ReconFusion: 3D Reconstruction with Diffusion Priors

Rundi Wu[1,2*] Ben Mildenhall[1*] Philipp Henzler[1] Keunhong Park[1] Ruiqi Gao[3] Daniel Watson[3]
Pratul P. Srinivasan[1] Dor Verbin[1] Jonathan T. Barron[1] Ben Poole[3] Aleksander Hołyński[1*]

[1]Google Research [2]Columbia University [3]Google DeepMind

[*] denotes equal contribution

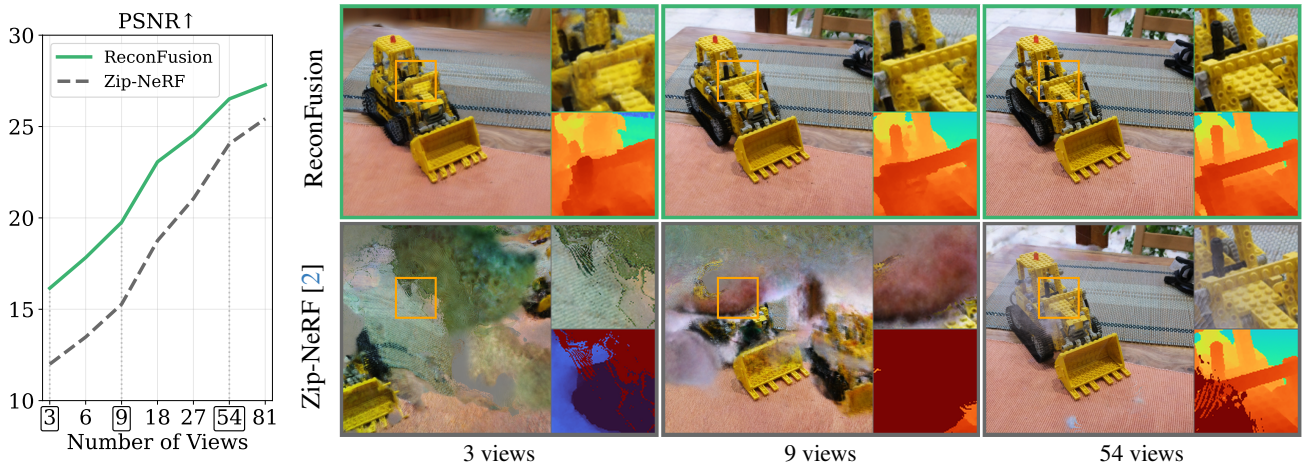


Figure 1. Methods for reconstructing a 3D scene from images, such as Neural Radiance Fields (NeRF), often exhibit artifacts when trained with few input views. ReconFusion uses a diffusion model trained for novel view synthesis to regularize NeRF optimization. When the reconstruction problem is severely underconstrained (3 and 9 views), this prior can greatly improve robustness and often prevent catastrophic failures. Even in the case of significantly more observations (54 views), ReconFusion improves quality and helps reduce "floater" artifacts common to volumetric reconstruction methods like NeRF. We encourage the reader to view the results at reconfusion.github.io to see the improvement our method can provide in few-view captures of real world scenes.

Abstract

3D reconstruction methods such as Neural Radiance Fields (NeRFs) excel at rendering photorealistic novel views of complex scenes. However, recovering a high-quality NeRF typically requires tens to hundreds of input images, resulting in a time-consuming capture process. We present ReconFusion to reconstruct real-world scenes using only a few photos. Our approach leverages a diffusion prior for novel view synthesis, trained on synthetic and multiview datasets, which regularizes a NeRF-based 3D reconstruction pipeline at novel camera poses beyond those captured by the set of input images. Our method synthesizes realistic geometry and texture in underconstrained regions while preserving the appearance of observed regions. We perform an extensive evaluation across various real-world datasets, including forward-facing and 360-degree scenes, demonstrating significant performance improvements over previous few-view NeRF reconstruction approaches. Please see our project page at reconfusion.github.io.

1. Introduction

Advances in 3D reconstruction have enabled the transformation of images of real-world scenes into 3D models which produce photorealistic renderings from novel viewpoints [26, 32]. Methods like NeRF [32] optimize a 3D representation whose renderings match observed input images at given camera poses. However, renderings from under-observed views are artifact-prone, particularly in less densely captured areas. As such, recovering a high-quality NeRF requires exhaustive scene capture, where each region is photographed from multiple angles multiple times.

NeRF's dense capture requirement poses a major challenge, necessitating tens to hundreds of images for even simple objects to ensure a clean reconstruction (Fig. 1). Many methods aim to reduce the reliance on dense captures by developing heuristic low-level regularizers based on reconstructed depth [11, 17, 40], visibility [27, 48], appearance [35, 60], or image-space frequencies [61]. However,

even the most effective methods show considerable degradation at novel viewpoints compared to denser captures.

With the recent success of diffusion models for image generation [21, 50], researchers have applied diffusion models to the task of novel view synthesis — modeling the distribution of unseen views given observations from known views [16, 29, 59]. While these models excel at generating realistic images from novel view points, they do not produce a single consistent 3D shape from a sparse set of input views. Existing work produces 3D models that are either trained per category [6, 16, 52, 62, 67], or are limited to single image inputs containing an object [29, 30, 47], preventing their use as a general prior for 3D scene reconstruction.

Our proposed method uses 2D image priors over novel views to enhance 3D reconstruction. We derive this prior from a diffusion model trained for novel view synthesis. Given multiple posed images of a scene, this model estimates the scene's appearance from novel viewpoints. As posed multiview data is limited (compared to massive single image datasets), we finetune our diffusion model from a pretrained latent diffusion model [42] on a mixture of real world and synthetic multiview image datasets: RealEstate10K [66], CO3D [38], MVImgNet [64], and Objaverse [10]. Once trained, this model is used to regularize a typical NeRF reconstruction pipeline by using an approach similar to score distillation sampling (SDS) [36].

Our approach outperforms existing baselines on several datasets of both forward-facing and unbounded 360° scenes. Furthermore, we show that our diffusion prior is an effective drop-in regularizer for NeRFs across a range of capture settings. In few-view scenarios with limited scene observations, it provides a strong prior for plausible geometry and appearance reconstruction. In denser capture settings, it helps reduce distracting "fog" and "floater" artifacts while preserving the appearance of well-sampled regions.

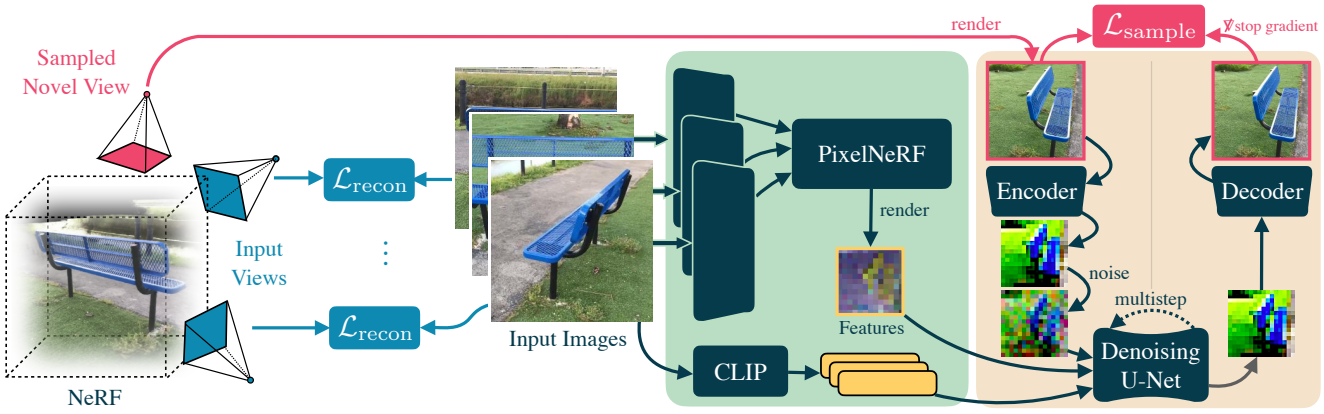
Many aspects of our pipeline have been explored in prior work. We contribute an end-to-end system that markedly improves 3D reconstruction quality, uniquely combining the challenges of developing a multiview-conditioned image diffusion model and integrating it into the NeRF optimization process, minimizing the need for rigorous capture.

2. Related Work

Few-view NeRF Minimizing NeRF's need for dense capture is crucial for democratizing 3D capture, and has motivated many works [11, 17, 22, 27, 35, 40, 46, 48, 49, 58, 60, 61]. Most existing methods focus on regularizing the geometry of the scene. DS-NeRF [11] utilizes sparse depth outputs from Structure-from-Motion (SfM) as supervision. DDP-NeRF [40] further uses a CNN to obtain dense depth supervision from sparse inputs. SimpleNeRF [49] regularizes appearance and geometry by training two additional models which respectively reduce positional encoding frequencies and remove view-dependent components. Similarly, FreeNeRF [61] demonstrates that simply regularizing the frequency range of NeRF's positional encoding features improves quality in few-view scenarios. RegNeRF [35] introduces a depth smoothness loss and a pre-trained normalizing flow color model to regularize the geometry and appearance of novel views. DiffusioNeRF [60] trains a diffusion model to regularize the distribution of RGB-D patches from perturbed viewpoints. GANeRF [41] trains a generator network to improve NeRF renderings and an image discriminator network to provide feedback that can be used to improve the reconstruction in a multiview-consistent manner. While all these methods regularize ambiguous geometry and appearance during NeRF optimization, they often fail on larger scenes when the view sparsity is extreme.

Regression models for view synthesis While NeRFs are optimized *per-scene*, other methods train feed-forward neural networks for generalized novel view synthesis. These networks leverage large collections of posed multiview data across many scenes [9, 12, 20, 43, 53, 55, 57, 63, 66]. Most methods lift the input images into a 3D representation, like using a plane sweep volume, and predict novel views in a feed-forward manner. They work well near the input views, but extrapolate poorly to ambiguous views, where the distribution of possible renderings becomes multi-modal.

Generative models for view synthesis Extrapolating beyond observed inputs for view synthesis requires generating unknown parts of the scene. Earlier works addressing this problem primarily leverage Generative Adversarial Networks (GANs) [4, 5, 13–15, 19, 33, 34, 45, 68]. More recent works use diffusion models, following their immense success on image generation [6, 16, 17, 24, 25, 29, 30, 39, 47, 52, 54, 62]. 3DiM [59] trains a pose-conditioned image-to-image diffusion model on synthetic ShapeNet data [7]. GeNVS [6] and SparseFusion [17] train on real-world multiview data [38] and further incorporate 3D geometry priors by conditioning on rendered features [51, 63]. While they show promising results for novel view synthesis, their models are category-specific and do not generalize to arbitrary scenes. Recently, Zero-1-to-3 [29] fine-tunes a large-scale pretrained diffusion model [42] on the synthetic Objaverse dataset [10] and achieves strong zero-shot generalization on real images. However, it only supports images of objects with clean backgrounds (versus full real scenes) and is limited to single-image inputs. ZeroNVS [44] further fine-tunes Zero-1-to-3 to enable single-image reconstruction of general scenes. Our approach is similar, but utilizes a PixelNeRF-based approach [63] for conditioning (similar to GeNVS [6]) to allow for any number of input images and provide more precise pose conditioning, and fine-tunes a pretrained image diffusion model on real-world multiview datasets [38, 64, 66], which combined facilitates few-view



(a) NeRF Training (b) Multi-View Conditioning (c) Diffusion Prior

Figure 2. (a) We optimize a NeRF to minimize a reconstruction loss $\mathcal{L}_{\text{recon}}$ between renderings and a limited set of input images, alongside a sample loss $\mathcal{L}_{\text{sample}}$ comparing renderings from random poses and with predictions by a diffusion model for those poses. (b) To generate the sample image, we use a PixelNeRF-style model [63] to fuse input image data, rendering a feature map for the sample view. (c) This feature map, merged with the noisy latent (computed by adding some amount of noise to the current NeRF rendering from that pose), is provided to a diffusion model. This model additionally uses CLIP embeddings of the input images via cross-attention, generating a decoded output sample. This sample is used to apply an image-space loss to the corresponding NeRF rendering.

reconstruction on arbitrary scenes.

Lifting 2D diffusion models for 3D generation Given the limited amount of 3D data available for training, recent works have attempted to leverage 2D diffusion models to generate 3D assets from a text prompt [28, 36, 56] or an input image [29, 47]. DreamFusion [36] proposed score distillation sampling (SDS), where a 2D diffusion model acts as a critic to supervise the optimization of a 3D model. SparseFusion [67] proposes multistep sampling where an image is sampled given a noisy encoding of the current rendering as a target for 3D reconstruction. We experiment with both approaches in our reconstruction pipeline.

3. ReconFusion

ReconFusion consists of a diffusion model trained for novel view synthesis and a 3D reconstruction procedure to make use of this diffusion model. We describe the details of the diffusion model training in Sec. 3.1 and how we use the diffusion model as a prior for 3D reconstruction in Sec. 3.2.

3.1. Diffusion Model for Novel View Synthesis

Given a set of posed images, we seek to learn a prior that can generate plausible novel views. If we can learn what the back or side of an object looks like given images of the front, we can use this to guide a 3D reconstruction process to recover a plausible 3D scene. Formally, we are given a set of input images $x^{\text{obs}} = \{x_i\}_{i=1}^N$, corresponding camera parameters $\pi^{\text{obs}} = \{\pi_i\}_{i=1}^N$, and a target camera for a novel view π, and want to learn the conditional distribution over the image x at the novel view: $p\big(x|x^{\text{obs}}, \pi^{\text{obs}}, \pi\big)$.

Diffusion Models We build on latent diffusion models (LDMs) [42] for their ability to efficiently model high resolution images. LDMs encode input images to a latent representation using a pretrained variational auto-encoder (VAE) $\mathcal{E}$. Diffusion is performed on these latents, where a denoising U-Net ϵ_θ maps noisy latents back to clean latents. During inference, this U-Net is used to iteratively denoise pure Gaussian noise to a clean latent. To recover an image, the latents are passed through a VAE decoder $\mathcal{D}$.

Conditioning Similar to Zero-1-to-3 [29], we start from an LDM trained for text-to-image generation, and additionally condition on input images and poses. Converting a text-to-image model into a posed images-to-image model requires augmenting the U-Net architecture with additional conditioning pathways. To modify the pretrained architecture for novel view synthesis from multiple posed images, we inject two new conditioning signals into the U-Net (see Fig. 2(b)). For high-level semantic information about the inputs, we use the CLIP [37] embedding of each input image (denoted e^{obs}) and feed this sequence of feature vectors into the U-Net via cross-attention. For relative camera pose and geometric information, we use a PixelNeRF [63] model R_ϕ to render a feature map f with the same spatial resolution as the latents from the target viewpoint π:

$$f = R_\phi\big(x^{\text{obs}}, \pi^{\text{obs}}, \pi\big)\,. \tag{1}$$

This rendered feature map f is a spatially aligned conditioning signal which implicitly encodes the relative camera transform. We concatenate f with the noisy latent along the channel dimension, and feed it into the denoising U-Net ϵ_θ. This feature map conditioning strategy is similar to

the one used in GeNVS [6], SparseFusion [67], and other recent works to better provide an accurate representation of the novel camera pose, as compared to attending over a direct embedding of the camera extrinsics and intrinsics themselves (an ablation study can be found in Sec. 4.3).

Training We freeze the weights of the pretrained encoder and decoder, initialize the U-Net parameters θ from pretrained weights, and optimize the modified architecture for view synthesis using the simplified diffusion loss [21]:

$$\mathcal{L}_{\text{Diff}}(\theta, \phi) = \mathbb{E}_{x,\pi,x^{\text{obs}},\pi^{\text{obs}},\epsilon,t} \left\| \epsilon - \epsilon_\theta(z_t, t, e^{\text{obs}}, f) \right\|^2, \quad (2)$$

where $t \in \{1, \dots, T\}$ is the diffusion timestep, $\epsilon \sim \mathcal{N}(0, I)$, $z_t = \alpha_t \mathcal{E}(x) + \sigma_t \epsilon$ is the noisy latent at that timestep, e^{obs} are the CLIP image embeddings for the input images x^{obs}, and f is the rendered feature map from PixelNeRF R_ϕ. In addition to the loss in Eqn. 2, we optimize the PixelNeRF parameters ϕ with a photometric loss:

$$\mathcal{L}_{\text{PixelNeRF}}(\phi) = \mathbb{E}_{x^{\text{obs}},\pi^{\text{obs}},x,\pi} \|c - x_\downarrow\|^2, \quad (3)$$

where c is an output of the PixelNeRF model (at the same resolution as the feature map f) and $x_\downarrow$ is the target image downsampled to the spatial resolution of z_t and f. This loss encourages PixelNeRF to reconstruct the RGB target image, which helps to avoid bad local minima where the diffusion model is unable to leverage the PixelNeRF inputs.

Due to the use of cross-attention and the design of PixelNeRF, both conditioning branches can take an arbitrary number and permutation of input images. This enables the model to be trained and evaluated with a variable number of observed posed images. While there are many ways to condition on images and poses, we found our design more effective than alternatives (see the ablation in Sec. 4.3).

3.2. 3D Reconstruction with Diffusion Priors

The trained diffusion model produces plausible single images for novel camera poses, but generated images are often inconsistent for different poses or random seeds. State-of-the-art NeRF methods produce 3D consistent 3D models, but often exhibit volumetric "floater" artifacts and inaccurate (or totally unrecognizable) geometry from novel views. To enable 3D reconstruction from a smaller number of posed inputs, we augment the state-of-the-art 3D reconstruction pipeline from Zip-NeRF [2] with a prior from our diffusion model trained for novel view synthesis.

Reconstruction loss NeRF-based methods optimize a randomly initialized 3D model to match a set of posed images. The NeRF parameters ψ are optimized by minimizing the reconstruction error between a rendered image $x = x(\psi, \pi^{\text{obs}})$ and an observed image x^{obs} at pose π^{obs}:

$$\mathcal{L}_{\text{Recon}}(\psi) = \mathbb{E}_{x^{\text{obs}},\pi^{\text{obs}}} \left[\ell(x(\psi, \pi^{\text{obs}}), x^{\text{obs}}) \right], \quad (4)$$

where ℓ is an image similarity loss function such as the ℓ_2-norm or a robust loss. This loss is only evaluated where we have observations, and thus the training procedure never views the 3D model from novel views.

Diffusion loss In addition, we seek to optimize the 3D model to produce realistic rendering at novel views unobserved in the inputs. To do so, we use a diffusion model which provides a prior on the distribution of plausible images of the scene. We distill this prior into a consistent 3D model by using a regularization loss derived from the diffusion model outputs. We experimented with several losses and discuss our multistep sampling approach below.

At each iteration, we sample a random view and generate an image from the diffusion model to produce a target image (see Fig. 2(a)). We can control how grounded the target image is to the current rendered image by starting the sampling process from an intermediate noise level. Specifically, we render an image $x(\psi, \pi)$ from a sampled novel viewpoint π, and encode and perturb it to a noisy latent z_t with noise level $t \sim \mathcal{U}[t_{\min}, t_{\max}]$. We then generate a sample from the latent diffusion model by running DDIM sampling [50] for k intermediate steps, uniformly spaced between the smallest noise level and t, yielding a latent sample z_0. This latent is decoded to produce a target image $\hat{x}_\pi = \mathcal{D}(z_0)$, which we use to supervise the rendering:

$$\mathcal{L}_{\text{sample}}(\psi) = \mathbb{E}_{\pi,t} \left[w(t) \left(\|x - \hat{x}_\pi\|_1 + \mathcal{L}_{\text{p}}(x, \hat{x}_\pi) \right) \right], \quad (5)$$

where $\mathcal{L}_{\text{p}}$ is the perceptual distance LPIPS [65], and $w(t)$ is a noise-level dependent weighting function. This diffusion loss is most similar to SparseFusion [67], and resembles the iterative dataset update strategy of Instruct-NeRF2NeRF [18], except we sample a new image at each iteration. We empirically found this approach to work better than score distillation sampling [36] (see supp. and Fig. 5).

Novel view selection Which views should we sample when using our diffusion prior? We do not want to place novel views inside of objects or behind walls, and the placement of views often depends on the scene content and type of capture. As in prior work such as RegNeRF [35], we wish to define a distribution based on the known input poses and capture pattern that will encompass a reasonable set of novel camera poses, roughly matching the positions from which we would expect to observe the reconstructed scene.

We achieve this by determining a base set or path of poses through the scene, which we can randomly sample and perturb to define a full pose distribution for novel views. In forward-facing captures such as LLFF [31] and DTU [23] or 360-degree captures such as mip-NeRF 360 [1], we define an elliptical path fit to the training views, facing toward the focus point (the point with minimum average distance to the training cameras' focal axes). In more unstructured captures such as CO3D [38] and RealEstate10K [66], we fit a B-spline to roughly follow the trajectory of the training

views. In either case, for each random novel view, we uniformly select one of the poses in the path and then perturb its position, up vector, and look-at point within some range. Please see the supplement for additional details.

3.3. Implementation Details

Our base diffusion model is a re-implementation of the Latent Diffusion Model [42] that has been trained on an internal dataset of image-text pairs with input resolution $512\times512\times3$ and a latent space with dimensions $64\times64\times8$. The encoder of our PixelNeRF is a small U-Net that takes as input an image of resolution 512×512 and outputs a feature map of resolution 64×64 with 128 channels (see the supplement for more details). We jointly train the PixelNeRF and finetune the denoising U-Net with batch size 256 and learning rate 10^{-4} for a total of $250k$ iterations. To enable classifier-free guidance (CFG), we set the input images to all zeros randomly with probability 10%.

We use Zip-NeRF [2] as our backbone and train the NeRF for a total of 1000 iterations. The reconstruction term $\mathcal{L}_{\text{recon}}$ uses the Charbonnier loss [8] as in Zip-NeRF. The weighting for $\mathcal{L}_{\text{sample}}$ is linearly decayed from 1 to 0.1 over training, and the classifier-free guidance scale used for sampling is set to 3.0. We fix $t_{\max}=1.0$ for all training steps, and linearly anneal $t_{\min}$ from 1.0 to 0.0. Regardless of t, we always sample the denoised image with $k=10$ steps. In practice, diffusion models for view synthesis can be conditioned on a small number of observed input images and poses. Given a target novel view, we select the 3 nearest camera positions from the observed inputs to condition the model. This enables our models to scale to large numbers of input images while selecting inputs that are most useful for the sampled novel view. Please refer to the supplementary materials for more implementation details.

4. Experiments

We evaluate ReconFusion on five real-world datasets to demonstrate the performance and generalizability of our approach for few-view 3D reconstruction (Sec. 4.2). We also perform several ablations on the components of the diffusion model and the 3D reconstruction procedure (Sec. 4.3). Finally, we demonstrate that our method improves NeRF reconstruction across a range of capture settings (Sec. 4.4). Additionally, we strongly advise the reader to view our supplementary video, as the visual difference in view synthesis results is significantly clearer in video than in still images.

4.1. Experiment Setup

Training Dataset To learn a generalizable diffusion prior for novel view synthesis, we train on a mixture of the synthetic Objaverse [10] dataset and three real-world datasets: CO3D [38], MVImgNet [64], and RealEstate10K [66]. For Objaverse, we render each 3D asset from 16 randomly sampled views at resolution 512×512 and composite the rendering onto a randomly selected solid color background. For the other three real-world captured datasets, we center crop and resize each frame to 512×512. For training, we sample 3 frames of the same scene as input views and sample another frame as the target view. Please refer to the supplementary materials for details about dataset mixing.

Evaluation Dataset We evaluate our method on scenes from multiple datasets with 3, 6 and 9 input views, which include in-distribution datasets (CO3D [38] and RealEstate10K [66]) and out-of-distribution datasets (LLFF [31], DTU [23] and mip-NeRF 360 [1]). LLFF and DTU are datasets of forward-facing scenes, where we adhere to the evaluation protocol of RegNeRF [35]. For the real-world object-centric scenes from CO3D we evaluate on a subset of 20 scenes from 10 categories. RealEstate10K contains video clips gathered from YouTube, and we sample 10 scenes (each with 100 frames) from its test set for evaluation. The mip-NeRF 360 dataset has 9 indoor or outdoor scenes each containing a complex central object or area and a detailed background. For CO3D and RealEstate10K, we select the input views evenly from all the frames and use every 8th of the remaining frames for evaluation. For the mip-NeRF 360 dataset, we retain its original test set and select the input views from the training set using a heuristic to encourage reasonable camera spacing and coverage of the central object (see supplement for details).

Baselines For evaluation datasets, we compare against the state-of-the-art dense-view NeRF model Zip-NeRF [2] (which is also the reconstruction pipeline used in our model), and state-of-the-art few-view NeRF regularization methods including DiffusioNeRF [60], FreeNeRF [61], and SimpleNeRF [49]. We also compare to ZeroNVS [44], concurrent work on novel view synthesis of scenes from a single image. To adapt it to multiview inputs, we use the input view closest to the sampled view as its input condition and denote this method as ZeroNVS*. On the CO3D dataset we additionally compare to SparseFusion [67]. Following their setup, we train the SparseFusion model in a category-specific manner. However, unlike the original implementation, which masks out the foreground object and sidesteps the difficulty of recovering the background of the scene, we use the whole unmasked image. Please refer to the supplement for more details about baselines.

4.2. Comparison Results

We report the quantitative results in Table 1 and show qualitative comparisons in Fig. 3. Please see the supplementary video for more visuals. Our backbone NeRF model, Zip-NeRF, often overfits to the input views and exhibits artifacts like "foggy" geometry or "floaters". State-of-the-art few-view NeRF regularization methods (DiffusioNeRF, FreeN-

Figure 3. A visual comparison of rendered images and depth maps on scenes from the RealEstate10K [66], LLFF [31], DTU [23], CO3D [38], and mip-NeRF 360 [1] datasets (input view count indicated in parentheses). Both the appearance and geometry of our method are of higher quality than the baselines in these examples–typical failure modes exhibited by the baselines include "floater" artifacts visible in depth maps, color artifacts or blurry low-fidelity geometry in minimally observed regions of the scenes, correct texture appearing in incorrect locations in the image, and so on. We encourage the reader to watch our supplementary video, as many of these differences are easier to identify with a moving camera trajectory.

Dataset Difficulty		Method	PSNR ↑ 3-view	PSNR ↑ 6-view	PSNR ↑ 9-view	SSIM ↑ 3-view	SSIM ↑ 6-view	SSIM ↑ 9-view	LPIPS ↓ 3-view	LPIPS ↓ 6-view	LPIPS ↓ 9-view
Easier	RealEstate10K	Zip-NeRF*	20.77	27.34	31.56	0.774	0.906	0.947	0.332	0.180	0.118
		DiffusioNeRF	19.12	24.18	27.78	0.710	0.808	0.869	0.444	0.344	0.282
		FreeNeRF	20.54	25.63	27.32	0.731	0.817	0.843	0.394	0.344	0.332
		SimpleNeRF	23.89	28.75	29.55	0.839	0.896	0.900	0.292	0.239	0.236
		ZeroNVS*	19.11	22.54	23.73	0.675	0.744	0.766	0.422	0.374	0.358
		Ours	25.84	29.99	31.82	0.910	0.951	0.961	0.144	0.103	0.092
	LLFF	Zip-NeRF*	17.23	20.71	23.63	0.574	0.764	0.830	0.373	0.221	0.166
		RegNeRF	19.08	23.09	24.84	0.587	0.760	0.820	0.374	0.243	0.196
		DiffusioNeRF	20.13	23.60	24.62	0.631	0.775	0.807	0.344	0.235	0.216
		FreeNeRF	19.63	23.72	25.12	0.613	0.773	0.820	0.347	0.232	0.193
		SimpleNeRF	19.24	23.05	23.98	0.623	0.737	0.762	0.375	0.296	0.286
		ZeroNVS*	15.91	18.39	18.79	0.359	0.449	0.470	0.512	0.438	0.416
		Ours	21.34	24.25	25.21	0.724	0.815	0.848	0.203	0.152	0.134
	DTU	Zip-NeRF*	9.18	8.84	9.23	0.601	0.589	0.592	0.383	0.370	0.364
		RegNeRF	19.39	22.24	24.62	0.777	0.850	0.886	0.203	0.135	0.106
		DiffusioNeRF	16.14	20.12	24.31	0.731	0.834	0.888	0.221	0.150	0.111
		FreeNeRF	20.46	23.48	25.56	0.826	0.870	0.902	0.173	0.131	0.102
		SimpleNeRF	16.25	20.60	22.75	0.751	0.828	0.856	0.249	0.190	0.176
		ZeroNVS*	16.71	17.70	17.92	0.716	0.737	0.745	0.223	0.205	0.200
		Ours	20.74	23.62	24.62	0.875	0.904	0.921	0.124	0.105	0.094
	CO3D	Zip-NeRF*	14.34	14.48	14.97	0.496	0.497	0.514	0.652	0.617	0.590
		DiffusioNeRF	15.65	18.05	19.69	0.575	0.603	0.631	0.597	0.544	0.500
		FreeNeRF	13.28	15.20	17.35	0.461	0.523	0.575	0.634	0.596	0.561
		SimpleNeRF	15.40	18.12	20.52	0.553	0.622	0.672	0.612	0.541	0.493
		SparseFusion	16.76	18.77	19.13	0.561	0.600	0.604	0.695	0.653	0.651
		ZeroNVS*	17.13	19.72	20.50	0.581	0.627	0.640	0.566	0.515	0.500
		Ours	19.59	21.84	22.95	0.662	0.714	0.736	0.398	0.342	0.318
Harder	mip-NeRF 360	Zip-NeRF*	12.77	13.61	14.30	0.271	0.284	0.312	0.705	0.663	0.633
		DiffusioNeRF	11.05	12.55	13.37	0.189	0.255	0.267	0.735	0.692	0.680
		FreeNeRF	12.87	13.35	14.59	0.260	0.283	0.319	0.715	0.717	0.695
		SimpleNeRF	13.27	13.67	15.15	0.283	0.312	0.354	0.741	0.721	0.676
		ZeroNVS*	14.44	15.51	15.99	0.316	0.337	0.350	0.680	0.663	0.655
		Ours	15.50	16.93	18.19	0.358	0.401	0.432	0.585	0.544	0.511

Table 1. Quantitative evaluation of few-view 3D reconstruction methods. Datasets are ordered in terms of sparsity from easier (novel views are close to observed views) to harder (novel views are far from observed views). Our method is the first to be evaluated on such a wide range of few-view real datasets. Despite this generality, we outperform all baselines across all domains. Baselines that we additionally tuned for the task of few-view reconstruction are indicated with *.

eRF and SimpleNeRF) are able to significantly improve the baseline quality on forward-facing scenes like LLFF and DTU. However, they fall short on 360-degree scenes (*e.g.* the CO3D dataset), where a large portion of the scene is undersampled or even unobserved due to the a much larger relative disparity between input views. On such scenes, ZeroNVS serves as a strong baseline as it often can reconstruct a complete 3D scene, but with limited visual fidelity. Our method outperforms all baselines on both in-distribution and out-of-distribution datasets, achieving state-of-the-art performance for few-view NeRF reconstructions.

4.3. Ablation Studies

In Table 2 and Fig. 4, we ablate two aspects of our diffusion model: the use of pretrained diffusion model weights (PT) and conditioning signal. To ablate "PT," we train the diffusion model from scratch. To ablate conditioning, in the `pose` experiment we replace our PixelNeRF module with a conditioning mechanism similar to ZeroNVS [44] (which itself extends Zero-1-to-3 [29]). This alternative simply concatenates the latents of all input images to the U-Net input, and represents the relative camera transforms as vectors (relative translation and rotation quaternion) which are concatenated with the CLIP embeddings. Both variants produce sampled images of lower quality, which subsequently

	PT	Condition	NeRF renders PSNR↑	SSIM↑	LPIPS↓	Diffusion samples PSNR↑	SSIM↑	LPIPS↓
In-domain	✓	`pose`	20.57	0.749	0.367	15.11	0.546	0.484
		`pixelnerf`	25.15	0.815	0.246	22.40	0.723	0.314
	✓	`pixelnerf`	25.34	0.823	0.232	24.05	0.751	0.281
Out-domain	✓	`pose`	17.28	0.521	0.458	12.18	0.244	0.599
		`pixelnerf`	19.82	0.580	0.383	16.46	0.420	0.452
	✓	`pixelnerf`	20.23	0.596	0.355	17.44	0.464	0.411

Table 2. We ablate two aspects of our model: pretrained diffusion weights (PT) and conditioning. For PT, we initialize the diffusion model weights from a pretrained text-to-image model. `pose` uses a pose conditioning similar to ZeroNVS [44] while `pixelnerf` uses our PixelNeRF conditioning (Sec. 3.1). We evaluate our model on both *in-domain* datasets (RealEstate10K, CO3D) and *out-of domain* datasets (LLFF, mip-NeRF 360).

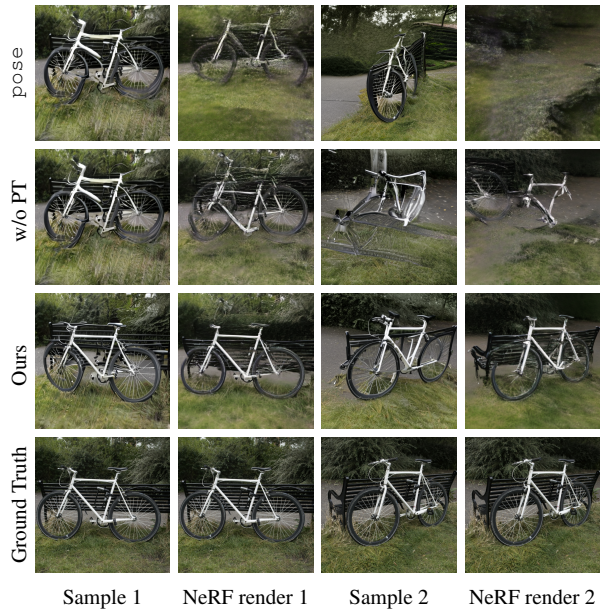


Figure 4. **Ablation of diffusion model on 3-view reconstruction.** We show two samples from the diffusion models, and renderings from the reconstructed NeRFs under the same viewpoints for three variants of the diffusion model: `pose`, without pretraining, and our full model. The samples from nearby poses are inconsistent due to randomness, but can be successfully reconciled into an underlying NeRF reconstruction.

degrades the NeRF reconstruction.

In Fig. 5, we ablate the choice of diffusion loss and find standard SDS results contain more artifacts, and the multistep diffusion loss effectively mitigates these artifacts. Additionally, annealing t_{min} leads to more details.

4.4. Scaling to More Views

To further investigate the effectiveness and robustness of our diffusion prior, we evaluate our method and the backbone Zip-NeRF under various numbers of input views. As

Figure 5. Comparing diffusion losses for 3D reconstruction. Note the "blotchy" texture on the placemat and background chair when using SDS, and improved background detail with annealing.

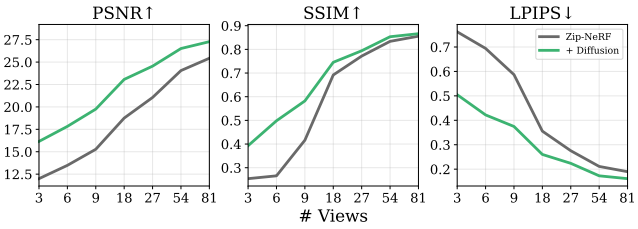


Figure 6. Our learned diffusion prior improves performance over the Zip-NeRF baseline up to as many as 81 input views on the `kitchenlego` scene from the mip-NeRF 360 dataset. Though most results in this paper focus on the challenging case of 3-9 input views, achieving a high-quality reconstruction of a real-world scene often requires significantly more images–this plot demonstrates that our diffusion prior reduces this requirement at all points along the capture density vs. quality curve.

the number of views increases, the input captures provide better coverage of the entire scene, resulting in less ambiguity. Therefore, we set the weighting factor for our diffusion loss ($\mathcal{L}_{\text{sample}}$) to be inversely proportional to the number of input views in this case. As shown in Fig. 6, our test set performance is consistently better than Zip-NeRF, indicating that our diffusion prior can serve as an effective drop-in regularizer across a range of capture settings.

5. Discussion

The goal of ReconFusion is to demonstrate the potential in piecing together two powerful building blocks. First, a state-of-the-art optimization-based 3D reconstruction pipeline, with an underlying 3D representation guaranteeing multiview consistency. And second, a powerful multiview-conditioned image diffusion model for generating plausible novel views, which can be used to guide reconstruction to avoid the artifacts resulting from an underconstrained inverse problem. Many current limitations are evident: the heavyweight diffusion model is costly and slows down reconstruction significantly; our current results demonstrate only limited 3D outpainting abilities compared to what our image model can hallucinate in 2D; tuning the balance of reconstruction and sample losses is tedious; etc.

However, this initial attempt at building such a system has already produced compelling results across a variety of scene types with significantly reduced view counts. We are

optimistic that it may possibly serve as a template for improvements in sparse reconstruction, as we move toward a future of ever more accessible 3D reconstruction techniques with dramatically reduced capture requirements.

Acknowledgements We would like to thank Arthur Brussee, Ricardo Martin-Brualla, Rick Szeliski, Peter Hedman, and Jason Baldridge for their valuable contributions in discussing the project and reviewing the manuscript, and Zhicheng Wang for setting up some of the data loaders necessary for our diffusion model training pipeline. We are grateful to Randy Persaud and Henna Nandwani for infrastructure support.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR*, 2022. 4, 5, 6

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 1, 4, 5, 13

[3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. InstructPix2Pix: Learning to Follow Image Editing Instructions. *CVPR*, 2023. 11

[4] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *CVPR*, 2021. 2

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. *CVPR*, 2022. 2

[6] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. GeNVS: Generative novel view synthesis with 3D-aware diffusion models. *arXiv*, 2023. 2, 4

[7] 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*, 2015. 2

[8] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. *ICIP*, 1994. 5

[9] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. *ICCV*, 2021. 2

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A Universe of Annotated 3D Objects. *CVPR*, 2023. 2, 5

[11] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free. *CVPR*, 2022. 1, 2

[12] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. *CVPR*, 2016. 2

[13] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. *3DV*, 2017. 2

[14] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *NeurIPS*, 2022.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 2014. 2

[16] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. NerfDiff: Single-image View Synthesis with NeRF-guided Distillation from 3D-aware Diffusion. *ICML*, 2023. 2

[17] Guangcong, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. SparseNeRF: Distilling Depth Ranking for Few-shot Novel View Synthesis. *ICCV*, 2023. 1, 2

[18] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. *ICCV*, 2023. 4

[19] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. *ICCV*, 2019. 2

[20] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised Learning of 3D Object Categories From Videos in the Wild. *ICCV*, 2021. 2

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *NeurIPS*, 2020. 2, 4

[22] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. *ICCV*, 2021. 2

[23] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large Scale Multi-view Stereopsis Evaluation. *CVPR*, 2014. 4, 5, 6

[24] Animesh Karnewar, Niloy J. Mitra, Andrea Vedaldi, and David Novotny. Holofusion: Towards photo-realistic 3d generative modeling. *ICCV*, 2023. 2

[25] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J. Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. *CVPR*, 2023. 2

[26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *SIGGRAPH*, 2023. 1

[27] Minseop Kwak, Jiuhn Song, and Seungryong Kim. GeCoNeRF: Few-shot Neural Radiance Fields via Geometric Consistency. *ICML*, 2023. 1, 2

[28] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation. *CVPR*, 2023. 3

[29] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-Shot One Image to 3D Object. *arXiv*, 2023. 2, 3, 7, 11

[30] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. SyncDreamer: Generating Multiview-consistent Images from a Single-view Image. *arXiv*, 2023. 2

[31] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *SIGGRAPH*, 2019. 4, 5, 6

[32] 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. *ECCV*, 2020. 1

[33] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. *ICCV*, 2019. 2

[34] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *CVPR*, 2021. 2

[35] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs. *CVPR*, 2022. 1, 2, 4, 5, 13

[36] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. *ICLR*, 2022. 2, 3, 4

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. *ICML*, 2021. 3

[38] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction. *ICCV*, 2021. 2, 4, 5, 6

[39] Xuanchi Ren and Xiaolong Wang. Look Outside the Room: Synthesizing A Consistent Long-Term 3D Scene Video from A Single Image. *CVPR*, 2022. 2

[40] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense Depth Priors for Neural Radiance Fields from Sparse Input Views. *CVPR*, 2022. 1, 2

[41] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. GANeRF: Leveraging Discriminators to Optimize Neural Radiance Fields. *arXiv preprint arXiv:2306.06044*, 2023. 2

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *CVPR*, 2022. 2, 3, 5

[43] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. *CVPR*, 2022. 2

[44] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, et al. ZeroNVS: Zero-Shot 360-Degree View Synthesis from a Single Real Image. *arXiv:2310.17994*, 2023. 2, 5, 7, 8, 13

[45] Katja Schwarz, Axel Sauer, Michael Niemeyer, Yiyi Liao, and Andreas Geiger. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. *NeurIPS*, 2022. 2

[46] Seunghyeon Seo, Donghoon Han, Yeonjin Chang, and Nojun Kwak. MixNeRF: Modeling a Ray with Mixture Density for Novel View Synthesis from Sparse Inputs. *CVPR*, 2023. 2

[47] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view Diffusion for 3D Generation. *arXiv*, 2023. 2, 3

[48] Nagabhushan Somraj and Rajiv Soundararajan. ViP-NeRF: Visibility Prior for Sparse Input Neural Radiance Fields. *SIGGRAPH*, 2023. 1, 2

[49] Nagabhushan Somraj, Adithyan Karanayil, and Rajiv Soundararajan. SimpleNeRF: Regularizing Sparse Input Neural Radiance Fields with Simpler Solutions. *SIGGRAPH Asia*, 2023. 2, 5, 13

[50] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. *ICLR*, 2020. 2, 4

[51] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. *ECCV*, 2022. 2

[52] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B Tenenbaum, Frédo Durand, William T Freeman, and Vincent Sitzmann. Diffusion with Forward Models: Solving Stochastic Inverse Problems Without Direct Supervision. *arXiv*, 2023. 2

[53] Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Representation and Rendering. *ICCV*, 2021. 2

[54] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. Consistent View Synthesis with Pose-Guided Diffusion Models. *CVPR*, 2023. 2

[55] Richard Tucker and Noah Snavely. Single-View View Synthesis With Multiplane Images. *CVPR*, 2020. 2

[56] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. *CVPR*, 2023. 3

[57] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering. *CVPR*, 2021. 2

[58] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs. *ICCV*, 2023. 2

[59] Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel View Synthesis with Diffusion Models. *ICLR*, 2022. 2

[60] Jamie Wynn and Daniyar Turmukhambetov. DiffusioNeRF: Regularizing neural radiance fields with denoising diffusion models. *CVPR*, 2023. 1, 2, 5, 13

[61] Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization. *CVPR*, 2023. 1, 2, 5, 13

[62] Paul Yoo, Jiaxian Guo, Yutaka Matsuo, and Shixiang Shane Gu. DreamSparse: Escaping from Plato's Cave with 2D Diffusion Model Given Sparse Views. *arXiv*, 2023. 2

[63] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields from One or Few Images. *CVPR*, 2021. 2, 3, 11

[64] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Tianyou Liang, Guanying Chen, Shuguang Cui, and Xiaoguang Han. MVImgNet: A Large-scale Dataset of Multi-view Images. *CVPR*, 2023. 2, 5

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CVPR*, 2018. 4

[66] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis using Multiplane Images. *SIGGRAPH*, 2018. 2, 4, 5, 6

[67] Zhizhuo Zhou and Shubham Tulsiani. SparseFusion: Distilling View-conditioned Diffusion for 3D Reconstruction. *CVPR*, 2023. 2, 3, 4, 5, 13

[68] 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. *NeurIPS*, 2018. 2

A. Diffusion Model Details

Our diffusion model is adapted from a pre-trained text-to-image latent diffusion model that maps $512 \times 512 \times 3$ inputs into a latent dimension of $64 \times 64 \times 8$. We modify this initial model to accept the necessary conditioning signals for text-free novel-view synthesis. We replace the inputs to the cross-attention pathway (which typically consist of a sequence of CLIP text embeddings) with the outputs of an additional dense layer. The input to this dense layer is a concatenated tensor consisting of (1) the unconditional CLIP text embedding (*i.e*. the empty string `""`), and (2) the CLIP image embeddings of each of the input conditioning frames. We initialize the weights of this dense layer such that it produces the unconditional CLIP text embedding at the start of fine-tuning. This mechanism is inspired by the fine-tuning process in Zero-1-to-3 [29]. Zero-1-to-3 fine-tunes from an *image variations* base model that has been previously fine-tuned to enable conditioning on CLIP image embeddings. We fine-tune directly from a text-conditioned model, but our architecture can learn image variation-like behavior through the dense layer. Furthermore, unlike Zero-1-to-3, our dense layer does not take pose as input, since the 3D transformation between the input conditioning frames and the target frame is applied through the PixelNeRF rendering process. In addition to the cross-attention modifications, we concatenate the outputs of the PixelNeRF model (a $64 \times 64 \times 131$ tensor consisting of RGB and features) to the input noise that is passed to the U-Net. As in prior work [3], we initialize the additional convolutional weights to zero such that the added inputs have no effect at the start of fine-tuning.

To enable classifier-free guidance on our added conditioning signals, we drop out all conditioning images for a training example with 10% probability. We drop out the CLIP and PixelNeRF conditioning pathways independently in order to enable separate guidance, although we found empirically that using the same guidance weight across both conditioning signals (*i.e*. performing joint CFG across both conditioning signals) produces optimal results. We train our model for $250{,}000$ iterations with a learning rate of 10^{-4} and a batch size of 128. Our training examples consist of 3 input conditioning images, 1 target image, and the corresponding relative poses between each input image and the target image. This data is sampled from CO3D, RealEstate10k, MVImgNet, and Objaverse with uniform probability. For Objaverse, we light the object with random environment maps and compose it onto a random solid background at each training iteration.

B. PixelNeRF Details

Our PixelNeRF module is inspired by but not identical to the architecture proposed in the original work [63]. The inputs are N images along with their camera poses (extrinsic and intrinsic matrices), along with a target camera pose. The output is an approximate rendering at the target camera pose (both RGB and feature channels), which is concatenated with the input to the diffusion U-Net to provide a strong conditioning signal that encodes the pose and image content of the target novel view. During inference, we typically resize (but do not crop) the inputs to PixelNeRF such that their shorter dimension is 512 pixels, since the model was trained on 512×512 resolution inputs. The output target image is always $64 \times 64 \times 131$ (3 RGB channels plus 128 feature channels), to match the latent resolution of the diffusion model.

The PixelNeRF module begins by passing all input images through a 2D U-Net to create feature images of equal spatial resolution with 128 channels. We then cast rays through each pixel of the target image, and sample 128 points along each ray from depth 0.5 to ∞ (uniform up to distance 1, then linear in disparity). We reproject these points into each of the input cameras and gather corresponding features from the feature images to make a gathered tensor of size $64 \times 64 \times N \times 128$. We append positionally-encoded 3D locations, as well as the mean and variance

(a) 3 input views (b) 6 input views (c) 9 input views (d) Ground truth

Figure 7. **More training views leads to better samples**. Here we show samples from the diffusion model at a novel viewpoint while varying the number of training images. In all cases, the diffusion model is given the three nearest images to the novel viewpoint. We see that as we increase the number of known scene observations, the expected distance to the nearest training view decreases, therefore increasing the fidelity of diffusion model samples.

(a) PixelNeRF RGB output (b) Diffusion model sample (c) PixelNeRF RGB output (d) Diffusion model sample

Figure 8. **PixelNeRF Visualization**. Here we show (a,c) a visualization of the 64×64 RGB component of the PixelNeRF output, and (b,d) the corresponding sample from the diffusion model, which is conditioned on the PixelNeRF outputs.

of these features over the N-long dimension corresponding to the number of inputs. A small MLP then processes this full tensor along the channel dimension to output a new set of features and weights. These weights are then used to compute a weighted sum along the N-long dimension, thereby producing a new tensor of size $64 \times 64 \times 128$. A second MLP then processes this summed tensor along the channel dimension to produce the final output of size $64 \times 64 \times (3 + 128)$.

All learned components (including the 2D U-Net used to extract image features) are initialized randomly and optimized jointly with the fine-tuned diffusion model U-Net. As mentioned in the main text, we apply an RGB reconstruction loss to encourage the PixelNeRF module to learn a useful conditioning signal. See Figure 8 for a visualization of our PixelNeRF model.

C. Dataset Details

For LLFF and DTU, we use the standard train/test splits proposed by earlier works. For RealEstate10k and CO3D, we select the training views evenly from all the frames and use every 8th of the remaining frames for evaluation. For the mip-NeRF 360 dataset we design a heuristic to choose a train split of views that are uniformly distributed around the hemisphere and pointed toward the central object of interest: We randomly sample 10^6 different 9-view splits and use the one that minimizes these heuristic losses, then further choose the 6- and 3-view splits to be subsets of the 9-view split.

We carefully rescale each dataset to be compatible with the near plane of 0.5 expected by the PixelNeRF module. DTU, CO3D, mip-NeRF 360 are rescaled by setting the "focus point" of the data to the origin and rescaling camera positions to fit inside a $[-1, 1]^3$ cube. RealEstate10k is pre-scaled by its creators to have a reasonable near distance of 1.0, so we simply multiply its camera positions by 0.5. LLFF similarly provides a near bound based on the COLMAP point cloud, which we use to rescale the data to allow a 0.5 near plane.

D. Baselines Details

Our Zip-NeRF [2] baseline has slight hyperparameter modifications from the original that better suit few-view reconstruction. This was done primarily to provide a maximally competitive baseline for our model, but these same hyperparameters are used by our model as well, and we observe a modest performance improvement due to them. In particular, we use:

- Distortion loss with weight 0.01,
- Normalized weight decay on the NGP grid parameters with strength 0.1,
- A smaller view-dependence network with width 32 and depth 1, to avoid overfitting,
- No hexagonal spiral control points, to accelerate rendering at the cost of introducing some aliasing,
- A downweighted density in the "contracted" region of space outside of the unit sphere, wherein we multiply the density emitted by Zip-NeRF by $|\det(\mathbf{J}_{\mathcal{C}}(\mathbf{x}))|$ (the isotropic scaling induced by the contraction function, see the supplement of Barron et al. [2]).

We find that this baseline performs competitively on forward-facing scenes such as LLFF and RealEstate10k, especially with 9 input views, but often produces many floaters or fails to reconstruct any meaningful geometry on the more difficult datasets. Further tuning and the addition of other heuristic regularizers (*e.g.*, the techniques used in RegNeRF or FreeNeRF) would likely improve results. However, the point of this model is to show baseline reconstruction performance with our diffusion model regularizer disabled, rather than to be a state-of-the-art few-view method. To re-emphasize this: the **only** difference between results labeled "Ours" and "Zip-NeRF" is that the diffusion regularizer weight is set to 0, all other hyperparameters are identical.

For RegNeRF [35] and FreeNeRF [61], we use the result images shared by the authors for the LLFF and DTU datasets, and run the authors' code for FreeNeRF on the other three datasets. For DiffusioNeRF [60], we use the authors' result images on the DTU dataset, and run their code on the other four datasets. Because SimpleNeRF [49] used different train/test splits for LLFF and DTU, we run the authors' code on all five datasets.

SparseFusion [67] was originally trained on CO3D using masked images of foreground objects. We re-train their models using the unmasked images in a category-specific manner, then use their 3D distillation pipeline to obtain the final rendered images.

ZeroNVS [44] is a concurrent work that trains a diffusion model for novel view synthesis of scenes from a single image. To evaluate its performance on multiview inputs, we modify its reconstruction pipeline by using the input view closest to the sampled random view for conditioning the diffusion model. For DTU and mip-NeRF 360 scenes (which they evaluate in their paper for the single input case), we follow their viewpoint selection strategy for sampling new views. For CO3D, we use the same strategy that is employed for the mip-NeRF 360 scenes. For RealEstate10K, we sample new views on a spline path fitted from the input views, then perturb them. For LLFF, we sample new views on a circle fitted from the input views, then perturb them.