semantic segmentation views from a single observation, which the 2D-only oracle model cannot even evaluate. However, due to performing 3D reconstruction in addition to semantic segmentation, the proposed method fails whenever 3D reconstruction fails. This may be the case for out-of-distribution objects. This failure mode is completely absent from the 2D oracle method. Please refer to the supplemental video for a detailed investigation of such cases.

For further intuition, we train the reference 2D U-Net on the same 30 image-semantic-label pairs that the proposed semi-supervised approach is trained on. In order to prevent the model from over-fitting, we use the validation set to perform a hyper-parameter search over dropout rates and use early-stopping. Despite using additional segmentation data beyond the 30 training examples in order to perform early-stopping and having access to the RGB novel-view oracle at test time, this U-Net baseline (Tab. 1, column 3) is outperformed by the proposed semi-supervised method. This baseline does not have the compact 3D multi-modal representation of the proposed method, and thus fails to generalize to other instances of the same class nor maintain 3D-consistent views.

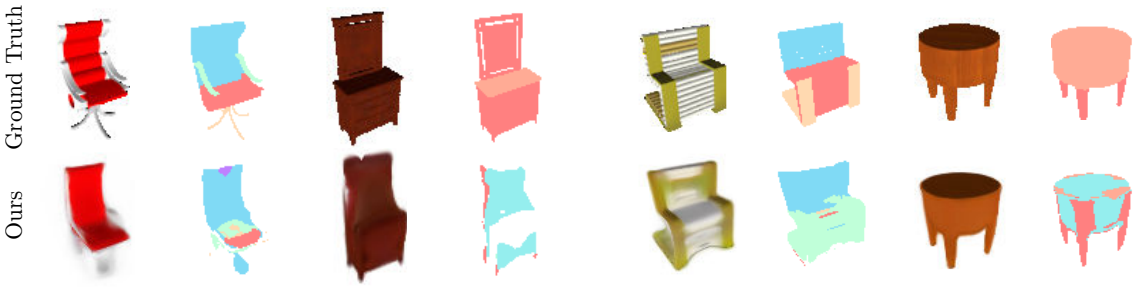

Figure 4.7: Failure cases.

4.3.3 Instance Interpolation.

Interpolating latent vectors inferred in the proposed framework amounts to jointly interpolating geometry, appearance and semantic information. Fig. 4.5 visualizes a latent-space interpolation of two chairs in the test set, both reconstructed from a single view by the proposed semi-supervised linear model. Geometry, appearance and semantic labels interpolate smoothly, demonstrating a tight coupling of these modalities.

4.3.4 3D reconstruction from semantic mask.

As an instantiation of the auto-decoder framework [58], inferring the neural scene representation of a novel object amounts to initializing and subsequently optimizing a new embedding vector to minimize reconstruction error. As the proposed method may be supervised by both semantic segmentation labels and RGB renderings, it also enables reconstruction of neural scene representations through either modality. Fig. 4.6 visualizes 3D reconstruction of a chair from a single posed segmentation mask, while Fig. 4.1 demonstrates reconstruction from a single posed color image.

4.3.5 Failure cases.

Fig. 4.7 displays failure cases of the proposed approach. The proposed approach inherits limitations and failure cases of scene representation networks, such as failure to reconstruct strong out-of-distribution samples or objects with small gaps or high-frequency geometric detail. In these cases, the semantic segmentation fails as well. In the semi-supervised regime, the linear classifier sometimes fails to assign the correct class even if geometry and appearance were reconstructed correctly, which we attribute to its limited representative power. We note that as both appearance-based 3D neural scene representation methods as well as semi-supervised representation learning methods further develop, these failure cases will improve.

4.4 Conclusion

In this chapter, we presented a 3D representation learning approach to joint reconstruction of appearance, geometry, and semantic labels. Our semi-supervised approach allows us to perform dense 3D semantic segmentation of a class of objects given as few as 30 human-annotated, posed semantic segmentation masks. At test time, this enables full 3D reconstruction and dense semantic segmentation from either posed RGB images, semantic segmentation masks, or both, from as few as a single observation. After reconstruction, the proposed approach enables multi-view consistent RGB and semantic view generation. We believe that this work outlines an exciting direction of extending representation learning methods into 3D and taking advantage of features that encode both shape and appearance. As both of these fields independently develop more powerful techniques, we expect that our proposed technique, which utilizes these methods collaboratively, will also improve.

Chapter 5

Conclusion

Scene representation is an essential aspect of artificial intelligence. Generative modeling with neural networks offers a powerful framework to leverage vast amounts of data in the form of weakly structured, unlabeled observations.

It is straightforward to assemble generative models that comply with the input-output behavior of the proposed self-supervised scene representation learning framework. Yet, existing methods for scene representation learning lacked an inductive bias on the 3D structure of scenes, and therefore failed to discover shape and geometry of scenes from limited data.

Our main contribution was a principled approach to equipping neural network models with such an inductive bias. We achieved this by factorizing the encoder and neural rendering algorithms into sub-routines that explicitly enforce multi-view and perspective geometry. By leveraging recent differentiable programming frameworks, it is straightforward to optimize the resulting algorithm in an end-to-end manner, thereby learning distributed representations of 3D environments. In Chapter 2, we demonstrated that this framework can enable high-quality novel view synthesis on real-world 3D environments. However, the proposed approach was limited by its reliance on the classic voxelgrid as its scene representation. By rethinking the nature of scene representations in a deep learning framework and replacing the classic voxelgrid with an implicit neural scene representation, we enabled the learning of priors over distributions of 3D environments, in turn enabling full 3D reconstruction of objects from only a single observation. Finally, we demonstrated that 3D structure and human assignment of semantic labels are closely linked, enabling us to exploit the features learned by Scene Representation Networks for semi-supervised semantic segmentation from an extremely limited dataset of labeled examples.

5.1 Discussion

Here, we would like to reflect on the nature and potential role of scene representation learning in computer vision itself. We consider the case when the only available observations are 2D images, and no measurements of scene geometry are available. We first make a trivial statement: Any 2D image observation captured with a camera is a *two-dimensional projection* of an otherwise *three-dimensional* scene.

This seemingly obvious insight raises important questions for current state-of-the-art approaches to computer vision. Existing approaches to major computer vision applications such as semantic segmentation, classification, object detection, object tracking, scene understanding, etc. rely almost exclusively on 2D feed-forward convolutional neural networks (CNNs). CNNs infer increasingly abstract 2D feature representations of an image via a composition of convolutional layers with normalization layers and nonlinearities. Following above insight, we might ask: How do CNNs account for the 3D nature of imaged environments?

The answer is that CNNs have *no* explicit notion of 3D structure of the scenes underlying the observations. The key inductive bias enforced by CNNs is 2D shift-invariance of features. We note that even this assumption, however, does not hold in 3D environments, where observations follow a perspective image formation model. For instance, when looking down a hallway with wooden tiling, perspective foreshortening of the wood texture yields a signal that is significantly different for parts of the floor close to the camera than for those that are further away. This lack of 3D inductive biases is demonstrated by the failure of CNN baselines in the previous chapters to correctly generate multi-view and perspectively consistent observations of even simple 3D scenes.

More generally, it is well-known that representations inferred by CNNs are not *equivariant* to 3D transformations such as 3D rotations, translations, scaling, changes in focal length, etc. Equivariance expresses the intuitive notion that whatever representation is inferred by the encoder should disentangle, for instance, the *scale* of an object from the *nature* of the object. Humans naturally have this ability: when presented with an image of small cat, we can easily infer that we are, in fact, observing a cat, albeit a small one. In contrast, a CNN that has been trained only on images of average-sized cats might fail entirely to recognize that the image displays a cat, and has no notion of its scale altogether. Though some recent work proposes methods to alleviate this shortcoming in CNNs [139], state-of-the-art image classification models do not account for this fact. As a result, CNNs require massive amounts of labeled training data to gain robustness to 3D transformations.

In contrast, the models discussed in this work are all endowed with a 3D inductive bias in the form of a 3D-structure-aware scene representation and an image formation model based on multi-view and perspective geometry. Specifically, Scene Representation Networks as discussed in Chapter 3 are fully equivariant to rigid body motion of the camera as well as modifications of the camera intrinsic parameters. This enables Scene Representation Networks to generalize to camera perspectives completely unobserved at training time, such as a zoomed-in view or rotation around

the camera's principle axis.

This suggests a drastically different approach to tackling existing computer vision applications, that can be summarized by the paradigm: **"All computer vision is 3D computer vision"**. Instead of engineering models that operate in image space without a well-defined scene representation, we argue to frame *all* vision models as models that operate on 3D-structured neural scene representations. In the case of single-image object detection, for instance, this would mean that the first step of the model is a *prior-based scene reconstruction* step performed by an observation encoder INFER, yielding a 3D-structure-aware neural scene representation that leverages learned prior information to resolve ambiguities. Subsequently, a classifier operates directly on the neural scene representation, *without access to the original image observations altogether*. In Chapter 4, we have seen one, albeit simple, instantiation of this framework: Instead of a 2D CNN performing semantic segmentation in image space, we exploit the power of scene representation networks to reconstruct a full 3D representation, and formulate semantic segmentation as an operation on this inferred representation. This inherits all properties of scene representation networks, such as equivariance to rigid-body motion of the camera.

Of course, existing scene representation learners —including the ones proposed in this work—are as of know extremely limited in the complexity of the scenes that they can learn effective priors over. This leads us to the final section of this work.

5.2 Future Work

While we have taken a significant step towards enabling neural networks to effectively infer properties of 3D environments, several open questions remain. First and foremost, DeepVoxels and Scene Representation Networks seem to suggest a trade-off in the learning of priors and photo-realism. While DeepVoxels are able to model real-world captures with high visual quality, Scene Representation Networks fall short of photorealism, and are constrained to rather simple 3D environments such as single objects or simple room environments. While follow-up work to scene representation networks such as Mildenhall et al. [140] has since demonstrated that implicit scene representations are capable of complete photo-realism, these methods have so far withstood the learning of priors over the parameters of the learned representation. As of today, to the best of the author's knowledge, no method has been proposed that allows *both* high-quality novel view synthesis *and* the learning of priors and thereby, inferring the scene representation from few observations. As a result, methods such as DeepVoxels or Mildenhall et al. [140] require rather many, densely sampled observations.

In this work, we rely on the availability of camera parameters for all image observations. While camera pose estimation is a well-understood research field and modern sparse bundle-adjustment algorithms can estimate camera parameters robustly and fairly efficiently, it is nevertheless desirable to leverage shape and appearance priors also for the camera parameter estimation itself. As all

proposed models are differentiable with respect to the camera parameters, future work may explore integration with recent differentiable camera parameter estimators [116].

Another promising direction of future research are the multitude of properties of natural 3D environments *other* than their apparent 3D structure, such as the dynamics governed by hamiltonian mechanics, light transport and its interactions with materials such as reflections, scattering and participating media, or the propagation of sound. Modern humans have an extraordinary intuitive understanding of these effects, enabling us to effectively integrate cues related to these effects into our model of a 3D environment.

As suggested in the previous section, a key direction of future research will be the application of scene representation learners to existing computer vision problems, such as classification, segmentation, object detection, etc. Another fascinating direction for future work, however, is the applications of scene representations in fields *other* than computer vision. In natural language, for instance, a neural representation may enable integration, persistence, and subsequent referencing of communicated information.

Lastly, it is unclear how precise models of environments have to be to support intelligent decision-making. To grasp an object, for instance, it may not be necessary to infer a representation that accurately models specular effects on its surface, and instead, an approximate representation of the object's geometry may suffice. This suggests future work that integrates the proposed 3D-structure-aware scene representation with an independent agent, such as a robot performing grasping or an autonomous vehicle.

Appendix A

DeepVoxels additional results

A.1 Results on two Additional Synthetic Scenes

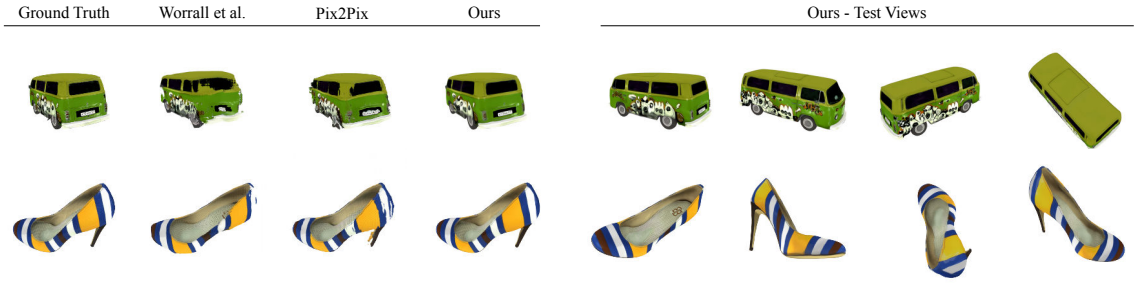

Figure A.1: Qualitative results on two additional objects.

	Bus	Shoe
	PSNR / SSIM	PSNR / SSIM
Nearest Neighbor	17.96 / 0.89	17.49 / 0.88
Tatarchenko et al. [4]	22.58 / 0.94	20.00 / 0.91
Worrall et al. [5]	19.30 / 0.91	20.34 / 0.91
Pix2Pix (Isola et al.) [6]	24.41 / 0.95	23.45 / 0.93
Ours	**31.78** / **0.98**	**33.70** / **0.98**

Table A.1: Quantitative comparison to four baselines on two additional scenes. Our approach obtains the best results in terms of PSNR and SSIM on all objects. See Fig. A.1 for qualitative results.

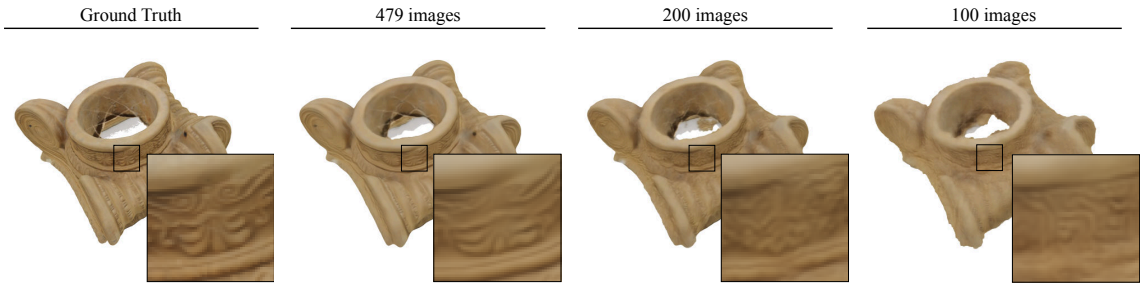

Figure A.2: Impact of number of training images on performance. From left to right: ground truth, models trained with 479, 200, and 100 images. While fine detail degrades with a decreasing number of images, the overall geometry stays coherent. Notably, on the pedestal dataset, we still outperform all baselines for 200 images with 30.65dB and with 26.15dB for 100 images, less than a fourth of the data.

A.2 Sensitivity to Number of Training Images

We investigate how the number of images in the training set impacts model performance. Figure A.2 shows novel views for a varying number of training images. Performance degrades gracefully with a decreasing number of images: While fine detail is reduced significantly, 3D geometry and rigid body motion is preserved.

A.3 Sensitivity to Volume Resolution

We demonstrate the impact of a smaller volume resolution on model performance. Figure A.3 shows novel views for a coarser discretization than the proposed 32 voxels per dimension. While high-frequency detail is degraded, 3D geometry stays consistent.

A.4 Sensitivity to Additive Rotational Noise

We demonstrate the impact of additive uniform random noise added to training poses. We trained the model on the pedestal with 1° and 5° random uniform rotation added to training poses. The model achieves 26.98dB at 1°, still out- performing all baselines, and 23.33dB at 5°, outperforming three baselines. See Fig. A.4 for examples. We note that our model has no trouble handling noisy poses obtained via bundle-adjustment.

A.5 Results on Real-World Captures

Here, we outline additional details on real-world data captured with a digital single-lens reflex camera. For each scene, we captured approximately 450 photographs. We use sparse bundle adjustment

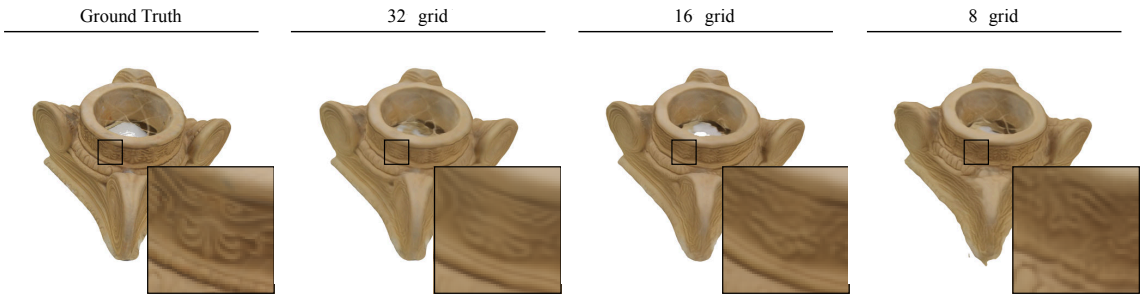

Figure A.3: Impact of volume resolution. From left to right: ground truth, models with grid resolution 32, 16 and 8 and PSNRs over the whole test set of 32.35dB, 30.13dB and 25.15dB. Quality deteriorates gracefully, with loss of fine detail but preservation of overall geometry.

Figure A.4: Impact of additive geometric noise in camera poses. From left to right: ground truth, no noise, 1° rotational noise, and 5° rotational noise and PSNRs over the whole test set of 32.35dB, 26.98dB and 23.33dB.

to estimate intrinsic and extrinsic camera parameters, as well as a sparse point cloud of keypoints to estimate the scale and center of gravity of the scene. Photographs were subsequently symmetrically center-cropped and downsampled to a resolution of 512×512 pixels. Zoom and focus were set at fixed values throughout the capture.

A.6 Failure Cases

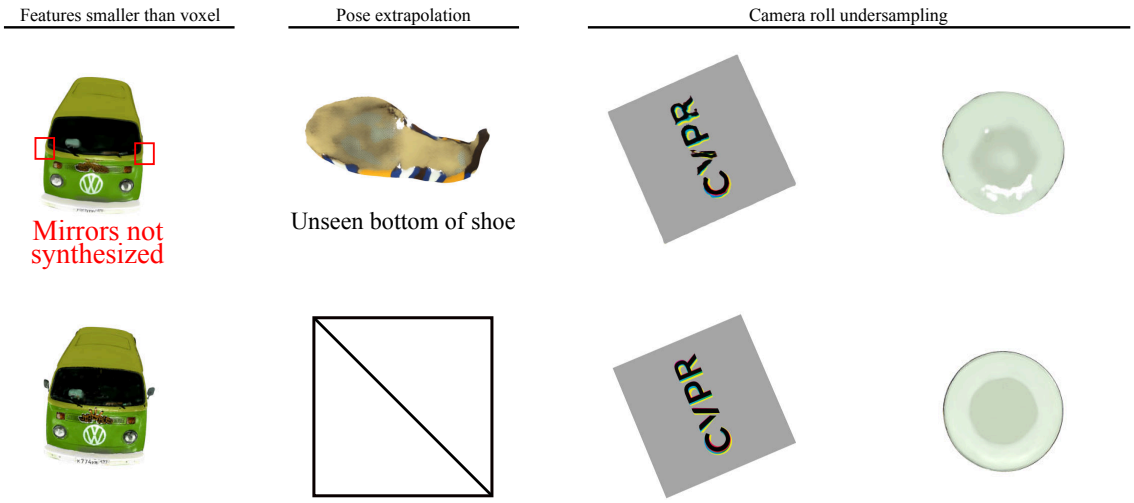

Figure A.5: Failure cases of the proposed method. From left to right: If features are significantly smaller than a voxel, our method fails to synthesize them. For strong pose extrapolation, our method may generate imagery with holes or views that are not multi-view consistent. Since our training data does not include variation in camera roll, object views are sampled only sparsely when seen from the top due to the gimbal lock. This may lead to multi-view inconsistencies when objects are seen from the top (see the "V" in the CVPR logo). For the vase, we found this may lead to "holes" in generated images (right) - this may be due to the similarity of the vase color to the background color.

A.7 DeepVoxels Submodule Architectures

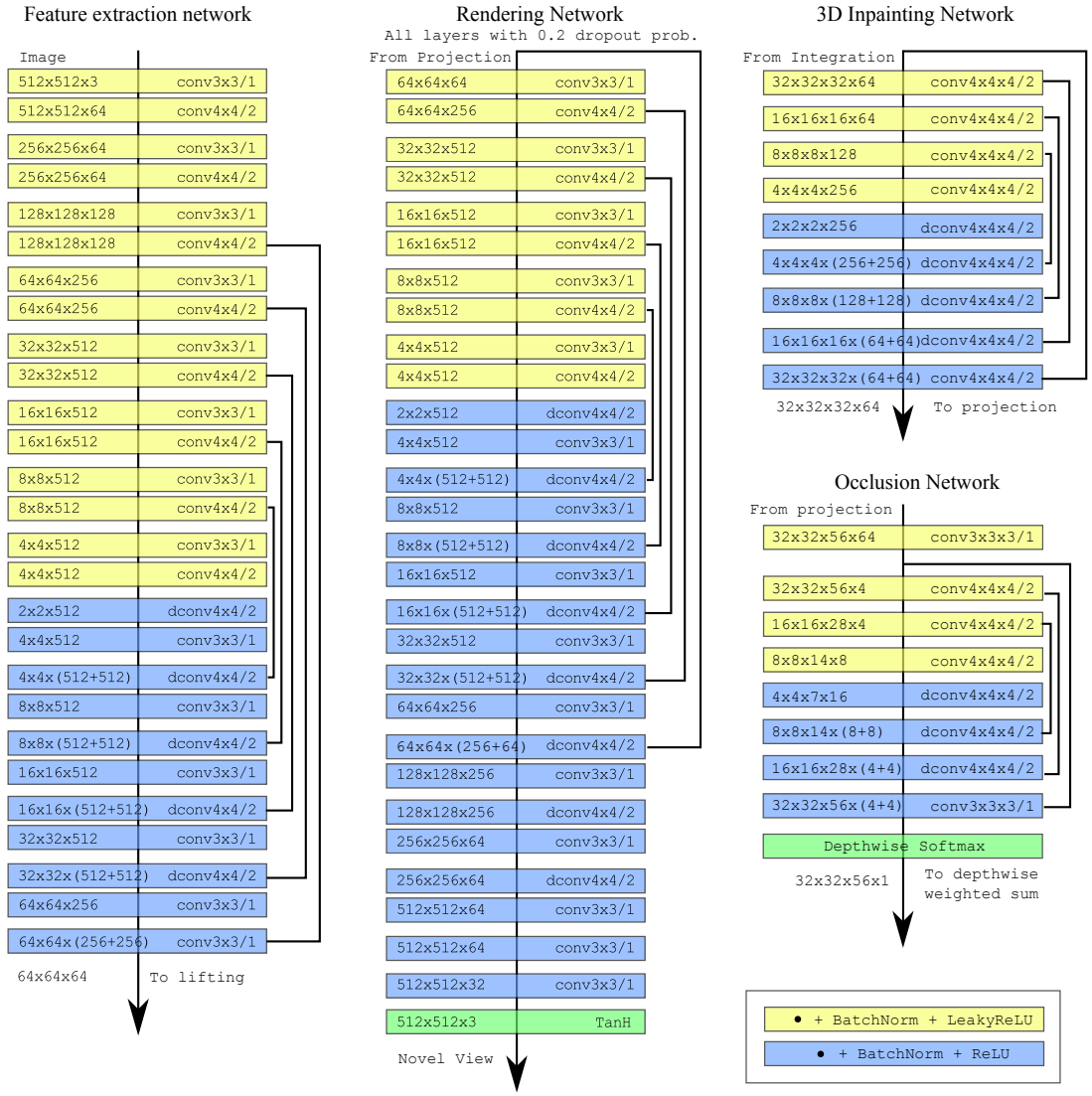

Figure A.6: Precise architectures of the feature extraction, rendering, inpainting and occlusion networks. They all follow the basic U-Net structure, while following general best practices in generative network architectures: Reflection padding instead of zero padding, kernel size divisible by stride.

A.8 Baseline Architecture Tatarchenko et al. [4]

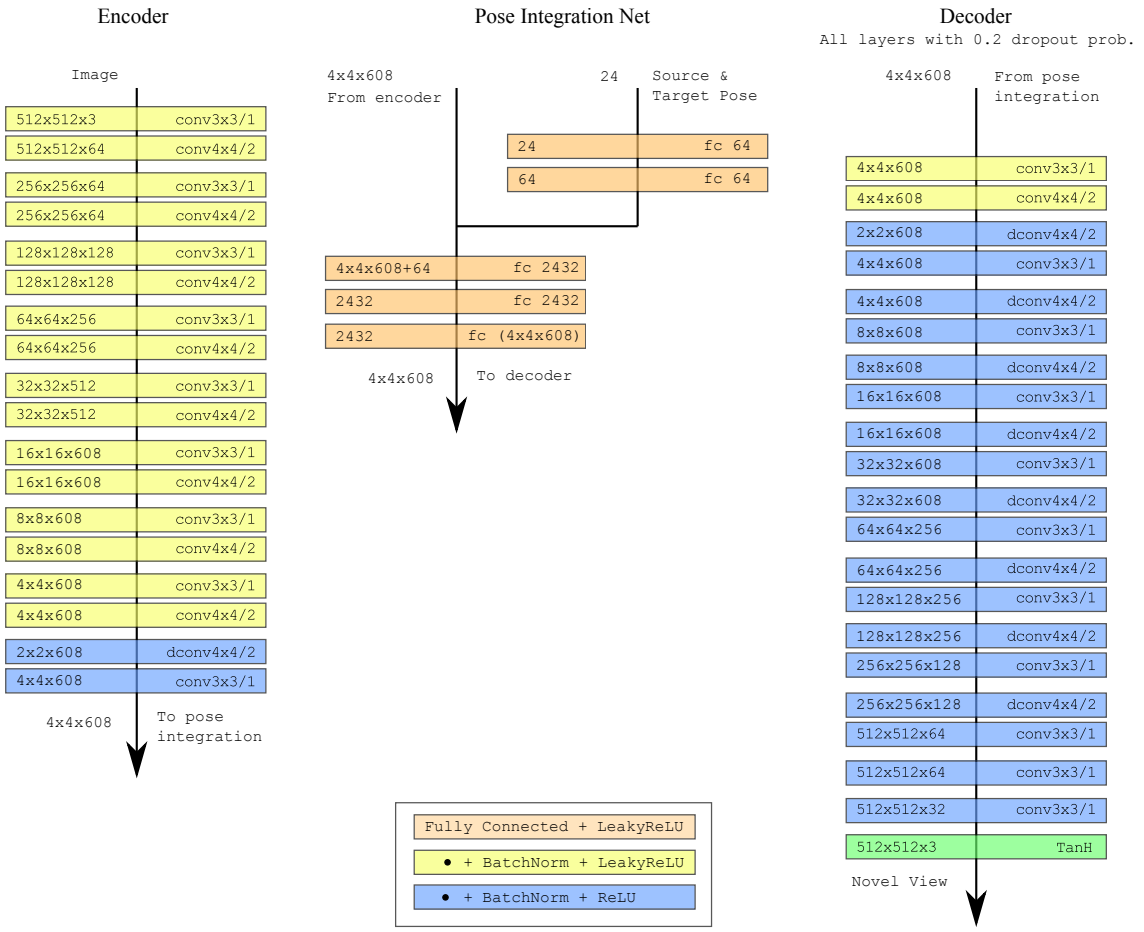

Figure A.7: Architectural details of the autoencoder baseline model with latent pose concatenation as proposed by Tatarchenko et al. [4].

A.9 Baseline Architecture Worrall et al. [5]

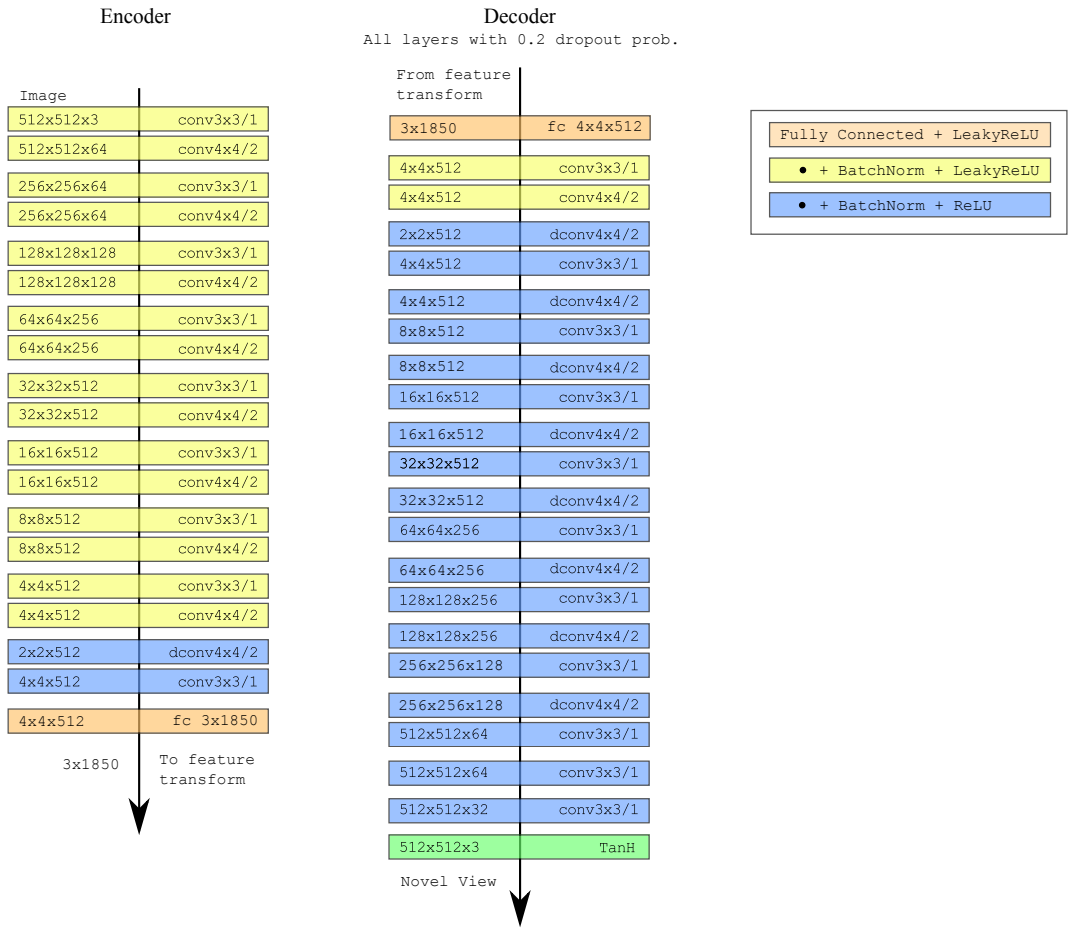

Figure A.8: Architectural details of the baseline model based on a rotation-equivariant latent space as proposed by Worrall et al. [5].

A.10 Baseline Architecture Pix2Pix (Isola et al. [6])

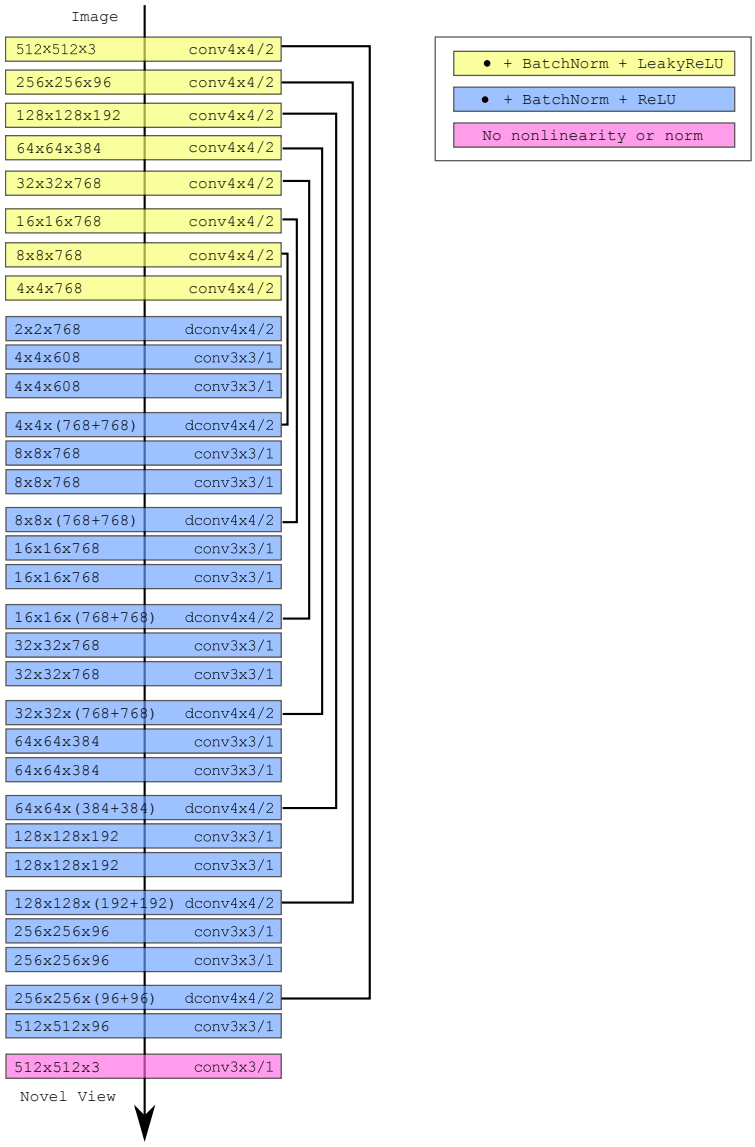

Figure A.9: Architectural details of the image-to-image translation baseline model based on Pix2Pix by Isola et al. [6].

A.11 Comparison of Ground-Truth Depth to Estimated Depth

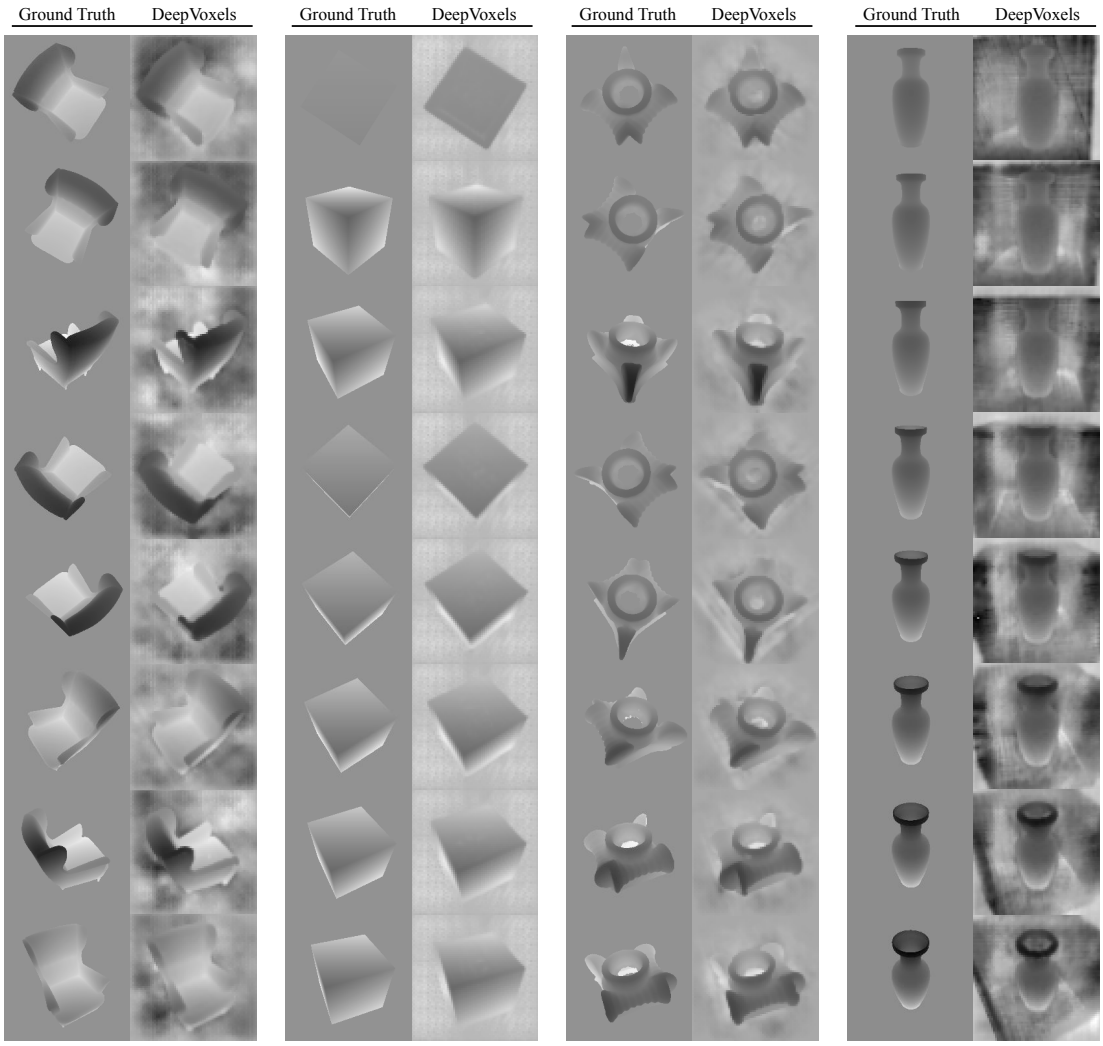

Figure A.10: Comparison of ground truth depth maps and the depth maps implicit in the DeepVoxels voxel visibility scores (upsampled from a resolution of 64×64 pixel). We note that these depth maps are learned in a fully unsupervised manner (at no time does our model see a depth map), and only arise out of the necessity to reason about voxel visibility. The background of the depth map is unconstrained in our model, which is why depth values may deviate from ground truth.

A.12 Pose Extrapolation

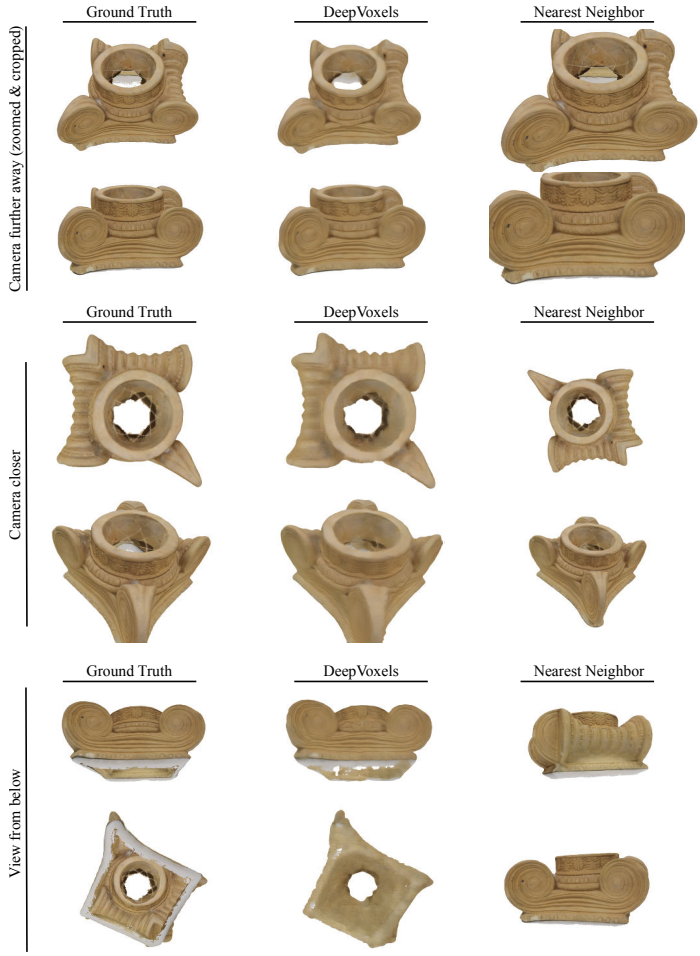

Figure A.11: Our training set comprises views sampled at random on the surface of the northern hemisphere. Images in each row are consistently scaled and cropped. We show views that require the model to extrapolate more aggressively - such as increasing the camera distance by a factor of 1.3 (top row), decreasing the camera distance by a factor of 0.75 (middle row) or leaving the northern hemisphere altogether and sampling from the southern hemisphere (bottom row). We show a comparison of ground truth (left column), our model output (center column), and the nearest neighbor in the training set (right column). For the proposed model, detail is lost especially in cases where the model has either never seen these points on the object (bottom row), or where details are seen from closeby for the first time (middle row). Generally, however, the performance degrades gracefully - rigid body motion and general geometry stay consistent, with loss in fine-scale detail and a few failures in occlusion reasoning.

Appendix B

Scene Representation Networks additional results

B.1 Additional Results on Neural Ray Marching

Computation of Normal Maps We found that normal maps visualize fine surface detail significantly better than depth maps (see Fig. B.1), and thus only report normal maps in the main submission. We compute surface normals as the cross product of the numerical horizontal and vertical derivatives of the depth map.

Ray Marching Progress Visualization The z-coordinates of running and final estimates of intersections in each iteration of the ray marcher in camera coordinates yield depth maps, which visualize every step of the ray marcher. Fig. B.1 shows two example ray marches, along with their final normal maps.

B.2 Comparison to DeepVoxels

We compare performance in single-scene novel-view synthesis with the recently proposed DeepVoxels architecture [1] on their four synthetic objects. DeepVoxels proposes a 3D-structured neural scene representation in the form of a voxel grid of features. Multi-view and projective geometry are hard-coded into the model architecture. We further report accuracy of the same baselines as in [1]: a Pix2Pix architecture [6] that receives as input the per-pixel view direction, as well as the methods proposed by [4] as well as by [5] and [104].

Table B.1 compares PSNR and SSIM of the proposed architecture and the baselines, averaged over all 4 scenes. We outperform the best baseline, DeepVoxels [1], by more than 3 dB. Qualitatively,

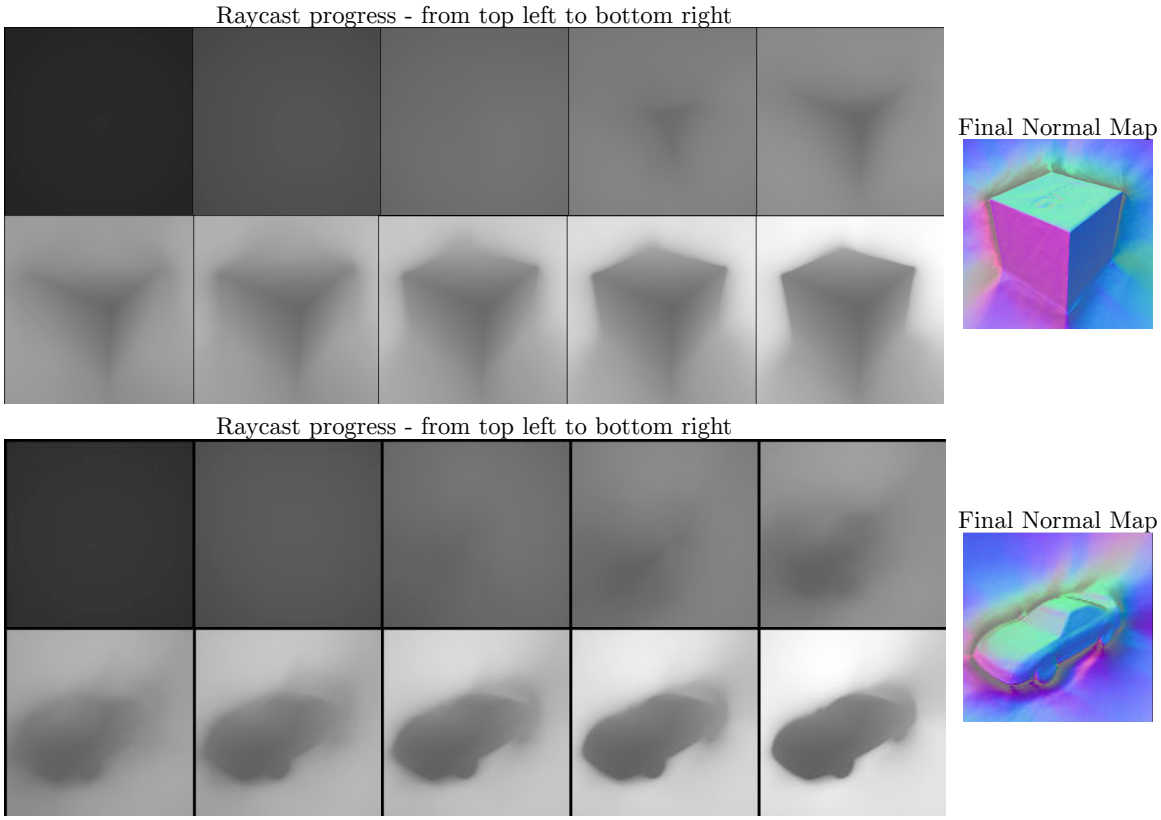

Figure B.1: Visualizations of ray marching progress and the final normal map. Note that the uniformly colored background does not constrain the depth - as a result, the depth is unconstrained around the silhouette of the object. Since the final normal map visualizes surface detail much better, we only report the final normal map in the main document.

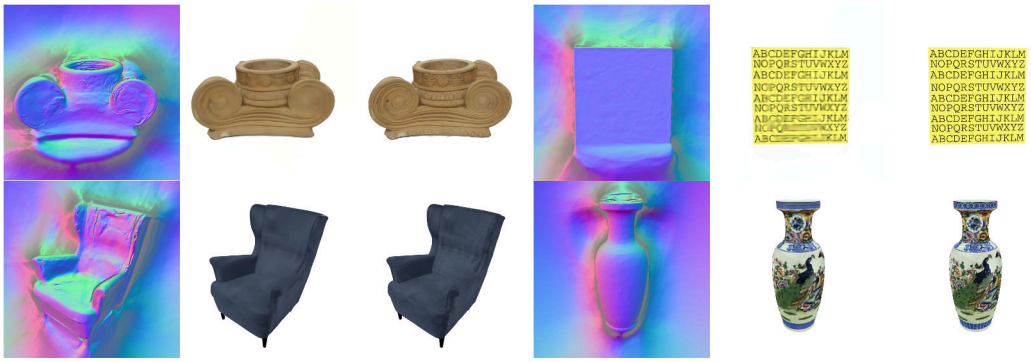

Figure B.2: Qualitative results on DeepVoxels objects. For each object: Left: Normal map of reconstructed geometry. Center: SRns output. Right: Ground Truth.

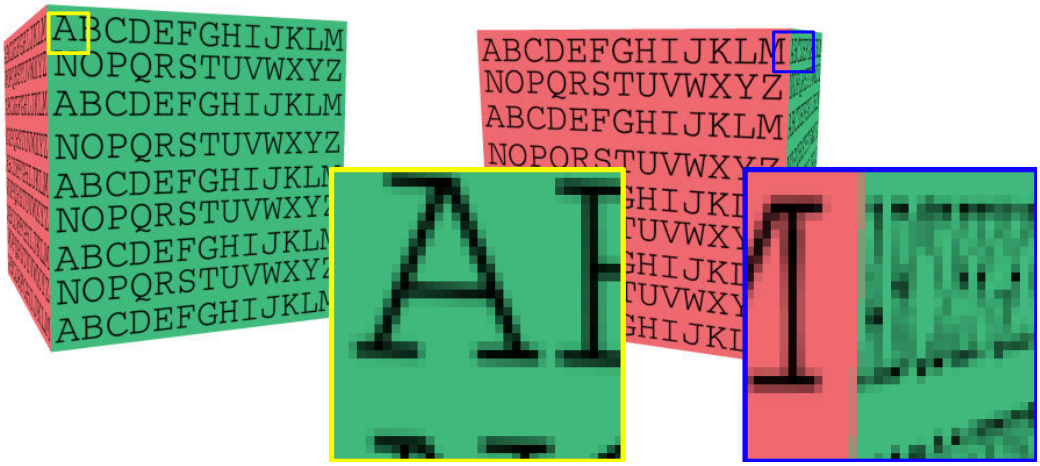

Figure B.3: Undersampled letters on the side of the cube (ground truth images). Lines of letters are less than two pixels wide, leading to significant aliasing. Additionally, the 2D downsampling as described in [1] introduced blur that is not multi-view consistent.

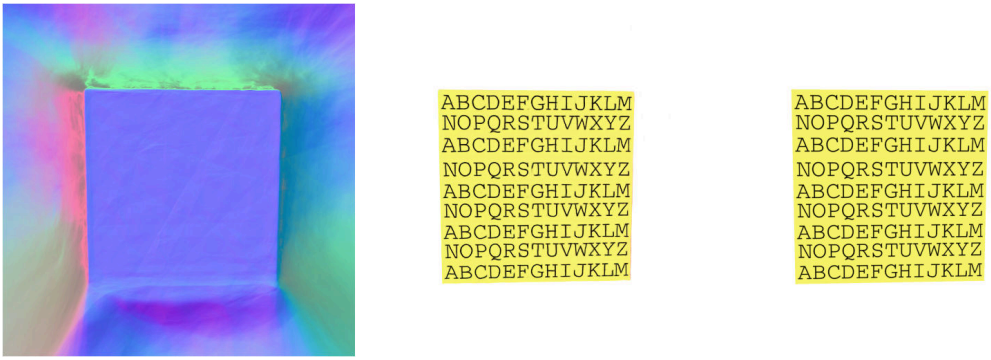

Figure B.4: By using a U-Net renderer similar to [1], we can reconstruct the undersampled letters. In exchange, we lose the guarantee of multi-view consistency. Left: Reconstructed normal map. Center: SRNs output. Right: ground truth.

	PSNR	SSIM
[4]	21.22	0.90
[5]	21.22	0.90
Pix2Pix [6]	23.63	0.92
DeepVoxels [1]	30.55	0.97
SRNs	**33.03**	**0.97**

Table B.1: Quantitative comparison to DeepVoxels [1]. With 3 orders of magnitude fewer parameters, we achieve a 3dB boost, with reduced multi-view inconsistencies.

DeepVoxels displays significant multi-view inconsistencies in the form of flickering artifacts, while the proposed method is almost perfectly multi-view consistent. We achieve this result with 550k parameters per model, as opposed to the DeepVoxels architecture with more than 160M free variables. However, we found that SRNs produce blurry output for some of the very high-frequency textural detail - this is most notable with the letters on the sides of the cube. Fig. B.3 demonstrates why this is the case. Several of the high-frequency textural detail of the DeepVoxels objects are heavily undersampled. For instance, lines of letters on the sides of the cube often only occupy a single pixel. As a result, the letters alias across viewing angles. This violates one of our key assumptions, namely that the same $(x, y, z) \in \mathbb{R}^3$ world coordinate always maps to the same color, independent of the viewing angle. As a result, it is impossible for our model to generate these details. We note that detail that is not undersampled, such as the CVPR logo on the top of the cube, is reproduced with perfect accuracy. However, we can easily accommodate for this undersampling by using a 2D CNN renderer. This amounts to a trade-off of our guarantee of multi-view consistency discussed in Sec. 3 of the main chapter with robustness to faulty training data. Fig. B.2 shows the cube rendered with a U-Net based renderer – all detail is replicated truthfully.

B.3 Reproducibility

In this section, we discuss steps we take to allow the community to reproduce our results. All models were evaluated on the test sets exactly once.

B.3.1 Architecture Details

Scene representation network Φ In all experiments, Φ is parameterized as a multi-layer perceptron (MLP) with ReLU activations, layer normalization before each nonlinearity [141], and four layers with 256 units each. In all generalization experiments in the main chapter, its weights ϕ are the output of the hypernetwork Ψ. In the DeepVoxels comparison (see Sec.B.2), where a separate Φ is trained per scene, parameters of ϕ are directly initialized using the Kaiming Normal method [142].

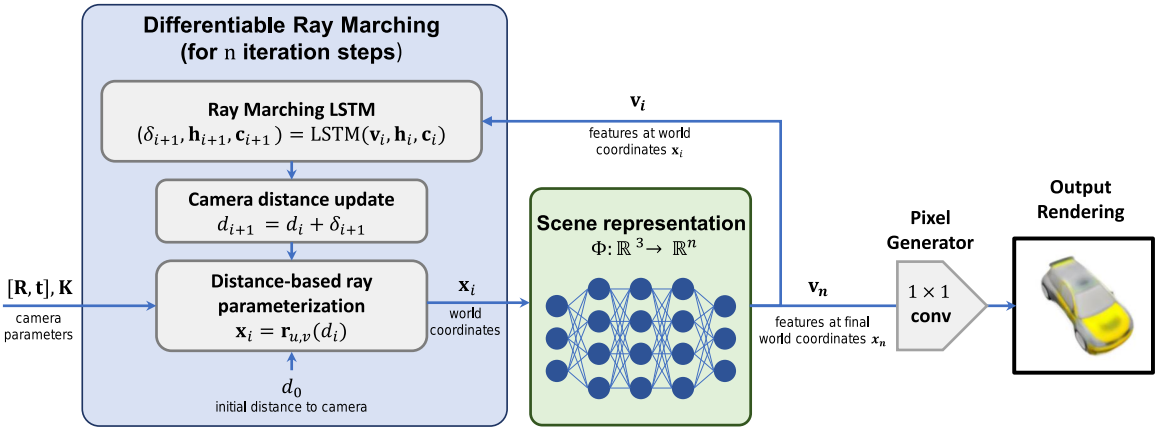

Figure B.5: Ray-marcher focused visualization of SRNs. At the heart of SRNs lies a continuous, 3D-aware neural scene representation, Φ, which represents a scene as a function that maps (x, y, z) world coordinates to a feature representation of the scene at those coordinates. To render Φ, a neural ray-marcher interacts with Φ via world coordinates along camera rays, parameterized via their distance d to the camera projective center. Ray Marching begins at a distance d_0 close to the camera. In each step, the scene representation network Φ is queried at the current world coordinates x_i. The resulting feature vector v_i is fed to the Ray Marching LSTM that predicts a step length δ_{i+1}. The world coordinates are updated according to the new distance to the camera, $d_{i+1} = d_i + \delta_{i+1}$. This is repeated for a fixed number of iterations, n. The features at the final world coordinates $v_n = \Phi(x_n)$ are then translated to an RGB color by the pixel generator.

Hypernetwork Ψ In generalization experiments, a hypernetwork Ψ maps a latent vector $\mathbf{z}_j$ to the weights of the respective scene representation ϕ_j. Each layer of Φ is the output of a separate hypernetwork. Each hypernetwork is parameterized as a multi-layer perceptron with ReLU activations, layer normalization before each nonlinearity [141], and three layers (where the last layer has as many units as the respective layer of Φ has weights). In the Shapenet and Shepard-Metzler experiments, where the latent codes $\mathbf{z}_j$ have length 256, hypernetworks have 256 units per layer. In the Basel face experiment, where the latent codes $\mathbf{z}_j$ have length 224, hypernetworks have 224 units per layer. Weights are initialized by the Kaiming Normal method, scaled by a factor 0.1. We empirically found this initialization to stabilize early training.

Ray marching LSTM In all experiments, the ray marching LSTM is implemented as a vanilla LSTM with a hidden state size of 16. The initial state is set to zero.

Pixel Generator In all experiments, the pixel generator is parameterized as a multi-layer perceptron with ReLU activations, layer normalization before each nonlinearity [141], and five layers with 256 units each. Weights are initialized with the Kaiming Normal method [142].

B.3.2 Time & Memory Complexity

Scene representation network Φ Φ scales as a standard MLP. Memory and runtime scale linearly in the number of queries, therefore quadratic in image resolution. Memory and runtime further scale linearly with the number of layers and quadratically with the number of units in each layer.

Hypernetwork Ψ Ψ scales as a standard MLP. Notably, the last layer of Ψ predicts all parameters of the scene representation Φ. As a result, the number of weights scales linearly in the number of weights of Φ, which is significant. For instance, with 256 units per layer and 4 layers, Φ has approximately 2×10^5 parameters. In our experiments, Ψ is parameterized with 256 units in all hidden layers. The last layer of Ψ then has approximately 5×10^7 parameters, which is the bulk of learnable parameters in our model. Please note that Ψ only has to be queried once to obtain Φ, at which point it could be discarded, as both the pixel generation and the ray marching only need access to the predicted Φ.

Differentiable Ray Marching Memory and runtime of the differentiable ray marcher scale linearly in the number of ray marching steps and quadratically in image resolution. As it queries Φ repeatedly, it also scales linearly in the same parameters as Φ.

Pixel Generator The pixel generator scales as a standard MLP. Memory and runtime scale linearly in the number of queries, therefore quadratic in image resolution. Memory and runtime further scale linearly with the number of layers and quadratically with the number of units in each layer.

B.3.3 Dataset Details

Shepard-Metzler objects We modified an open-source implementation of a Shepard-Metzler renderer `https://github.com/musyoku/gqn-dataset-renderer.git` to generate meshes of Shepard-Metzler objects, which we rendered using Blender to have full control over camera intrinsic and extrinsic parameters consistent with other presented datasets.

Shapenet v2 cars We render each object from random camera perspectives distributed on a sphere with radius 1.3 using Blender. We disabled specularities, shadows and transparencies and used environment lighting with energy 1.0. We noticed that a few cars in the dataset were not scaled optimally, and scaled their bounding box to unit length. A few meshes had faulty vertices, resulting in a faulty bounding box and subsequent scaling to a very small size. We discarded those 40 out of 2473 cars.

Shapenet v2 chairs We render each object from random camera perspectives distributed on a sphere with radius 2.0 using Blender. We disabled specularities, shadows and transparencies and used environment lighting with energy 1.0.

Faces dataset We use the Basel Face dataset to generate meshes with different identities at random, where each parameter is sampled from a normal distribution with mean 0 and standard deviation of 0.7. For expressions, we use the blendshape model of [143], and sample expression parameters uniformly in $(-0.4, 1.6)$.

DeepVoxels dataset We use the dataset as presented in [1].

B.3.4 SRNs Training Details

General details

Multi-Scale training Our per-pixel formulation naturally allows us to train in a coarse-to-fine setting, where we first train the model on downsampled images in a first stage, and then increase the resolution of images in stages. This allows larger batch sizes at the beginning of the training, which affords more independent views for each object, and is reminiscent of other coarse-to-fine approaches [73].

Solver For all experiments, we use the ADAM solver with $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Implementation & Compute We implement all models in PyTorch. All models were trained on single GPUs of the type RTX6000 or RTX8000.

Hyperparameter search Training hyperparameters for SRNs were found by informal search – we did not perform a systematic grid search due to the high computational cost.

Per-experiment details

For a resolution of 64×64, we train with a batch size of 72. Due to the memory complexity being quadratic in the image sidelength, we decrease the batch size by a factor of 4 when we double the image resolution. λ_{depth} is always set to 1×10^{-3} and λ_{latent} is set to 1. The ADAM learning rate is set to 4×10^{-4} if not reported otherwise.

Shepard-Metzler experiment We directly train our model on images of resolution 64×64 for 352 epochs.

Shapenet cars We train our model in 2 stages. We first train on a resolution of 64×64 for 5k iterations. We then increase the resolution to 128×128. We train on the high resolution for 70 epochs. The ADAM learning rate is set to 5×10^{-5}.

Shapenet chairs We train our model in 2 stages. We first train on a resolution of 64×64 for 20k iterations. We then increase the resolution to 128×128. We train our model for 12 epochs.

Basel face experiments We train our model in 2 stages. We first train on a resolution of 64×64 for 15k iterations. We then increase the resolution to 128×128 and train for another 5k iterations.

DeepVoxels experiments We train our model in 3 stages. We first train on a resolution of 12×128 with a learning rate of 4×10^{-4} for 20k iterations. We then increase the resolution to 256×256, and lower the learning rate to 1×10^{-4} and train for another 30k iterations. We then increase the resolution to 512×512, and lower the learning rate to 4×10^{-6} and train for another 30k iterations.

B.4 Relationship to per-pixel autoregressive methods

With the proposed per-pixel generator, SRNs are also reminiscent of autoregressive per-pixel architectures, such as PixelCNN and PixelRNN [70, 111]. The key difference to autoregressive per-pixel architectures lies in the modeling of the probability $p(\mathcal{I})$ of an image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$. PixelCNN and PixelRNN model an image as a one-dimensional sequence of pixel values $\mathcal{I}_1, ..., \mathcal{I}_{H \times W}$, and estimate their joint distribution as

$$p(\mathcal{I}) = \prod_{i=1}^{H \times W} p(\mathcal{I}_i | \mathcal{I}_1, ..., \mathcal{I}_{i-1}). \tag{B.1}$$

Instead, conditioned on a scene representation Φ, pixel values are conditionally independent, as our approach independentaly and deterministically assigns a value to each pixel. The probability of observing an image $\mathcal{I}$ thus simplifies to the probability of observing a scene Φ under extrinsic $\mathbf{E}$ and intrinsic $\mathbf{K}$ camera parameters

$$p(\mathcal{I}) = p(\Phi)p(\mathbf{E})p(\mathbf{K}). \tag{B.2}$$

This conditional independence of single pixels conditioned on the scene representation further motivates the per-pixel design of the rendering function Θ.

B.5 Baseline Discussions

B.5.1 Deterministic Variant of GQN

Deterministic vs. Non-Deterministic [11] propose a powerful probabilistic framework for modeling uncertainty in the reconstruction due to incomplete observations. However, here, we are exclusively interested in investigating the properties of the scene representation itself, and this submission discusses SRNs in a purely deterministic framework. To enable a fair comparison, we thus implement a deterministic baseline inspired by the Generative Query Network [11]. We note that the results obtained in this comparison are not necessarily representative of the performance of the unaltered Generative Query Network. We leave a formulation of SRNs in a probabilistic framework and a comparison to the unaltered GQN to future work.

Architecture As representation network architecture, we choose the "Tower" representation, and leave its architecture unaltered. However, instead of feeding the resulting scene representation $\mathbf{r}$ to a convolutional LSTM architecture to parameterize a density over latent variables $\mathbf{z}$, we instead directly feed the scene representation $\mathbf{r}$ to a generator network. We use as generator a deterministic, autoregressive, skip-convolutional LSTM C, the deterministic equivalent of the generator architecture proposed in [11]. Specifically, the generator can be described by the following equations:

$$\text{Initial state} \qquad (\mathbf{c}_0, \mathbf{h}_0, \mathbf{u}_0) = (\mathbf{0}, \mathbf{0}, \mathbf{0}) \tag{B.3}$$

$$\text{Pre-process current canvas} \qquad \mathbf{p}_l = \kappa(\mathbf{u}_l) \tag{B.4}$$

$$\text{State update} \qquad (\mathbf{c}_{l+1}, \mathbf{h}_{l+1}) = C(\mathbf{E}, \mathbf{r}, \mathbf{c}_l, \mathbf{h}_l, \mathbf{p}_l) \tag{B.5}$$

$$\text{Canvas update} \qquad \mathbf{u}_{l+1} = \mathbf{u}_l + \Delta(\mathbf{h}_{l+1}) \tag{B.6}$$

$$\text{Final output} \qquad \mathbf{x} = \eta(\mathbf{u}_L), \tag{B.7}$$

with timestep l and final timestep L, LSTM output $\mathbf{c}_l$ and cell $\mathbf{h}_l$ states, the canvas $\mathbf{u}_l$, a downsampling network κ, the camera extrinsic parameters $\mathbf{E}$, an upsampling network Δ, and a 1×1 convolutional layer η. Consistent with [11], all up- and downsampling layers are convolutions of size 4×4 with stride 4. To account for the higher resolution of the Shapenet v2 car and chair images, we added a further convolutional layer / transposed convolution where necessary.

Training On both the cars and chairs datasets, we trained for $180,000$ iterations with a batch size of 140, taking approximately 6.5 days. For the lower-resolution Shepard-Metzler objects, we trained for $160,000$ iterations at a batch size of 192, or approximately 5 days.

Testing For novel view synthesis on the training set, the model receives as input the 15 nearest neighbors of the novel view in terms of cosine similarity. For two-shot reconstruction, the model

receives as input whichever of the two reference views is closer to the novel view in terms of cosine similarity. For one-shot reconstruction, the model receives as input the single reference view.

B.5.2 Tatarchenko et al.

Architecture We implement the exact same architecture as described in [4], with approximately $70 \cdot 10^6$ parameters.

Training For training, we choose the same hyperparameters as proposed in [4]. As we assume no knowledge of scene geometry, we do not supervise the model with a depth map. As we observed the model to overfit, we stopped training early based on model performance on the held-out, official Shapenet v2 validation set.

Testing For novel view synthesis on the training set, the model receives as input the nearest neighbor of the novel view in terms of cosine similarity. For two-shot reconstruction, the model receives as input whichever of the two reference views is closer to the novel view. Finally, for one-shot reconstruction, the model receives as input the single reference view.

B.5.3 Worrall et al.

Architecture Please see Fig. B.6 for a visualization of the full architecture. The design choices in this architecture (nearest-neighbor upsampling, leaky ReLU activations, batch normalization) were made in accordance with [5].

Training For training, we choose the same hyperparameters as proposed in [5].

Testing For novel view synthesis on the training set, the model receives as input the nearest neighbor of the novel view in terms of cosine similarity. For two-shot reconstruction, the model receives as input whichever of the two reference views is closer to the novel view. Finally, for one-shot reconstruction, the model receives as input the single reference view.

B.6 Differentiable Ray-Marching in the context of classical renderers

The proposed neural ray-marcher is inspired by the classic sphere tracing algorithm [110]. Sphere tracing was originally developed to render scenes represented via analytical signed distance functions. It is defined by a special choice of the step length: each step has a length equal to the signed distance to the closest surface point of the scene. Since this distance is only zero on the surface of the scene,

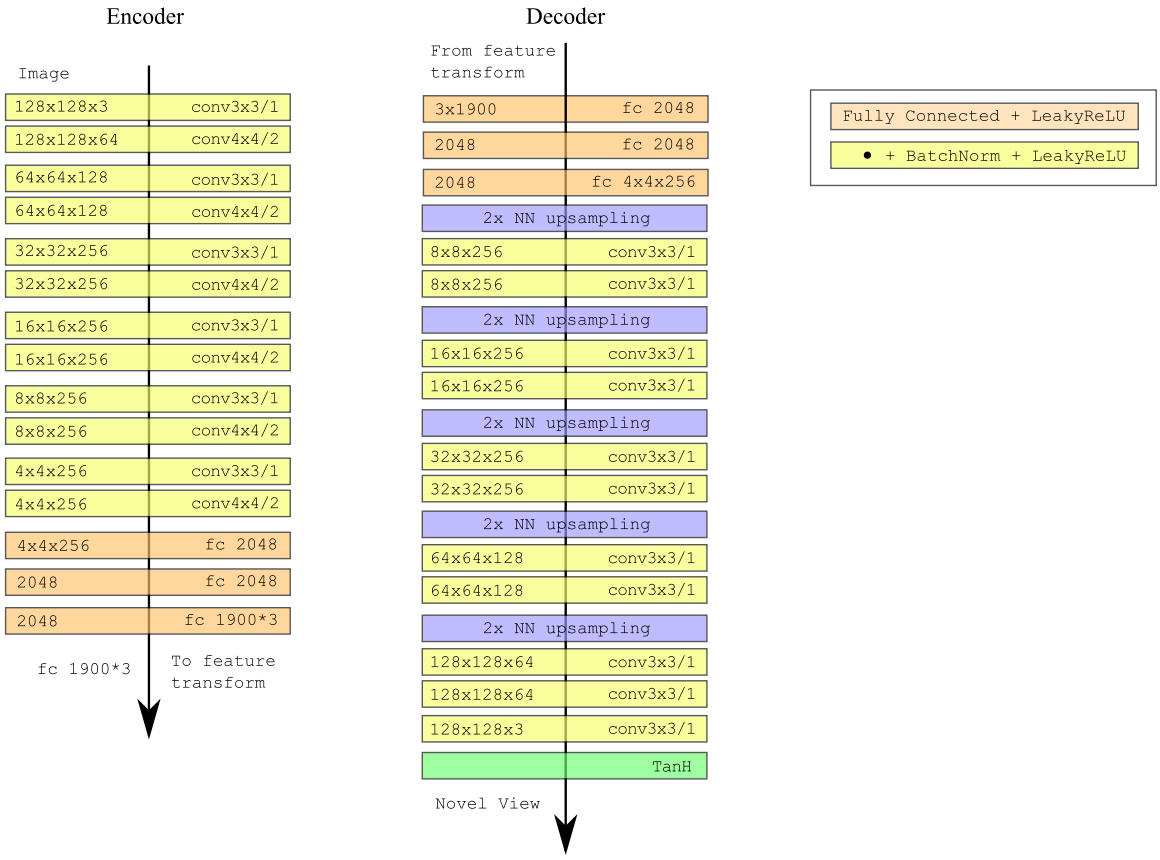

Figure B.6: Architecture of the baseline method proposed in [5].

the algorithm takes non-zero steps until it has arrived at the surface, at which point no further steps are taken. A major downside of sphere-tracing is its weak convergence guarantee: Sphere tracing is only guaranteed to converge for an infinite number of steps. This is easy to see: For any fixed number of steps, we can construct a scene where a ray is parallel to a close surface (or falls through a slim tunnel) and eventually intersects a scene surface. For any constant number of steps, there exists a surface parallel to the ray that is so close that the ray will not reach the target surface. In classical sphere-tracing, this is circumvented by taking a large number of steps that generally take the intersection estimate within a small neighborhood of the scene surface – the color at this point is then simply defined as the color of the closest surface. However, this heuristic can still fail in constructed examples such as the one above. Extensions of sphere tracing propose heuristics to modifying the step length to speed up convergence [111]. The Ray-Marching LSTM instead has the ability to learn the step length. The key driver of computational and memory cost of the proposed rendering algorithm is the ray-marching itself: In every step of the ray-marcher, for every pixel, the scene representation ϕ is evaluated. Each evaluation of ϕ is a full forward pass through a multi-layer perceptron. See B.3.2 for an exact analysis of memory and computational complexity of the different components.

Other classical rendering algorithms usually follow a different approach. In modern computer graphics, scenes are often represented via explicit, discretized surface primitives - such as is the case in meshes. This allows rendering via rasterization, where scene geometry is projected onto the image plane of a virtual camera in a single step. As a result, rasterization is computationally cheap, and has allowed for real-time rendering that has approached photo-realism in computer graphics.

However, the image formation model of rasterization is not appropriate to simulate physically accurate image formations that involve proper light transport, view-dependent effects, participating media, refraction, translucency etc. As a result, physics-based rendering usually uses ray-tracing algorithms, where for each pixel, a number of rays are traced from the camera via all possible paths to light sources through the scene. If the underlying scene representations are explicit, discrete representations – such as meshes – the intersection testing required is again cheap. Main drivers of computational complexity in such systems are then the number of rays that need to be traced to appropriately sample all paths to lights sources that contribute to the value of a single pixel.

In this context, the proposed ray-marcher can be thought of as a sphere-tracing-inspired ray-tracer for implicitly defined scene geometry. It does not currently model multi-bounce ray-tracing, but could potentially be extended in the future (see B.8).

B.7 Trade-offs of the Pixel Generator vs. CNN-based renderers

As described in the main chapter, the pixel generator comes with a guarantee of multi-view consistency compared to a 2D-CNN based rendering network. On the flip side, we cannot make use of progress in the design of novel CNN architectures that save memory by introducing resolution bottlenecks and skip connections, such as the U-Net [93]. This means that the pixel generator is comparably memory-hungry, as each layer operates on the full resolution of the image to be generated. Furthermore, CNNs have empirically been demonstrated to be able to generate high-frequency image detail easily. It is unclear what the limitations of the proposed pipeline are with respect to generating high-frequency textural detail. We note that the pixel generator is not a necessary component of SRNs, and can be replaced by a classic 2D-CNN based renderer, as we demonstrate in B.2.

B.8 Future work

Applications outside of vision. SRNs have promising applications outside of vision. Neural scene representations are a core aspect of artificial intelligence, as they allow an agent to model its environment, navigate, and plan interactions. Thus, natural applications of SRNs lie in robotic manipulation or as the world model of an independent agent.

Extending SRNs to other image formation models. SRNs could be extended to other image formation models, such as computer tomography or magnetic resonance imaging. All that is required is a differentiable forward model of the image formation. The ray-marcher could be adapted accordingly to integrate features along a ray or to sample at pre-defined locations. For image formation models that observe scenes directly in 3D, the ray-marcher may be left out completely, and ϕ may be sampled directly.

Probabilistic formulation. An interesting avenue of future work is to extend SRNs to a probabilistic model that can infer a probability distribution over feasible scenes consistent with a given set of observations. In the following, we formulate one such approach, very similar to the formulation of Kumar et al. [61], which is in turn based on the work of Eslami et al. [11]. Please note that this formulation is not experimentally verified in the context of SRNs and is described here purely to facilitate further research in this direction.

Formally, the model can be summarized as:

$$r_i = M(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i) \tag{B.8}$$

$$r = \sum_i r_i \tag{B.9}$$

$$z \sim P_\Theta(z|r) \tag{B.10}$$

$$\phi = \Psi(z) \tag{B.11}$$

$$\mathcal{I} = \Theta(\Phi_\phi, \mathbf{E}, \mathbf{K}) \tag{B.12}$$

We assume that we are given a set of instance datasets $\mathcal{D} = \{\mathcal{C}_j\}_{j=1}^M$, where each $\mathcal{C}_j$ consists of tuples $\{(\mathcal{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^N$. For a single scene $\mathcal{C}$ with n observations, we first replicate and concatenate the camera pose $\mathbf{E}_i$ and intrinsic parameters $\mathbf{K}_i$ of each observations to the image channels of the corresponding 2D image $\mathcal{I}_i$. Using a learned convolutional encoder M, we encode each of the n observations to a code vector r_i. These code vectors r_i are then summed to form a permutation-invariant representation of the scene r. Via an autoregressive DRAW model [144], we form a probability distribution P_θ that is conditioned on the code vector r and sample latent variables z. z is decoded into the parameters of a scene representation network, ϕ, via a hypernetwork $\Psi(z) = \phi$. Lastly, via our differentiable rendering function Θ, we can render images $\mathcal{I}$ from Φ_ϕ as described in the main chapter. This allows to train the full model end-to-end given only 2D images and their camera parameters. We note that the resulting optimization problem is intractable and requires the optimization of an evidence lower bound via an approximate posterior, which we do not derive here – please refer to [61]. Similarly to [61], this formulation will lead to multi-view consistent renderings of each scene, as the scene representation Φ stays constant across queries of Θ.

View- and lighting-dependent effects, translucency, and participating media. Another exciting direction for future work is to model further aspects of realistic scenes. One such aspect is view- and lighting dependent effects, such as specularities. For fixed lighting, the pixel generator could receive as input the direction of the camera ray in world coordinates, and could thus reason about the view-dependent color of a surface. To model simple lighting-dependent effects, the pixel generator could further receive the light ray direction as an input (assuming no occlusions). Lastly, the proposed formulation could also be extended to model multiple ray bounces in a ray-casting framework. To model translucency and participating media, the ray-marcher could be extended to sum features along a ray instead of only sampling a feature at the final intersection estimate.

Bibliography

[1] ©IEEE. Reprinted, with permission, from Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. CVPR*, 2019.

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

[3] Amit Kohli, Vincent Sitzmann, and Gordon Wetzstein. Inferring semantic information with 3d neural scene representations. *arXiv preprint arXiv:2003.12673*, 2020.

[4] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR abs/1511.06702*, 1(2):2, 2015.

[5] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Interpretable transformations with encoder-decoder networks. In *Proc. ICCV*, volume 4, 2017.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, pages 5967–5976, 2017.

[7] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, June 2018.

[8] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. In *Nature*, 2015.

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

[12] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[13] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proc. ICCV Workshops*, pages 298–372, 2000.

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[15] Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research , 1980.

[16] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

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

[18] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. ECCV*, 2016.

[19] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Trans. on Graphics (SIGGRAPH)*, volume 25, pages 835–846, 2006.

[20] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building rome in a day. In *Proc. CVPR*, pages 72–79, 2009.

[21] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. PAMI*, 32(8):1362–1376, 2010.

[22] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. on Graphics*, 32(6):169, 2013.

[23] David Gallup, Marc Pollefeys, and Jan-Michael Frahm. 3d reconstruction using an n-layer heightmap. In *Joint Pattern Recognition Symposium*, pages 1–10. Springer, 2010.

[24] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[25] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *Proc. NeurIPS*, 2019.

[26] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *Proc. ICLR*, 2017.

[27] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Proc. NIPS*, pages 3581–3589, 2014.

[28] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[30] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Proc. NIPS*, pages 766–774, 2014.

[31] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[32] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proc. ECCV*, pages 649–666. Springer, 2016.

[33] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proc. CVPR*, pages 6874–6883, 2017.

[34] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *Proc. ECCV*, pages 535–553. Springer, 2016.

[35] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proc. ICCV*, pages 37–45, 2015.

[36] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, pages 1413–1421, 2015.

[37] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, pages 1422–1430, 2015.

[38] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Proc. NIPS*, pages 365–376, 2017.

[39] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. CVPR*, pages 2626–2634, 2017.

[40] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proc. NIPS*, pages 82–90, 2016.

[41] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *3DV*, pages 402–411. IEEE Computer Society, 2017.

[42] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. CVPR*, 2016.

[43] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proc. CVPR*, 2018.

[44] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Proc. NIPS*. 2016.

[45] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. ECCV*, 2016.

[46] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. CVPR*, 2017.

[47] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. ICCV*, pages 2107–2115, 2017.

[48] C. Haene, S. Tulsiani, and J. Malik. Hierarchical surface prediction. *Proc. PAMI*, pages 1–1, 2019.

[49] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. CVPR*, 2017.

[50] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *Proc. ICML*, pages 40–49, 2018.

[51] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *Proc. ECCV*, 2016.

[52] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. on Graphics*, 37(4):65:1–65:12, 2018.

[53] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. In *Proc. CVPR*, 2018.

[54] Dominic Jack, Jhony K. Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frédéric Maire, and Anders Eriksson. Learning free-form deformations for 3d object reconstruction. *CoRR*, 2018.

[55] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. CVPR*, pages 3907–3916, 2018.

[56] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, pages 371–386, 2018.

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

[58] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, pages 165–174, 2019.

[59] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. *Proc. ICCV*, 2019.

[60] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *Proc. NeurIPS*, 2019.

[61] Ananya Kumar, SM Ali Eslami, Danilo Rezende, Marta Garnelo, Fabio Viola, Edward Lockhart, and Murray Shanahan. Consistent jumpy predictions for videos and scenes. *arXiv preprint arXiv:1807.02033*, 2018.

[62] Ferran Alet, Adarsh K Jeewajee, Maria Bauza, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Graph element networks: adaptive, structured computation and memory. In *Proc. ICML*, 2019.

[63] Yunchao Liu, Zheng Wu, Daniel Ritchie, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to describe scenes with programs. In *Proc. ICLR*, 2019.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[65] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[66] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2013.

[67] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *Proc. ICLR Workshops*, 2015.

[68] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. NeurIPS*, pages 10236–10245, 2018.

[69] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Proc. NIPS*, pages 4797–4805, 2016.

[70] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proc. ICML*, 2016.

[71] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014.

[72] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. ICML*, 2017.

[73] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *Proc. ICLR*, 2018.

[74] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proc. ECCV*, 2016.

[75] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. ICLR*, 2016.

[76] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014.

[77] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, 2017.

[78] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[79] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018.

[80] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Proc. NIPS*. 2016.

[81] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Proc. NIPS*. 2015.

[82] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *Proc. ICANN*, 2011.

[83] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *Proc. AAAI*, 2018.

[84] A. Yuille and D. Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10:301–308, 2006.

[85] Thomas G. Bever and David Poeppel. Analysis by synthesis: A (re-)emerging program of research for language and vision. *Biolinguistics*, 4(2):174–200, 2010.

[86] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Proc. NIPS*. 2015.

[87] Jimei Yang, Scott Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Proc. NIPS*, 2015.

[88] Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash K. Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proc. CVPR*, 2015.

[89] Hsiao-Yu Fish Tung, Adam W. Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *Proc. ICCV*.

[90] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *Proc. CVPR*, 2017.

[91] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. ICLR*, 2018.

[92] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Proc. NIPS*, pages 2017–2025. 2015.

[93] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[94] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. IROS*, page 922 928, September 2015.

[95] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Proc. NIPS*, pages 1696–1704, 2016.

[96] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE Trans. PAMI*, 39(4):692–705, 2017.

[97] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. *Proc. ECCV*, 2018.

[98] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[99] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proc. NIPS*, pages 9605–9616, 2018.

[100] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proc. CVPR*, 2018.

[101] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2014.

[102] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004.

[103] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Proc. NIPS Workshops*, 2017.

[104] Taco S Cohen and Max Welling. Transformation properties of learned visual representations. *Proc. ICLR*, 2015.

[105] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. *Proc. CVPR*, 2019.

[106] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Proc. NIPS*. 2018.

[107] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: image generation with disentangled 3d representations. In *Proc. NIPS*, pages 118–129, 2018.

[108] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Proc. NIPS*, pages 2802–2812, 2018.

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

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

[111] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Proc. NIPS*, 2016.

[112] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[113] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *Proc. ICLR*, 2017.

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

[115] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. Ieee, 2009.

[116] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. In *Proc. ICLR*, 2019.

[117] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *Proc. ICLR*, 2019.

[118] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[119] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2020.

[120] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. *arXiv preprint arXiv:2003.08981*, 2020.

[121] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Proc. NIPS*, pages 3856–3866, 2017.

[122] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *Proc. NeurIPS*, pages 15486–15496, 2019.

[123] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[124] O. Chapelle, B. Scholkopf, and A. Zien, Eds. Semi-supervised learning. *IEEE Trans. on Neural Networks*, 20(3):542–542, 2009.

[125] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Proc. NeurIPS*, pages 15509–15519, 2019.

[126] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, pages 3431–3440, 2015.

[127] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *Trans. PAMI*, 40(4):834–848, 2017.

[128] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proc. ICCV*, pages 1529–1537, 2015.

[129] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[130] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proc. ICCV*, pages 2650–2658, 2015.

[131] Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *Proc. IROS*, pages 598–605. IEEE, 2017.

[132] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Proc. ACCV*, pages 213–228. Springer, 2016.

[133] Jinghua Wang, Zhenhua Wang, Dacheng Tao, Simon See, and Gang Wang. Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks. In *Proc. ECCV*, pages 664–679. Springer, 2016.

[134] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proc. ECCV*, pages 452–468, 2018.

[135] Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi, and Philip Torr. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Trans. on Graphics (TOG)*, 34(5):154, 2015.

[136] Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor A Prisacariu, Olaf Kähler, David W Murray, Shahram Izadi, Patrick Pérez, et al. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Proc. ICRA*, pages 75–82. IEEE, 2015.

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

[138] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proc. CVPR*, pages 909–918, 2019.

[139] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. *arXiv preprint arXiv:2002.12880*, 2020.

[140] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[141] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[142] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034, 2015.

[143] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. CVPR*, pages 2387–2395, 2016.

[144] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *Proc. ICML*, 2015.