DiffusioNeRF: Regularizing Neural Radiance Fields with Denoising Diffusion Models

Jamie Wynn   Daniyar Turmukhambetov

Niantic

www.github.com/nianticlabs/diffusionerf

Abstract

Under good conditions, Neural Radiance Fields (NeRFs) have shown impressive results on novel view synthesis tasks. NeRFs learn a scene's color and density fields by minimizing the photometric discrepancy between training views and differentiable renderings of the scene. Once trained from a sufficient set of views, NeRFs can generate novel views from arbitrary camera positions. However, the scene geometry and color fields are severely under-constrained, which can lead to artifacts, especially when trained with few input views.

To alleviate this problem we learn a prior over scene geometry and color, using a denoising diffusion model (DDM). Our DDM is trained on RGBD patches of the synthetic Hypersim dataset and can be used to predict the gradient of the logarithm of a joint probability distribution of color and depth patches. We show that, these gradients of logarithms of RGBD patch priors serve to regularize geometry and color of a scene. During NeRF training, random RGBD patches are rendered and the estimated gradient of the log-likelihood is backpropagated to the color and density fields. Evaluations on LLFF, the most relevant dataset, show that our learned prior achieves improved quality in the reconstructed geometry and improved generalization to novel views. Evaluations on DTU show improved reconstruction quality among NeRF methods.

1. Introduction

Neural radiance fields, neural implicit surfaces, and coordinate-based scene representations are proving valuable for novel view synthesis and 3D reconstruction tasks. NeRFs [17] learn a *specific scene's* appearance as a multi-layer perceptron that predicts density and color, when given any 3D point and a viewing direction.

This volume representation allows differentiable rendering from arbitrary views, where predicted color contributions along a ray are alpha-composited according to the den-
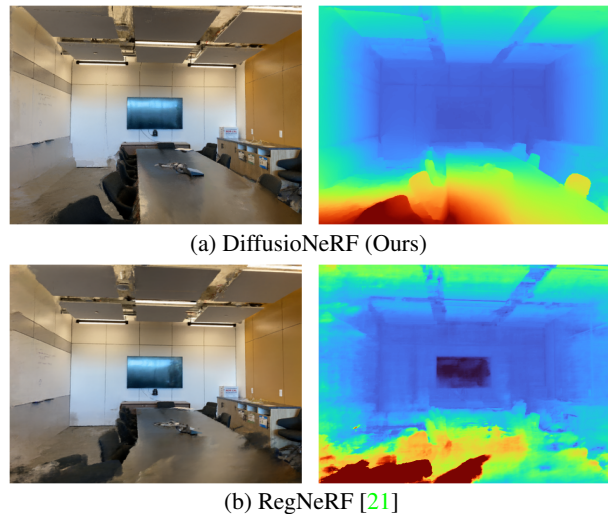

(a) DiffusioNeRF (Ours)

(b) RegNeRF [21]

Figure 1. Image and depth map rendered from a test view. All NeRF models were trained with **3** views of the LLFF [16] dataset's "Room" scene. Our priors encourage NeRF to explain the TV and table geometry with flat surfaces in the density field, and to explain the view-dependent color changes with the color field.

sity predictions.

The model is trained with the aim of faithfully reconstructing images captured with known camera poses. Even when trained with just a photometric reconstruction loss, NeRFs show impressive generalization capabilities, inspiring novel applications in virtual and augmented reality, and visual special effects.

However, with small numbers or even with large numbers of input views, the scene color and geometry fields are severely under-constrained. Indeed, an infinite number of NeRFs can explain all training views. In practice, NeRFs can generate low-quality and physically implausible geometries and surface appearances. For example, "floaters" are one common artifact, where the fitted density field contains clouds of semi-transparent material floating in mid-air that would look reasonable in 2D once rendered from training views, but look implausible from novel views.

Various hand-crafted regularizers and learned priors have been proposed to tackle these issues: hand-engineered priors to constrain the scene geometry [2,21], learned priors that force plausible renderings from arbitrary views [21], and methods that use single image depth and normal estimation [38,46] to provide high-level constraints on the estimated scene geometry. However, there are no approaches that learn a joint probability distribution of the scene geometry and color.

Our contribution is leveraging denoising diffusion models (DDMs) as a learned prior over color and geometry. Specifically, we use an existing synthetic dataset to generate a dataset of RGBD patches to train our DDM. DDMs do not predict a probability for RGBD patch distribution. Rather, they provide the gradient of the log-probability of the RGBD patch distribution, *i.e.* the negative direction to the noise predicted by DDM is equivalent to moving towards the modes of the RGBD patch distribution. As NeRFs are trained with stochastic gradient descent, gradients of log-probabilities are sufficient, as they can be back-propagated to NeRF networks during training to act as a regularizer; probabilities are not required for this purpose. We demonstrate that the DDM gradient encourages NeRFs to fit density and color fields that are more physically plausible on the LLFF and DTU datasets.

2. Related work

Geometry modeling The geometry of the scene can be modeled as a density field [17], occupancy field [22,23] or signed distance field [40,43,44]. Geometry models can be rendered using differentiable surface/volumetric rendering, so that the training loss for a NeRF model is the photometric reconstruction loss [17]. Signed distance fields also require regularization with an Eikonal loss [6] to constrain the distance field to be valid. Our regularizer operates on rendered color and depth patches, so it can be applied to any geometry representation.

Field representation NeRFs [17] represent geometry with a multi-layer perceptron that is queried with a 3D coordinate. Positional encoding of coordinates, where coordinate values are evaluated with sinusoids at different frequencies, allows modeling of high-frequency density signals with MLPs [35]. Alternatively, [7,29] encode scalar opacity and spherical harmonic coefficients in a sparse voxel representation, and shows that novel views can be synthesized without MLPs. Similarly, Neural Sparse Voxel Fields [15] stores feature encodings in a sparse voxel octree structure that can be trilinearly interpolated and passed through an MLP to predict density and color, thus improving the modeling capacity and rendering speed of NeRFs. MVSNeRF [3] predicts a volume of feature encodings by constructing a 3D cost volume and processing it with 3D CNNs. Density and color MLPs trilinearly interpolate the feature encoding vol-

ume to train NeRFs. The 3D CNN can be pretrained on a large number of scenes, which allows faster convergence on novel scenes.

Instant Neural Graphics Primitives [19] uses multi-scale hash tables to store feature encodings of all coordinates in a fixed memory block. This allows storing features at varying spatial resolutions, and consequently reduces the size of the MLP that models geometry and color. With a GPU-optimized implementation, Instant NGP can train NeRFs in minutes without quality degradation. Our contribution is in priors used for NeRF optimization, and hence our method is agnostic to the underlying geometry representation. As Instant NGP is fast to train and render, we use it as a backbone for our experiments.

Density regularization Mip-NeRF 360 [2] proposes a density regularizer that encourages compactness of the density along conical frustums. In addition to our learned regularizer, we use [2] density regularizer as it helps to sharpen the distribution of densities along sampled rays.

Regularization with loss terms Loss terms to regularize NeRFs can play an important role in the final result, as they provide additional supervision to under-constrained geometry and color fields. Some regularizers are hand-crafted to encourage depth and normal smoothness, *e.g.* [2,21,23,48]. In [11], a semantic loss is introduced to make high-level semantic attributes consistent across renderings from random views. In [27] a loss term regularizes rendered depth maps with depths estimated using Structure-from-Motion and depth completion methods. MonoSDF [46] regularizes occupancy fields with loss terms that incorporate depth and normals maps predicted with a single-image depth prediction model. Similarly, [38] introduces loss terms that use a single-image normal prediction model to regularize rendered normal maps. While all these approaches introduce high-level geometric supervision to NeRFs, the predicted depth and normals are fixed during NeRF fitting and hence the depth and normal models provide a unimodal prior over geometry. Furthermore, the additional supervision is not adapted to the NeRF reconstructions and hence the monocular depth and normal predictions are trusted blindly.

Regularization with Normalizing Flows RegNeRF [21] uses a 2D depth patch smoothness prior and a normalizing flow model as a learned prior over 2D RGB patches. The color patches are rendered while fitting the NeRF and a term proportional to the log probability density assigned to the patch by the normalizing flow model is added to the loss function.

However, the underlying cause of NeRF's dramatic performance degradation in the few-view case is that the geometry is poor, so we argue that it is preferable to regularize the geometry directly, rather than indirectly via RGB patches. By learning a distribution over RGBD patches we also benefit from the fact that color and depth are strongly
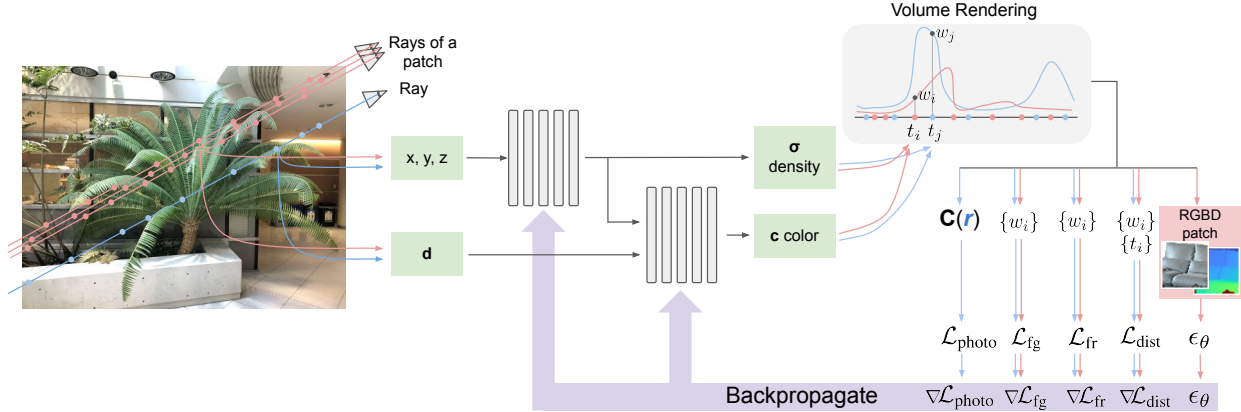
Figure 2. Illustration of our method. The scene is sampled with training-view rays and rays originating from random patches. Color and density are predicted by MLP for the 3D points sampled along the rays. Volumetric rendering is used to estimate expected color $\mathbf{C}(\mathbf{r})$, depth $\mathbf{D}(\mathbf{r})$ as well as weights of color contributions $\{w_i\}$ and positions of samples $\{t_i\}$. These estimates are used to compute gradients of losses that are backpropagated to color and density MLPs. DDM model ϵ_θ uses RGBD patches to predict color and density gradients that are passed to MLPs directly. Instant NGP's multi-scale hash table of feature encodings is not illustrated for simplicity.

correlated, and therefore attempting to regularize them separately discards information.

RegNeRF [21] uses MLPs to model color and density fields, hence during NeRF training the patch rendering cost can extend NeRF training time substantially. Thus, RegNeRF renders 8×8 patches for the prior model, which severely limits the amount of context visible to the normalizing flow model. We use Instant NGP for our NeRF representation, which has a fast rendering time, allowing us to model priors over 48×48 patches.

Normalizing flows are generative models that learn to transform a simple probability distribution into a more complex data distribution [13]. The model is built of blocks that fulfil the requirements of (i) preserving the number of dimensions of input and output features; (ii) being invertible, *i.e.* the input to the block can be calculated from the output; and (iii) the Jacobian of each block must be tractable so that the log probability density can be computed. These constraints can lead to trade-offs in which model expressiveness is sacrificed for tractability. Diffusion models do not have such constraints on their structures and may therefore be more suitable to model data priors.

Denoising Diffusion Models DDMs [8, 20, 31] are powerful generative models that learn to estimate gradients of the log data distribution. Once trained, Langevin dynamics sampling [42] can be used to generate novel samples by performing a sequence of denoising steps starting from a random sample of a standard Gaussian distribution. Denoising Diffusion Models have successfully been used to learn and sample images [8, 34], video [9], speech [4, 14], *etc*. Recently, multiple DDM-based models were proposed for the task of text-to-image synthesis, *e.g.* DALL-E 2 [25] and Imagen [28]. Concurrently to our work, Dreamfusion [24]

has incorporated Imagen into NeRF optimization to generate novel 3D assets from a text input. Unlike our work, they use DDMs to guide optimization of NeRFs to match input text, while we use DDMs to regularize NeRFs given input training images.

3. Method

We start by covering preliminaries like NeRF and DDM training. Next, we describe the relation of DDMs to the gradient of the log-likelihood of the data, and show how we incorporate DDMs as NeRF regularizers. An overview of the our method is shown in Fig. 2.

3.1. NeRFs

Given a set of images of a scene $\mathcal{I}$ with camera intrinsic parameters and poses, we are interested in optimizing a density field $\sigma : \mathbb{R}^3 \to \mathbb{R}_+$ and color field $\mathbf{c} : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3_{[0,1]}$, where the density field can be evaluated at any 3D coordinate $(x, y, z) \in \mathbb{R}^3$ and the color field can be evaluated at any 3D coordinate and viewing direction $\mathbf{d} \in \mathbb{S}^2$.

The density and color fields can be used to synthesize views of the scene from arbitrary cameras using differentiable rendering techniques. The expected color $C(\mathbf{r})$ of a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ can be estimated using discrete samples $t_{0:N}$ (where $t_{i+1} > t_i > 0$), so

$$\mathbf{C}(\mathbf{r}) \cong \sum_{i=1}^{N} w_i \mathbf{c}(\mathbf{r}(t_i), \mathbf{d}) + \left(1 - \sum_{i=1}^{N} w_i\right) \mathbf{c}_{\text{bg}}, \quad (1)$$

where the weights of color contributions are $w_i = T(t_i)\rho(t_i)$, defined with

$$\rho(t_i) = 1 - \exp(-\sigma(\mathbf{r}(t_i))(t_{i+1} - t_i)) \quad (2)$$

and

$$T(t_i) = \prod_{j=1}^{i-1}(1 - \rho(t_j)) \qquad (3)$$

is the accumulated transmittance function, *i.e.* the probability of the ray $\mathbf{r}(t)$ starting at camera center $\mathbf{o}$ and reaching coordinate $\mathbf{r}(t_i)$ without being absorbed. The $\mathbf{c}_{\text{bg}}$ is the background color, which we set to white.

Similarly, one can compute the expected depth as

$$\mathbf{D}(\mathbf{r}) = \frac{\sum_{i=1}^{N} w_i t_i}{\sum_{i=1}^{N} w_i}. \qquad (4)$$

The density and color fields are optimized to reduce the photometric reconstruction loss, *e.g.* the L2 difference between input images and renderings from the same views is

$$\mathcal{L}_{\text{photo}}(\sigma, \mathbf{c}) = \sum_{i=1}^{\mathcal{I}} ||I_i - \mathbf{C}_i||_2. \qquad (5)$$

The weights of color contributions w_i in Eq. 5 can be regularized to have compact distribution [2]:

$$\mathcal{L}_{\text{dist}} = \frac{1}{D(\mathbf{r})}\left(\sum_{i,j} w_i w_k \left|\frac{t_i + t_{i+1}}{2} - \frac{t_j + t_{j+1}}{2}\right| \right.$$
$$\left. + \frac{1}{3}\sum_{i=1}^{N} w_i^2(t_{i+1} - t_i)\right), \qquad (6)$$

where we deviate from the original formulation by dividing through by the expected depth for the ray, which has the effect of increasing the strength of this regularizer for geometry that is close to the camera.

We also encourage the weights to sum to unity, because in real scenes we always expect a ray to be absorbed fully by the scene geometry:

$$\mathcal{L}_{\text{fg}} = \left(1 - \sum_{i=1}^{N} w_i\right)^2. \qquad (7)$$

In the few-view case, NeRFs frequently collapse to a degenerate solution in which each camera is fully or partially "covered up" with a copy of the corresponding training image. To prevent this, we introduce a regularization approach in which the placement of density that is contained in only one view frustum is penalized as

$$\mathcal{L}_{\text{fr}} = \sum_i w_i \mathbf{1}(n_i <= 1), \qquad (8)$$

where n_i is the number of training view frustums in which the point along the ray $\mathbf{r}(t_i)$ is contained, so that only weights which lie in fewer than two training frustums are included in the sum. This reflects our prior that most of the scene should be within the frustum of more than one of the training views.

Combining these geometric regularizers into a loss function already gives a very strong baseline,

$$\mathcal{L}_{\text{geom}} = \mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\mathcal{L}_{\text{fg}} + \lambda_{\text{fr}}\mathcal{L}_{\text{fr}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}. \qquad (9)$$

The λ coefficients control the contributions of the regularizers. In our experiments we refer to this combination of losses as our "geometric baseline".

3.2. Score functions and DDMs

Per Bayes' theorem, the *a posteriori* probability of density and color fields given training views $\mathcal{I}$ is

$$p(\sigma, \mathbf{c}|\mathcal{I}) \propto p(\mathcal{I}|\sigma, \mathbf{c})p(\sigma, \mathbf{c}), \qquad (10)$$

where we drop the normalizing constant since it depends only on $\mathcal{I}$. The log-posterior is

$$\log(p(\mathcal{I}|\sigma, \mathbf{c})) + \log(p(\sigma, \mathbf{c})). \qquad (11)$$

In practice, we are interested in maximizing $p(\sigma, \mathbf{c}|\mathcal{I})$ with stochastic gradient descent, which only requires computation of the gradient of the log-likelihood $\nabla_{\sigma,\mathbf{c}} \log(p(\mathcal{I}|\sigma, \mathbf{c}))$ and the gradient of the log-prior $\nabla_{\sigma,\mathbf{c}} \log(p(\sigma, \mathbf{c}))$, *i.e.* the score function. Notice that explicit computation of the probabilities of the density and color fields $p(\sigma, \mathbf{c})$ is not required. Below, we describe how DDMs are learned and their relation to the score function.

The forward diffusion process progressively adds small Gaussian noise to a data sample $\mathbf{x}_0 \sim q(\mathbf{x})$ to produce progressively noisier versions, so

$$\mathbf{x}_\tau = \sqrt{\alpha_\tau}\mathbf{x}_{\tau-1} + \sqrt{\beta_\tau}\epsilon_{\tau-1}, \qquad (12)$$

where $\epsilon_{\tau-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\alpha_\tau = 1 - \beta_\tau$, *i.e.* the variances $\{\beta_\tau\}_{\tau=1}^{\mathcal{T}}$ control the noise schedule. As the noise function is Gaussian, it follows from the reparameterization trick that

$$q(\mathbf{x}_\tau|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_\tau; \sqrt{\bar{\alpha}_\tau}\mathbf{x}_0, (1 - \bar{\alpha}_\tau\mathbf{I})), \qquad (13)$$

where $\bar{\alpha}_\tau = \prod_{s=0}^{\tau} \alpha_s$, allowing efficient generation of noised samples for arbitrary τ. As $\mathcal{T} \to \infty$ the distribution of noised samples $\mathbf{x}_{\mathcal{T}}$ is equivalent to an isotropic unit Gaussian.

The DDM [8, 20, 31] is tasked to learn the reverse diffusion process:

$$p(\mathbf{x}_{\tau-1}|\mathbf{x}_\tau) = \mathcal{N}(\mathbf{x}_{\tau-1}; \mu(\mathbf{x}_\tau, \tau), \tilde{\beta}_\tau\mathbf{I})), \qquad (14)$$

where $\tilde{\beta}_\tau = (1 - \bar{\alpha}_{\tau-1})\beta_\tau/(1 - \bar{\alpha}_\tau)$.

Since $\mathbf{x}_\tau$ is available as input to $\mu(\mathbf{x}_\tau, \tau)$, the mean $\mu(\mathbf{x}_\tau, \tau)$ can be computed by predicting noise $\epsilon_{\tau-1}$ from the noised input [8]:

$$\mu(\mathbf{x}_\tau, \tau) = \frac{1}{\sqrt{\alpha_\tau}}\left(\mathbf{x}_\tau - \frac{\beta_\tau}{\sqrt{1 - \bar{\alpha}_\tau}}\epsilon_\theta(\mathbf{x}_\tau, \tau)\right), \qquad (15)$$
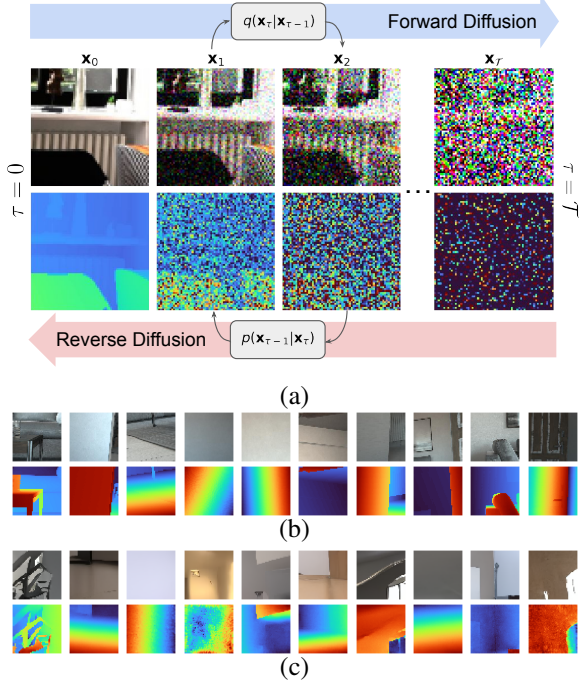
Figure 3. (a) Illustration of forward and reverse diffusion processes. (b) Example RGBD patches in the training set of the DDM model extracted from Hypersim dataset. (c) Example RGBD patches generated with our DDM model trained on Hypersim dataset. Depths are shown as normalized inverse depths for visualization purposes. The noise in the samples is due to noise that is injected during the sampling process.

using a neural network $\epsilon_\theta(\mathbf{x}_\tau, \tau)$.

Thus, one can learn the reverse diffusion process by training a neural network $\epsilon_\theta(\mathbf{x}_\tau, \tau)$ to estimate noise given a noised input and noise-level using the loss function:

$$\mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_\tau}{2\alpha_\tau(1 - \bar{\alpha}_\tau)} ||\epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_\tau}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_\tau}\epsilon, \tau \right) || \right],$$
(16)

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Fig. 3 (a) illustrates the forward and backwards processes.

Importantly, it was shown in [8, 37] that a DDM noise estimator has a connection to score matching [10, 32, 33] and is proportional to the score function:

$$\epsilon_\theta(\mathbf{x}_\tau, \tau) \propto -\nabla_\mathbf{x} \log p(\mathbf{x}).$$
(17)

Hence, taking steps in the negative direction to the noise predicted by the model is equivalent to moving towards the modes of the data distribution. This can be used to generate samples from the data distribution using Langevin dynamics [8, 32, 42].

In this work, we want to use a DDM model as a score function estimator to regularize NeRF reconstructions according to eq. 11. Hence, we model a prior over $(\sigma, \mathbf{c})$ by

modeling the score function over the distribution of RGBD patches $\epsilon_\theta(\{\mathbf{C}(\mathbf{r}), \mathbf{D}(\mathbf{r})|\mathbf{r} \in P\})$, where P is a set of rays that pass through a random 48×48 patch of pixels cast from a random camera. To allow control of the magnitude of the gradients, we further normalize the output of $\epsilon_\theta(\{\mathbf{C}(\mathbf{r}), \mathbf{D}(\mathbf{r})|\mathbf{r} \in P\})$, and refer to this regularization function as ϵ_θ (see supplementary for details).

To train our DDM we use *Hypersim* [26], a photorealistic synthetic dataset for indoor scene understanding with ground truth images and depth maps. Specifically, we sample 48×48 patches of images and depth maps to generate training data for the DDM (removing problematic images and scenes as per dataset instructions); see Fig. 3(b) for examples. Fig. 3(c) shows samples of RGBD patches generated by our DDM model. The quality of samples indicates that DDM successfully learns the data distribution of the RGBD Hypersim patches.

3.3. Regularizing NeRFs with DDMs

The gradient of the log-posterior (11), which forms our loss function, is

$$\nabla \log p(\sigma, \mathbf{c}|\mathcal{I}) = \nabla \log p(\sigma, \mathbf{c}) + \nabla \log p(\mathcal{I}|\sigma, \mathbf{c}).$$
(18)

By plugging (17) into the above, we can use a diffusion model as a prior over $(\sigma, \mathbf{c})$. For the second term on the RHS we use loss in eq 9, resulting in the following gradient for our loss function:

$$\nabla \mathcal{L} = \nabla \mathcal{L}_{\text{photo}} + \lambda_{\text{fg}} \nabla \mathcal{L}_{\text{fg}} + \lambda_{\text{fr}} \nabla \mathcal{L}_{\text{fr}} + \lambda_{\text{dist}} \nabla \mathcal{L}_{\text{dist}} - \lambda_{\text{DDM}} \epsilon_\theta,$$
(19)

where λ_{DDM} controls the weight of the our regularizer.

During NeRF optimization we compute the gradient of the loss as per eq. 19 and backpropagate as usual to obtain gradients for the NeRF density and color field parameters.

3.4. Implementation Details

We use the training protocol of [8, 39] to train our DDM model. We optimize the DDM for 650,000 steps with batch size 32 on 1 GPU.

We use the torch-ngp [36] implementation of Instant NGP [19] with the tiny-cuda-nn [18] back-end as the NeRF model for our experiments. NeRFs are optimized for 12,000 steps, where the first 2500 steps are optimized with $\lambda_{\text{dist}} = 0$ and the diffusion time parameter τ smoothly interpolates from 0.1 to 0, hence we set $\bar{\alpha}_\tau = \cos(0.5\pi(\tau + 0.008)/1.008)$ and other variables are derived accordingly. By scheduling τ this way the diffusion model is conditioned to expect progressively less noisy inputs as the NeRF trains and generates increasingly more accurate colors and depths. After 3000 steps, λ_{dist} linearly increases from 0 until it reaches its maximum value at 8000 steps, where the maximum value is 1×10^{-4} for the DTU dataset and 1.5×10^{-5} for the LLFF dataset. We empirically

	Method	Setting	PSNR ↑			SSIM ↑			LPIPS ↓			Average ↓		
			3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
LLFF	mip-NeRF [1]	Optimized per Scene	14.62	20.87	24.26	0.351	0.692	0.805	0.495	0.255	0.172	0.246	0.114	0.073
	DietNeRF [11]	Optimized per Scene	14.94	21.75	24.28	0.370	0.717	0.801	0.496	0.248	0.183	0.240	0.105	0.073
	PixelNeRF ft [45]	DTU + ft per Scene	16.17	17.03	18.92	0.438	0.473	0.535	0.512	0.477	0.430	0.217	0.196	0.163
	MVSNeRF ft [3]	DTU + ft per Scene	17.88	19.99	20.47	0.584	0.660	0.695	0.327	0.264	0.244	0.157	0.122	0.111
	RegNeRF [21]	Optimized per Scene	19.08	21.10	24.86	0.587	0.760	0.820	0.336	0.206	0.161	0.146	0.086	0.067
	Geometric Baseline	Optimized per Scene	19.88	24.28	25.10	0.590	0.765	0.802	0.192	0.101	0.084	0.118	0.071	0.060
	DiffusioNeRF (Ours)	Optimized per Scene	19.79	23.79	25.02	0.568	0.747	0.785	0.209	0.114	0.096	0.127	0.075	0.064
DTU	mip-NeRF [1]	Optimized per Scene	8.68	16.54	23.58	0.571	0.741	0.879	0.353	0.198	0.092	0.323	0.148	0.056
	DietNeRF [11]	Optimized per Scene	11.85	20.63	23.83	0.633	0.778	0.823	0.314	0.201	0.173	0.243	0.101	0.068
	PixelNeRF ft [45]	DTU + ft per Scene	18.95	20.56	21.83	0.710	0.753	0.781	0.269	0.223	0.203	0.125	0.104	0.090
	MVSNeRF ft [3]	DTU + ft per Scene	18.54	20.49	22.22	0.769	0.822	0.853	0.197	0.155	0.135	0.113	0.089	0.069
	RegNeRF [21]	Optimized per Scene	18.89	22.20	24.93	0.745	0.841	0.884	0.190	0.117	0.089	0.112	0.071	0.047
	Geometric Baseline	Optimized per Scene	13.60	16.43	22.01	0.661	0.759	0.853	0.212	0.147	0.071	0.185	0.092	0.056
	DiffusioNeRF (Ours)	Optimized per Scene	16.20	20.34	25.18	0.698	0.818	0.883	0.160	0.093	0.046	0.135	0.052	0.033

Table 1. DiffusioNeRF *vs*. SOTA in novel view synthesis task on LLFF and DTU datasets with few input views [21, 45]. We report scores on PSNR, SSIM, LPIPS and Average metrics averaged over all 8 scenes when NeRFs are fitted with 3, 6 and 9 training views. For each view/metric combination the **first** and underlined second scores are highlighted.

found that this schedule of τ and regularization weights produces best results. On a single Nvidia A100 GPU our NeRF model trains in approximately 30 minutes per scene.

Furthermore, 25% of the time we use a training pose for patch rendering, and sample the RGB component of the RGBD patch directly from the training image. This is helpful in the early stages, when NeRF renderings are not yet accurate.

4. Experiments

Datasets We experiment on two datasets: LLFF and DTU.

The *LLFF* [16] dataset has 8 scenes with 20-62 images per scene captured with a handheld camera. The scenes are reconstructed with COLMAP [30] to estimate camera intrinsics, camera poses and the 3D bounds of the scenes. A few images are used for training and test images are used to evaluate novel view synthesis quality. We select LLFF for evaluations as it allows comparison against other SOTA NeRF models, such as RegNeRF [21].

The *DTU* [12] dataset consists of images of objects placed on a table against black background. Images and depth maps are captured with structured light scanner mounted on an industrial robot arm. The dataset provides images, poses, and ground truth point clouds for evaluation.

For novel-view synthesis with few view setting on DTU, we use the test set of 15 scans of PixelNeRF [45], allowing comparison against other methods.

We use the test set of 15 scans defined in [23, 43, 46] to evaluate geometry quality, *e.g.* via the surface method of evaluation as described in UNISURF [23]. Traditionally, geometry estimated by the density field of a NeRF may not allow accurate surface reconstruction compared to occupancy and SDF-based approaches [23], which score higher on DTU, *e.g.* [23, 43, 44, 46].

Metrics For the task of novel-view synthesis, hold-out

views of the scene are used as ground truth to compare against synthesized views. Image similarity metrics such as PSNR, SSIM [41] and LPIPS [47] are measured for each test view and average score per each scene is reported. We also report an "Average" score, specifically the geometric mean of the three metrics as per [1]: $\sqrt[3]{10^{-\text{PSNR}/10} \cdot \sqrt{1-\text{SSIM}} \cdot \text{LPIPS}}$.

For the geometry estimation task, we convert an isosurface of the density field into a mesh using the marching cubes. The mesh is culled to retain only parts that are visible in at least one training view and the background surfaces are masked out. We then sample the mesh to generate a point cloud, and report the average chamfer $L1$ distance between the estimated and ground truth point clouds.

4.1. Evaluations

Table 1 show a comparison of our geometric baseline and our model against SOTA methods on LLFF and DTU datasets when trained with 3, 6 and 9 views. When the number of views is low, the regularizer can have a large impact on the final result, which allows easier comparison of regularizers. As seen from table 1, the geometric baseline and our method both score favorably to other methods, achieving best scores in PSNR, LPIPS and Average metrics. Our geometric baseline has higher metrics on LLFF, however there are artifacts in the generated test views that can be seen in Fig. 4. Our diffusion model-based method generates more plausible depths compared to the geometric baseline, see section 4.2. One side-effect is over-smoothing of thin-structures (*e.g.* the top row in Fig. 4). It is also noteworthy that test views contain parts of the scene that are not visible in any of the training views. These occluded parts of the scene can impact reconstruction scores significantly (see supplementary for details).

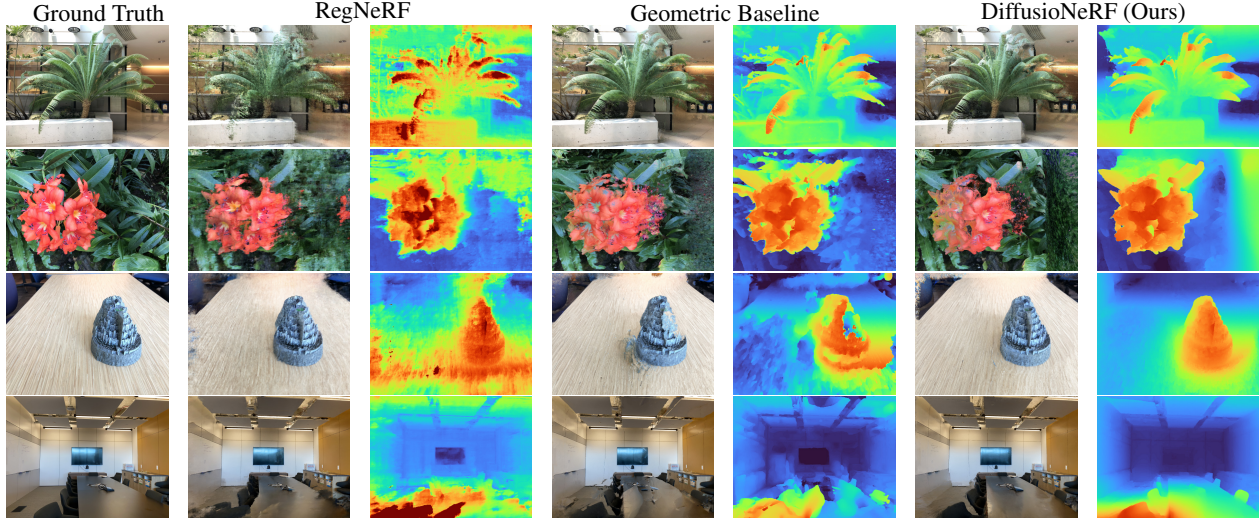Table 2 shows an evaluation of reconstruction quality on

Figure 4. Qualitative results for the task of novel view synthesis on LLFF dataset. NeRF models are trained with 3 views and rendered from one of test views. Our DDM model encourages more realistic geometry as seen in the depth maps.

15 scans of the DTU dataset when NeRFs are fitted with all views. In the large number of views regime, the priors are less important as training views provide more information about the scene. Nevertheless, the priors should not introduce any undesirable artifacts and can help with ambiguous regions such as textureless table. Despite DDM being trained on images of indoor room-sized scenes, it shows good generalization to the object-centric reconstruction task. Our density-based method performs adequately when compared to occupancy and SDF-based methods.

In Fig. 5 the qualitative results indicate that density based methods struggle with shiny objects (rows 2 and 4) but can have higher fidelity geometry on diffuse and textured surfaces (rows 1 and 3). The textured regions alone are not sufficient for high quality output, *e.g.* our geometric baseline struggles to complete the geometry of a house in row 1, and our DDM model provides a complementary signal to the geometric regularizers resulting in fewer holes and smoother surfaces.

4.2. Ablation studies

In table 3 we show contributions of each of our optimization terms evaluated on LLFF and DTU datasets for novel

SDF-based Methods	Mean Chamfer-$L1\downarrow$	NeRF-based Methods	Mean Chamfer-$L1\downarrow$
UNISURF [23]	1.02	Instant NGP [19]	1.71
NeuS [40]	0.84	NeRF [17]	1.49
VolSDF [43]	0.86	Geometric Baseline	1.36
MonoSDF [46]	0.73	**DiffusioNeRF**	1.21

Table 2. DiffusioNeRF *vs*. SOTA in geometry reconstruction on the DTU dataset with all views [5].

view synthesis and reconstruction quality. As reported, the geometric baseline scores favorably on the LLFF dataset, but has issues in geometry as reflected in DTU scores. Qualitative results in Fig. 4 demonstrate that the geometry estimated by the geometric baseline is not realistic, even if the appearance scores are high. Our DDM-based approach improves on DTU scores, but its performance on the novel view synthesis metrics is hampered by its tendency to introduce details in areas of the scene that are not pictured in any training view.

In table 3 we also show ablations of some of the finer details of our model. This table suggests that a model trained on 24×24 patches outperforms a model trained on 48×48 patches on LLFF, but underperforms on DTU.

The ablations show the significance of feeding patches from input images to DDM 25% of the time during NeRF fitting. It can be especially important early on, when rendered patches are very different from input images.

Unsurprisingly, reducing the amount of training data for the DDM (only using 20% of the Hypersim scenes) slightly reduces the scores. The RGB-only regularization with DDMs is similar to RegNeRF's normalizing flow model regularization, but with larger patch sizes. Interestingly, the RGBD regularizer trained with 20% of the data is still better than the RGB-only regularizer that was trained with 100% of the data. The last two rows of the ablation show that careful scheduling of τ and DDM gradient weights is necessary to produce good results. This is an active area of research, having previously been noted in [24]. The DDM weight λ_{DDM} trades off the accuracy of reconstruction around thin structures against the overall depth smoothness.
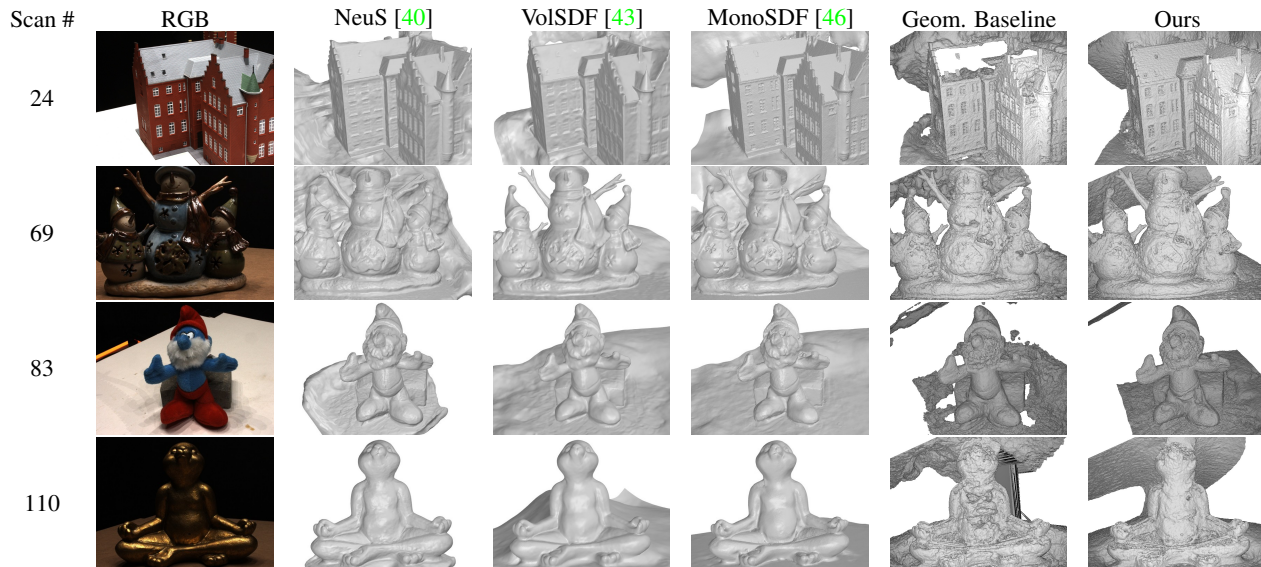
Figure 5. Qualitative comparison of our method against SOTA on geometry reconstruction evaluated on DTU dataset.

Method	LLFF Average ↓			DTU Average ↓			Chamfer-$L1$ ↓
	3-view	6-view	9-view	3-view	6-view	9-view	All views
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}}$	0.210	0.128	0.090	0.203	0.142	0.119	2.87
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\nabla\mathcal{L}_{\text{fg}}$	0.210	0.128	0.090	0.195	0.126	0.092	1.71
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\nabla\mathcal{L}_{\text{fg}} + \lambda_{\text{fr}}\nabla\mathcal{L}_{\text{fr}}$	0.135	0.089	0.072	0.215	0.128	0.093	1.71
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\nabla\mathcal{L}_{\text{fg}} + \lambda_{\text{fr}}\nabla\mathcal{L}_{\text{fr}} - \lambda_{\text{DDM}}\epsilon_\theta$	0.145	0.085	0.066	0.190	0.097	0.072	1.67
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\nabla\mathcal{L}_{\text{fg}} + \lambda_{\text{fr}}\nabla\mathcal{L}_{\text{fr}} + \lambda_{\text{dist}}\nabla\mathcal{L}_{\text{dist}}$	0.118	0.071	0.060	0.185	0.092	0.056	1.36
$\nabla\mathcal{L} = \nabla\mathcal{L}_{\text{photo}} + \lambda_{\text{fg}}\nabla\mathcal{L}_{\text{fg}} + \lambda_{\text{fr}}\nabla\mathcal{L}_{\text{fr}} + \lambda_{\text{dist}}\nabla\mathcal{L}_{\text{dist}} - \lambda_{\text{DDM}}\epsilon_\theta$	0.127	0.075	0.064	0.135	0.052	0.033	1.21
DDM regularizer using 24x24 patches	0.126	0.074	0.061	0.195	0.068	0.043	1.22
24x24 patch DDM & NeRF fitted with $4 \times \lambda_{\text{DDM}}$	0.129	0.074	0.062	0.260	0.080	0.050	1.22
Patches from input images are not given to DDM	0.139	0.078	0.066	0.159	0.063	0.049	1.91
DDM trained with 20% of Hypersim scenes	0.132	0.078	0.066	0.163	0.057	0.035	1.65
RGB-only DDM regularizer	0.134	0.083	0.070	0.189	0.081	0.058	1.31
$\tau = 0$ (no schedule) during NeRF fitting	0.137	0.081	0.067	0.152	0.055	0.042	1.31
NeRF fitted with $4 \times \lambda_{\text{DDM}}$	0.146	0.088	0.076	0.220	0.134	0.071	2.56

Table 3. Ablation study of our method. Note that for DTU, λ_{fr} is set to 0, hence the 2nd and 3rd rows have identical scores on DTU. Geometric baseline corresponds to the model in the 5th row.

5. Conclusions

In this paper we address the problem of regularization of NeRFs. Our approach uses a DDM trained on RGBD patches to approximate a score function, *i.e.* the gradient of the logarithm of an RGBD patch distribution. Experimentally, we demonstrate that the proposed regularization scheme improves performance on novel view synthesis and 3D reconstruction.

While we show regularization using color and depth patches as input, the proposed framework is versatile and can be used to regularize the 3D voxel grid of densities, density weights sampled along the ray, *etc*. Indeed, instead of generating RGBD patches, we can generate 3D voxel blocks of densities to train a DDM and use it during NeRF optimization to regularize the density field directly. Early results are shown in the supplementary materials.

One avenue of future work is formulating a principled combination of the DDM gradient with the NeRF objective to avoid heuristics-based τ and gradient scheduling.

Our work is focused on NeRF optimization, however the general approach of using DDMs as a regularizer could potentially be used for other tasks that are optimized with gradient descent, *e.g.* self-supervised monocular depth estimation [5], or self-supervised stereo matching [49, 50].

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 *ICCV*, 2021. 6

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, 2022. 2, 4

[3] 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. In *ICCV*, 2021. 2, 6

[4] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation. In *ICLR*, 2020. 3

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017. 7, 8

[6] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *ICML*, 2020. 2

[7] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV*, 2021. 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020. 3, 4, 5

[9] Jonathan Ho, Tim Salimans, Alexey A Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video Diffusion Models. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022. 3

[10] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 2005. 5

[11] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *ICCV*, 2021. 2, 6

[12] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large Scale Multi-view Stereopsis Evaluation. In *CVPR*, 2014. 6

[13] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE TPAMI*, 2020. 3

[14] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *ICLR*, 2020. 3

[15] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. In *NeurIPS*, 2020. 2

[16] 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 TOG*, 2019. 1, 6

[17] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 1, 2, 7

[18] Thomas Müller. Tiny CUDA Neural Network Framework, 2021. github.com/nvlabs/tiny-cuda-nn. 5, 8

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 2022. 2, 5, 7

[20] Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *ICML*, 2021. 3, 4

[21] 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 *CVPR*, 2022. 1, 2, 3, 6

[22] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*, 2020. 2

[23] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *ICCV*, 2021. 2, 6, 7

[24] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*, 2023. 3, 7

[25] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-conditional Image Feneration with CLIP Latents. *arXiv preprint arXiv:2204.06125*, 2022. 3

[26] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. In *ICCV*, 2021. 5

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

[28] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *NeurIPS*, 2022. 3

[29] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*, 2022. 2

[30] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 6

[31] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *ICML*, 2015. 3, 4

[32] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *NeurIPS*, 2019. 5

[33] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *Uncertainty in Artificial Intelligence*, 2020. 5

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*, 2021. 3

[35] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*, 2020. 2

[36] Jiaxiang Tang. Torch-NGP: a PyTorch implementation of Instant-NGP, 2022. github.com/ashawkey/torch-ngp. 5, 8

[37] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 2011. 5

[38] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. NeuRIS: Neural Reconstruction of Indoor Scenes Using Normal Priors. In *ECCV*, 2022. 2

[39] Phil Wang. Denoising Diffusion Probabilistic Model in Pytorch, 2022. github.com/lucidrains/denoising-diffusion-pytorch. 5, 8

[40] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *NeurIPS*, 2021. 2, 7, 8

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE TIP*, 2004. 6

[42] Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, 2011. 3, 5

[43] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. In *NeurIPS*, 2021. 2, 6, 7, 8

[44] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *NeurIPS*, 2020. 2, 6

[45] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 6

[46] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. In *NeurIPS*, 2022. 2, 6, 7, 8

[47] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018. 6

[48] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination. *ACM TOG*, 2021. 2

[49] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-Supervised Learning for Stereo Matching with Self-Improving Ability. *arXiv preprint arXiv:1709.00930*, 2017. 8

[50] Chao Zhou, Hong Zhang, Xiaoyong Shen, and Jiaya Jia. Unsupervised Learning of Stereo Matching. In *ICCV*, 2017. 8