# pixelNeRF: Neural Radiance Fields from One or Few Images

Alex Yu Vickie Ye Matthew Tancik Angjoo Kanazawa
UC Berkeley

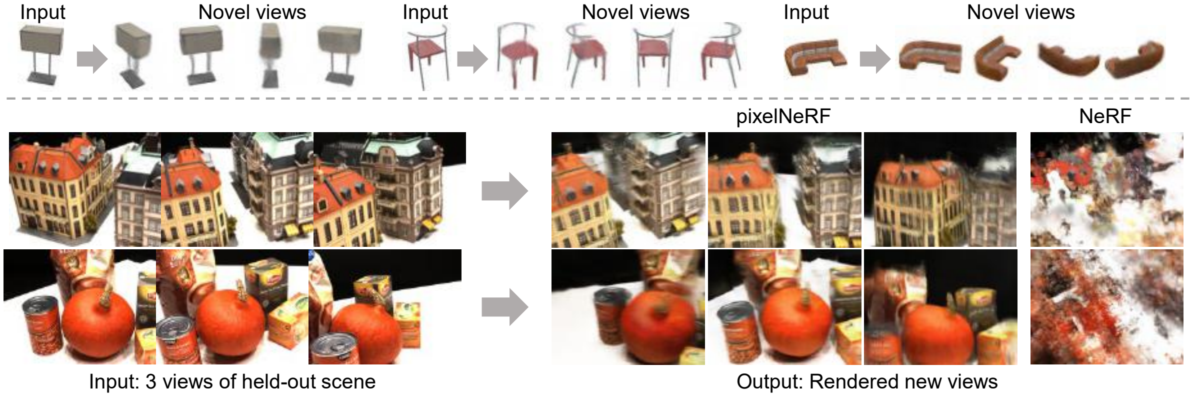


Figure 1: **NeRF from one or few images.** We present pixelNeRF, a learning framework that predicts a Neural Radiance Field (NeRF) representation from a single (top) or few posed images (bottom). PixelNeRF can be trained on a set of multi-view images, allowing it to generate plausible novel view synthesis from very few input images without test-time optimization (bottom left). In contrast, NeRF has no generalization capabilities and performs poorly when only three input views are available (bottom right).

## Abstract

*We propose pixelNeRF, a learning framework that predicts a continuous neural scene representation conditioned on one or few input images. The existing approach for constructing neural radiance fields [27] involves optimizing the representation to every scene independently, requiring many calibrated views and significant compute time. We take a step towards resolving these shortcomings by introducing an architecture that conditions a NeRF on image inputs in a fully convolutional manner. This allows the network to be trained across multiple scenes to learn a scene prior, enabling it to perform novel view synthesis in a feed-forward manner from a sparse set of views (as few as one). Leveraging the volume rendering approach of NeRF, our model can be trained directly from images with no explicit 3D supervision. We conduct extensive experiments on ShapeNet benchmarks for single image novel view synthesis tasks with held-out objects as well as entire unseen categories. We further demonstrate the flexibility of pixelNeRF by demonstrating it on multi-object ShapeNet scenes and real scenes from the DTU dataset. In all cases, pixelNeRF outperforms current state-of-the-art baselines for novel view synthesis and single image 3D reconstruction. For the video and code, please visit the project website:* `https://alexyu.net/pixelnerf`.

## 1. Introduction

We study the problem of synthesizing novel views of a scene from a sparse set of input views. This long-standing problem has recently seen progress due to advances in differentiable neural rendering [27, 20, 24, 40]. Across these approaches, a 3D scene is represented with a neural network, which can then be rendered into 2D views. Notably, the recent method neural radiance fields (NeRF) [27] has shown impressive performance on novel view synthesis of a specific scene by implicitly encoding volumetric density and color through a neural network. While NeRF can render photorealistic novel views, it is often impractical as it requires a large number of posed images and a lengthy per-scene optimization.

In this paper, we address these shortcomings by proposing pixelNeRF, a learning framework that enables predicting NeRFs from one or several images in a feed-forward manner. Unlike the original NeRF network, which does not make use of any image features, pixelNeRF takes spatial image features aligned to each pixel as an input. This image conditioning allows the framework to be trained on a set of multi-view images, where it can learn scene priors to perform view synthesis from one or few input views. In contrast, NeRF is unable to generalize and performs poorly when few input images are available, as shown in Fig. 1.

Specifically, we condition NeRF on input images by first computing a fully convolutional image feature grid from the input image. Then for each query spatial point $\mathbf{x}$ and viewing direction $\mathbf{d}$ of interest in the view coordinate frame, we sample the corresponding image feature via projection and bilinear interpolation. The query specification is sent along with the image features to the *NeRF network* that outputs density and color, where the spatial image features are fed to each layer as a residual. When more than one image is available, the inputs are first encoded into a latent representation in each camera's coordinate frame, which are then pooled in an intermediate layer prior to predicting the color and density. The model is supervised with a reconstruction loss between a ground truth image and a view rendered using conventional volume rendering techniques. This framework is illustrated in Fig. 2.

PixelNeRF has many desirable properties for few-view novel-view synthesis. First, pixelNeRF can be trained on a dataset of multi-view images without additional supervision such as ground truth 3D shape or object masks. Second, pixelNeRF predicts a NeRF representation in the camera coordinate system of the input image instead of a canonical coordinate frame. This is not only integral for generalization to unseen scenes and object categories [42, 38], but also for flexibility, since no clear canonical coordinate system exists on scenes with multiple objects or real scenes. Third, it is fully convolutional, allowing it to preserve the spatial alignment between the image and the output 3D representation. Lastly, pixelNeRF can incorporate a variable number of posed input views at test time without requiring any test-time optimization.

We conduct an extensive series of experiments on synthetic and real image datasets to evaluate the efficacy of our framework, going beyond the usual set of ShapeNet experiments to demonstrate its flexibility. Our experiments show that pixelNeRF can generate novel views from a single image input for both category-specific and category-agnostic settings, even in the case of unseen object categories. Further, we test the flexibility of our framework, both with a new multi-object benchmark for ShapeNet, where pixelNeRF outperforms prior approaches, and with simulation-to-real transfer demonstration on real car images. Lastly, we test capabilities of pixelNeRF on real images using the DTU dataset [14], where despite being trained on under 100 scenes, it can generate plausible novel views of a real scene from three posed input views.

## 2. Related Work

**Novel View Synthesis.** The long-standing problem of novel view synthesis entails constructing new views of a scene from a set of input views. Early work achieved photorealistic results but required densely captured views of the scene [19, 11]. Recent work has made rapid progress toward photorealism for both wider ranges of novel views and sparser sets of input views, by using 3D representations based on neural networks [27, 23, 26, 39, 43, 7]. However, because these approaches fit a single model to each scene, they require many input views and substantial optimization time per scene.

| | NeRF | DISN | ONet | DVR | SRN | Ours |
|---|---|---|---|---|---|---|
| Learns scene prior? | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Supervision | 2D | 3D | 3D | 2D | 2D | 2D |
| Image features | ✗ | Local | Global | Global | ✗ | Local |
| Allows multi-view? | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |
| View space? | - | ✗ | ✗ | ✗ | ✗ | ✓ |

Table 1: **A comparison with prior works reconstructing neural scene representations.** The proposed approach learns a scene prior for one or few-view reconstruction using only multi-view 2D image supervision. Unlike previous methods in this regime, we do not require a consistent canonical space across the training corpus. Moreover, we incorporate local image features to preserve local information which is in contrast to methods that compress the structure and appearance into a single latent vector such as Occupancy Networks (ONet) [25] and DVR [28].

There are methods that can predict novel view from few input views or even single images by learning shared priors across scenes. Methods in the tradition of [36, 3] use depth-guided image interpolation [56, 10, 33]. More recently, the problem of predicting novel views from a single image has been explored [45, 49, 37, 5]. However, these methods employ 2.5D representations, and are therefore limited in the range of camera motions they can synthesize. In this work we infer a 3D volumetric NeRF representation, which allows novel view synthesis from larger baselines.

Sitzmann et al. [40] introduces a representation based on a continuous 3D feature space to learn a prior across scene instances. However, using the learned prior at test time requires further optimization with known absolute camera poses. In contrast, our approach is completely feed-forward and only requires relative camera poses. We offer extensive comparisons with this approach to demonstrate the advantages our design affords. Lastly, note that concurrent work [44] adds image features to NeRF. A key difference is that we operate in view rather than canonical space, which makes our approach applicable in more general settings. Moreover, we extensively demonstrate our method's performance in few-shot view synthesis, while GRF shows very limited quantitative results for this task.

**Learning-based 3D reconstruction.** Advances in deep learning have led to rapid progress in single-view or multi-view 3D reconstruction. Many approaches [15, 12, 48, 55, 39, 34, 51, 25, 32] propose learning frameworks with various 3D representations that require ground-truth 3D models for supervision. Multi-view supervision [52, 46, 21, 22, 40, 28, 8, 2] is less restrictive and more ecologically plausible. However, many of these methods [52, 46, 21, 22, 28] require object masks; in contrast, pixelNeRF can be trained from images alone, allowing it to be applied to scenes of two objects without modification.

Most single-view 3D reconstruction methods condition neural 3D representations on input images. The majority employs global image features [29, 6, 28, 25, 8], which, while memory efficient, cannot preserve details that are present in the image and often lead to retrieval-like results. Spatially-aligned *local* image features have been shown to achieve detailed reconstructions from a single view [51, 34]. However, both of these methods require 3D supervision. Our method is inspired by these approaches, but only requires multi-view supervision.

Within existing methods, the types of scenes that can be reconstructed are limited, particularly so for object-centric approaches (e.g. [48, 21, 12, 46, 39, 55, 25, 51, 28]). CoReNet [32] reconstructs scenes with multiple objects via a voxel grid with offsets, but it requires 3D supervision including the identity and placement of objects. In comparison, we formulate a scene-level learning framework that can in principle be trained to scenes of arbitrary structure.

**Viewer-centric 3D reconstruction** For the 3D learning task, prediction can be done either in a viewer-centered coordinate system, i.e. *view space*, or in an object-centered coordinate system, i.e. *canonical space*. Most existing methods [51, 25, 28, 40] predict in canonical space, where all objects of a semantic category are aligned to a consistent orientation. While this makes learning spatial regularities easier, using a canonical space inhibits prediction performance on unseen object categories and scenes with more than one object, where there is no pre-defined or well-defined canonical pose. PixelNeRF operates in view-space, which has been shown to allow better reconstruction of unseen object categories in [38, 2], and discourages the memorization of the training set [42]. We summarize key aspects of our approach relative to prior work in Table 1.

## 3. Background: NeRF

We first briefly review the NeRF representation [27]. A NeRF encodes a scene as a continuous volumetric radiance field $f$ of color and density. Specifically, for a 3D point $\mathbf{x} \in \mathbb{R}^3$ and viewing direction unit vector $\mathbf{d} \in \mathbb{R}^3$, $f$ returns a differential density $\sigma$ and RGB color $\mathbf{c}$: $f(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$.

The volumetric radiance field can then be rendered into a 2D image via

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt \tag{1}$$

where $T(t) = \exp\big(-\int_{t_n}^{t} \sigma(s)\, ds\big)$ handles occlusion. For a target view with pose $\mathbf{P}$, a camera ray can be parameterized as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, with the ray origin (camera center) $\mathbf{o} \in \mathbb{R}^3$ and ray unit direction vector $\mathbf{d} \in \mathbb{R}^3$. The integral is computed along $\mathbf{r}$ between pre-defined depth bounds $[t_n, t_f]$. In practice, this integral is approximated with numerical quadrature by sampling points along each pixel ray.

The rendered pixel value for camera ray $\mathbf{r}$ can then be compared against the corresponding ground truth pixel value, $\mathbf{C}(\mathbf{r})$, for all the camera rays of the target view with pose $\mathbf{P}$. The NeRF rendering loss is thus given by

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{P})} \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2 \tag{2}$$

where $\mathcal{R}(\mathbf{P})$ is the set of all camera rays of target pose $\mathbf{P}$.

**Limitations** While NeRF achieves state of the art novel view synthesis results, it is an optimization-based approach using geometric consistency as the sole signal, similar to classical multiview stereo methods [1, 35]. As such each scene must be optimized individually, with no knowledge shared between scenes. Not only is this time-consuming, but in the limit of single or extremely sparse views, it is unable to make use of any prior knowledge of the world to accelerate reconstruction or for shape completion.

## 4. Image-conditioned NeRF

To overcome the NeRF representation's inability to share knowledge between scenes, we propose an architecture to condition a NeRF on spatial image features. Our model is comprised of two components: a fully-convolutional image encoder $E$, which encodes the input image into a pixel-aligned feature grid, and a NeRF network $f$ which outputs color and density, given a spatial location and its corresponding encoded feature. We choose to model the spatial query in the input view's camera space, rather than a canonical space, for the reasons discussed in § 2. We validate this design choice in our experiments on unseen object categories (§ 5.2) and complex unseen scenes (§ 5.3). The model is trained with the volume rendering method and loss described in § 3.

In the following, we first present our model for the single view case. We then show how this formulation can be easily extended to incorporate multiple input images.

### 4.1. Single-Image pixelNeRF

We now describe our approach to render novel views from one input image. We fix our coordinate system as the *view space* of the input image and specify positions and camera rays in this coordinate system.

Given a input image $\mathbf{I}$ of a scene, we first extract a feature volume $\mathbf{W} = E(\mathbf{I})$. Then, for a point on a camera ray $\mathbf{x}$, we retrieve the corresponding image feature by projecting $\mathbf{x}$ onto the image plane to the image coordinates $\pi(\mathbf{x})$ using

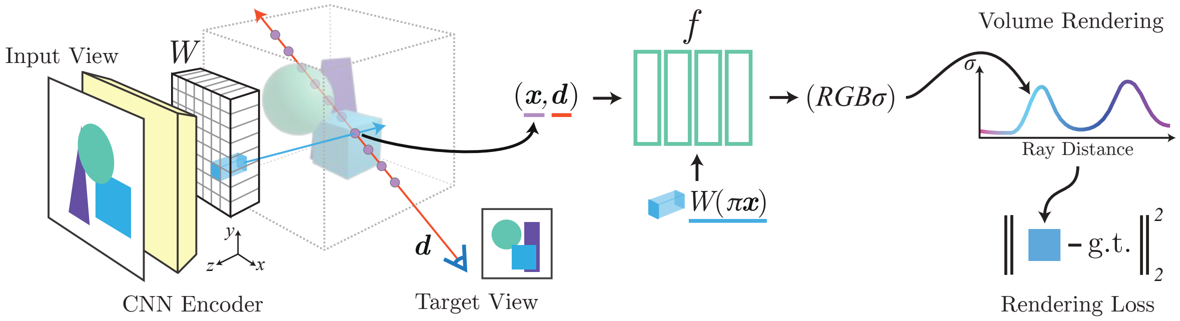


Figure 2: **Proposed architecture in the single-view case.** For a query point $\mathbf{x}$ along a target camera ray with view direction $\mathbf{d}$, a corresponding image feature is extracted from the feature volume $\mathbf{W}$ via projection and interpolation. This feature is then passed into the NeRF network $f$ along with the spatial coordinates. The output RGB and density value is volume-rendered and compared with the target pixel value. The coordinates $\mathbf{x}$ and $\mathbf{d}$ are in the camera coordinate system of the input view.

known intrinsics, then bilinearly interpolating between the pixelwise features to extract the feature vector $\mathbf{W}(\pi(\mathbf{x}))$. The image features are then passed into the NeRF network, along with the position and view direction (both in the input view coordinate system), as

$$f(\gamma(\mathbf{x}), \mathbf{d}; \mathbf{W}(\pi(\mathbf{x}))) = (\sigma, \mathbf{c}) \tag{3}$$

where $\gamma(\cdot)$ is a positional encoding on $\mathbf{x}$ with 6 exponentially increasing frequencies introduced in the original NeRF [27]. The image feature is incorporated as a residual at each layer; see § 5 for more information. We show our pipeline schematically in Fig. 2.

In the few-shot view synthesis task, the query view direction is a useful signal for determining the importance of a particular image feature in the NeRF network. If the query view direction is similar to the input view orientation, the model can rely more directly on the input; if it is dissimilar, the model must leverage the learned prior. Moreover, in the multi-view case, view directions could serve as a signal for the relevance and positioning of different views. For this reason, we input the view directions at the beginning of the NeRF network.

## 4.2. Incorporating Multiple Views

Multiple views provide additional information about the scene and resolve 3D geometric ambiguities inherent to the single-view case. We extend our model to allow for an arbitrary number of views at test time, which distinguishes our method from existing approaches that are designed to only use single input view at test time. [8, 55] Moreover, our formulation is independent of the choice of world space and the order of input views.

In the case that we have multiple input views of the scene, we assume only that the relative camera poses are known. For purposes of explanation, an arbitrary world coordinate system can be fixed for the scene. We denote the $i$th input image as $\mathbf{I}^{(i)}$ and its associated camera transform from the world space to its view space as $\mathbf{P}^{(i)} = \begin{bmatrix}\mathbf{R}^{(i)} & \mathbf{t}^{(i)}\end{bmatrix}$.

For a new target camera ray, we transform a query point $\mathbf{x}$, with view direction $\mathbf{d}$, into the coordinate system of each input view $i$ with the world to camera transform as

$$\mathbf{x}^{(i)} = \mathbf{P}^{(i)}\mathbf{x}, \quad \mathbf{d}^{(i)} = \mathbf{R}^{(i)}\mathbf{d} \tag{4}$$

To obtain the output density and color, we process the coordinates and corresponding features in each view coordinate frame independently and aggregate across the views within the NeRF network. For ease of explanation, we denote the initial layers of the NeRF network as $f_1$, which process inputs in each input view space separately, and the final layers as $f_2$, which process the aggregated views.

We encode each input image into feature volume $\mathbf{W}^{(i)} = E(\mathbf{I}^{(i)})$. For the view-space point $\mathbf{x}^{(i)}$, we extract the corresponding image feature from the feature volume $\mathbf{W}^{(i)}$ at the projected image coordinate $\pi(\mathbf{x}^{(i)})$. We then pass these inputs into $f_1$ to obtain intermediate vectors:

$$\mathbf{V}^{(i)} = f_1\left(\gamma(\mathbf{x}^{(i)}), \mathbf{d}^{(i)}; \mathbf{W}^{(i)}\left(\pi(\mathbf{x}^{(i)})\right)\right). \tag{5}$$

The intermediate $\mathbf{V}^{(i)}$ are then aggregated with the average pooling operator $\psi$ and passed into a the final layers, denoted as $f_2$, to obtain the predicted density and color:

$$(\sigma, \mathbf{c}) = f_2\left(\psi\left(\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(n)}\right)\right). \tag{6}$$

In the single-view special case, this simplifies to Equation 3 with $f = f_2 \circ f_1$, by considering the view space as the world space. An illustration is provided in the supplemental.

# 5. Experiments

We extensively demonstrate our approach in three experimental categories: 1) existing ShapeNet [4] benchmarks for category-specific and category-agnostic view synthesis, 2) ShapeNet scenes with unseen categories and multiple objects, both of which require geometric priors instead of recognition, as well as domain transfer to real car photos and 3) real scenes from the DTU MVS dataset [14].

**Baselines** For ShapeNet benchmarks, we compare quantitatively and qualitatively to SRN [40] and DVR [28], the current state-of-the-art in few-shot novel-view synthesis and 2D-supervised single-view reconstruction respectively. We use the 2D multiview-supervised variant of DVR. In the category-agnostic setting (§ 5.1.2), we also include grayscale rendering of SoftRas [21] results. [1] In the experiments with multiple ShapeNet objects, we compare with SRN, which can also model entire scenes.

For the experiment on the DTU dataset, we compare to NeRF [27] trained on sparse views. Because NeRF is a test-time optimization method, we train a separate model for each scene in the test set.

**Metrics** We report the standard image quality metrics PSNR and SSIM [57] for all evaluations. We also include LPIPS [54], which more accurately reflects human perception, in all evaluations except in the category-specific setup (§ 5.1.1). In this setting, we exactly follow the protocol of SRN [40] to remain comparable to prior works [41, 50, 9, 8, 44], for which source code is unavailable.

**Implementation Details** For the image encoder $E$, to capture both local and global information effectively, we extract a feature pyramid from the image. We use a ResNet34 backbone pretrained on ImageNet for our experiments. Features are extracted prior to the first $4$ pooling layers, upsampled using bilinear interpolation, and concatenated to form latent vectors of size $512$ aligned to each pixel.

To incorporate a point's corresponding image feature into the NeRF network $f$, we choose a ResNet architecture with a residual modulation rather than simply concatenating the feature vector with the point's position and view direction. Specifically, we feed the encoded position and view direction through the network and add the image feature as a residual at the beginning of each ResNet block. We train an independent linear layer for each block residual, in a similar manner as AdaIn and SPADE [13, 30], a method previously used with success in [25, 28]. Please refer to the supplemental for additional details.

## 5.1. ShapeNet Benchmarks

We first evaluate our approach on category-specific and category-agnostic view synthesis tasks on ShapeNet.

### 5.1.1 Category-specific View Synthesis Benchmark

We perform one-shot and two-shot view synthesis on the "chair" and "car" classes of ShapeNet, using the protocol and dataset introduced in [40]. The dataset contains 6591

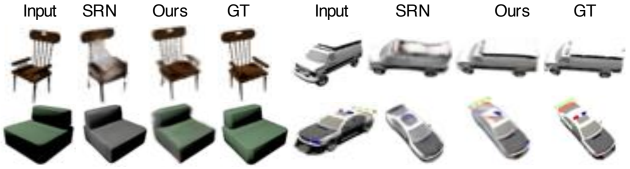


Figure 3: **Category-specific single-view reconstruction benchmark**. We train a separate model for cars and chairs and compare to SRN. The corresponding numbers may be found in Table 2.

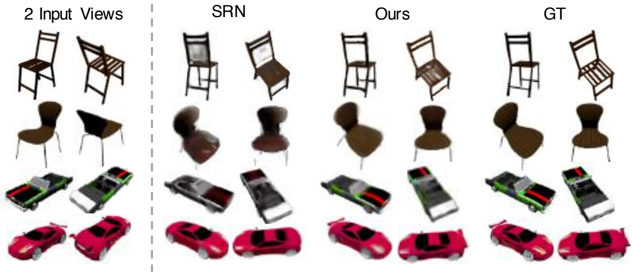


Figure 4: **Category-specific** 2**-view reconstruction benchmark**. We provide two views (left) to each model, and show two novel view renderings in each case (right). Please also refer to Table 2.

| | | 1-view | | 2-view | |
|---|---|---|---|---|---|
| | | PSNR | SSIM | PSNR | SSIM |
| Chairs | GRF [44] | 21.25 | 0.86 | 22.65 | 0.88 |
| | TCO [41] * | 21.27 | 0.88 | 21.33 | 0.88 |
| | dGQN [9] | 21.59 | 0.87 | 22.36 | 0.89 |
| | ENR [8] * | 22.83 | - | - | - |
| | SRN [40] | 22.89 | 0.89 | 24.48 | 0.92 |
| | Ours * | **23.72** | **0.91** | **26.20** | **0.94** |
| Cars | SRN [40] | 22.25 | 0.89 | 24.84 | 0.92 |
| | ENR [8] * | 22.26 | - | - | - |
| | Ours * | **23.17** | **0.90** | **25.66** | **0.94** |

Table 2: **Category-specific 1- and 2-view reconstruction**. Methods marked * do not require canonical poses at test time. In all cases, a single model is trained for each category and used for both 1- and 2-view evaluation. Note ENR is a 1-view only model.

| | 1-view | | | 2-view | | |
|---|---|---|---|---|---|---|
| | ↑ PSNR | ↑ SSIM | ↓ LPIPS | ↑ PSNR | ↑ SSIM | ↓ LPIPS |
| − Local | 20.39 | 0.848 | 0.196 | 21.17 | 0.865 | 0.175 |
| − Dirs | 21.93 | 0.885 | 0.139 | 23.50 | 0.909 | 0.121 |
| Full | **23.43** | **0.911** | **0.104** | **25.95** | **0.939** | **0.071** |

Table 3: **Ablation studies for ShapeNet chair reconstruction.** We show the benefit of using local features over a global code to condition the NeRF network (−Local vs Full), and of providing view directions to the network (−Dirs vs Full).

[1] Color inference is not supported by the public SoftRas code.

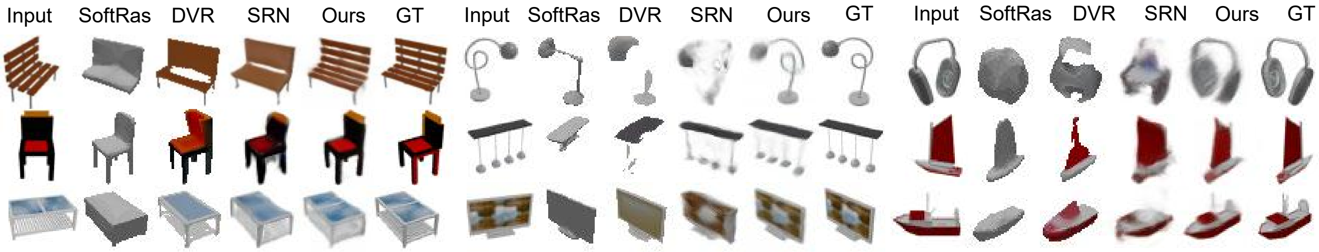


Figure 5: **Category-agnostic single-view reconstruction.** Going beyond the SRN benchmark, we train a single model to the 13 largest ShapeNet categories; we find that our approach produces superior visual results compared to a series of strong baselines. In particular, the model recovers fine detail and thin structure more effectively, even for outlier shapes. Quite visibly, images on monitors and tabletop textures are accurately reproduced; baselines representing the scene as a single latent vector cannot preserve such details of the input image. SRN's test-time latent inversion becomes less reliable as well in this setting. The corresponding quantitative evaluations are available in Table 4. Due to space constraints, we show objects with interesting properties here. Please see the supplemental for sampled results.

| | | plane | bench | cbnt. | car | chair | disp. | lamp | spkr. | rifle | sofa | table | phone | boat | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↑ PSNR | DVR | 25.29 | 22.64 | 24.47 | 23.95 | 19.91 | 20.86 | 23.27 | 20.78 | 23.44 | 23.35 | 21.53 | 24.18 | 25.09 | 22.70 |
| | SRN | 26.62 | 22.20 | 23.42 | 24.40 | 21.85 | 19.07 | 22.17 | 21.04 | 24.95 | 23.65 | 22.45 | 20.87 | 25.86 | 23.28 |
| | Ours | **29.76** | **26.35** | **27.72** | **27.58** | **23.84** | **24.22** | **28.58** | **24.44** | **30.60** | **26.94** | **25.59** | **27.13** | **29.18** | **26.80** |
| ↑ SSIM | DVR | 0.905 | 0.866 | 0.877 | 0.909 | 0.787 | 0.814 | 0.849 | 0.798 | 0.916 | 0.868 | 0.840 | 0.892 | 0.902 | 0.860 |
| | SRN | 0.901 | 0.837 | 0.831 | 0.897 | 0.814 | 0.744 | 0.801 | 0.779 | 0.913 | 0.851 | 0.828 | 0.811 | 0.898 | 0.849 |
| | Ours | **0.947** | **0.911** | **0.910** | **0.942** | **0.858** | **0.867** | **0.913** | **0.855** | **0.968** | **0.908** | **0.898** | **0.922** | **0.939** | **0.910** |
| ↓ LPIPS | DVR | 0.095 | 0.129 | 0.125 | 0.098 | 0.173 | 0.150 | 0.172 | 0.170 | 0.094 | 0.119 | 0.139 | 0.110 | 0.116 | 0.130 |
| | SRN | 0.111 | 0.150 | 0.147 | 0.115 | 0.152 | 0.197 | 0.210 | 0.178 | 0.111 | 0.129 | 0.135 | 0.165 | 0.134 | 0.139 |
| | Ours | **0.084** | **0.116** | **0.105** | **0.095** | **0.146** | **0.129** | **0.114** | **0.141** | **0.066** | **0.116** | **0.098** | **0.097** | **0.111** | **0.108** |

Table 4: **Category-agnostic single-view reconstruction.** Quantitative results for category-agnostic view-synthesis are presented, with a detailed breakdown by category. Our method outperforms the state-of-the-art by significant margins in all categories.

chairs and 3514 cars with a predefined split across object instances. All images have resolution 128 × 128.

A single model is trained for each object class with 50 random views per object instance, randomly sampling either one or two of the training views to encode. For testing, We use 251 novel views on an Archimedean spiral for each object in the test set of object instances, fixing 1-2 informative views as input. We report our performance in comparison with state-of-the-art baselines in Table 2, and show selected qualitative results in Fig. 4. We also include the quantitative results of baselines TCO [41] and dGQN [9] reported in [40] where applicable, and the values available in the recent works ENR [8] and GRF [44] in this setting.

PixelNeRF achieves noticeably superior results despite solving a problem *significantly harder* than SRN because we: 1) use feed-forward prediction, without test-time optimization, 2) do not use ground-truth absolute camera poses at test-time, 3) use view instead of canonical space.

**Ablations.** In Table 3, we show the benefit of using local features and view directions in our model for this category-specific setting. Conditioning the NeRF network on pixel-aligned local features instead of a global code (−Local vs Full) improves performance significantly, for both single and two-view settings. Providing view directions (−Dirs vs Full) also provides a significant boost. For these ablations, we follow an abbreviated evaluation protocol on ShapeNet chairs, using 25 novel views on the Archimedean spiral.

### 5.1.2 Category-agnostic Object Prior

While we found appreciable improvements over baselines in the simplest category-specific benchmark, our method is by no means constrained to it. We show in Table 4 and Fig. 5 that our approach offers a much greater advantage in the *category-agnostic* setting of [21, 28], where we train a single model to the 13 largest categories of ShapeNet. Please see the supplemental for randomly sampled results.

We follow community standards for 2D-supervised methods on multiple ShapeNet categories [28, 16, 21] and use the renderings and splits from Kato et al. [16], which provide 24 fixed elevation views of 64 × 64 resolution for each object instance. During both training and evaluation, a random view is selected as the input view for each object and shared across all baselines. The remaining 23 views are used as target views for computing metrics (see § 5).

## 5.2. Pushing the Boundaries of ShapeNet

Taking a step towards reconstruction in less controlled capture scenarios, we perform experiments on ShapeNet data in three more challenging setups: 1) unseen object categories, 2) multiple-object scenes, and 3) simulation-to-real

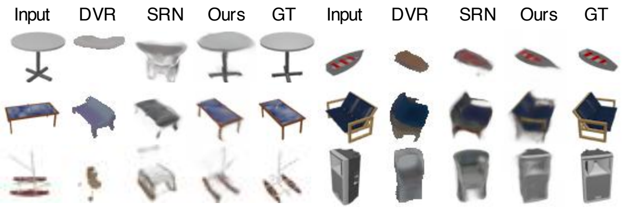


Figure 6: **Generalization to unseen categories.** We evaluate a model trained on planes, cars, and chairs on 10 unseen ShapeNet categories. We find that the model is able to synthesize reasonable views even in this difficult case.

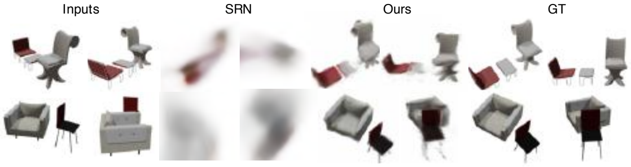


Figure 7: **360°view prediction with multiple objects.** We show qualitative results of our method compared with SRN on scenes composed of multiple ShapeNet chairs. We are easily able to handle this setting, because our prediction is done in view space; in contrast, SRN predicts in canonical space, and struggles with scenes that cannot be aligned in such a way.

| | Unseen category | | | Multiple chairs | | |
|---|---|---|---|---|---|---|
| | ↑ PSNR | ↑ SSIM | ↓ LPIPS | ↑ PSNR | ↑ SSIM | ↓ LPIPS |
| DVR | 17.72 | 0.716 | 0.240 | - | - | - |
| SRN | 18.71 | 0.684 | 0.280 | 14.67 | 0.664 | 0.431 |
| Ours | **22.71** | **0.825** | **0.182** | **23.40** | **0.832** | **0.207** |

Table 5: **Image quality metrics for challenging ShapeNet tasks.** (Left) Average metrics on 10 unseen categories for models trained on only planes, cars, and chairs. See the supplemental for a breakdown by category. (Right) Average metrics for two-view reconstruction for scenes with multiple ShapeNet chairs.

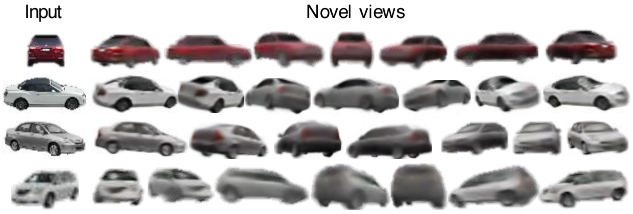


Figure 8: **Results on real car photos.** We apply the car model from § 5.1.1 directly to images from the Stanford cars dataset [18]. The background has been masked out using PointRend [17]. The views are rotations about the view-space vertical axis.

transfer on car images. In these settings, successful reconstruction requires geometric priors; recognition or retrieval alone is not sufficient.

**Generalization to novel categories.** We first aim to reconstruct ShapeNet categories which were not seen in training. Unlike the more standard category-agnostic task described in the previous section, such generalization is impossible with semantic information alone. The results in Table 5 and Fig. 6 suggest our method learns intrinsic geometric and appearance priors which are fairly effective even for objects quite distinct from those seen during training.

We loosely follow the protocol used for zero-shot cross-category reconstruction from [55, **?**]. Note that our baselines [40, 28] do not evaluate in this setting, and we adapt them for the sake of comparison. We train on the airplane, car, and chair categories and test on 10 categories unseen during training, continuing to use the Kato et al. renderings described in § 5.1.2.

**Multiple-object scenes.** We further perform few-shot 360° reconstruction for scenes with multiple randomly placed and oriented ShapeNet chairs. In this setting, the network cannot rely solely on semantic cues for correct object placement and completion. The priors learned by the network must be applicable in an arbitrary coordinate system. We show in Fig. 7 and Table 5 that our formulation allows us to perform well on these simple scenes without additional design modifications. In contrast, SRN models scenes in a canonical space and struggles on held-out scenes.

We generate training images composed with 20 views randomly sampled on the hemisphere and render test images composed of a held out test set of chair instances, with 50 views sampled on an Archimedean spiral. During training, we randomly encode two input views; at test-time, we fix two informative views across the compared methods. In the supplemental, we provide example images from our dataset as well as additional quantitative results and qualitative comparisons with varying numbers of input views.

**Sim2Real on Cars.** We also explore the performance of pixelNeRF on real images from the Stanford cars dataset [18]. We directly apply car model from § 5.1.1 without any fine-tuning. As seen in Fig. 8, the network trained on synthetic data effectively infers shape and texture of the real cars, suggesting our model can transfer beyond the synthetic domain.

Synthesizing the 360° background from a single view is nontrivial and out of the scope for this work. For this demonstration, the off-the-shelf PointRend [17] segmentation model is used to remove the background.

## 5.3. Scene Prior on Real Images

Finally, we demonstrate that our method is applicable for few-shot *wide baseline* novel-view synthesis on real scenes in the DTU MVS dataset [14]. Learning a prior for view synthesis on this dataset poses significant challenges: not only does it consist of more complex scenes, without clear semantic similarities across scenes, it also contains inconsistent backgrounds and lighting between scenes. More-

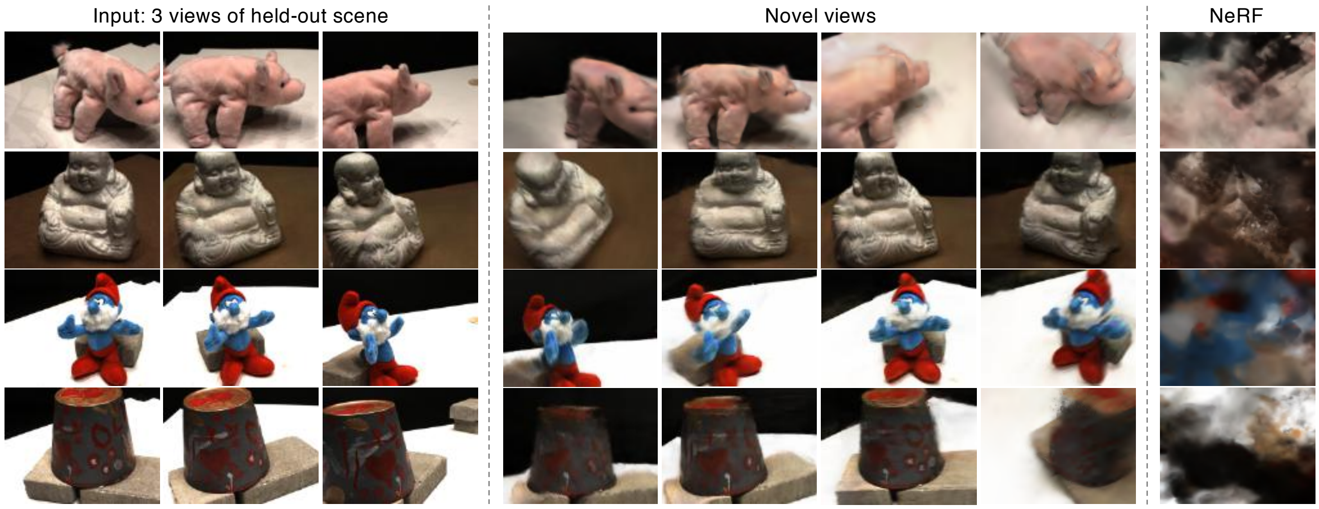


Figure 9: **Wide baseline novel-view synthesis on a real image dataset.** We train our model to distinct scenes in the DTU MVS dataset [14]. Perhaps surprisingly, even in this case, our model is able to infer novel views with reasonable quality for held-out scenes without further test-time optimization, all from only three views. Note the train/test sets share no overlapping scenes.

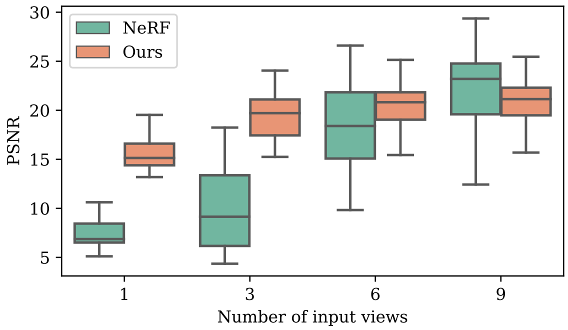


Figure 10: **PSNR of few-shot feed-forward DTU reconstruction.** We show the quantiles of PSNR on DTU for our method and NeRF, given 1, 3, 6, or 9 input views. Separate NeRFs are trained per scene and number of input views, while our method requires only a single model trained with 3 encoded views.

over, under 100 scenes are available for training. We found that the standard data split introduced in MVSNet [53] contains overlap between scenes of the training and test sets. Therefore, for our purposes, we use a different split of 88 training scenes and 15 test scenes, in which there are no shared or highly similar scenes between the two sets. Images are down-sampled to a resolution of $400 \times 300$.

We train one model across all training scenes by encoding 3 random views of a scene. During test time, we choose a set of fixed informative input views shared across all instances. We show in Fig. 9 that our method can perform view synthesis on the held-out test scenes. We further quantitatively compare the performance of our feed-forward model with NeRF optimized to the same set of input views in Fig. 10. Note that training each of 60 NeRFs took 14 hours; in contrast, pixelNeRF is applied to new scenes immediately without any test-time optimization.

## 6. Discussion

We have presented pixelNeRF, a framework to learn a scene prior for reconstructing NeRFs from one or a few images. Through extensive experiments, we have established that our approach can be successfully applied in a variety of settings. We addressed some shortcomings of NeRF, but there are challenges yet to be explored: 1) Like NeRF, our rendering time is slow, and in fact, our runtime increases linearly when given more input views. Further, some methods (e.g. [28, 21]) can recover a mesh from the image enabling fast rendering and manipulation afterwards, while NeRF-based representations cannot be converted to meshes very reliably. Improving NeRF's efficiency is an important research question that can enable real-time applications. 2) As in the vanilla NeRF, we manually tune ray sampling bounds $t_n, t_f$ and a scale for the positional encoding. Making NeRF-related methods scale-invariant is a crucial challenge. 3) While we have demonstrated our method on real data from the DTU dataset, we acknowledge that this dataset was captured under controlled settings and has matching camera poses across all scenes with limited viewpoints. Ultimately, our approach is bottlenecked by the availability of large-scale wide baseline multi-view datasets, limiting the applicability to datasets such as ShapeNet and DTU. Learning a general prior for 360° scenes in-the-wild is an exciting direction for future work.

## Acknowledgements

We thank Shubham Goel and Hang Gao for comments on the text. We also thank Emilien Dupont and Vincent Sitzmann for helpful discussions.

## References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, pages 72–79, 2009.

[2] Miguel Angel Bautista, Walter Talbott, Shuangfei Zhai, Nitish Srivastava, and Joshua M. Susskind. On the generalization of learning-based 3d reconstruction. In *WACV*, pages 2180–2189, January 2021.

[3] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432, 2001.

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], 2015.

[5] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, pages 4090–4100, 2019.

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

[7] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *CVPR*, pages 7830–7839, 2020.

[8] Emilien Dupont, Miguel Angel Bautista, Alex Colburn, Aditya Sankar, Carlos Guestrin, Joshua Susskind, and Qi Shan. Equivariant neural rendering. In *ICML*, 2020.

[9] S. Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari Morcos, Marta Garnelo, Avraham Ruderman, Andrei Rusu, Ivo Danihelka, Karol Gregor, David Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360:1204–1210, 06 2018.

[10] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deep stereo: Learning to predict new views from the world's imagery. In *CVPR*, pages 5515–5524, 2016.

[11] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[12] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018.

[13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[14] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, pages 406–413, 2014.

[15] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017.

[16] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[17] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020.

[18] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[19] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996.

[20] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. In *NeurIPS*, 2020.

[21] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019.

[22] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *NeurIPS*, 2019.

[23] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019.

[24] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021.

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

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

[27] 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. In *ECCV*, 2020.

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020.

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

[30] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.

Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Stefan Popov, Pablo Bauszat, and Vittorio Ferrari. CoReNet: Coherent 3d scene reconstructionfrom a single rgb image. In *ECCV*, 2020.

[33] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, pages 623–640, 2020.

[34] S. Saito, Z. Huang, R. Natsume, S. Morishima, H. Li, and A. Kanazawa. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, pages 2304–2314, 2019.

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

[36] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, 1998.

[37] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020.

[38] Daeyun Shin, Charless Fowlkes, and Derek Hoiem. Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction. In *CVPR*, 2018.

[39] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. DeepVoxels: Learning persistent 3d feature embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

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

[41] 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.

[42] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, pages 3405–3414, 2019.

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

[44] Alex Trevithick and Bo Yang. GRF: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020.

[45] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020.

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

[47] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[48] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[49] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020.

[50] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Interpretable transformations with encoder-decoder networks. In *ICCV*, pages 5737–5746, 2017.

[51] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, pages 490–500, 2019.

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

[53] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

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

[55] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Joshua B Tenenbaum, William T Freeman, and Jiajun Wu. Learning to Reconstruct Shapes from Unseen Classes. In *NeurIPS*, 2018.

[56] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, pages 286–301, 2016.

[57] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

# Appendix

## A. Additional Results

In this section, we provide additional qualitative and quantitative results for several key experiments. The reader is encouraged to refer to the video and website for a richer, animated presentation of qualitative results.

### A.1. Category-agnostic ShapeNet: Random Results

We show *randomly* sampled results for the category-agnostic setting (§ 5.1.2) in Fig. 11, Fig. 12, and Fig. 13. Specifically, we sample 6 uniformly random objects for each of the 13 largest ShapeNet categories and show comparisons to the baselines [21, 28, 40] as in the main paper. Two random views are selected from the 24 available views to be source and target views respectively.

### A.2. Generalization to novel categories

In Table 6 we show a detailed breakdown of metrics by category on unseen categories, as promised in the main paper.

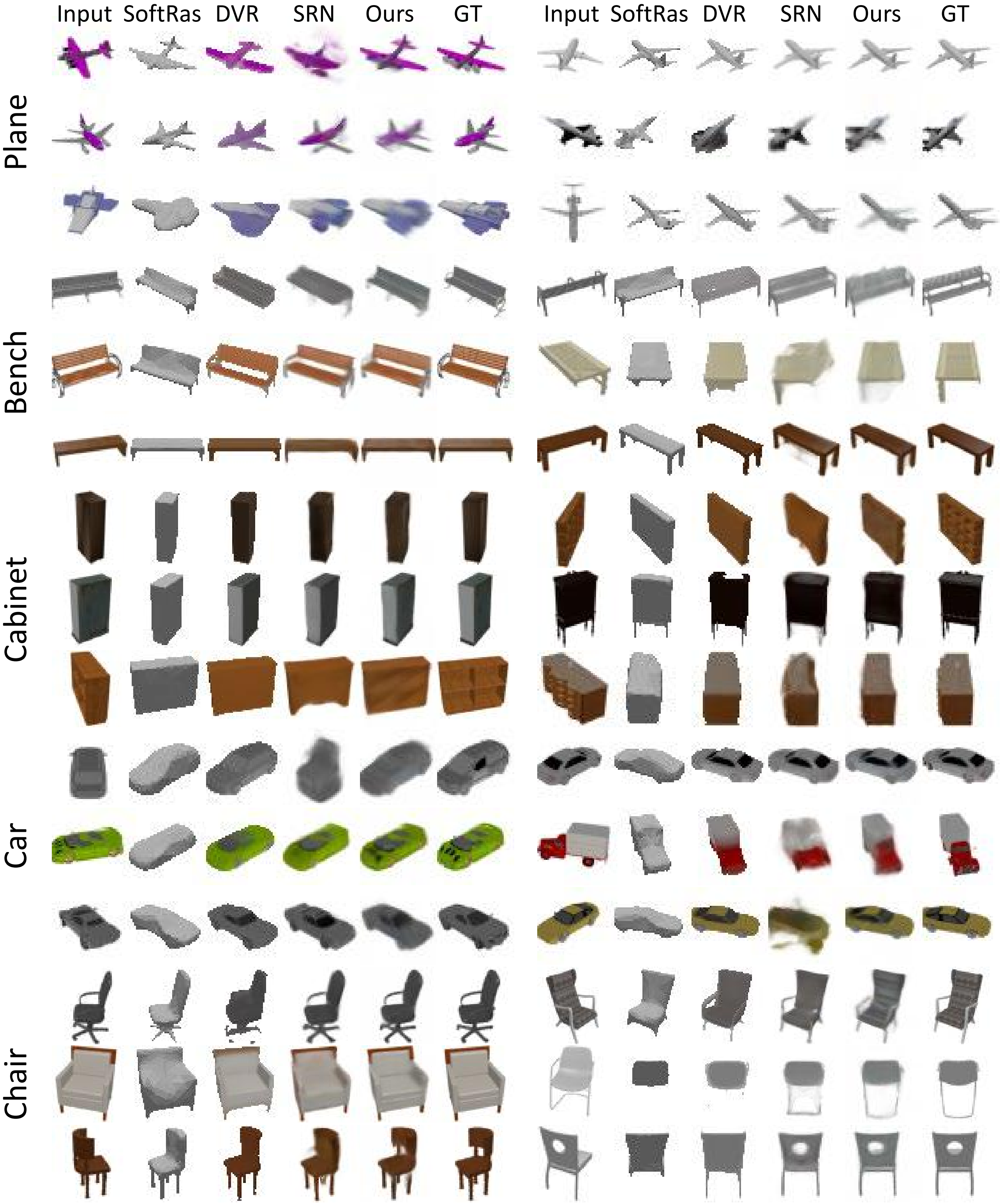


Figure 11: **Randomly sampled results.** part 1

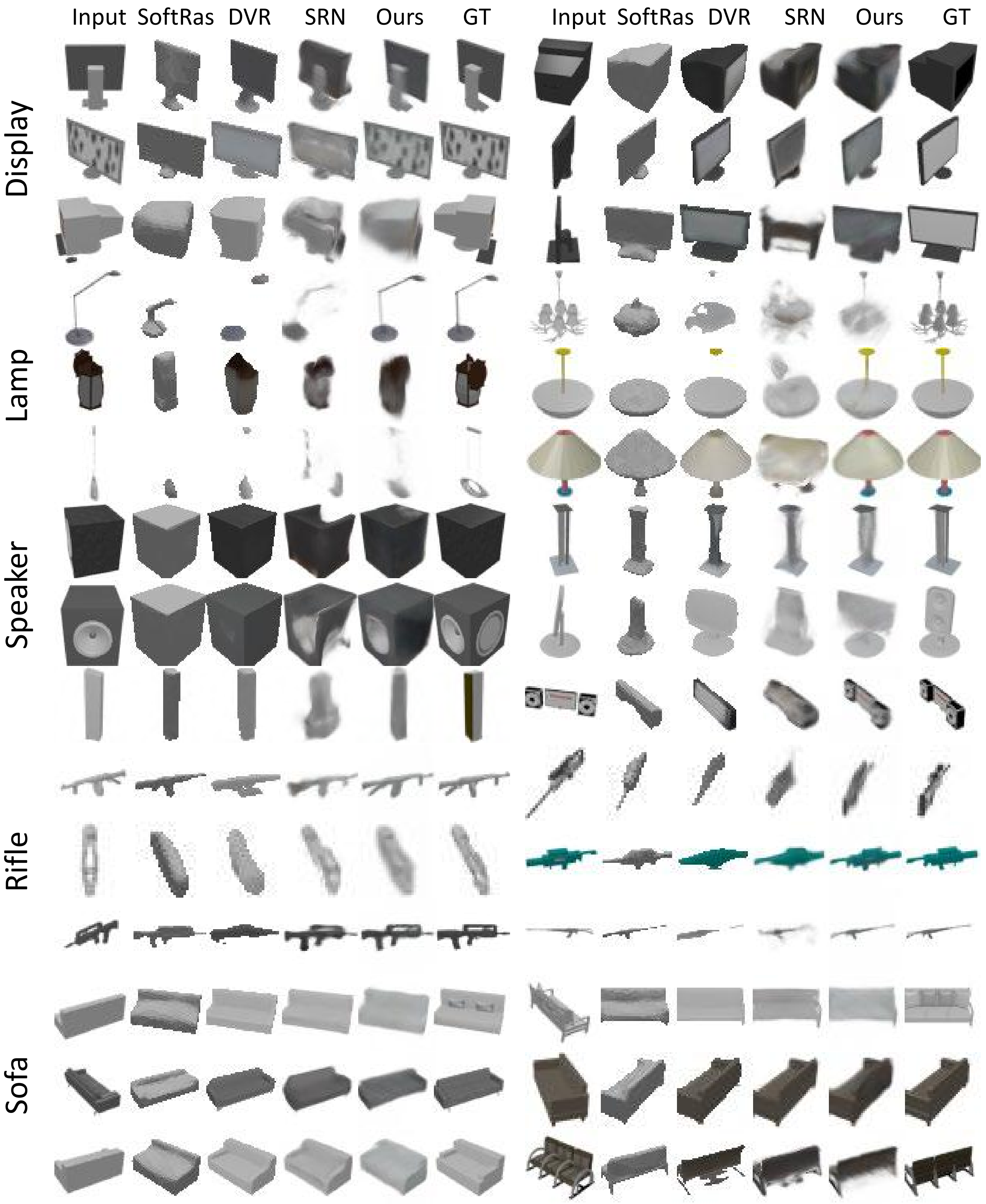


Figure 12: **Randomly sampled results.** part 2

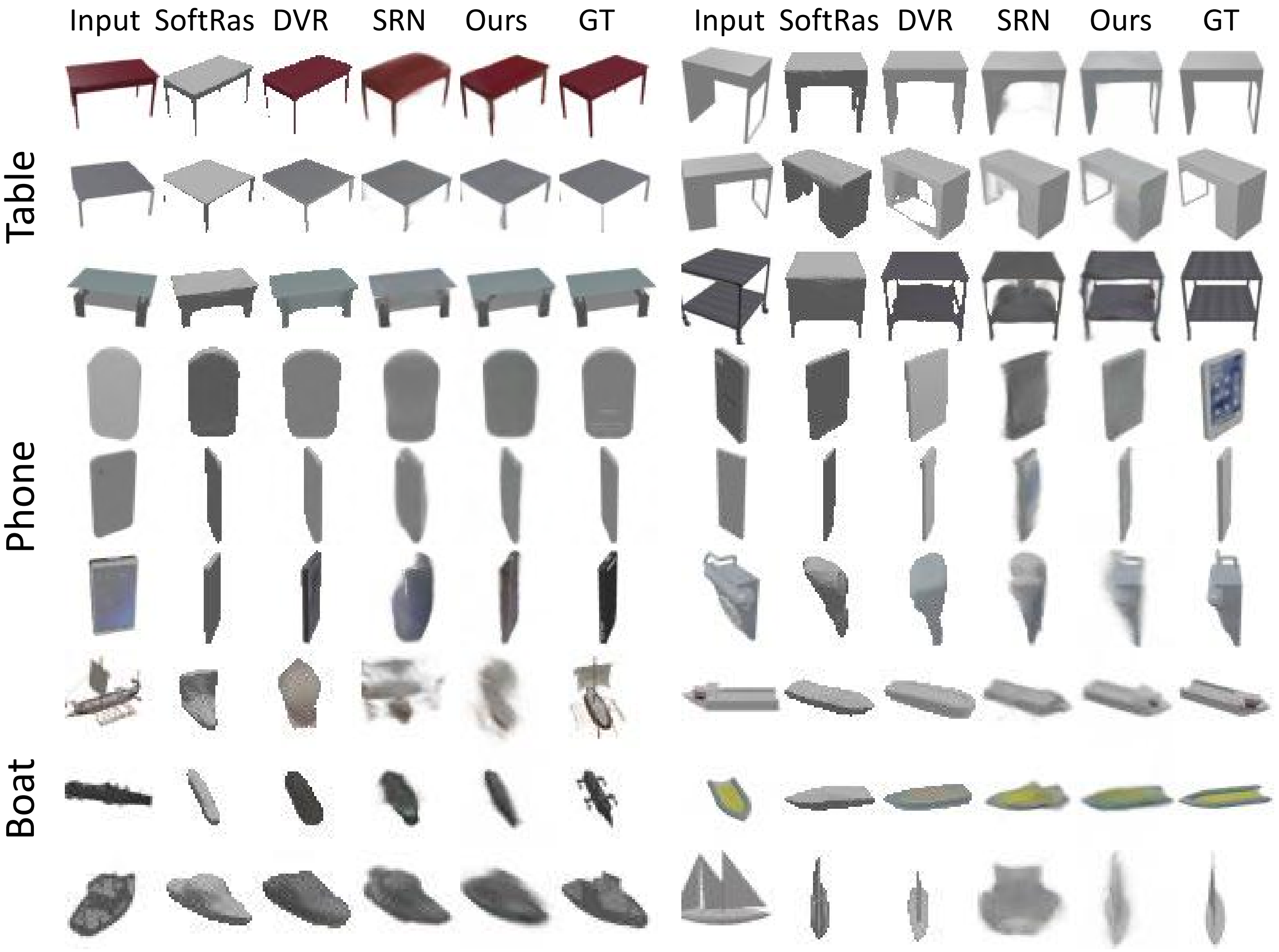


Figure 13: **Randomly sampled results.** part 3

### A.3. Two-object Scenes

We show samples from our rendered dataset in Fig. 14. An analysis of performance as more views become available is in Table 7, for our method when compared with SRN. We also show *randomly sampled* results of scenes when given two input views in Figure 15. We train our model using two random views, and give the model either one, two, or three fixed informative views during inference.

### A.4. DTU

In Fig. 16, we show quantitative results for each scene as well as renderings of of all test scenes not shown in the main paper.

In Table 8 we provide means and standard deviations of metrics for our method and NeRF on the DTU test set, with 1, 3, 6, 9 views. The PSNR here was plotted in Fig. 10 of the main paper

## B. Reproducibility

### B.1. Implementation Details

Here we describe implementation details in the interest of reproducibility. A general remark is that due to the high compute cost, we did not spend significant effort to tune the architecture or training procedure, and it is possible that variations can perform better, or that smaller models may suffice.

**Encoder** $E$ As briefly discussed in the main paper, we use a ResNet34 backbone and extract a feature pyramid by taking the feature maps prior to the first pooling operation and after the first ResNet 3 layers. For a $H \times W$ image, the feature maps have shapes

1. $64 \times H/2 \times W/2$
2. $64 \times H/4 \times W/4$

| | | bench | cbnt. | disp. | lamp | spkr. | rifle | sofa | table | phone | boat | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↑ PSNR | DVR | 18.37 | 17.19 | 14.33 | 18.48 | 16.09 | 20.28 | 18.62 | 16.20 | 16.84 | 22.43 | 17.72 |
| | SRN | 18.71 | 17.04 | 15.06 | 19.26 | 17.06 | 23.12 | 18.76 | 17.35 | 15.66 | 24.97 | 18.71 |
| | Ours | **23.79** | **22.85** | **18.09** | **22.76** | **21.22** | **23.68** | **24.62** | **21.65** | **21.05** | **26.55** | **22.71** |
| ↑ SSIM | DVR | 0.754 | 0.686 | 0.601 | 0.749 | 0.657 | 0.858 | 0.755 | 0.644 | 0.731 | 0.857 | 0.716 |
| | SRN | 0.702 | 0.626 | 0.577 | 0.685 | 0.633 | 0.875 | 0.702 | 0.617 | 0.635 | 0.875 | 0.684 |
| | Ours | **0.863** | **0.814** | **0.687** | **0.818** | **0.778** | **0.899** | **0.866** | **0.798** | **0.801** | **0.896** | **0.825** |
| ↓ LPIPS | DVR | 0.219 | 0.257 | 0.306 | 0.259 | 0.266 | 0.158 | 0.196 | 0.280 | 0.245 | 0.152 | 0.240 |
| | SRN | 0.282 | 0.314 | 0.333 | 0.321 | 0.289 | 0.175 | 0.248 | 0.315 | 0.324 | 0.163 | 0.280 |
| | Ours | **0.164** | **0.186** | **0.271** | **0.208** | **0.203** | **0.141** | **0.157** | **0.188** | **0.207** | **0.148** | **0.182** |

Table 6: **Generalization to novel categories**. Expanding on Table 5 in the main paper, we show quantitative results with a breakdown by category.

| | 1-view | | | 2-view | | | 3-view | | |
|---|---|---|---|---|---|---|---|---|---|
| | ↑ PSNR | ↑ SSIM | ↓ LPIPS | ↑ PSNR | ↑ SSIM | ↓ LPIPS | ↑ PSNR | ↑ SSIM | ↓ LPIPS |
| SRN | 13.76 | 0.658 | 0.422 | 14.28 | 0.660 | 0.432 | 14.67 | 0.664 | 0.431 |
| Ours | **20.15** | **0.767** | **0.274** | **23.40** | **0.832** | **0.207** | **23.68** | **0.800** | **0.206** |

Table 7: **Performance on synthetic two-object dataset with increasing number of views at test time.** Image quality metrics for SRN and our method, when increasing the number of views given at test time.

3. $128 \times H/8 \times W/8$
4. $256 \times H/16 \times W/16$

These are upsampled bilinearly to $H/2 \times W/2$ and concatenated into a volume of size $512 \times H/2 \times W/2$. For a $64 \times 64$ image, to avoid losing too much resolution, we skip the first pooling layer, so that the image resolutions are at $1/2, 1/2, 1/4, 1/8$ of the input rather than $1/2, 1/4, 1/8, 1/16$. We use ImageNet pretrained weights provided through PyTorch.

**NeRF network** $f$ We employ a fully-connected ResNet architecture with 5 ResNet blocks and width 512, similar to that in [28]. To enable arbitrary number of views as input, we aggregate across the source-views after block 3 using an average-pooling operation. This architecture is illustrated in Fig. 18. We remark that due to computational cost, we did not tune this architecture very much in practice.

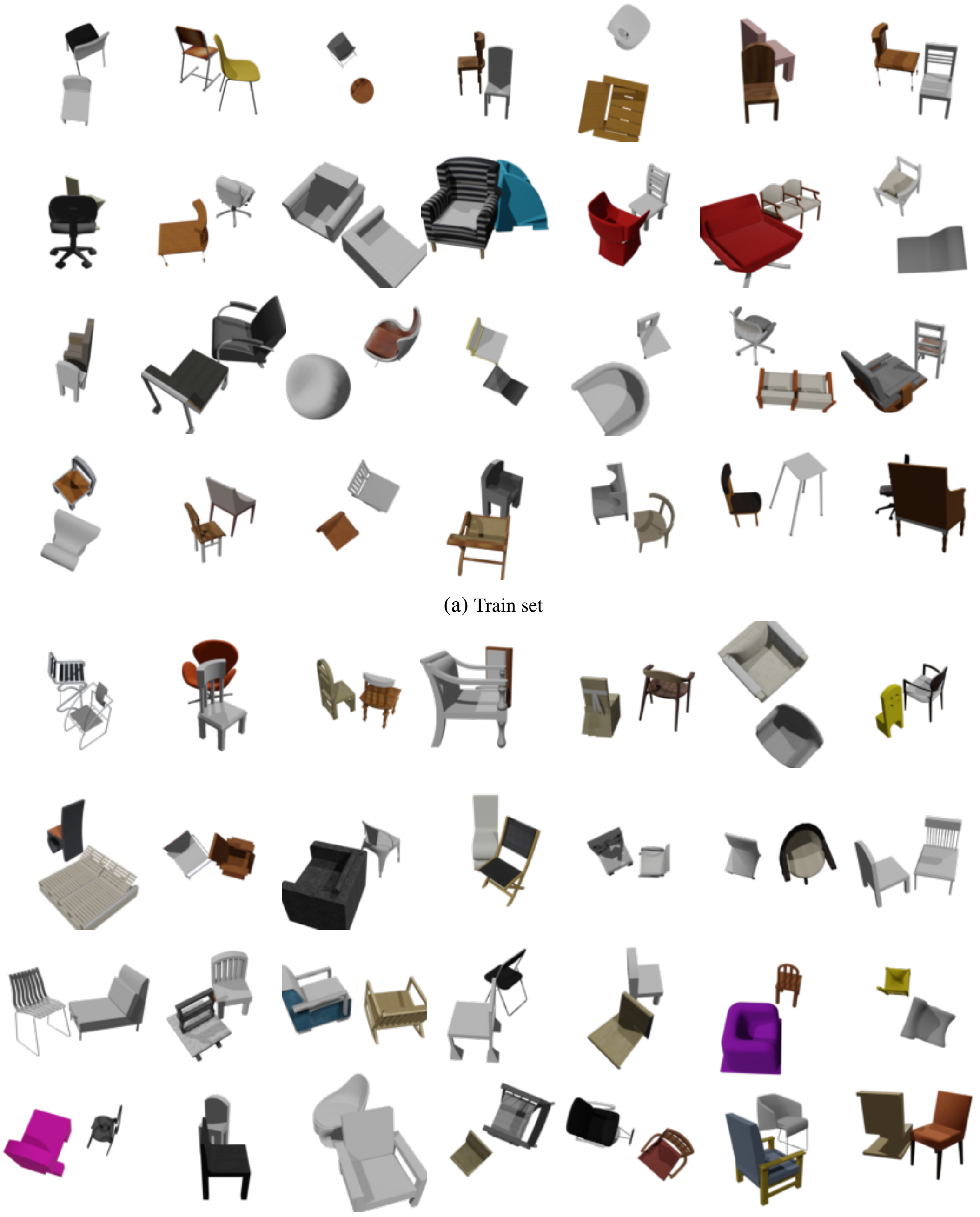
**Hierarchical volume sampling** To improve the sampling efficiency, in practice, we also use *coarse* and *fine* NeRF networks $f_c, f_f$ as in the vanilla NeRF [27], both of which share an identical architecture described above. Note that the encoder $E$ is not duplicated.

More precisely, we use 64 stratified uniform and 16 importance samples, and additionally take 16 fine samples with a normal distribution (SD 0.01) around the expected ray termination (i.e. depth) from the coarse model, to further promote denser sampling near the surface.

**NeRF rendering hyperparameters** We use positional encoding $\gamma$ from NeRF for the spatial coordinates, with exponentially increasing frequencies:

$$\gamma(\mathbf{x}) = \begin{pmatrix} \sin(2^0 \omega \mathbf{x}) \\ \cos(2^0 \omega \mathbf{x}) \\ \sin(2^1 \omega \mathbf{x}) \\ \cos(2^1 \omega \mathbf{x}) \\ \vdots \\ \sin(2^{L-1} \omega \mathbf{x}) \\ \cos(2^{L-1} \omega \mathbf{x}) \end{pmatrix} \tag{7}$$

Note that we do not apply the encoding to the view directions. In all experiments, we set $L = 6$. We also concatenate the input coordinates along the encoding as in the NeRF implementation. $\omega$ is a scaling factor, set (rather arbitrarily) to 1.5 for the single-category, category-agnostic ShapeNet experiments as well as the DTU experiment, and to 2.0 for the multi-object experiment. While the exponent base can be tuned, in practice we left it at 2 as in NeRF.



(a) Train set

(b) Test set

Figure 14: Randomly sampled images from the synthetic two-object scene dataset

The sampling bounds were set manually for each dataset. They were $[1.25, 2.75]$ for ShapeNet chairs, $[0.8, 1.8]$ for ShapeNet cars, $[1.2, 4.0]$ for Kato et al. [16] renderings (category agnostic, novel category), $[4.0, 9.0]$ for our rendered 2-object dataset, and $[0.1, 5.0]$ for input.

We use a white background color in NeRF to match the ShapeNet renderings, except in the DTU setup where a black background is used.

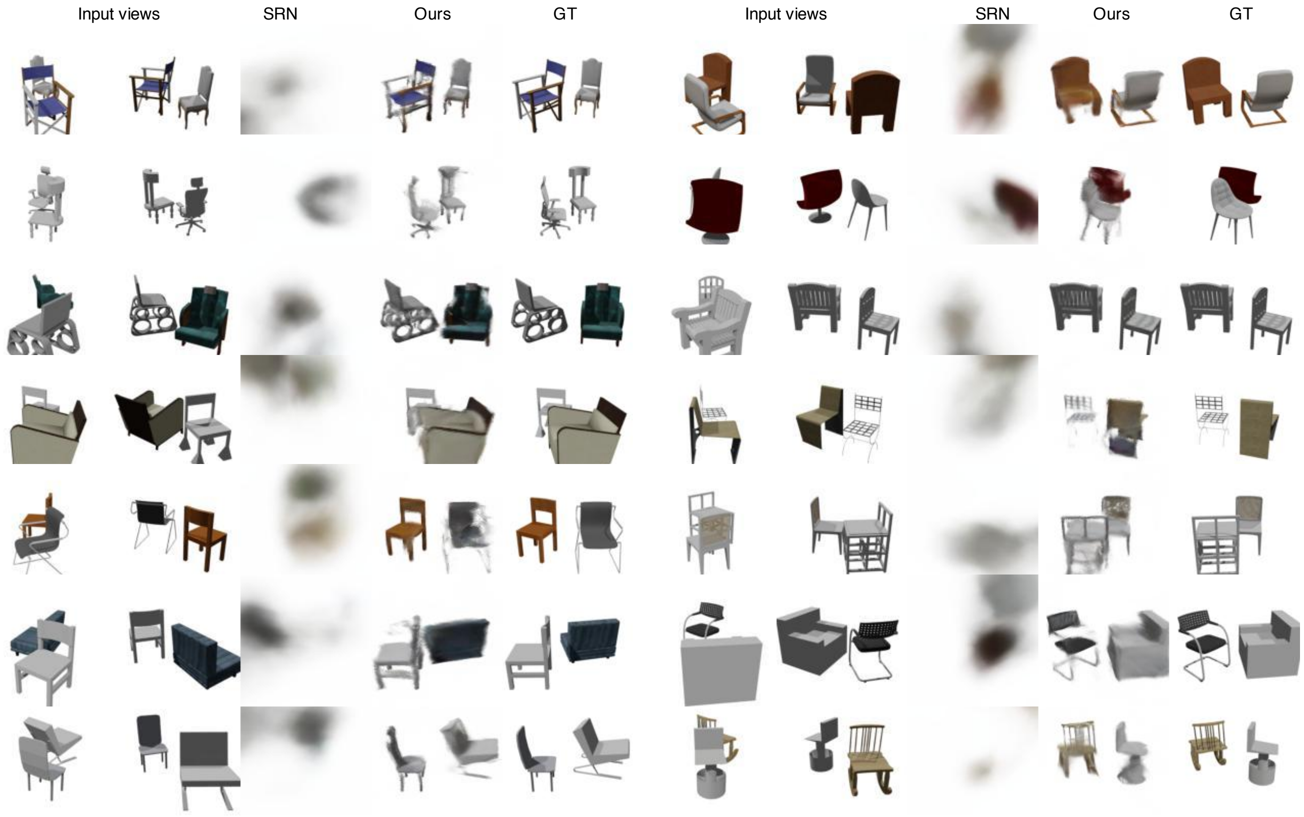


Figure 15: Randomly sampled results for two object scenes, when given two input views.

| | | 1 View | | | 3 View | | | 6 View | | | 9 View | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS |
| Ours | Mean | **15.55** | **0.537** | **0.535** | **19.33** | **0.695** | **0.387** | **20.43** | **0.732** | 0.361 | 21.10 | 0.758 | 0.337 |
| | SD | 1.87 | 0.127 | 0.081 | 2.59 | 0.131 | 0.105 | 2.66 | 0.115 | 0.102 | 2.71 | 0.102 | 0.094 |
| NeRF | Mean | 8.00 | 0.286 | 0.703 | 9.85 | 0.374 | 0.622 | 18.59 | 0.719 | **0.347** | **22.14** | **0.820** | **0.262** |
| | SD | 3.20 | 0.093 | 0.055 | 4.69 | 0.173 | 0.137 | 4.72 | 0.177 | 0.133 | 4.33 | 0.131 | 0.109 |

Table 8: **DTU aggregate metrics vs. NeRF**. Expanding on Fig. 10 in the main paper, we compare our method to NeRF on DTU test scenes quantitatively. Recall higher is better for PSNR and SSIM, while lower is better for LPIPS. Note that PixelNeRF is a feed-forward method, while a NeRF was optimized for 14 hours for each scene and set of input views.

**Model implementation** We implement all models using the PyTorch [31] framework.

## B.2. Experimental Details

We first provide general details about the metrics and training procedure common to all experiments, then present more specific details for each experimental setting in subsections.

**Metric details** We use PSNR and SSIM from the scikit-image [47] package as in SRN [40], whereas LPIPS is computed with the code provided by the LPIPS authors [54] after normalizing the pixel values to the $[-1, 1]$ range. We use the VGG network version of LPIPS following NeRF [27].

**Training** For all experiments, we take the learning rate to be $10^{-4}$. We use a batch size of $4$ instances and 128 rays

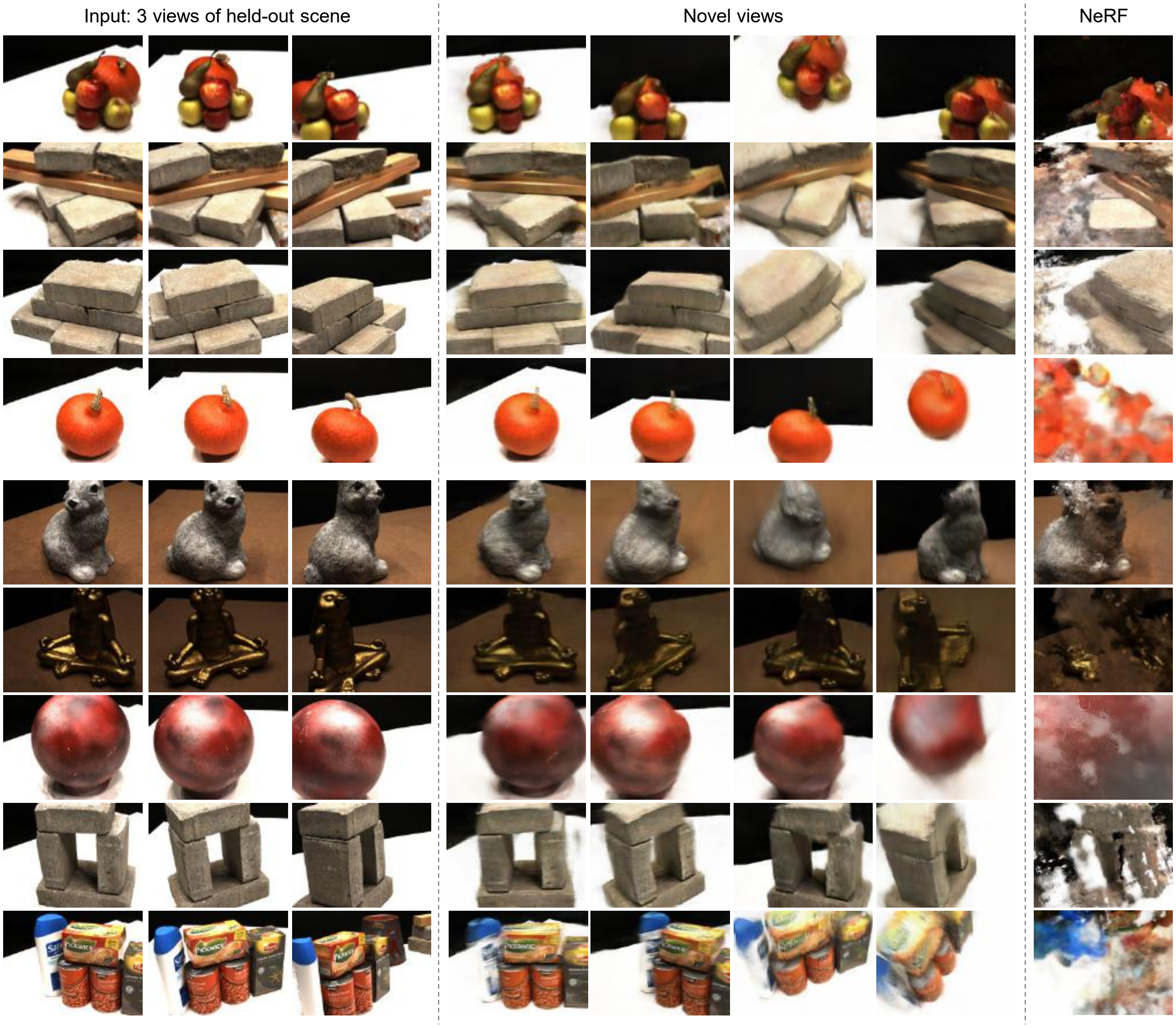


Figure 16: **Additional DTU results.** Views from the remaining 9 scenes are shown.

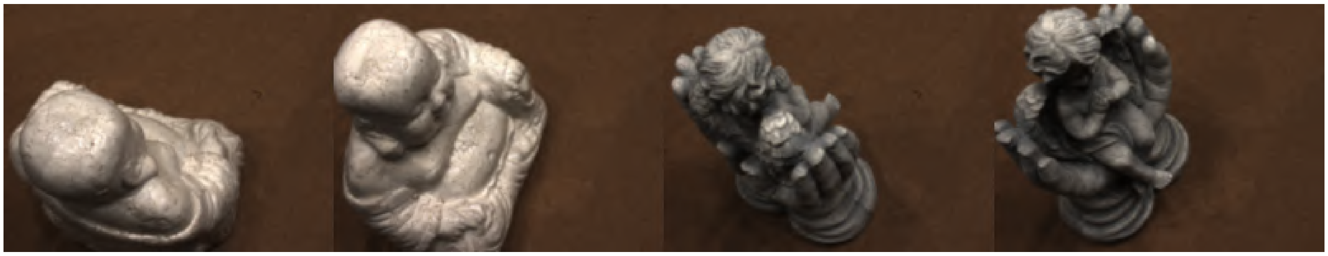


Figure 17: **DTU split overlap.** The first and third scans (115, 119) are from the standard DTU training set from MVSNet, while the second and fourth (114, 118) are from the test set. In our split, highly similar scenes are either all placed in the same set or discarded.

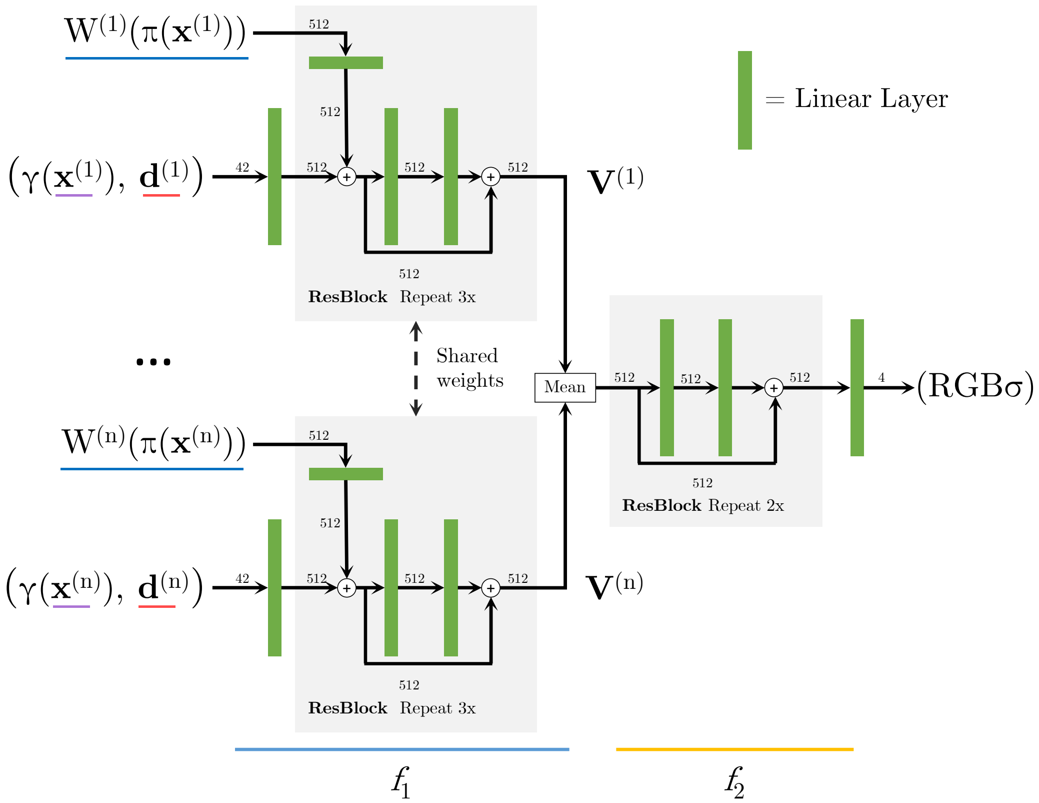


Figure 18: **Multi-view NeRF Network Architecture.** We use notation established in § 5.1.2 of the main paper, where $\gamma$ denotes a positional encoding with 6 exponentially increasing frequencies. Each linear layer is followed by a ReLU activation. Note that in the single-view case, $f_1$ and $f_2$ can be considered a single ResNet $f = f_2 \circ f_1$.

per instances.

| **Full Name** | cabinet | display | speaker |
|---|---|---|---|
| **Abbreviation** | cbnt. | disp. | spkr. |

Table 9: ShapeNet category name abbreviations.

### B.2.1 Single-category ShapeNet

We train for 400000 iterations, which took roughly 6 days on a single Titan RTX. For efficiency, we sample rays from within a tight bounding box around the object for the first 300000 iterations, after which we remove the bounding box to avoid background artifacts. Further, we use 2 input views for the first 300000 iterations and after that, we randomly choose to take either 1 or 2 views as input to encourage the model to work with either 1 or 2 views.

SRN's evaluation protocol is followed: in the 1-view case, we use view 64 as input, and in the 2-view case, we use views 64 and 128.

**Baselines** For SRN [40], we use the pretrained chair model from the public GitHub repository. Note that SRN requires a test-time training step (latent inversion) to generate result images; we apply latent inversion for 170000 iterations for both the 1-view and 2-view cases for chairs.

Recall that, due to a camera sampling bug, we use an updated car dataset provided by the SRN author. Thus, we follow instructions in the Github to train a model on the new dataset; we train for 400000 iterations and apply latent inversion for 100000 iterations for each of the 1-view and 2-view cases. Note the quantitative results we report are slightly lower than that in [8] in the single-view case, but substantially higher than in the original SRN paper, which used the bugged renderings. For the remaining baselines, we only report numbers from the relevant papers on the same task.

### B.2.2 Category-agnostic ShapeNet

We train our model for 800000 iterations on the entire training set, where rays are sampled from within a tight bounding box for the first 400000 iterations. This took about 6 days on an RTX 2080Ti.

**Evaluation protocol** As discussed in the main paper, we evaluate on the test split from [16] as provided by DVR [28]. To ensure fairness, we sampled a random input view to encode for each object and use this view for all baselines as well.

**Baselines** For DVR [28], we use the pretrained 2D multiview-supervised model from the public GitHub and the provided rendering code (in `render.py`). For Soft-Ras [21], we similarly use the pretrained ShapeNet model from the public GitHub repo and obtain images using their renderer library.

Since SRN [40] did not originally evaluate in this setting, we train a model for this category-agnostic setting using the public code. We train for 1 million iterations and perform latent inversion for 260000 iterations, taking about 14 days on a Titan RTX in total.

### B.2.3 Generalization to Novel Categories

We train our model for 680000 iterations across all instances of 3 categories: airplane, car, and chair. Rays are sampled from within a tight bounding box for the first 400000 iterations. This took about 5 days on a GTX 1080Ti.

**Evaluation protocol** Since there are more than 25000 objects in the 10 remaining categories, it would be computationally prohibitive to evaluate on all of them. For our purposes, we sample 25% of the objects from each category for testing, using the remaining for validation. The protocol otherwise remains the same as in the category-agnostic model (§ B.2.2).

**Baselines** We train SRN and DVR as in § B.2.2. For DVR [28] we turned off the use of visual hull depth for sampling, since this information was not provided for all instances of the dataset shipped with DVR.

### B.2.4 Two-object Scenes

We generate more complex synthetic scenes consisting of two ShapeNet chairs. We subdivide ShapeNet chairs into 2715 training instances and 1101 test instances. We generate scenes by randomly placing instances within each split around the origin, rotated randomly about each object's z-axis, and render 128 × 128 resolution images. Per instance, we render 20 training views sampled binned uniform on the hemisphere, and 50 testing views sampled on an Archimedean spiral, similar to the SRN protocol.

We compare with SRN [40] as our baseline on this task, using the publicly available code. We note that SRN performs prediction in a canonical object-centric coordinate system, and used this version for this task. We train a model for evaluation on this two-object dataset using one, two, and three input views. We first train the model for 1 million iterations. Then for each number of input views, we fix the set of input views per instance and perform latent inversion for 150,000 iterations.

### B.2.5 Sim2Real on Real Car Images

We use car images from the Stanford Cars dataset [18]. PointRend [17] is applied to the images to obtain foreground masks and bounding boxes. After removing the background with this mask, the image is then translated and rescaled so that the center of the bounding box is at the center of the image and the shorter side of the bounding box is 1/4 of the image side length, 128. This normalization heuristic is motivated by the observation that the shorter side roughly corresponds to the height or width of the car, which is a more constant quantity than the length.

For evaluation, we set the camera pose to identity and use the same sampling strategy and bounds as at train time for the single-category cars model.

### B.2.6 Real Images on DTU

A single model is trained on the 88 training scenes. We use exposure level 3 only. Note that while there are several views per scene with incorrect exposure throughout the DTU dataset, we did not remove them for training purposes. At each training step, a random color jitter augmentation is applied equally to all views of each object.

**Dataset details** While we solve a very different task from MVSNet [53] which predicts depth maps from short-baseline views and is 2.5D supervised, we considered using the MVSNet [53] DTU split to conform to standards for training on DTU. However, we found that the the split contained effective overlap across the train/val/test sets, making it a poor benchmark of cross-scene generalization, as shown in Fig. 17. For our purposes, we created a different split to avoid this issue: we use scans 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, 114 for testing and all other scans except 1, 2, 7, 25, 26, 27, 29, 39, 51, 54, 56, 57, 58, 73, 83, 111, 112, 113, 115, 116, 117 for training.

We downsampled all DTU images to $400 \times 300$ and adjusted the world scale of all scans by a factor or $1/300$.

**Evaluation protocol** We separately evaluate using 1, 3, 6, 9 informative input views and calculate image metrics with the remaining views. Specifically, we selected views 25, 22, 28, 40, 44,48, 0, 8, 13 for input, taking a prefix of this list when less than 9 views are used. We exclude the views with bad exposure (they are 3, 4, 5, 6, 7, 16, 17, 18, 19, 20, 21, 36, 37, 38, 39) for testing.

**Baselines** We train a total of 60 NeRFs for comparison, one for each scene and number of input views, using the original NeRF TensorFlow code. Each NeRF is trained for 400000 iterations with ray batch size 128, which takes about 14 hours on an RTX Titan, to ensure convergence.

We found that NeRF did not converge in 5 cases when given 6 or 9 views, including in the case of smurf (scan 82) shown in the video. This is possibly due to exposure variation in the DTU dataset. For these scenes, we initialized the model to the trained weights for the same scene with 3 less views and train for about 200000 additional iterations to get reasonable results.