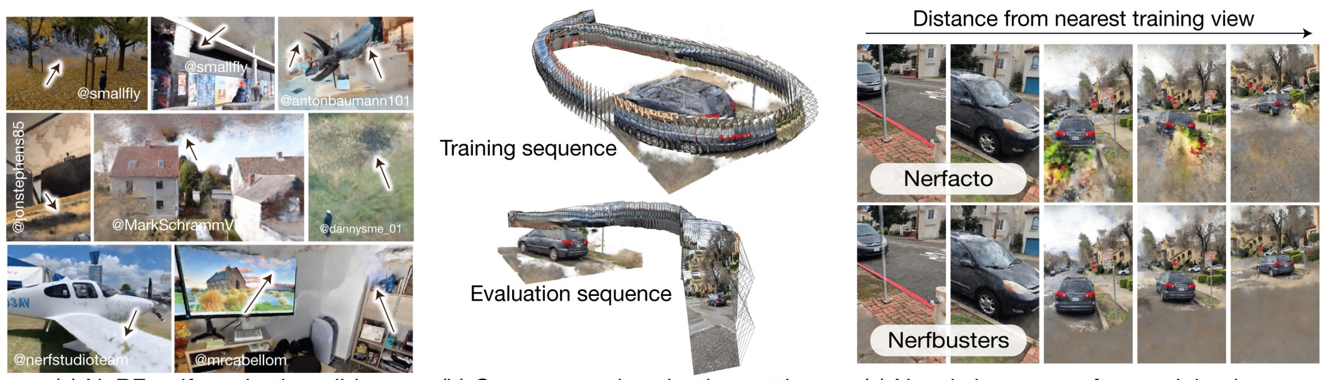
Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs

Frederik Warburg* Ethan Weber* Matthew Tancik
Aleksander Holynski Angjoo Kanazawa

University of California, Berkeley



(a) NeRF artifacts in-the-wild (b) Our proposed evaluation setting (c) Novel views away from training images

Figure 1: **Nerfbusters.** Rendering NeRFs at novel views far away from training views can result in artifacts, such as floaters or bad geometry. These artifacts are prevalent in in-the-wild captures (a) but are rarely seen in NeRF benchmarks, because evaluation views are often selected from the same camera path as the training views. We propose a new dataset of in-the-wild captures and a more realistic evaluation procedure (b), where each scene is captured by two paths: one for training and one for evaluation. We also propose Nerfbusters, a 3D diffusion-based method that improves scene geometry and reduces floaters (c), significantly improving upon existing regularizers in this more realistic evaluation setting.

Abstract

Casually captured Neural Radiance Fields (NeRFs) suffer from artifacts such as floaters or flawed geometry when rendered outside the input camera trajectory. Existing evaluation protocols often do not capture these effects, since they usually only assess image quality at every 8th frame of the training capture. To aid in the development and evaluation of new methods in novel-view synthesis, we propose a new dataset and evaluation procedure, where two camera trajectories are recorded of the scene: one used for training, and the other for evaluation. In this more challenging in-the-wild setting, we find that existing hand-crafted regularizers do not remove floaters nor improve scene geometry. Thus, we propose a 3D diffusion-based method that leverages local 3D priors and a novel density-based score distillation sampling loss to discourage artifacts during NeRF optimization. We show that this data-driven prior removes floaters and improves scene geometry for casual captures.

*Denotes equal contribution

1. Introduction

Casual captures of Neural Radiance Fields (NeRFs) [20] are usually of lower quality than most captures shown in NeRF papers. When a typical user (e.g., a hobbyist) captures a NeRFs, the ultimate objective is often to render a fly-through path from a considerably different set of viewpoints than the originally captured images. This large viewpoint change between training and rendering views usually reveals *floater* artifacts and bad geometry, as shown in Fig. 1a. One way to resolve these artifacts is to teach or otherwise encourage users to more extensively capture a scene, as is commonly done in apps such as Polycam[1] and Luma[2], which will direct users to make three circles at three different elevations looking inward at the object of interest. However, these capture processes can be tedious, and

[1]https://poly.cam/
[2]https://lumalabs.ai/

furthermore, users may not always follow complex capture instructions well enough to get an artifact-free capture.

Another way to clean NeRF artifacts is to develop algorithms that allow for better out-of-distribution NeRF renderings. Prior work has explored ways of mitigating artifacts by using camera pose optimization [38, 13] to handle noisy camera poses, per-image appearance embeddings to handle changes in exposure [15], or robust loss functions to handle transient occluders [32]. However, while these techniques and others show improvements on standard benchmarks, most benchmarks focus on evaluating image quality at held-out frames from the training sequence, which is not usually representative of visual quality at novel viewpoints. Fig. 1c shows how the Nerfacto method starts to degrade as the novel-view becomes more extreme.

In this paper, we propose both (1) a novel method for cleaning up casually captured NeRFs and (2) a new evaluation procedure for measuring the quality of a NeRF that better reflects rendered image quality at novel viewpoints. Our proposed evaluation protocol is to capture two videos: one for training a NeRF, and a second for novel-view evaluation (Fig. 1b). Using the images from the second capture as ground-truth (as well as depth and normals extracted from a reconstruction on *all* frames), we can compute a set of metrics on visible regions where we expect the scene to have been reasonably captured in the training sequence. Following this evaluation protocol, we capture a new dataset with 12 scenes, each with two camera sequences for training and evaluation.

We also propose *Nerfbusters*, a method aimed at improving geometry for everyday NeRF captures by improving surface coherence, cleaning up floaters, and removing cloudy artifacts. Our method learns a local 3D geometric prior with a diffusion network trained on synthetic 3D data and uses this prior to encourage plausible geometry during NeRF optimization. Compared to global 3D priors, local geometry is simpler, category-agnostic, and more repeatable, making it suitable for arbitrary scenes and smaller-scale networks (a 28 Mb U-Net effectively models the distribution of all plausible surface patches). Given this data-driven, local 3D prior, we use a novel unconditional Density Score Distillation Sampling (DSDS) loss to regularize the NeRF. We find that this technique removes floaters and makes the scene geometry crisper. To the best of our knowledge, we are the first to demonstrate that a learned local 3D prior can improve NeRFs. Empirically, we demonstrate that Nerfbusters achieves state-of-the-art performance for casual captures compared to other geometry regularizers.

We implement our evaluation procedure and Nerfbusters method in the open-source Nerfstudio repository [35]. The code and data can be found at `https://ethanweber.me/nerfbusters`.

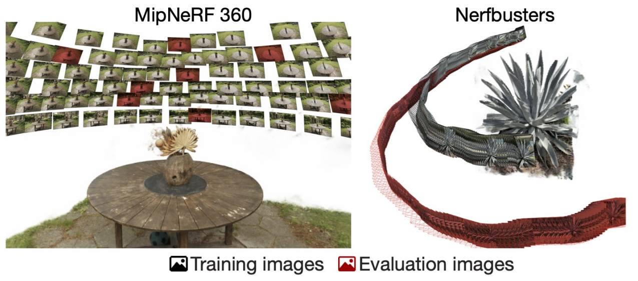


Figure 2: **Evaluation protocols.** Current evaluation of NeRFs (e.g., MipNeRF 360) measures render quality at every 8th frame of the captured (training) trajectory, thus only testing the model's ability to render views with small viewpoint changes. In contrast, we propose a new evaluation protocol, where two sequences are captured of the same scene: one for training and one for evaluation. Please see the supplementary material for plots showing the training and evaluation sequences for various NeRF datasets, including our proposed Nerfbuster Dataset.

2. Related Work

Evaluating NeRFs in-the-wild. Early works in neural rendering [18], including NeRF [20], established an evaluation protocol for novel view synthesis, where every 8th frame from a camera trajectory is used for evaluation. Most follow-up works have adapted this protocol and demonstrated impressive results on forward-facing scenes in LLFF [19], synthetic scenes [20], or 360 scenes [2, 29]. In these datasets, the training and evaluation views share camera trajectories, thus the methods are evaluated only for small viewpoint changes, as illustrated in Fig. 2. In contrast, we propose to record two camera trajectories, one for training and one for evaluation. **??** compares existing datasets (synthetic scenes [20], LLFF [19], MipNeRF 360 [2], and Phototourism [11]) with the proposed Nerfbuster dataset. We visualize the training and evaluation poses for each scene and quantify the difficulty of each dataset by computing the average rotation and translation difference between evaluation images and their closest training images. We find that viewpoint changes are very limited, and the proposed Nerfbuster dataset is much more challenging. Recently, Gao *et al.* [8] revisited the evaluation process for dynamic NeRFs, also highlighting shortcomings in dynamic NeRF evaluation. NeRFs for extreme viewpoint changes and few-shot reconstruction have been explored on ShapeNet [4], DTU [10], and CO3D [29], where a few or just a single view is available during training. These works focus on the generalization and hallucination of unseen regions, and either assume a category-specific prior [45, 43] or focus on simple scenes [43]. In contrast, our casual captures setting assumes that a $10-20$ second video is available at training time, better reflecting how people capture NeRFs. We then evaluate fly-throughs with extreme novel views on an entirely different video sequence, as illustrated in Fig. 2.

Diffusion models for 3D. Recently, several works have proposed the use of diffusion models for 3D generation or manipulation [27, 37, 22, 41]. These approaches can be divided into (1) methods that distill priors from existing 2D text-to-image diffusion models into a consistent 3D representation [27, 37], and (2) methods that train a diffusion model to explicitly model 3D objects or scenes [22]. These directions are complementary, where the former benefits from the sheer size of image datasets, and the latter from directly modeling 3D consistency. DreamFusion [27] proposes Score Distillation Sampling (SDS), where the text-guided priors from a large text-to-image diffusion model can be used to estimate the gradient direction in optimization of a 3D scene. We take inspiration from the SDS optimization procedure but instead adapt it to supervise NeRF densities in an unconditional manner, directly on 3D density values. As the underlying model, we train a 3D diffusion model on local 3D cubes extracted from ShapeNet [4] objects. We find the distribution of geometry (surfaces) within local cubes is significantly simpler than 2D natural images or global 3D objects, reducing the need for conditioning and high guidance weights. To the best of our knowledge, we are the first to suggest a learned 3D prior for category-agnostic, unbounded NeRFs.

Data-driven local 3D priors. Approaches for learning 3D geometry can be divided into local and global approaches, where global approaches reason about the entire scene, and local approaches decompose the scene into local surface patches. We learn a prior over local geometric structures, as local structures are simple, category-agnostic, and more repeatable than global structures [3, 21]. DeepLS [3] proposes to decompose a DeepSDF [26] into local shapes, and finds that this simplifies the prior distribution that the network learns. Similarly, AutoSDF [21] learns a local 3D prior and tries to learn the distribution over a 3D scene with an autoregressive transformer. We are inspired by their approach, but use a diffusion model rather than a VQ-VAE [28, 36], and show that the learned prior can be used to regularize NeRF geometry.

Regularizers in NeRFs. Our work can be seen as a regularizer for NeRFs. Most existing regularizers are handcrafted priors that encourage smoothness and discourage non-empty space. Plenoxels [7] proposed a Total-Variation (TV) in 3D that penalizes the large changes between neighboring voxels. TV has also been applied in 2D rendered images in RegNeRF [25] and on the basis of factorized plenoptic fields [5, 6]. Plenoctrees [42] proposed a sparsity loss that penalizes densities from randomly queried 3D locations in space. This sparsity loss removes densities unnecessary in explaining the training views. To avoid penalizing all densities equally, MIP-NeRF 360 [2] proposes a distortion loss on accumulated weights that encourages surfaces to be sharp. Concurrent and most similar to our work, DiffusionRF [41] proposes a data-driven RGB-D diffusion prior. The method trains a diffusion model on synthetic RGB-D examples and uses the learned prior to regularize a NeRF. In contrast to the proposed local 3D diffusion prior, operating in 2.5D comes with several disadvantages, namely 1) occlusions are not modeled, 2) the joint distribution of RGB-D images is more complex than that of 3D occupancy (and as a result requires more data for generalization), 3) unlike a 3D diffusion model, it is not by definition 3D-consistent, *i.e.*, the view consistency has to come from the NeRF rather than the regularizer.

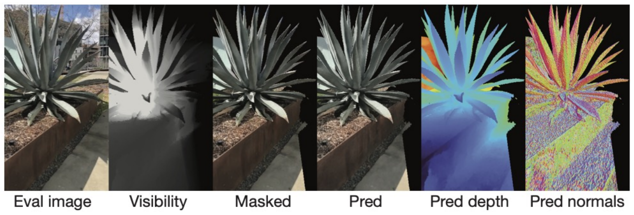


Figure 3: **Evaluation capture.** Here we show the data used in our evaluation protocol. The evaluation trajectory is a separate capture that is held out during optimization of the NeRF. Individual components shown here are further described in Sec. 3.

3. Evaluation Procedure

We propose an evaluation protocol that captures two videos, one for training and one for evaluating a NeRF. Training videos should be around $10-20$ seconds which are indicative of what a user might do when prompted to scan an object or scene. Anything longer than this may reduce the appeal and practicality of using NeRFs. The second video represents the novel view that a user may wish to render. The second video is only used as ground truth and does not change how users currently interact with NeRFs. We record 12 scenes (two videos each) in this way to construct our Nerfbusters Dataset. All videos were taken with a hand-held phone to approximate the casual capture setup.

Evaluating on casual captures. The steps to create our evaluation data can be boiled down to the following straightforward steps:

1. Record a video to train a NeRF (training split)
2. Record a second video with a viewpoint change (evaluation split)
3. Extract images from both videos and run SfM on all images
4. Train a "pseudo ground truth" model on both splits and save depth, normal, and visibility maps for the evaluation split.
5. Train your proposed method on the training split and evaluate with the evaluation split images and their pseudo ground truth maps.

In Fig. 3, we show an evaluation image and its visibility, depth, and normal maps. The pseudo ground truth is high quality since it has been trained together with the first video. The visibility map is computed by taking the depth map, back-projecting each pixel into a 3D point, and then counting how many times that 3D point is seen from a training viewpoint. This dataset with associated visibility masks and processing code can be found at https://ethanweber.me/nerfbusters.

Masking valid regions. Rendering extreme novel views exposes part of the scene that was not captured in the training views. As most existing NeRFs are not designed to hallucinate completely unseen views, we only evaluate regions observed in the train capture trajectories using visibility masks. More specifically, we mask out regions that are either (1) not seen by any training views (i.e., where the visibility map is zero) or (2) are predicted to be too far away (i.e., predicted depth $>$ distance threshold). We set this threshold to two times the largest distance between any two camera origins in both the training and evaluation splits. In the Nerfstudio codebase, this corresponds to a value of 2 because camera poses are scaled to fit within a box with bounds (-1,-1,-1) and (1,1,1).

Coverage. Because we mask out pixels by both visibility [8] and depth, we report "coverage" which is the percent of evaluated pixels within the visible regions, commonly reported in depth completion [44, 39, 40]. For example, removing all densities and predicting infinite depth would result in zero coverage.

Image quality and geometry metrics. We use masked versions of PSNR, SSIM, and LPIPS for image quality. We also report on depth (MSE and mean abs. disparity difference) and normals (mean and median degrees, and the percent of valid pixels with normals $<$ 30 degrees). We report averages for all images in the Nerfbusters Dataset in Sec. 5.

4. Nerfbusters

Our method consists of two steps. First, we train a diffusion model to denoise local 3D cubes. This model is trained on synthetic data and learns a prior over local 3D shapes. Second, we apply this local prior to real 3D scenes represented by NeRFs. We do this by querying densities in local cubes in the scene and using a novel Density Score Distillation Score (DSDS) loss to regularize our implicit scene representation. This prior improves reconstructions in regions with sparse supervision signals and removes floaters. Fig. 5 provides an overview of our pipeline.

4.1. Data-driven 3D prior

Following the recent process in the context of denoising generative diffusion models [33, 34, 24, 30, 27], we formulate our generative model as a denoising diffusion probabilistic model (DDPM) [9], which iteratively denoises a

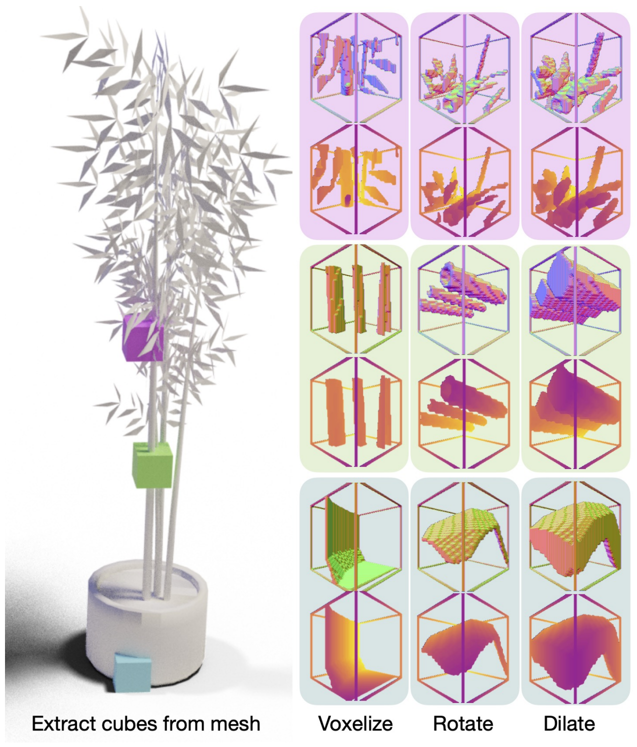


Figure 4: **Training data for Nerfbusters diffusion model.** Given a mesh, we extract local cubes scaled $1-10\%$ of the mesh size. We voxelize these cubes with resolution 32^3, and augment them with random rotations and random dilation. We illustrate each step with renderings of depth and normals from three cubes. The synthetic scenes from Shapenet offer a high variety in local cubes, containing both flat surfaces, round shapes, and fine structures.

voxelized $32 \times 32 \times 32$ cube x of occupancy. Our diffusion model ϵ_θ is trained with

$$\mathcal{L}_{\text{Diff}} = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|_2^2 \tag{1}$$

where $t \sim \mathcal{U}(0, 1000)$, $\epsilon \sim \mathcal{N}(0, I)$ and $\bar{\alpha}$ follows a linear schedule that determines the amount of noise added at timestep t. We implement our diffusion model as a small 3D U-Net [31] with three downsampling layers that double the number of channels per downsampling. We train the model on synthetic 3D cubes from ShapeNet and find that a small U-Net with only 7.2 M parameters (28MB) is sufficient to learn a local 3D prior over shapes.

4.2. Curate synthetic 3D cubes

We train our diffusion model on local cubes sampled from ShapeNet [4], illustrated in Fig. 4. We sample a random ShapeNet mesh and extract N local meshes at the boundary with sizes between 1-10$\%$ of the mesh min and max vertices. We voxelize these local meshes into cubes with a resolution of 32^3. We then augment the cubes with random rotations and dilation. This data processing pipeline is fast and performed online during training to increase the

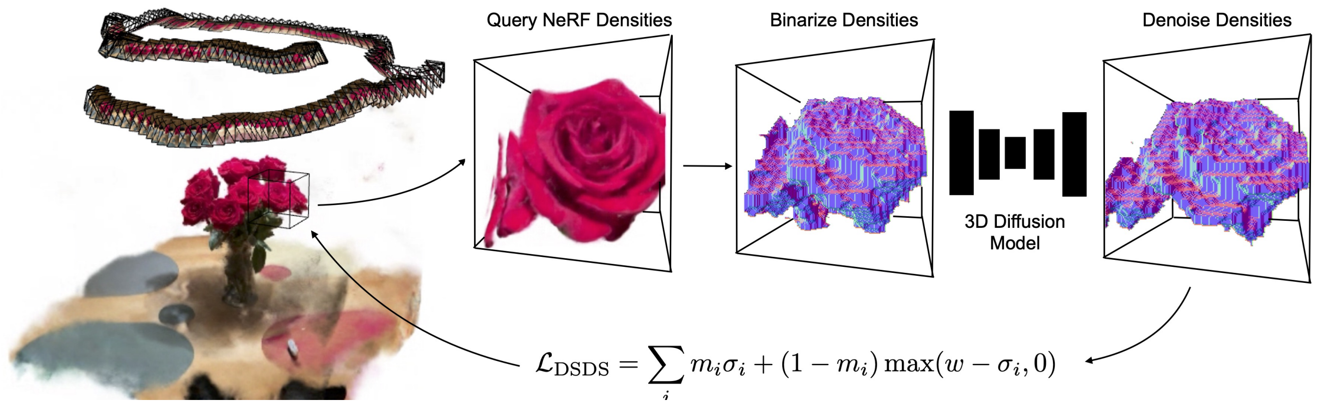

Query NeRF Densities
Binarize Densities
Denoise Densities
3D Diffusion Model
$\mathcal{L}_{\text{DSDS}} = \sum_i m_i\sigma_i + (1-m_i)\max(w-\sigma_i, 0)$

Figure 5: **Method overview.** We learn a local 3D prior with a diffusion model that regularizes the 3D geometry of NeRFs. We use importance sampling to query a 32^3 cube of NeRF densities. We binarize these densities and perform one single denoising step using a pre-trained 3D diffusion model. With these denoised densities, we compute a density score distillation sampling (DSDS) that penalizes NeRF densities where the diffusion model predicts empty voxels and pushes the NeRF densities above the target w where the diffusion model predicts occupied voxels $m = \mathbb{1}\{x_0 < 0\}$.

diversity of 3D cubes. We find that adjusting the thickness of the surface with dilation rather than infilling the mesh is faster and better defined for non-watertight meshes. Fig. 4 illustrates the large diversity in the local cubes—some contain flat surfaces (bottom of the vase), round shapes (stem), and fine structures (leaves).

4.3. Applying 3D prior in-the-wild

We represent a 3D scene with a Neural Radiance Field (NeRF), [20] which takes a 3D point as input and outputs color and density, and is trained with differentiable volume rendering [20, 16]. We build on the Nerfacto model from Nerfstudio [35] that combines recent progress in NeRFs including hash grid encoding [23], proposal sampling [1], per-image-appearance optimization [15], and scene contraction [1]. Although Nerfacto has been optimized for in-the-wild image captures, it still reveals floaters when rendered from novel views. To address these issues, we leverage the pre-trained Nerfbusters diffusion model. We propose a novel sampling strategy that samples cubes from non-empty regions and a Density Score Distillation Sampling (DSDS) loss that distills the diffusion prior into the NeRF. As a result, our approach yields better scene geometry.

Importance sampling cubes. Since the NeRF represents a density field, we can query voxelized cubes in 3D space at any size, location, and resolution. For an efficient sampling of the location of the 3D cubes, we propose to store a low-resolution occupancy grid of either *accumulation weights* or *densities*. We sample the location of the 3D cubes from the distribution of this low-resolution occupancy grid. Storing *accumulation weights* in the occupancy grid yields cubes sampled mostly on frequently seen surfaces. Whereas, storing *densities* in the occupancy grid enables sampling of occluded regions. In practice, we clamp densities to one, to avoid a few densities dominating the sampling probability. We apply an exponential moving average (EMA) decay on the grid to update the occupancy grid when floaters are deleted. This importance sampling method comes with almost no added cost since we store the densities or weights along the rays already used for volume rendering, and use a small 20^3 occupancy grid. Following the sampling of a cube center location, we proceed to sample cubes of resolution 32^3 and 1-10$\%$ of the scene.

Density Score Distillation Sampling (DSDS). Our diffusion model is trained on discretized synthetic data in $\{-1, 1\}$ indicating free or occupied space, respectively. NeRF densities, on the other hand, are in $[0, \infty)$, where low densities indicate free space and larger densities mean more occupied space. In practice, we observe that densities less than 0.01 are mostly free space, whereas occupied space have density values ranging from $[0.01, 2000]$. We propose a Density Score Distillation Sampling (DSDS) loss that handles the domain gap between the densities without exploiting gradients.

Given a cube of NeRF densities σ, we discretize the densities $x_t = 1$ if $\sigma > \tau$ else -1 at time t, where τ is a hyperparameter that decides at what density to consider a voxel for empty or occupied. The Nerfbusters diffusion model then predicts the denoised cube x_0. The timestep t is a hyperparameter that determines how much noise the diffusion model should remove and can be interpreted as a learning rate. In practice, we choose a small $t \in [10, 50]$. With the denoised cube x_0, we penalize NeRF densities that the diffusion model predicts as empty or increase densities that the diffusion model predicts as occupied with

$$\mathcal{L}_{\text{DSDS}} = \sum_i m_i\sigma_i + (1-m_i)\max(w-\sigma_i, 0) \qquad (2)$$

where $m = \mathbb{1}\{x_0 < 0\}$ is a mask based on the de-

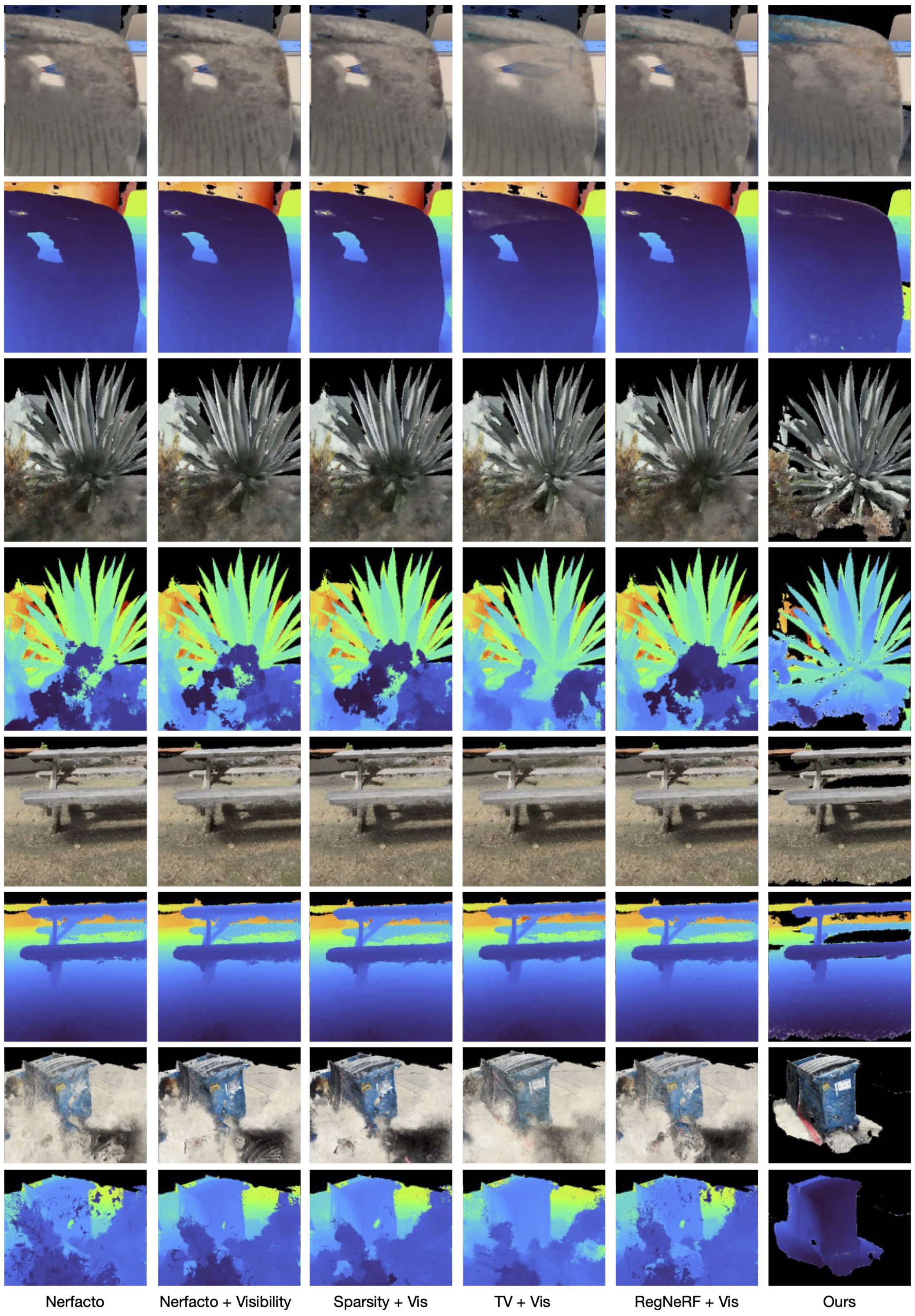


Figure 6: **Qualitative results.** NeRFs suffer from floaters and bad geometry when rendered away from training views. Our proposed diffusion prior fills holes (first rows), removes floaters (second and fifth row), and improves geometry (all). Please see the supplementary material for video results on our evaluation splits.

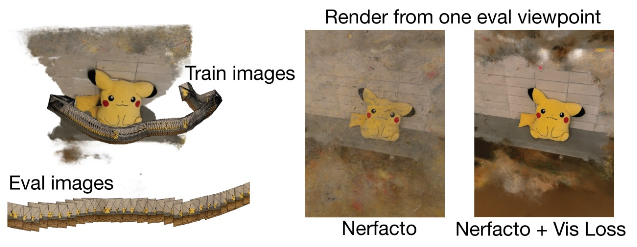


Figure 7: **Visibility loss.** Our visibility loss enables stepping behind or outside the training camera frustums. We accomplish this by supervising densities to be low when not seen by at least one training view. Other solutions would be to store an occupancy grid [23] or compute ray-frustum intersection tests during rendering. Our solution is easy to implement and applicable to any NeRF.

noised predictions. We penalize densities where the diffusion model predicts emptiness and increase densities where the model predicts occupancy. w is a hyperparameter that determines how much to increase the densities in occupied space. The $\max$ operator ensures that no loss is applied if an occupied voxel already has a density above w. Similar to SDS [27, 37], the DSDS loss distills the diffusion prior with a single forward pass and without backpropagating through the diffusion model.

Why not just... use a differentiable function to convert densities to the valid range of the diffusion model, then compute the SDS loss [27, 37], and then backpropagate through the activation function? This would require a function $s : \sigma \rightarrow x_t$ to map $s(0) = -1$, $s(\tau) = 0$, and $s(2\tau) = 1$, where τ is the crossing value where densities begin to be occupied. A scaled and shifted sigmoid function or a clamped linear function satisfies these requirements, but both have very steep gradients in some regions and no gradients in other regions, resulting in issues when backpropagating. In contrast, DSDS has gradients for any density predicted to be empty or occupied. In practice, we set $\tau = w = 0.01$ meaning our method deletes densities at points predicted to be empty and otherwise leaves the points unconstrained for the NeRF RGB loss to freely optimize.

Why not just... use accumulated weights, which are in the range [0, 1]? Weights are more well-behaved than densities but more expensive to compute as they require shooting a ray through the scene, evaluating and accumulating the densities along a ray. This results in significantly more function calls, but more fundamentally, requires one to specify a view from which to shoot the rays. This limits the diffusion prior to improving regions that are visible regions from the chosen view. A similar issue arises when using 2D or 2.5D priors [25, 41], where they may not regularize occluded regions unless viewpoints are chosen in a scene-specific way.

4.4. Visibility Loss

Our proposed local 3D diffusion model improves scene geometry and removes floaters, but it requires decent start-

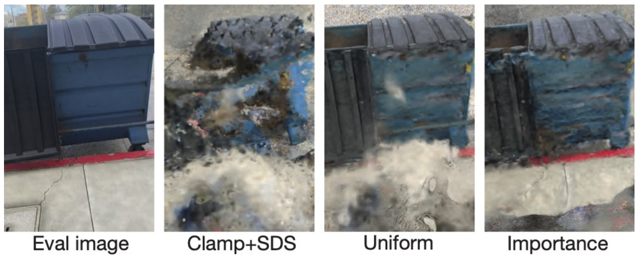


Figure 8: **Ablations results.** Using a simple activation function and SDS results in a not-well-behaved gradient signal, increasing the number of floaters in the scene. Importance sampling more effectively applies the 3D cube loss in space, cleaning up floaters and improving the scene geometry.

ing densities since it operates locally and thus needs contextual information to ground its denoising steps. To this end, we propose a simple loss that penalizes densities at 3D locations that are not seen by multiple training views. We find this simple regularizer effective in removing floaters from regions outside the convex hull of the training views. We define our visibility loss as

$$\mathcal{L}_{\text{vis}} = \sum_i V(q_i) f_\sigma(q_i) \tag{3}$$

where $f_\sigma(q_i) = \sigma_i$ is the NeRF density at the 3D location q_i, and $V(q_i) = \mathbb{1}\{\sum_{j=1} v_{ij} < 1\}$ indicates if the location is not visible from any training views. We approximate the visibility $v_{ij} \in \{0, 1\}$ of the i'th 3D location in the j'th training view with a frustum check. This approximation does not handle occlusions, instead overestimates the number of views a location is visible from. This loss penalizes densities in regions not seen by training images.

In practice, we implement this by defining a single tight sphere around our training images and render batches of rays that shoot from a random location on the sphere surface, through the center of the scene, and far off into the distance. We render rays with Nerfacto and apply this loss to the sampled points. Nerfacto uses a proposal sampler [2] to importance sample around surfaces, so our loss is effective in quickly culling away any floating artifacts with high density outside visible regions. See Fig. 7 for a qualitative result where we render from behind training images.

5. Experiments in-the-wild

We follow our proposed protocol described in detail in Sec. 3. We apply different regularizers as a post-processing approach to clean up NeRFs and also run ablations on our proposed method.

Implementation details. For each experiment, we use the Nerfacto model within the Nerfstudio [35] codebase. We turn off pose estimation for evaluation purposes and then train Nerfacto for 30K iterations which takes up to half

	PSNR ↑	SSIM ↑	LPIPS ↓	Depth ↓	Disp. ↓	Mean ° ↓	Median ° ↓	% 30° ↑	Coverage ↑
Nerfacto Pseudo GT	25.98	0.8591	0.1019	0.0	0.0	0.0	0.0	1.0	0.893
Nerfacto	17.00	0.5267	0.3800	126.277	1.510	60.63	54.638	0.254	**0.896**
+ Visibility Loss	17.81	0.5538	0.3432	100.057	1.041	57.73	51.335	0.280	0.854
+ Vis + Sparsity [42]	17.81	0.5536	0.3445	92.168	1.145	57.77	51.399	0.280	0.854
+ Vis + TV [7]	17.84	0.5617	0.3409	74.015	0.382	61.93	56.164	0.242	0.843
+ Vis + RegNeRF [25]	17.49	0.5396	0.3585	182.447	1.200	59.39	53.267	0.268	0.858
+ Vis + DSDS (Ours)	**17.99**	**0.6060**	**0.2496**	**54.453**	**0.114**	**54.77**	**47.981**	**0.295**	0.630

Table 1: **Quantitative evaluation.** NeRFs suffer when rendered away from the training trajectories. Existing regularizers do not suffice to improve the geometry. Nerfbusters learns a local 3D prior with a diffusion model, which removes floaters and improves the scene geometry. Results are averaged across 12 scenes.

Cube sampling strategies					
	PSNR	SSIM	Disp.	Mean °	Cov.
Uniform	14.61	0.4276	10.288	61.52	**0.886**
Densities σ	**16.46**	**0.5086**	**0.081**	**49.21**	0.606
Weights	15.86	0.4466	0.112	53.09	0.634
Activation functions					
	PSNR	SSIM	Disp.	Mean °	Cov.
Clamp+SDS	12.53	0.2652	2.065	87.33	**1.000**
Sigmoid+SDS	12.53	0.2652	2.065	87.33	**1.000**
$\sigma_\mathcal{T}$+DSDS	**15.86**	**0.4466**	**0.112**	**53.09**	0.634
Cube size range as % of scene					
	PSNR	SSIM	Disp.	Mean °	Cov.
1-20%	**17.05**	**0.5005**	**0.083**	54.87	0.600
10-20%	16.93	0.4884	0.090	**50.78**	**0.640**
1-10%	15.86	0.4466	0.112	53.09	0.634

Table 2: **Ablation study.** Ablation on the "garbage" scene for different settings of using our 3D prior as a NeRF loss. Cube sampling refers to uniformly sampling the entire scene versus importance sampling with accumulated weights or densities.

an hour. We then fine-tune from this checkpoint with different regularizer methods. We compare the proposed method with vanilla Nerfacto, Nerfacto with the proposed visibility loss, Nerfacto with our visibility loss and 3D sparsity loss [42], 3D TV regularization [7], and 2D TV which is RegNeRF [25]. Our implementations also use the distortion loss [2] which is on by default with Nerfacto. All methods are effective within the first 1K iterations of fine-tuning (~4 minutes on an NVIDIA RTX A5000 for Nerfbusters), but we train for 5K iterations. For the 3D baselines, we sample 40 32^3 cubes per iteration and for the 2D baseline RegNeRF, we render ten 32^2 patches. The usual NeRF reconstruction loss is also applied during fine-tuning with 4096 rays per batch.

Results. Tab. 1 shows that visibility loss improves vanilla Nerfacto across all quality metrics. Existing hand-crafted regularizers do not improve upon this baseline. In contrast, our learn local diffusion prior removes floaters and improves the scene geometry, yielding state-of-the-art results on these challenging casual captures. The proposed method deletes floaters, and thus we find that it has lower coverage than the baselines. **??** shows per scene results. Fig. 6 shows a qualitative comparison of the methods for both indoor and outdoor scenes. We find that our method improves geometry by completing holes (see the chair in the first row), removing floaters (see in front of century plant in the second row and garbage truck in the fourth row), and sharping geometry (see the under the bench in the third row).

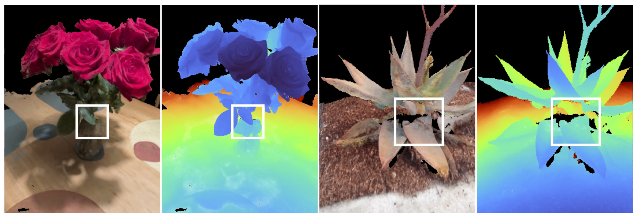


Figure 9: **Limitations.** The proposed model only operates on densities, which comes with some limitations. We find that it cannot distinguish floaters from transparent objects (left). It does not hallucinate texture and thus ends up removing regions that are occluded in all training views (right).

Ablations of our 3D prior on real data We ablate our method on the "garbage" scene (Tab. 2). We find that the cube sampling strategies (i.e., where to apply the diffusion prior) are important, and using the proposed importance sampling with densities yields the best performance. Fig. 8 compares uniform sampling with importance sampling (using densities). Importance sampling samples less empty space, and thus is more effective at cleaning up floaters and scene geometry. We compare the proposed DSDS loss against SDS with either a scaled and shifted sigmoid or a clamped sigmoid that satisfies our requirements (see Sec. 4.3). We find the gradients do not flow well through this activation function resulting in a distorted scene with many floaters (see Fig. 8 left). We also ablate the cube sizes used cubes size ranging from 1% to 20% of the scene scale. We find that our method is relatively robust to the cube sizes, yielding a trade-off between removing more with larger cubes and removing less with smaller cubes.

6. Conclusion and future work

Transparent objects. NeRFs are able to represent transparent objects by assigning low densities to the transparent object. These transparent densities behave similarly to floaters, and it requires semantic information to distinguish the two. Since our local diffusion prior does not have semantic information, it removes transparent objects as illustrated in the vase in Fig. 9.

Hallucinating texture. The proposed method cleans geometry but cannot edit texture, as our method operates on densities. This means that we can remove regions that contain floaters or fill holes, but we cannot colorize these regions. We leave colorization and inpainting low-confidence regions to future work, where 2D diffusion priors [27, 17] or 3D-consistent inpainting [14, 12] may be relevant.

Conclusion. We propose a new evaluation procedure of Neural Radiance Fields (NeRFs) that better encompasses how artists, designers, or hobbyists use the technology. We present a dataset with 12 captures recorded with two camera trajectories each, one used for training and one for evaluation. We find that current hand-crafted regularizers are insufficient when NeRFs are rendered away from the training trajectory. We propose a data-driven, local 3D diffusion prior, Nerfbusters, that removes floaters and improves the scene geometry. We have implemented our proposed evaluation procedure and method in the widely adopted codebase Nerfstudio and will release it for the benefit of the community.

Acknowledgements

We thank our colleagues for their discussions and feedback throughout the project, especially those within the Kanazawa AI Research (KAIR) lab and those part of the Nerfstudio development team. We thank Kamyar Salahi, Abhik Ahuja, Jake Austin, Xinyang Han, and Alexander Kristoffersen for helping to improve paper drafts.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 5

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 3, 7, 8

[3] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 608–625. Springer, 2020. 3

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 3, 4

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. 3

[6] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv preprint arXiv:2301.10241*, 2023. 3

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3, 8

[8] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *Advances in Neural Information Processing Systems*, 2022. 2, 4

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 4

[10] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 2

[11] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 129(2):517–547, 2021. 2

[12] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pages 515–534. Springer, 2022. 9

[13] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021. 2

[14] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14458–14467, 2021. 9

[15] 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 *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 7210–7219, 2021. 2, 5

[16] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 5

[17] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Realfusion: 360 {\deg} reconstruction of any object from a single image. *arXiv preprint arXiv:2302.10663*, 2023. 9

[18] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[19] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[20] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 5

[21] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 306–315, 2022. 3

[22] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. *arXiv preprint arXiv:2212.01206*, 2022. 3

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 5, 7

[24] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 4

[25] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5480–5490, June 2022. 3, 7, 8

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3

[27] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3, 4, 7, 9

[28] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 3

[29] 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. In *International Conference on Computer Vision*, 2021. 2

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 4

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 4

[32] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. *arXiv preprint arXiv:2302.00833*, 2023. 2

[33] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 4

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 4

[35] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, et al. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023. 2, 5, 7

[36] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 3

[37] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022. 3, 7

[38] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2

[39] Frederik Warburg, Daniel Hernandez-Juarez, Juan Tarrio, Alexander Vakhitov, Ujwal Bonde, and Pablo F Alcantarilla. Self-supervised depth completion for active stereo. *IEEE Robotics and Automation Letters*, 7(2):3475–3482, 2022. 4

[40] Frederik Warburg, Michael Ramamonjisoa, and Manuel López-Antequera. Sparseformer: Attention-based depth completion network. *arXiv preprint arXiv:2206.04557*, 2022. 4

[41] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *arxiv*, 2023. 3, 7

[42] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 3, 8

[43] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[44] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018. 4

[45] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. *arXiv preprint arXiv:2212.00792*, 2022. 2