SparseFusion: Distilling View-conditioned Diffusion for 3D Reconstruction

Zhizhuo Zhou Shubham Tulsiani
Carnegie Mellon University
{zhizhuo, shubhtuls}@cmu.edu
https://sparsefusion.github.io/

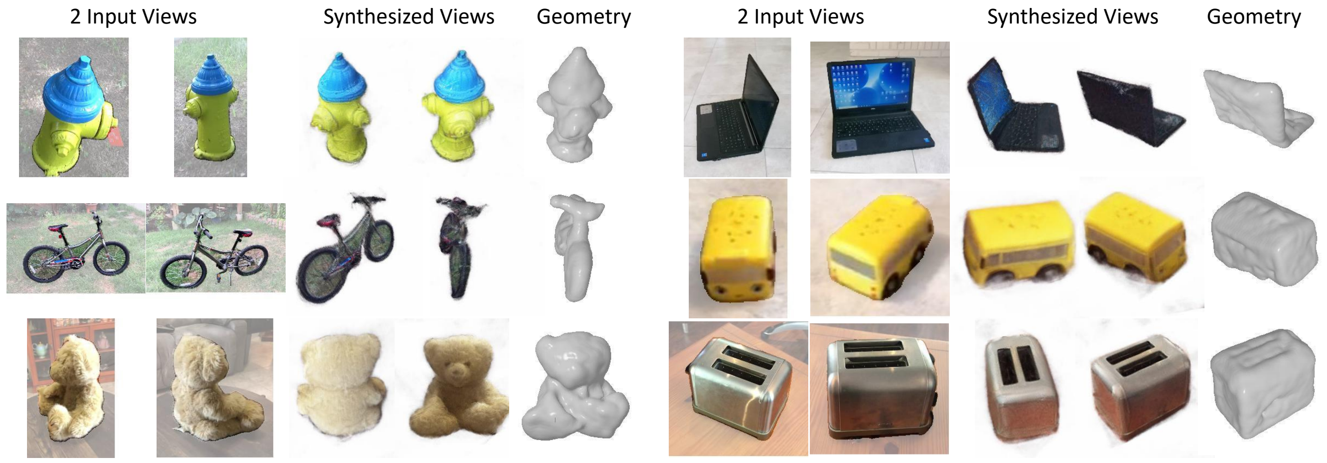

Figure 1. **Sparse-view Reconstruction.** We present SparseFusion, an approach for 3D reconstruction given a few (*e.g.* just two) segmented input images with known relative pose. SparseFusion is able to generate a 3D consistent neural scene representation, enabling us to render novel views and extract the underlying geometry, while being able to generate detailed and plausible structures in uncertain or unobserved regions (*e.g.* front of hydrant, teddy's face, back of laptop, or left side of toybus). Please see project page for 360-degree visualizations.

Abstract

We propose SparseFusion, a sparse view 3D reconstruction approach that unifies recent advances in neural rendering and probabilistic image generation. Existing approaches typically build on neural rendering with re-projected features but fail to generate unseen regions or handle uncertainty under large viewpoint changes. Alternate methods treat this as a (probabilistic) 2D synthesis task, and while they can generate plausible 2D images, they do not infer a consistent underlying 3D. However, we find that this trade-off between 3D consistency and probabilistic image generation does not need to exist. In fact, we show that geometric consistency and generative inference can be complementary in a mode-seeking behavior. By distilling a 3D consistent scene representation from a view-conditioned latent diffusion model, we are able to recover a plausible 3D representation whose renderings are both accurate and realistic. We evaluate our approach across 51 categories in the CO3D dataset and show that it outperforms existing methods, in both distortion and perception metrics, for sparse-view novel view synthesis.

1. Introduction

Consider the two images of the teddybear shown in Figure 1 and try to imagine the underlying 3D object. Relying on the direct visual evidence in these images, you can easily infer that the teddybear is white, has a large head, and has small arms. Even more remarkably, you can imagine beyond the directly visible to estimate a *complete* 3D model of this object *e.g.* forming a mental model of the teddy's face with (likely black) eyes even though these were not observed. In this work, we build a computational approach that can similarly predict 3D from just a few images – by integrating visual measurements and priors via probabilistic modeling and then seeking likely 3D modes.

A growing number of recent works have studied the related tasks of *sparse-view* 3D reconstruction and novel-view synthesis, *i.e.* inferring 3D representations and/or synthesizing novel views of an object given just a few (typically 2-3) images with known relative camera poses. By leveraging data-driven priors, these approaches can learn to efficiently leverage multi-view cues and infer 3D from sparse views. However, they still yield blurry predictions under large viewpoint changes and cannot hallucinate plausible content in unobserved regions. This is because they do not account for the uncertainty in the outputs *e.g.* the unobserved nose of a teddybear maybe either red or black, but these methods, by reducing inference to independent pixel-wise or point-wise predictions, cannot model such variation.

In this work, we propose to instead model the *distribution* over the possible images given observations from some context views and an arbitrary query viewpoint. Leveraging a geometrically-informed backbone that computes pixel-aligned features in the query view, our approach learns a (conditional) diffusion model that can then infer detailed plausible novel-view images. While this probabilistic image synthesis approach allows the generation of higher quality image outputs, it does not directly yield a 3D representation of underlying the object. In fact, the (independently) sampled outputs for each query view often do not even correspond to a consistent underlying 3D *e.g.* if the nose of the teddybear is unobserved in context views, one sampled query view may paint it red, while another one black.

To obtain a consistent 3D representation, we propose a *Diffusion Distillation* technique that 'distills' the predicted distributions into an instance-specific 3D representation. We note that the conditional diffusion model not only gives us the ability to sample novel-view images, but also to (approximately) compute the likelihood of a generated one. Using this insight, we optimize an instance-specific (neural) 3D representation by maximizing the diffusion-based likelihood of its renderings. We show that this leads to a mode-seeking optimization that results in more accurate and realistic renderings, while also recovering a 3D-consistent representation of the underlying object. We demonstrate our approach on over 50 real-world categories from the CO3D dataset and show that our method allows recovering accurate 3D and novel views given as few as 2 images as input – please see Figure 1 for sample results.

2. Related Work

Instance-specific Reconstruction from Multiple Views. Leveraging Structure-from-Motion [29, 33] to recover camera viewpoints, early Multi-view-Stereo (MVS) [5, 30] methods could recover dense 3D outputs. Recent neural incarnations of these [17, 43, 44] use volumetric rendering to learn a compact neural scene representation. Follow up works [2, 4, 18] seek to make the training and rendering orders of magnitudes faster. However, these methods require many input views, making them impractical for real world applications. While some works [8, 19, 47] seek to reduce the input views required, they still do not make predictions for unseen regions.

Single-view 3D Reconstruction. The ability to predict 3D geometry (and appearance) beyond the visible is a key goal for single-view 3D prediction methods. While these approaches have pursued prediction of different 3D representations *e.g.* volumetric [3, 6, 10, 39, 45], mesh-based [7, 12], or neural implicit [14, 16, 40] 3D, the use of a single input image fundamentally limits the details that can be predicted. Moreover, these methods do not prioritize view-synthesis as a goal. While our approach similarly learns data driven inference, we aim for a more detailed reconstruction and high quality novel-view renderings.

	Single-instance				Re-projection				Latent			Ours
	NeRF [17]	RegNeRF [19]	VolSDF [43]	NeRS [47]	IBRNet [42]	PixelNeRF [46]	NerFormer [22]	GPNR [35]	LFN [31]	SRT [28]	ViewFormer [13]	SparseFusion
1) Real data	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓
2) Sparse-views	×	✓	✓	✓	×	✓	✓	×	✓	✓	✓	✓
3) 3D consistent	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
4) Generalization	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓
5) Generate unseen	×	×	×	×	×	×	×	×	×	×	✓	✓

Table 1. **Comparison with prior methods.** The rows indicate whether each method: 1) has been demonstrated on real world data, 2) works with sparse (2-6) input views, 3) generates geometrically consistent views, 4) generalizes to new scene instances, and 5) hallucinates unseen regions.

Generalizable view Synthesis from Fewer Views. Novel view synthesis (NVS), while similar to reconstruction, has slightly different roots. Earlier works [37, 50] frame NVS as a 2D problem, using deep networks to make predictions from global encodings. Recent approaches combine deep networks with various rendering formulations [28,31,32]. Strong performing approaches often leverage re-projected features from input views with volumetric rendering [22, 38, 46] or image based rendering [1, 35, 42]. While feature re-projection methods are 3D consistent, they regress to the mean and fail to produce perceptually sharp outputs. Another line of work [13, 24] revisits NVS as a probabilistic 2D generation task, using newer generative backbones to offer better perceptual quality at the cost of larger distortion and 3D consistency. See Table 1 for a comparison of our method against existing approaches.

Diffusion Models. Several works extend upon denoising diffusion models [11, 34] to achieve impressive applications, such as generating images from text [21,27] and placing a foreground object in different backgrounds [26]. In this work, we leverage these class of models for (probabilistic) novel view synthesis while using geometry-aware features as conditioning. Inspired by the impressive results in DreamFusion [20] which optimized 3D scenes using text-conditioned diffusion models, we propose a view-conditioned diffusion distillation mechanism to similarly extract 3D modes in the sparse view reconstruction task.

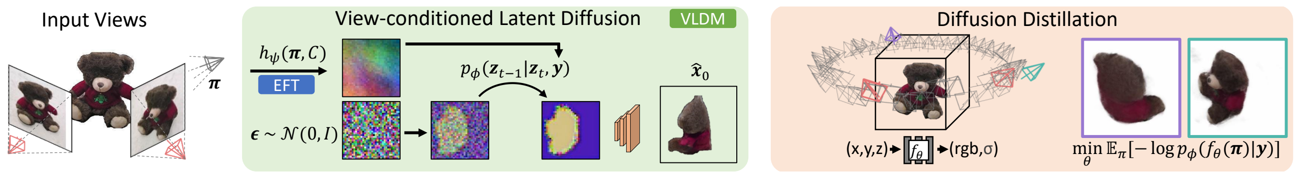
Input Views
$\boldsymbol{\pi}$
View-conditioned Latent Diffusion
VLDM
$h_\psi(\boldsymbol{\pi}, C)$
EFT
$p_\phi(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{y})$
$\hat{\boldsymbol{x}}_0$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
Diffusion Distillation
(x,y,z)→f_θ→(rgb,σ)
$\min_\theta \mathbb{E}_\pi[-\log p_\phi(f_\theta(\boldsymbol{\pi})|\boldsymbol{y})]$

Figure 2. **Overview of SparseFusion.** SparseFusion comprises of two core components: a view-conditioned latent diffusion model (VLDM) and a diffusion distillation process that optimizes an Instant NGP [18, 36]. We use VLDM to model $p(\boldsymbol{x}|\boldsymbol{\pi}, C)$.

3. Background: Denoising Diffusion

Our method adopts and optimizes through denoising diffusion models [11], and here we give a brief summary of the key formulations used, and refer the reader to the appendix for further details.

Training Objective. One can learn denoising diffusion models by optimizing a variational lower bound on the log-likelihood of the observed data. Conveniently, this reduces to a training framework where one adds (time-dependent) noise to a data point $\boldsymbol{x}_0$, and then trains a network $\boldsymbol{\epsilon}_\phi$ to predict this noise given the noisy data point $\boldsymbol{x}_t$.

$$\mathcal{L}_{DM} = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon},t}\left[w_t \, ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t, t)||^2\right]$$
$$\text{where } \boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}; \;\; \boldsymbol{\epsilon} \sim \mathcal{N}(0,1) \tag{1}$$

Here, $\bar{\alpha}_t$ is a scheduling hyper-parameter, and the weights w_t depend on this learning schedule, but are often set to 1 to simplify the objective.

Interpretation as Reconstruction Error. The above noise prediction objective, which represents a bound on the log likelihood, can also be viewed as a reconstruction error. Concretely, given a noisy $\boldsymbol{x}_t$, the network prediction $\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t, t)$ can be interpreted as yielding a reconstruction for the original input, where the learning objective can we rewritten as a reconstruction error:

$$\hat{\boldsymbol{x}}_{0,t} = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t, t)}{\bar{\alpha}_t} \tag{2}$$

$$\mathcal{L}_{DM} = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon},t}\left[w'_t \, ||\hat{\boldsymbol{x}}_{0,t} - \boldsymbol{x}_0||^2\right] \tag{3}$$

While the above summary focused on unconditional diffusion models, they can be easily extended to infer conditional distributions $p(\boldsymbol{x}|\boldsymbol{y})$ by additionally using $\boldsymbol{y}$ as an input for the noise prediction network $\boldsymbol{\epsilon}_\phi$.

4. Approach

Given sparse-view observations of an object (typically 2-3 images with masked foreground) with known camera viewpoints, our approach aims to infer a (3D) representation capable of synthesizing novel views while also capturing the geometric structure. However, as aspects of the object maybe unobserved and its geometry difficult to precisely infer, direct prediction of 3D or novel views leads to implausibly blurry outputs in regions of uncertainty.

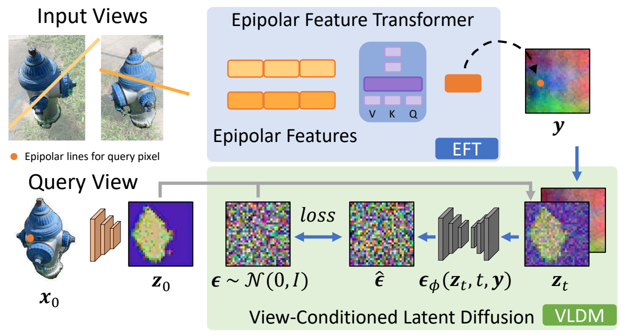
Input Views
Epipolar Feature Transformer
Epipolar lines for query pixel
V K Q
Epipolar Features
EFT
$\boldsymbol{y}$
Query View
loss
$\boldsymbol{x}_0$
$\boldsymbol{z}_0$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\hat{\boldsymbol{\epsilon}}$
$\boldsymbol{\epsilon}_\phi(\boldsymbol{z}_t, t, \boldsymbol{y})$
$\boldsymbol{z}_t$
View-Conditioned Latent Diffusion
VLDM

Figure 3. **View-conditioned Diffusion.** We show a diagram of our view-conditioned latent diffusion model. VLDM is conditioned on features $\boldsymbol{y}$, which is predicted by EFT.

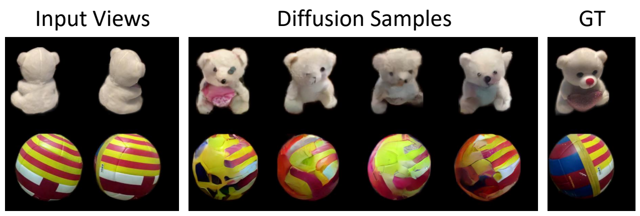

Figure 4. **Diffusion Samples.** Given the same input features, the reverse sampling process of diffusion model results in different predictions for unseen regions.

To enable plausible and 3D-consistent predictions, we instead take a two step approach as outlined in Figure 2. First, we learn a probabilistic view-synthesis model that, using geometry-guided diffusion, can model the *distribution* of images from query views given the sparse-view context (Section 4.1). While this allows generating detailed and diverse outputs, the obtained renderings lack 3D consistency. To extract a 3D representation, we propose a 3D neural distillation process that 'distills' the predicted view distributions into a consistent 3D mode (Section 4.2).

4.1. Geometry-guided Probabilistic View Synthesis

Given a target view pose $\boldsymbol{\pi}$ along with a set of reference images and their relative poses $C \equiv (\boldsymbol{x}_m, \boldsymbol{\pi}_m)$, we want to model the conditional distribution $p(\boldsymbol{x}|\boldsymbol{\pi}, C)$, from which

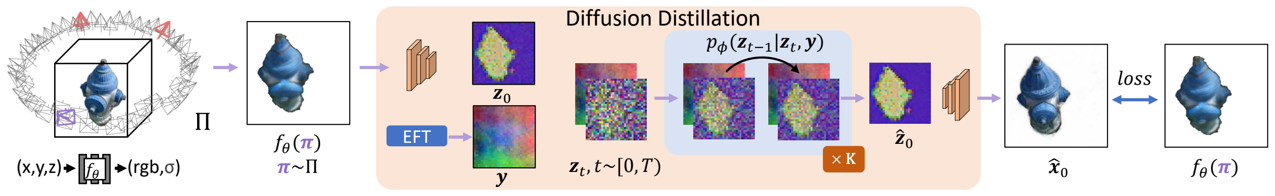

Figure 5. **Diffusion Distillation Diagram.** We optimize the parameters θ of an Instant NGP network such that rendered images $f_\theta(\boldsymbol{\pi})$ from $\boldsymbol{\pi} \sim \Pi$ are similar to VLDM predictions $\hat{\boldsymbol{x}}_0$, effectively seeking a mode in $p_\phi(\boldsymbol{x}|\boldsymbol{\pi}, C)$.

we can synthesize an image $\hat{\boldsymbol{x}}$. We illustrate our approach to modeling this distribution in Figure 3. First, we use an epipolar feature transformer (EFT) inspired by [35] as feature extractor to obtain a low resolution feature grid $\boldsymbol{y}$ in the view space of $\boldsymbol{\pi}$ given the context C. In conjunction, we train a view-conditioned latent diffusion model (VLDM) that models the distribution over novel-view images condition on these geometry-aware features.

4.1.1 Epipolar Feature Transformer

We build upon GPNR [35] to extract features from context C. GPNR learns a feedforward network, $g_\psi(\boldsymbol{r}, C)$, that predicts color given a query ray $\boldsymbol{r}$ by extracting features along its epipolar lines in all context images and aggregating the them with transformers. We make several modifications to GPNR to suit our needs. First, we replace the patch projection layer with a ResNet18 [9] convolutional encoder as we found the lightweight patch encodings, while suitable for small baseline view synthesis, are not robust under the sparse-view setting. Furthermore, we modify the last layer to predict both a RGB value and a feature vector. We denote the RGB branch as g_ψ and the feature branch as h_ψ. We refer to our modified epipolar patch-based feature transformer as **EFT** and present its color branch as a strong baseline.

We train the color branch of the EFT to minimize a simple reconstruction loss in Eq. 4, where $\boldsymbol{r}$ is a query ray sampled from $\boldsymbol{\pi}$, C is the set of reference images and their relative poses, and $I(\boldsymbol{r})$ is the ground truth pixel value.

$$\mathcal{L}_{EFT} = \sum_{\boldsymbol{r} \in R(\boldsymbol{\pi})} ||g_\psi(\boldsymbol{r}, C) - I(\boldsymbol{r})||^2 \tag{4}$$

4.1.2 View-conditioned Latent Diffusion Model

While EFT can directly predict novel views, the pixelwise prediction mechanism does not allow it to model the underlying probability distribution, thus resulting in blurry mean-seeking predictions under uncertainty. To model the distribution over plausible images we train a view-conditioned diffusion model to estimate $p(\boldsymbol{x}|\boldsymbol{\pi}, C)$ while using EFT as a geometric feature extractor. Instead of directly modeling the distribution in pixel space, we find it computationally efficient to do so in a lower-resolution latent space. We let $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$ and $x = \mathcal{D}(\boldsymbol{z})$ where $\boldsymbol{x}$ is an image and $\boldsymbol{z}$ is the corresponding latent code. Please see the appendix for details.

Given target view $\boldsymbol{\pi}$ and a set of input images C, we extract a 32 by 32 feature grid $\boldsymbol{y} = h_\psi(\boldsymbol{\pi}, C)$ using the EFT backbone. We train our VLDM to recover ground truth image latent $\boldsymbol{z_0}$ conditioned on $\boldsymbol{y}$. Following diffusion model training conventions [11, 21, 34], we optimize a simplified variational lower bound in Eq. 5.

$$\mathcal{L}_{VLDM} = \mathbb{E}_{\boldsymbol{z}, \boldsymbol{\epsilon}\ \mathcal{N}(0,1), t, \boldsymbol{y}} \left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{z_t}, t, \boldsymbol{y})||^2 \right] \tag{5}$$

Figure 3 shows a diagram of the training setup. Our VLDM model allows us to approximate $p(\boldsymbol{x}|\boldsymbol{\pi}, C)$, and enables drawing multiple sample predictions. In Figure 4, we see variations in VLDM predictions. Nevertheless, all predictions are plausible explanations for the target view given that majority of it is unseen.

4.2. Extracting 3D Modes via Diffusion Distillation

While the proposed VLDM gives us the ability to hallucinate unseen regions and make realistic predictions under uncertainty, it does not output a 3D representation. In fact, as it models the distribution over images, the views sampled from the VLDM do not (and should not!) necessarily correspond to a single underlying 3D interpretation. How can we then obtain an output 3D representation while preserving the high-quality of renderings?

3D Inference as Neural Mode Seeking. Our key insight is that the VLDM model not only allows us to sample plausible novel views, but the modeled distribution also gives us a mechanism to approximate the likelihood of a generated novel view. Building on this insight, we propose to distill the VLDM predictions to obtain an instance-specific 3D neural scene representation f_θ, such as NeRF [17] or Instant NGP (INGP) [18]. Intuitively, we want to arrive at a solution for f_θ such that its renderings $\boldsymbol{x} \equiv f_\theta(\boldsymbol{\pi})$ from arbitrary viewpoints $\boldsymbol{\pi}$ are likely under the conditional distribution modeled by the VLDM $p_\phi(\boldsymbol{x}|\boldsymbol{\pi}, C)$:

$$\min_\theta \mathbb{E}_{\boldsymbol{\pi}\sim\Pi} \ -\log\ p_\phi(f_\theta(\boldsymbol{\pi})|\boldsymbol{\pi}, C) \tag{6}$$

where we minimize the negative log-likelihood for images rendered with f_θ over cameras sampled from a prior camera distribution Π (constructed by assuming a circular camera trajectory and that all cameras look at a common center).

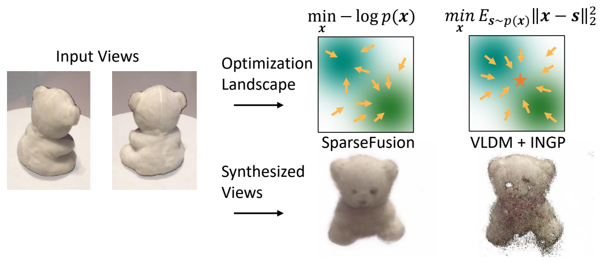

Figure 6. **Mode Seeking Visualization.** We show qualitative comparison between a mode-seeking (SparseFusion) and a mean-seeking (VLDM+INGP) objective.

We terms this process as 'neural mode seeking' as it encourages a representation which maximizes likelihood as opposed to minimizing distance to samples (mean seeking).

Neural Mode Seeking via Diffusion Distillation. Given a learned diffusion model, the reconstruction objective (Eq. 3) yields a bound on the log-likelihood of a data point $\mathbf{x}$. This approximation yields a simple mechanism for computing the likelihood of a (rendered) image $f_\theta(\boldsymbol{\pi})$ to be used in the mode-seeking optimization (Eq. 6):

$$-\log p(\boldsymbol{x}_0) \approx \mathbb{E}_{\boldsymbol{\epsilon},t}\left[w_t||\boldsymbol{z}_0 - \hat{\boldsymbol{z}}_{0,t}||^2\right] + C \tag{7}$$

where $\boldsymbol{z}_0 = \mathcal{E}(f_\theta(\boldsymbol{\pi}))$ is the latent of the rendered image, $t \sim (0, T]$, and $\hat{\boldsymbol{z}}_{0,t}$ is the predicted latent variable (analogous to $\hat{\boldsymbol{x}}_{0,t}$ in Eq. 2). Intuitively, this objective implies that if, after adding noise to obtain $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$, the denoising diffusion model predicts $\hat{\boldsymbol{x}}_0$ close to the original input, one has reached a mode under $p(\boldsymbol{x})$. We visualize the behavior of mode seeking versus mean seeking in Figure 6.

Multi-step Denoising and Image-space Reconstruction. In practice, we find that this single-step objective in Eq. 7 (*i.e.* directly predicting a reconstruction $\hat{\boldsymbol{z}}_{0,t}$ from $\boldsymbol{z}_0$) gives high-variance estimates. To make optimization more stable, we instead perform a multi-step denoising – instead of directly predicting $\hat{\boldsymbol{z}}_{0,t}$, we adaptively use multiple timesteps (up to 50 steps) $\mathcal{T} = (t_1, \cdots, t_k, t)$, and successively predict $\hat{\boldsymbol{z}}_{t_{k-1},t_k}$ (via [15]) *i.e.* predict a denoised estimate for time t_{k-1} given a sample from time t_k. We denote this reconstruction as $\hat{\boldsymbol{z}}_{0,\mathcal{T}}$ to highlight the multiple-step reconstruction. We find that using $\hat{\boldsymbol{z}}_{0,\mathcal{T}}$ instead of $\hat{\boldsymbol{z}}_{0,t}$ significantly improves the mode-seeking optimization. Finally, instead of computing the reconstruction error in latent space, we compute the objective in image space. Combining these modifications with the above objective, our final objective for optimizing for neural mode seeking with view conditioned diffusion models can be expressed as:

$$\mathcal{L}_{Distillation} = \mathbb{E}_{\boldsymbol{\pi},\boldsymbol{\epsilon},t}\left[w_t||f_\theta(\boldsymbol{\pi}) - \hat{\boldsymbol{x}}_{0,\mathcal{T}}||^2\right] \tag{8}$$

where $\hat{\boldsymbol{x}}_{0,\mathcal{T}} = \mathcal{D}(\hat{\boldsymbol{z}}_{0,\mathcal{T}})$, and $\hat{\boldsymbol{z}}_{0,\mathcal{T}}$ is the multi-step reconstruction from $\boldsymbol{z}_t$ – which is obtained by adding noise to $\boldsymbol{z}_0 = \mathcal{E}(f_\theta(\boldsymbol{\pi}))$. While $\hat{z}$ in the above objective does (indirectly) depend on the neural representation f_θ, we follow [20] in ignoring this dependence when computing parameter gradients. We show a diagram of multi-step denoising diffusion distillation in Figure 5.

5. Experiments

We demonstrate our approach on a challenging real world multi-view dataset CO3Dv2 [22], across 51 diverse categories. First, we compare SparseFusion against prior works, highlighting the benefit of our approach in sparse view settings. Then, we show the importance of diffusion distillation and its probabilistic mode-seeking formulation.

5.1. Experimental Setup

Dataset. We perform experiments on CO3Dv2 [22], a multi-view dataset of real world objects annotated with relative camera poses and foreground masks. We use the specified *fewview-train* and *fewview-dev* splits for training and evaluation. Since SparseFusion optimizes an instance-specific Instant NGP, it is computationally prohibitive to evaluate on all evaluation scenes. Instead, we perform most experiments on a ***core subset*** of 10 categories proposed by [22], evaluating 10 scenes per category. Furthermore, we demonstrate that SparseFusion extends to diverse categories by evaluating 5 scenes per category across 51 categories.

Baselines. We compare SparseFusion against current state-of-the-art methods. We first compare against *PixelNeRF* [46], a feature re-projection method. We adapt *PixelNeRF* to CO3Dv2 dataset and train category-specific models on the 10 categories of the *core subset*, each for 300k steps. We also compare against *NerFormer* [22], another feature re-projection method. We use category-specific models provided by the authors for all 51 categories. Moreover, we compare against *ViewFormer*[1] [13], an autoregressive image generation method, using models provided by the authors. Lastly, we present components of SparseFusion, EFT and VLDM, as strong baselines.

Metrics. We report standard image distortion metric PSNR and perception metric LPIPS [49]. We recognize that neither metric is perfect for ambiguous cases of novel view synthesis; PSNR derives from MSE and favors mean color prediction while LPIPS favors perceptual details.

Implementation Details. For EFT, we use a ResNet18 [9] backbone and three groups of transformer encoders with

[1]Only category-agnostic CO3Dv1 weights are compatible with our evaluation. We use the 10-category weights for our *core subset* experiments and all-category weights for our all category experiments. Despite this difference, the comparative results of ViewFormer against our baselines are consistent with the comparisons reported in their original paper.

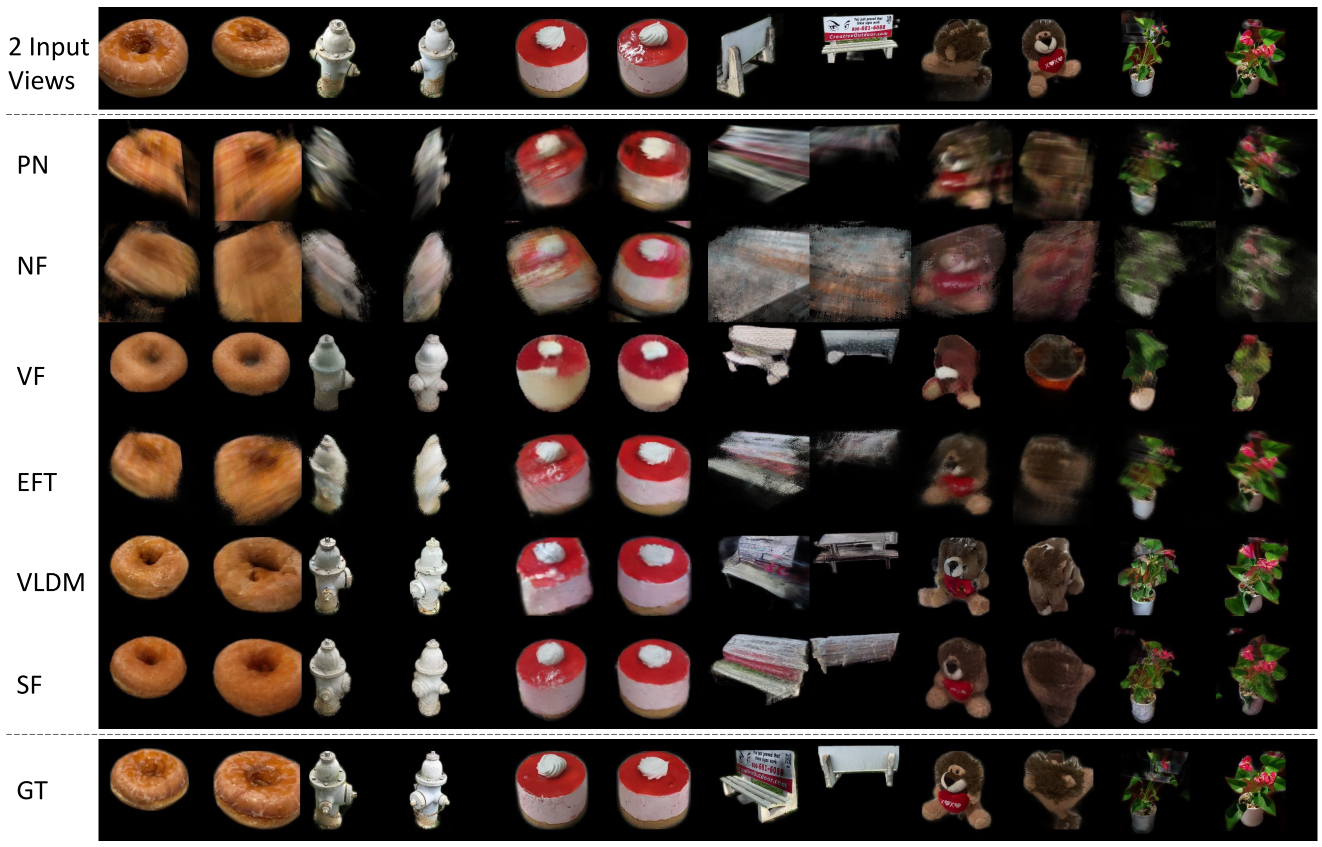

Figure 7. **View Synthesis Qualitative Results.** We show view synthesis results with 2 input views on donut, hydrant, cake, bench, teddybear, and plant categories. We visualize 2 novel views per instance with PixelNeRF (PN), NerFormer (NF), ViewFormer (VF), EFT, VLDM, and finally, SparseFusion (SF). Corresponding numbers can be found in Table 2.

Table 2. **Detailed View Synthesis Benchmark.** We show 2-view category-specific metrics on 10 CO3D categories from the *core subset*. We show PSNR ↑ and LPIPS ↓ averaged across 10 scenes per category.

	Donut		Apple		Hydrant		Vase		Cake		Ball		Bench		Suitcase		Teddybear		Plant	
	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
PixelNeRF [46]	20.9	0.30	20.0	0.35	19.0	0.27	21.3	0.26	18.3	0.37	18.5	0.36	17.7	0.35	21.7	0.30	18.5	0.35	19.3	0.36
NerFormer [22]	20.3	0.34	19.5	0.33	18.2	0.30	17.7	0.34	16.9	0.44	16.8	0.35	15.9	0.44	20.0	0.39	15.8	0.43	17.8	0.45
ViewFormer[1] [13]	19.3	0.29	20.1	0.26	17.5	0.22	20.4	0.21	17.3	0.33	18.3	0.31	16.4	0.30	21.0	0.26	15.5	0.32	17.8	0.31
EFT	21.5	0.31	22.0	0.29	21.6	0.22	21.1	0.25	19.9	0.33	21.4	0.29	**17.8**	0.34	**23.0**	0.26	19.8	0.30	**20.4**	0.31
VLDM	20.1	0.25	21.3	0.22	20.1	0.18	20.2	0.20	18.9	0.30	20.3	0.25	16.6	0.29	21.3	0.23	17.9	0.27	18.9	0.27
SparseFusion	**22.8**	**0.22**	**22.8**	**0.20**	**22.3**	**0.16**	**22.8**	**0.18**	**20.8**	**0.28**	**22.4**	**0.22**	16.7	**0.28**	22.2	**0.22**	**20.6**	**0.24**	20.0	**0.25**

4 layers each. We use 256 hidden dimensions for all layers. For VLDM, we freeze the VAE from [23] that encodes 256x256 images to 32x32 latents with channel dimension of 4. We construct a 400M parameter denosing UNet similar to [25, 27] for the probabilistic modeling. We jointly train category-specific EFT and VLDM model, using Eq. 4 and Eq. 5, on all categories in CO3Dv2. We use a batch size of 2 and train for 100K iterations.

For diffusion distillation, we use a PyTorch implementation of Instant NGP [18, 36]. Due to memory constraints, we render images at 128x128 and upsample to 256x256 before performing diffusion distillation. For each instance, we optimize Instant NGP for 3,000 steps. During the first 1,000 steps, we optimize rendering loss on input images and predicted EFT images from a circular camera trajectory to initialize a rough volume. During the next 2,000 steps, we perform diffusion distillation. Reconstructing a single instance takes roughly an hour on an A5000 gpu.

5.2. Reconstruction on Real Images

Core Subset: 2-view. We show 2-view category specific reconstruction results for the 10 *core subset* categories. We evaluate metrics on the first 10 scenes of each category. For each scene, we load 32 linearly spaced views, from which we randomly sample two input views and evaluate

Table 3. **View Synthesis on 10 Categories.** We benchmark view synthesis results on the 10 categories with 2, 3, and 6 input views.

	2 Views		3 Views		6 Views	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
PixelNeRF [46]	19.52	0.327	20.67	0.293	22.47	0.241
NerFormer [22]	17.88	0.382	18.54	0.367	19.99	0.332
ViewFormer[1] [13]	18.37	0.282	18.91	0.275	19.72	0.266
EFT	20.85	0.289	**22.71**	0.262	**24.57**	0.210
VLDM	19.55	0.247	20.85	0.225	22.35	0.201
SparseFusion	**21.34**	**0.225**	22.35	**0.216**	23.74	**0.200**

Table 4. **View Synthesis on 51 Categories.** We benchmark novel view synthesis on all CO3D categories with 2 input views.

	2 Views	
	PSNR ↑	LPIPS ↓
NerFormer [22]	18.44	0.365
ViewFormer[1] [13]	18.91	0.265
EFT	**21.44**	0.281
VLDM	19.85	0.229
SparseFusion	21.20	**0.223**

on the remaining 30 unseen views. The input and evaluation views are held constant across methods. We report category-specific PSNR and LPIPS in Table 2. We show qualitative comparisons in Figure 7.

SparseFusion outperforms all other methods in LPIPS, only losing out in PSNR for 3 categories. Despite PSNR favoring mean predicting methods, SparseFusion achieves higher PSNR in 7 categories. The strong performance of SparseFusion is reflected in the qualitative comparison. Existing methods either predict a blurry view for unseen regions or a perceptually reasonable view that disregards 3D consistency. SparseFusion predicts views that are both perceptually reasonable and geometrically consistent.

Core Subset: Varying Views. We examine performance of the different methods as we increase the number of input views. As the number of input views increase, more regions are observed, giving an advantage to methods that explicitly use feature re-projection. We evaluate 2, 3, and 6 view reconstruction on the *core subset* categories and show PSNR and LPIPS in Table 3.

We see feature re-projection methods improve drastically with more input views as the need for hallucination of unseen region decreases. EFT outperforms SparseFusion in PSNR for the 3-view and 6-view setting. However, SparseFusion still remains competitive in PSNR while being significantly better in LPIPS. Moreover, SparseFusion outperforms all of the current state-of-the-art methods in both PSNR and LPIPS for 2, 3, and 6 view reconstruction.

All Categories: 2-views. We compare against NerFormer and ViewFormer on all 51 categories to demonstrate that

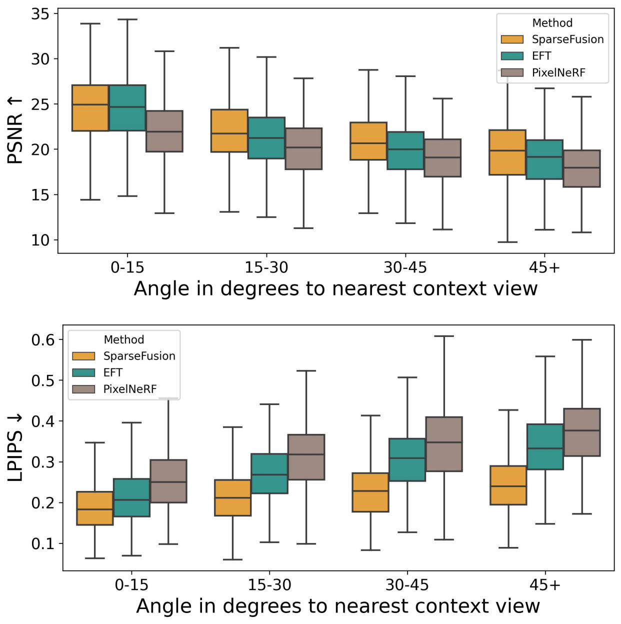

Figure 8. **Metrics Binned by Viewpoint Change.** We show metrics binned by the angle of query camera to the nearest context view. Results are aggregated from Table 2.

SparseFusion performs well on diverse categories. We evaluate with 2 random input views on the first 5 scenes of each category for all 51 categories, and report the averaged metrics in Table 4. While EFT edges out in PSNR, SparseFusion achieves the lowest LPIPS while remaining competitive in PSNR. Existing methods, NerFormer and ViewFormer perform significantly worse. We show qualitative results of SparseFusion on diverse categories in Figure 9, where in addition to 3 synthesized novel views, we also visualize the underlying geometry by extracting an isosurface via marching cubes.

Failure Modes. We show failure modes on the bottom row of Figure 9. On the bottom left, SparseFusion fails to reconstruct a good geometry for the black suitcase. As Instant NGP is trained to output a default black color for the background, the neural representation sometimes fails to disambiguate black foreground from black background. On the bottom right, we see SparseFusion propagating a dataset bias for the category, remote. Since most remote images are TV remotes, SparseFusion attempts to make the video game controller a TV remote.

5.3. Additional Analysis

Performance binned by viewpoint changes. We investigate the relationship between magnitude of viewpoint change and reconstruction performance. We analyze SparseFusion, EFT, and PixelNeRF results on the *core subset* and visualize PSNR and LPIPS binned by angle in de-

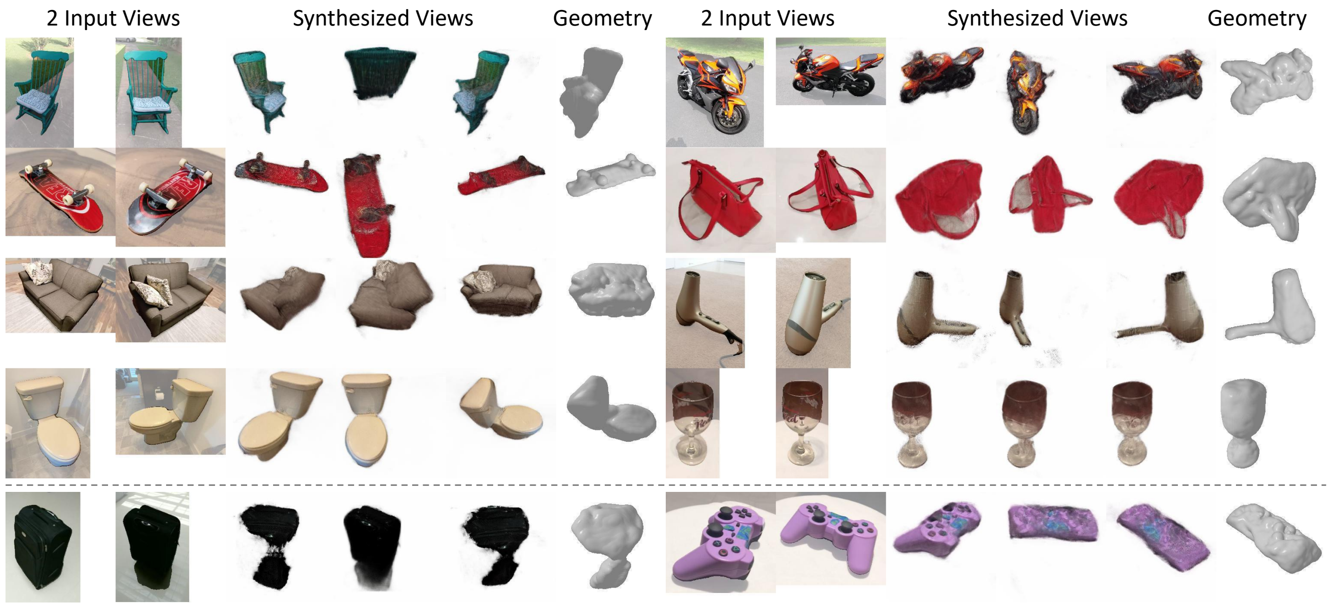

Figure 9. **Reconstruction on Diverse Categories.** We show SparseFusion reconstructions on a subset of the 51 CO3D categories. We also show a couple of failure modes on the last row. Please see project page for more samples and 360-degree visualizations.

Table 5. **The importance of mode seeking.** We show metrics when EFT and VLDM are naively used to optimize Instant NGP [18] in a mean seeking behavior, versus the mode seeking optimization in SparseFusion. We average across 10 scenes of hydrants with 2 input views.

Backbone	Method	PSNR ↑	LPIPS ↓
EFT	base	21.58	0.224
	base w/ INGP	21.53	0.223
VLDM	base	20.05	0.178
	base w/ INGP	20.62	0.230
	SparseFusion w/o multi-step	17.10	0.212
	SparseFusion	**22.26**	**0.159**

grees to the nearest context view in Figure 8. We show that for small viewpoint changes, SparseFusion performs better in LPIPS and competitively in PSNR against EFT. As viewpoint change increases, feature re-projection methods fall off quite fast while SparseFusion remains more robust and performs relatively better.

Importance of Mode Seeking. We compare the diffusion distillation formulation against a naive method to obtain a neural representation given a view synthesis method (VLDM or EFT). Concretely, we obtain several rendered samples ($\{\hat{I}, \hat{\pi}\}$) from the base view synthesis method given the context views C, and simply train an INGP to fit a 3D representation to these.

We present the results in Table 5, and see no significant change when we fit INGP to EFT renderings because EFT predicts consistent mean outputs. However, when we fit INGP to VLDM predictions, we see that perceptual quality decreases. We show a qualitative example in Figure 6 and also illustrate a toy 2D scenario which explains this drop due to mean seeking where averaging over conflicting samples leads to a poor reconstruction. However, when we optimize INGP using the diffusion distillation objective, both PSNR and LPIPS improve, underscoring the importance of mode seeking. We also ablate the importance of the multi-step denoising in Table 5 and see that it is critical for our method's performance.

6. Discussion

We presented an approach for inferring 3D neural representations from sparse-view observations. Unlike prior methods that struggled to deal with uncertainty, our approach allowed predicting 3D-consistent representations with plausible and realistic outputs even in unobserved regions. While we believe our work represents a significant step forward in recovering detailed 3D from casually captured images, a few challenges still remain. A key limitation of our work (as well as prior methods) is the reliance on known (relative) camera poses across the observations, and while there have been recent promising advances [48], this remains a challenging task in general. Additionally, our approach requires optimizing instance-specific neural fields and is computationally expensive. Finally, while our work introduced the view-conditioned diffusion distillation in context of sparse-view reconstruction, we believe even single-view 3D prediction approaches can benefit from leveraging similar objectives.

Acknowledgements

We thank Naveen Venkat, Mayank Agarwal, Jeff Tan, Paritosh Mittal, Yen-Chi Cheng, and Nikolaos Gkanatsios for helpful discussions and feedback. We also thank David Novotny and Jonáš Kulhánek for sharing pretrained models for NerFormer and ViewFormer, respectively. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (DGE1745016, DGE2140739).

References

[1] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. In *CVPR*, 2022. 2

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2

[3] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 2

[4] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[5] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 2009. 2

[6] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 2

[7] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019. 2

[8] Shubham Goel, Georgia Gkioxari, and Jitendra Malik. Differentiable stereopsis: Meshes from multiple views using differentiable rendering. In *CVPR*, 2022. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 5, 11

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

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 2, 3, 4, 11, 12

[12] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2

[13] Jonáš Kulhánek, Erik Derner, Torsten Sattler, and Robert Babuška. Viewformer: Nerf-free neural rendering from few images using transformers. In *ECCV*, 2022. 2, 5, 6, 7

[14] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. In *NeurIPS*, 2020. 2

[15] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022. 5, 13

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2

[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. 2, 4

[18] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 2022. 2, 3, 4, 6, 8, 12

[19] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2

[20] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2, 5

[21] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *URL: https://doi. org/10.48550/arXiv*, 2204, 2022. 2, 4

[22] 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 *ICCV*, 2021. 2, 5, 6, 7

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 6, 12

[24] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021. 2

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 6

[26] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arxiv:2208.12242*, 2022. 2

[27] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2, 6, 12

[28] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, 2022. 2

[29] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2

[30] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006. 2

[31] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021. 2

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

[33] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *IJCV*, 2008. 2

[34] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2, 4, 11

[35] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*, 2022. 2, 4, 11

[36] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 3, 6, 12, 13

[37] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016. 2

[38] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *ICCV*, 2021. 2

[39] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 2

[40] Kalyan Alwala Vasudev, Abhinav Gupta, and Shubham Tulsiani. Pre-train, self-train, distill: A simple recipe for supersizing 3d reconstruction. In *CVPR*, 2022. 2

[41] Phil Wang. Implementation of imagen, google's text-to-image neural network, in pytorch, 2022. https://github.com/lucidrains/imagen-pytorch. 12

[42] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[43] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 2

[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

[45] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. Shelf-supervised mesh prediction in the wild. In *CVPR*, 2021. 2

[46] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 5, 6, 7

[47] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *NeurIPS*, 2021. 2

[48] Jason Y. Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose: Predicting probabilistic relative rotation for single objects in the wild. In *ECCV*, 2022. 8

[49] 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. 5

[50] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016. 2

Appendix

A. Extended Background: Denoising Diffusion

Denoising diffusion probabilistic models [11] approximate a distribution $p(\boldsymbol{x})$ over real data by reversing a Markov chain of diffusion steps, starting from Gaussian noise at $\boldsymbol{x}_T$ to a realistic image at $\hat{\boldsymbol{x}}_0$. See [11] for details.

Forward Process. The forward diffusion process, which incrementally adds noise to a real image $\boldsymbol{x}_0$ until the image becomes Gaussian noise $\boldsymbol{x}_T$, is defined in Eq. 9. Forward variance β is usually defined by a fixed schedule.

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}) \tag{9}$$

Reverse Process. The reverse diffusion process reverses the noise added in the forward process, effectively denoising a noisy image. When we generate a sample from a diffusion model, we apply the reverse process T times from $t = T$ to $t = 1$. The reverse process is defined in Eq. 10, where posterior mean $\mu_\phi(\boldsymbol{x}_t, t)$ is predicted from a network and posterior variance σ^2 follows a fixed schedule (though other works such as [34] also learn σ^2 with a network).

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\mu_\phi(\boldsymbol{x}_t, t), \sigma^2\boldsymbol{I}) \tag{10}$$

Posterior Mean. Prior works [11,34] have found that parameterizing the neural network to predict $\boldsymbol{\epsilon}$ instead of $\boldsymbol{x}_{t-1}$ or $\boldsymbol{x}_0$ works better in practice. We write posterior mean in terms of $\boldsymbol{\epsilon}$ in Eq. 11 where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \Pi_{s=1}^t \alpha_s$.

$$\mu_\phi(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t, t)) \tag{11}$$

As mentioned in the main text, this parametrization leads to a training framework where one adds (time-dependent) noise to a data point $\boldsymbol{x}_0$, and then trains the network $\boldsymbol{\epsilon}_\phi$ to predict this noise given the noisy data point $\boldsymbol{x}_t$.

$$\begin{gathered}\mathcal{L}_{DM} = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon},t}\left[w_t\,||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t, t)||^2\right] \\ \text{where } \boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon};\quad \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)\end{gathered} \tag{12}$$

In this work, we use conditional diffusion models to infer distributions of the form $p(\boldsymbol{x}|\boldsymbol{y})$ by additionally using $\boldsymbol{y}$ as an input for the noise prediction network $\boldsymbol{\epsilon}_\phi(\boldsymbol{x}, \boldsymbol{y}, t)$.

B. Implementation Details

We provide detailed implementation and training details for all components of SparseFusion.

B.1. Epipolar Feature Transformer

Overview. Epipolar feature transformer is a feed-forward network that first gathers features along the epipolar lines of input images before aggregating them through a series of transformers. EFT is inspired by the GPNR approach by Suhail *et al.* [35], but we modify the feature extractor backbone to better suit the sparse-view setup and additionally use epipolar features for conditional diffusion. We describe our implementation below.

Notation: Let g_ψ be the RGB branch and h_ψ be the feature branch.

Inputs: $C \equiv (\boldsymbol{x}_m, \boldsymbol{\pi}_m)$, a set of input images with known camera poses and a query pose $\boldsymbol{\pi}$ – note that the poses are w.r.t. an arbitrary world-coordinate system and we only use their relative configuration.

Outputs: an RGB image $\boldsymbol{x}$ and a feature grid $\boldsymbol{y}$ corresponding to the query viewpoint $\boldsymbol{\pi}$.

Feature Extractor Backbone. Given input views $C \equiv (\boldsymbol{x}_m, \boldsymbol{\pi}_m)$ where $\boldsymbol{x}_m$ is the m^{th} masked (black background) input image of shape (256, 256, 3). We use ResNet18 [9] as our backbone to extract pixel-aligned features by concatenating intermediate features from the first 4 layer groups of ResNet18, using bilinear upsampling to ensure all features are 128 by 128. For each image $\boldsymbol{x}_m$, we arrive at a feature grid of shape (128, 128, 512).

Epipolar Points Projector. Given a query camera $\boldsymbol{\pi}$, each pixel in its image plane corresponds to some ray. Our Epipolar Transformer seeks to infer per-pixel colors or features, and does so by processing each ray using the multi-view projections of points along it. For each ray $\boldsymbol{r}$ (parameterized by its origin and direction), we project 20 points along the ray direction with depth values linearly spaced between z_near and z_far. We set z_near to $s - 5$ and z_far to $s + 5$ where s is the average distance from scene cameras to origin computed per scene. The 20 points, with shape (20, 3), are then projected into the screen space of each of the m input cameras, giving us epipolar points with shape (M, 20, 2). We use bilinear sampling to sample image features at the epipolar points, giving us combined epipolar features of shape (M, 20, 512) per ray. This becomes the input to our epipolar feature transformer.

Epipolar Feature Transformer. EFT aggregates the epipolar features from a single ray with a series of three transformers to predict an RGB pixel color and a 256-dimension feature. We visualize the EFT in Figure 10. We show details of the transformers in Table 6. All transformer encoders have hidden and output dimensions of 256. Both the depth aggregator and view aggregator transformers are

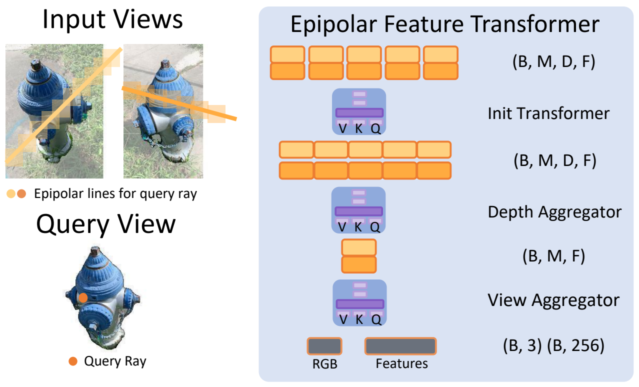

Figure 10. **Epipolar Feature Transformer** We show a diagram of EFT. This module processes each query ray independently, using a transformer to aggregate the projected features across views and across possible depths. For each ray, the output is a predicted RGB color (used as a baseline prediction method), and a pixel-aligned feature (used as conditioning in the diffusion model).

followed by a weighted average operation, where the output features from the transformers are multiplied by a weight, which sums to 1 along the sequence length dimension. The relative weights are predicted by a linear layer before passing through softmax. This effectively performs weighted averaging along the sequence dimension.

Table 6. **EFT Configuration.** We use default PyTorch hyperparameters for each layer. B is number of rays. M is the number of input views. D is the number of epipolar feature samples along the ray. D is 20.

Transformer	Layers	Sequence Dims / Dim	Output Shape
Init Transformer	Transformer Encoder x4	M	(B, M, D, 256)
Depth Aggregator	Transformer Encoder x4	D	(B, M, D, 256)
	Linear + Softmax	D	(B, M, D, 1)
	Weighted Average		(B, M, 256)
View Aggregator	Transformer Encoder x4	M	(B, M, 256)
	Linear + Softmax	M	(B, M, 1)
	Weighted Average		(B, 256)
Color Branch	Linear		(B, 3)

The inputs to the transformer are the sampled features concatenated with additional ray and depth encodings. Given a point along the query ray $\boldsymbol{r}_q$ at depth d, we denote by $\boldsymbol{p}_{md}$ its projection in the m^{th} context view. In addition to the pixel-aligned feature $\boldsymbol{f}_{md}$ (described in previous paragraph), we also concatenate encodings of the query ray $\boldsymbol{r}_q$, the depth $\boldsymbol{d}$, and the ray $\boldsymbol{r}_{md}$ connecting the m^{th} camera center to the 3D point. We use plucker coordinates to represent each ray, and compute harmonic embeddings for each to $(\boldsymbol{r}_q, \boldsymbol{r}_{md}, \boldsymbol{d})$ (using 6 harmonic functions) before concatenating them with $\boldsymbol{f}_{md}$ to form the input tokens to the transformer.

Training Procedure. We can train the color branch of EFT as a standalone novel view synthesis baseline. In our work, EFT is jointly trained with VLDM. Please see supplementary Section B.2 for details.

B.2. View-conditioned Diffusion Model

Overview. View-conditioned diffusion model is a latent diffusion model that conditions on a pixel-aligned feature grid $\boldsymbol{y}$.

Notation: Let ϵ_ϕ be the denoising UNet, $\mathcal{E}$ be the VAE encoder, and $\mathcal{D}$ be the VAE decoder.

VAE. We use the VAE from Stable Diffusion [23]. We use the provided v1-3 weights and keep the VAE frozen for all experiments. We use (256, 256, 3) RGB images as input, and the VAE encodes them into latents of shape (32, 32, 4). We refer readers to [23] for more details.

Denoising UNet. Our 400M parameter UNet roughly follows [27]. We construct our UNet using code from [41] with the parameters in Table 7.

Table 7. **UNet Parameters.** We provide parameters for our UNet.

Parameter	Value
channels	4
dim	256
dim_mults	(1,2,4,4)
num_resnet_blocks	(2,2,2,2)
layer_attns	(False, False, False, True)
cond_images_channels	256

The UNet comprises of 4 down-sampling blocks, a middle block, and 4 up-sampling blocks. We show the input and output shape for the modules of the UNet in Table 8. We refer readers to [41] for UNet details. We disable all text conditioning and cross attention mechanisms; instead, we concatenate EFT features, $\boldsymbol{y}$, with image latents, $\boldsymbol{z}_t$. These EFT features are computed for the of 32×32 rays corresponding to the patch centers.

Training Procedure. We train with batch size of 2, randomly chosen number of input views between 2-5, and learning rate of 5e-5 using Adam optimizer with default hyperparameters for 100K steps. We optimize both the UNet weights and also the EFT weights. We optimize the UNet and feature branch of EFT with the simplified variational lower bound [11]. We optimize the color branch of EFT with pixel-wise reconstruction loss.

B.3. Diffusion Distillation

Overview. We optimize a 3D neural scene representation, Instant NGP [18, 36], with our VLDM.

Notation: Let f_θ be the volumetric Instant NGP renderer, $p_\phi(\boldsymbol{z}_{0:T}|\boldsymbol{\pi}, C)$ be the multi-step denoising process that estimates $\hat{\boldsymbol{z}}_0$. Let Π be an instance-specific camera distribution.

Table 8. **UNet Blocks.** We outline the modules in our denoising UNet.

Modules	Block	Output Shape
Input		(B, 260, 32, 32)
Init. Conv	InitBlock	(B, 256, 32, 32)
Down 1	DownBlock	(B, 256, 16, 16)
Down 2	DownBlock	(B, 512, 8, 8)
Down 3	DownBlock	(B, 1024, 4, 4)
Down 4	DownBlock Self-attention	(B, 1024, 4, 4) (B, 1024, 4, 4)
Middle	MiddleBlock	(B, 1024, 4, 4)
Up 1	UpBlock Self-attention	(B, 1024, 8, 8) (B, 1024, 8, 8)
Up 2	UpBlock	(B, 512, 16, 16)
Up 3	UpBlock	(B, 256, 32, 32)
Up 4	UpBlock	(B, 256, 32, 32)
Final Conv.	Conv2D	(B, 4, 32, 32)

Instant NGP. We use the PyTorch Instant NGP implementation from [36]. We set scene bounds to 4 with desired hashgrid resolution of 8,192. We use a small 3 layer MLP with hidden dimension of 64 to predict RGB and density. We do not use view direction as input.

Camera Distribution. Given a set of input cameras $C_I \equiv (\boldsymbol{\pi}_m)$ and a query camera $\boldsymbol{\pi}_q$, we first find the look-at point P_{at} by finding the nearest point to all $m+1$ rays originating from camera centers. Then, we fit a circle O in 3D space with center being the mean of all camera centers. Let the normal of circle O be $\boldsymbol{n}$. To sample a camera, we first sample a point P_i on O and jitter the angle between $\overline{P_{at}P_i}$ and $\boldsymbol{n}$ by $\mathcal{N}(0, 0.17)$ radians to get jittered point $P_i^{'}$. We then construct a camera $\boldsymbol{\pi}$ with center $P_i^{'}$ looking at P_{at}.

Multi-step Diffusion Guidance. Given a rendered image $\boldsymbol{x}_0$, we encode it to obtain $\boldsymbol{z}_0$. Then, we uniformly sample $t \sim (0, T]$ and construct a noisy image latent $\boldsymbol{z}_t$. We perform multi-step denoising to obtain $\hat{\boldsymbol{z}}_0$ by iteratively sampling $\hat{\boldsymbol{z}}_{t_{k-1}} \sim p_\phi(\boldsymbol{z}_{t_{k-1}} | \hat{\boldsymbol{z}}_{t_k}, y)$ on an interval of time steps $\mathcal{T} = (t_1, ..., t_k, t)$ using a linear multi-step method [15]. We construct $\mathcal{T}$ by linearly spacing $k+1$ time steps between $(0, t]$. We define k with a simple scheduler:

$$k + 1 = \left\{ \begin{array}{ll} \frac{100t}{T}, & \text{if } t \leq \frac{T}{2} \\ 50, & \text{if } t > \frac{T}{2} \end{array} \right\} \tag{13}$$

Finally, given $\hat{\boldsymbol{z}}_0$, we get the predicted image $\hat{\boldsymbol{x}}_0 = \mathcal{D}(\hat{\boldsymbol{z}}_0)$. We do not compute gradients through multi-step diffusion and treat $\hat{\boldsymbol{x}}_0$ as a detached tensor.

Distillation Details. We perform 3,000 steps of distillation, optimizing weights of the MLP θ with Adam optimizer and learning rate 5e-4. During each step of diffusion distillation, we sample $\boldsymbol{\pi} \sim \Pi$ and render an image $\boldsymbol{x}_0 = f_\theta(\boldsymbol{\pi})$. For the first 1,000 steps, we compute rendering loss between $f_\theta(\boldsymbol{\pi})$ and $g_\psi(\boldsymbol{\pi}|C)$. During the remaining steps, we compute loss between $f_\theta(\boldsymbol{\pi})$ and $\hat{\boldsymbol{x}}_0$. To avoid out-of-memory error, we render images at reduced resolution (128, 128) and apply bilinear up-sampling before performing multi-step diffusion. In addition, we compute rendering loss between $f_\theta(\boldsymbol{\pi}_m)$ and $\boldsymbol{x}_m$ on all m input images. Optimizing a single scene takes roughly 1 hour on an A5000 GPU.

C. Ethics and Discussion

Compared to existing novel view synthesis methods, SparseFusion is more computationally expensive. This poses a hardware limitation for potential downstream tasks and may also increase emissions. Additionally, SparseFusion relies on view-conditioned diffusion models (VLDM), which are trained on multi-view datasets. VLDMs are good at representing their training data, potentially learning harmful biases that will propagate to reconstructed 3D scenes. While our current use case for reconstructing static objects from CO3D categories does not present ethical concerns, adapting SparseFusion to humans or animals requires more thorough examination of bias present in the training data.