**HOLODIFFUSION: Training a 3D Diffusion Model using 2D Images**

Animesh Karnewar*†
UCL
a.karnewar@ucl.ac.uk
Andrea Vedaldi
Meta AI
vedaldi@meta.com
David Novotny*
Meta AI
dnovotny@meta.com
Niloy J. Mitra
UCL
n.mitra@ucl.ac.uk

https://holodiffusion.github.io

**Abstract**

Diffusion models have emerged as the best approach for generative modeling of 2D images. Part of their success is due to the possibility of training them on millions if not billions of images with a stable learning objective. However, extending these models to 3D remains difficult for two reasons. First, finding a large quantity of 3D training data is much more complex than for 2D images. Second, while it is conceptually trivial to extend the models to operate on 3D rather than 2D grids, the associated cubic growth in memory and compute complexity makes this infeasible. We address the first challenge by introducing a new diffusion setup that can be trained, end-to-end, with only posed 2D images for supervision; and the second challenge by proposing an image formation model that decouples model memory from spatial memory. We evaluate our method on real-world data, using the CO3D dataset which has not been used to train 3D generative models before. We show that our diffusion models are scalable, train robustly, and are competitive in terms of sample quality and fidelity to existing approaches for 3D generative modeling.

1. Introduction

Diffusion models have rapidly emerged as formidable generative models for images, replacing others (e.g., VAEs, GANs) for a range of applications, including image colorization [47], image editing [35], and image synthesis [9, 21]. These models explicitly optimize the likelihood of the training samples, can be trained on millions if not billions of images, and have been shown to capture the underlying model distribution better [9] than previous alternatives.

A natural next step is to bring diffusion models to 3D data. Compared to 2D images, 3D models facilitate direct manipulation of the generated content, result in perfect view consistency across different cameras, and allow object placement using direct handles. However, learning 3D diffusion models is hindered by the lack of a sufficient volume of 3D data for training. A further question is the choice of representation for the 3D data itself (e.g., voxels, point clouds, meshes, occupancy grids, etc.). Researchers have proposed 3D-aware diffusion models for point clouds [33],

* Indicates equal contribution.
† part of this work was done during an internship at MetaAI.
volumetric shape data using wavelet features [22] and novel view synthesis [61]. They have also proposed to distill a pretrained 2D diffusion model to generate neural radiance fields of 3D objects [30, 45]. However, a diffusion-based 3D generator model trained using only 2D image for supervision is not available yet.

In this paper, we contribute HOLODIFFUSION, the first unconditional 3D diffusion model that can be trained with only real posed 2D images. By posed, we mean different views of the same object with known cameras, for example, obtained by means of structure from motion [49].

We make two main technical contributions: (i) We propose a new 3D model that uses a hybrid explicit-implicit feature grid. The grid can be rendered to produce images from any desired viewpoint and, since the features are defined in 3D space, the rendered images are consistent across different viewpoints. Compared to utilizing an explicit density grid, the feature representation allows for a lower resolution grid. The latter leads to an easier estimation of the probability density due to a smaller number of variables. Furthermore, the resolution of the grid can be decoupled from the resolution of the rendered images. (ii) We design a new diffusion method that can learn a distribution over such 3D feature grids while only using 2D images for supervision. Specifically, we first generate intermediate 3D-aware features conditioned only on the input posed images. Then, following the standard diffusion model learning, we add noise to this intermediate representation and train a denoising 3D UNet to remove the noise. We apply the denoising loss as photometric error between the rendered images and the Ground-Truth training images. The key advantage of this approach is that it enables training of the 3D diffusion model from 2D images, which are abundant, sidestepping the difficult problem of procuring a huge dataset of 3D models for training.

We train and evaluate our method on the Co3Dv2 [46] dataset where HOLODIFFUSION outperforms existing alternatives both qualitatively and quantitatively.

2. Related Work

2.1. Image-conditioned 3D Reconstruction

Neural and differentiable rendering. Neural rendering [56] is a class of algorithms that partially or entirely use neural networks to approximate the light transport equation.

The 2D versions of neural rendering include variants of pix2pix [23], deferred neural rendering [57], and their follow-up works. The common theme in all these methods is that a post-processor neural network (usually a CNN) maps neural feature images into photorealistic RGB images.

The 3D versions of neural rendering have recently been popularized by NeRF [36], which uses a Multi Layer Perceptron (MLP) to model the parameters of the 3D scene (radiance and occupancy) and a physically-based rendering procedure (Emission-Absorption raymarching). NeRF solves the inverse rendering problem where, given many 2D images of a scene, the aim is to recover its 3D shape and appearance. The success of NeRF gave rise to many follow-up works [1, 2, 36, 58, 68]. While the latter uses MLPs to represent the occupancy and radiance of the underlying scene, different representations were explored in [7, 13, 27, 31, 38, 55, 59, 66].

Few-view reconstruction. In many cases, dense image supervision is unavailable, and one has to condition the reconstruction on a small number of scene views instead. Since 3D reconstruction from few-views is ambiguous, recent methods aid the reconstruction with 3D priors learned by observing many images of an object category. Works like CMR [25], C3DM [43], and UMR [29] learn to predict the parameters of a mesh by observing images of individual examples of the object category. DOVE [63] also predicts meshes, but additionally leverages stronger constraints provided by videos of the deformable objects.

Others [17, 67] aimed at learning to fit NeRF given only a small number of views; they do so by sampling image features at the 2D projections of the 3D ray samples. Later works [46, 60] have improved this formulation by using transformers to process the sampled features. Finally, ViewFormer [28] drops the rendering model and learns a fully implicit transformer-based new-view synthesizer. Recently, BANMo [65] reconstructed deformable objects with a signed distance function.

2.2. 3D Generative Models

3D Generative adversarial networks. Early 3D generative models leveraged adversarial learning [12] as the primary form of supervision. PlatonicGAN [16] learns to generate colored 3D shapes from an unstructured corpus of images belonging to the same category, by rendering a voxel grid from random viewpoints so that an adversarial discriminator cannot distinguish between the renders and natural images sampled from a large uncurated database. PrGAN [10] differs from PlatonicGAN by focusing only on rendering untextured 3D shapes. To deal with the large memory footprint of voxels, HoloGAN [39] adjusts PlatonicGAN by rendering a low-resolution 2D feature image of a coarse voxel grid, followed by a 2D convolutional decoder mapping the feature render to the final RGB image. The results are, however, not consistent with camera motion.

Inspired by the success of NeRF [37], GRAF [50] also trains in a data setting similar to PlatonicGAN [16] but, differently from PlatonicGAN, represents each generated scene with a neural radiance field. The GRAF pipeline was subsequently improved by PiGAN [6], leveraging a SIREN-based [52] architecture. Similar to HoloGAN, StyleNeRF [14] first renders a radiance feature field followed by
Figure 2. Method overview. Our HOLODIFFUSION takes as input video frames for category-specific videos \( \{ s^i \} \) and trains a diffusion-based generative model \( D_\theta \). The model is trained with only posed image supervision \( \{(I^i_j, P^i_j)\} \), without access to 3D ground-truth. Once trained, the model can generate view-consistent results from novel camera locations. Please refer to Sec. 3 for details.

3.1. Diffusion Models

Given \( N \) i.i.d. samples \( \{ x^i \}_{i=1}^N \) from (an unknown) data distribution \( p(x) \), the task of generative modeling is to find the parameters \( \theta \) of a parametric model \( p_\theta(x) \) that best approximate \( p(x) \). Diffusion models are a class of likelihood-based models centered on the idea of defining a forward diffusion (noising) process \( q(x_t|x_{t-1}) \), for \( t \in [0, T] \). The noising process converts the data samples into pure noise,
\[ i.e., \, q(x_T) \approx q(x_T | x_{T-1}) = \mathcal{N}(0, I). \] The model then learns the reverse process \( p(x_{t-1} | x_t) \), which iteratively converts the noise samples back into data samples starting from the purely Gaussian sample \( x_T \).

The Denoising Diffusion Probabilistic Model (DDPM) [20], in particular, defines the noising transitions using a Gaussian distribution, setting

\[
q(x_t | x_{t-1}) := \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t)I). \tag{1}
\]

Samples can be easily drawn from this distribution by using the reparameterization trick:

\[
x_t := \sqrt{\alpha_t} x_{t-1} + \sqrt{1 - \alpha_t} \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I). \tag{2}
\]

One similarly defines the reverse denoising step using a Gaussian distribution:

\[
p_{\theta}(x_{t-1} | x_t) := \mathcal{N}(x_{t-1}; \sqrt{\alpha_t} \mathcal{D}_\theta(x_t, t), (1 - \alpha_t)I), \tag{3}
\]

where, the \( \mathcal{D}_\theta \) is the denoising network with learned parameters \( \theta \). The sequence \( \alpha_t \) defines the noise schedule as:

\[
\alpha_t = 1 - \beta_t, \quad \beta_t \in [0, 1], \quad s.t. \beta_t > \beta_{t-1} \forall t \in [0, T]. \tag{4}
\]

We use a linear time schedule with \( T = 1000 \) steps.

The denoising must be applied iteratively for sampling the target distribution \( p(x) \). However, for training the model we can draw samples \( x_t \) directly from \( q(x_t | x_0) \) as:

\[
x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon \quad \text{where} \quad \epsilon = \prod_{i=0}^{t} \alpha_i. \tag{5}
\]

It is common to use the network \( \mathcal{D}_\theta(x_t, t) \) to predict the noise component \( \epsilon \) instead of the signal component \( x_{t-1} \) in eq. (2); which has the interpretation of modeling the score of the marginal distribution \( q(x_t) \) up to a scaled constant [20, 32]. Instead, we employ the “\( x_0 \)-formulation” from eq. (3), which has recently been explored in the context of diffusion model distillation [48] and modeling of text-conditioned videos using diffusion [19]. The reasoning behind this design choice will become apparent later.

**Training.** Training the “\( x_0 \)-formulation” of a diffusion model \( \mathcal{D}_\theta \) comprises minimizing the following loss:

\[
\mathcal{L} = \| \mathcal{D}_\theta(x_t, t) - x_0 \|^2, \tag{6}
\]

encouraging \( \mathcal{D}_\theta \) to denoise sample \( x_t \sim \mathcal{N}(\sqrt{\alpha_t} x_0, (1 - \alpha_t)I) \) to predict the clean sample \( x_0 \).

**Sampling.** Once the denoising network \( \mathcal{D}_\theta \) is trained, sampling can be done by first starting with pure noise, i.e., \( x_T \sim \mathcal{N}(0, I) \), and then iteratively refining \( T \) times using the network \( \mathcal{D}_\theta \), which terminates with a sample from target data distribution \( x_0 \sim q(x_0) = p(x) \):

\[
x_{t-1} \sim \mathcal{N}(\sqrt{\alpha_{t-1}} \mathcal{D}_\theta(x_t, t), (1 - \alpha_{t-1})I). \tag{7}
\]

### 3.2. Learning 3D Categories by Watching Videos

**Training data.** The input to our learning procedure is a dataset of \( N \in \mathbb{N} \) video sequences \( \{s^i\}_{i=1}^N \), each depicting an instance of the same object category (e.g., car, carrot, teddy bear). Each video \( s^i = (I^i_j, P_j^s)_{j=1}^{N_{\text{frames}}} \) comprises \( N_{\text{frames}} \) pairs \( (I^i_j, P_j^s) \), each consisting of an RGB image \( I^i_j \in \mathbb{R}^{3 \times H \times W} \) and its corresponding camera pose \( P_j^s \in \mathbb{R}^{4 \times 4} \), represented as a \( 4 \times 4 \) camera matrix.

Our goal is to train a generative model \( p(V) \) where \( V \) is a representation of the shape and appearance of a 3D object; furthermore, we aim to learn this distribution using only the 2D training videos \( \{s^i\}_{i=1}^N \).

**3D feature grids.** As 3D representation \( V \) we pick 3D feature voxel grids \( V \in \mathbb{R}^{d^V \times S \times S \times S} \) of size \( S \in \mathbb{N} \) containing \( d^V \)-dimensional latent feature vectors. Given the voxel grid \( V \) representing the object from a certain video \( s \), we can reconstruct any frame \( (I^i_j, P_j^s) \in s \) of the video as \( I^i_j = r_\gamma(V, P_j^s) \) by the means of the rendering function \( r_\gamma(V, P_j^s) : \mathbb{R}^{d^V \times S \times S \times S} \times \mathbb{R}^{4 \times 4} \rightarrow \mathbb{R}^{3 \times H \times W} \), where \( \gamma \) is the feature parameters (see Sec. 3.4 for details).

Next, we discuss how to build a diffusion model for the distribution \( p(V) \) of feature grids. One might attempt to directly apply the methodology of Sec. 3.1, setting \( x = V \), but this does not work because we have no access to ground-truth feature grids \( V \) for training; instead, these 3D models must be inferred from the available 2D videos while training. We solve this problem in the next section.

### 3.3. Bootstrapped Latent Diffusion Model

In this section, we show how to learn the distribution \( p(V) \) of feature grids from the training videos \( s \) alone. In what follows, we use the symbol \( V \) as a shorthand for \( p(V) \).

The training videos provide RGB images \( I \) and their corresponding camera poses \( P \), but no sample feature grids \( V \) from the target distribution \( V \). As a consequence, we also have no access to the noised samples \( V_i \sim \mathcal{N}(\sqrt{\alpha} V_0, (1 - \alpha)I) \) required to evaluate the denoising objective eq. (6) and thus learn a diffusion model.

To solve this issue, we introduce the BLDM (Bootstrapped Latent Diffusion Model). BLDM can learn the denoiser-cum-generator \( \mathcal{D}_\theta \) given samples \( V \sim \mathcal{V} \) from an auxiliary distribution \( \mathcal{V} \), which is closely related but not identical to the target distribution \( \mathcal{V} \).

**The auxiliary samples \( \mathcal{V} \).** As shown in fig. 2, our idea is to obtain the auxiliary samples \( \mathcal{V} \) as a (learnable) function of the corresponding training videos \( s \). To this end, we use a design strongly inspired by Warp-Conditioned-Embedding (WCE) [17], which demonstrated compelling performance for learning 3D object categories. Specifically, given a training video \( s \) containing frames \( I_j \), we generate a grid \( \mathcal{V} \in \mathbb{R}^{d^V \times S \times S \times S} \) of auxiliary features \( \mathcal{V}_{mnos} \in [-1, 1]^{d^V} \).
by projecting the 3D coordinate $x_{mno}^V$ of the each grid element $(m, n, o)$ to every video frame $I_i$, sampling corresponding 2D image features, and aggregating those into a single $d_V$-dimensional descriptor per grid element. The 2D image features are extracted by a frozen encoder (we use the ResNet-32 encoder $E$ [15]). This process is detailed in the supplementary material.

**Auxiliary denoising diffusion objective.** The standard denoising diffusion loss eq. (6) is unavailable in our case because the data samples $V$ are unavailable. Instead, we leverage the “$x_0$-formulation” of diffusion to employ an alternative diffusion objective which does not require knowledge of $V$. Specifically, we replace eq. (6) with a photometric loss

$$L_{\text{photo}} := \|r_{\mathcal{Z}}(D_0(\bar{V}_i, t), P_j) - I_j\|^2,$$  

(8)

which compares the rendering $r_{\mathcal{Z}}(D_0(\bar{V}_i, t), P_j)$ of the denoising $D_0(\bar{V}_i, t)$ of the noised auxiliary grid $\bar{V}_i$ to the (known) image $I_j$ with pose $P_j$. Equation (8) can be computed because the image $I_j$ and camera parameters $P_j$ are known and $\bar{V}_i$ is derived from the auxiliary sample $\bar{V}$, whose computation is given in the previous section.

**Train/test denoising discrepancy.** Our denoiser $D_0$ takes as input a sample $\bar{V}_i$ from the noised auxiliary distribution $\mathcal{V}_i$ instead of the noised target distribution $\mathcal{V}_t$. While this allows to learn the denoising model by minimizing eq. (8), it prevents us from drawing samples from the model at test time. This is because, during training, $D_0$ learns to denoise the auxiliary samples $\bar{V} \in \mathcal{V}$ (obtained through fusing image features into a voxel-grid), but at test time we need instead to draw target samples $V \in \mathcal{V}$ as specified by eq. (7) per sampling step. We address this problem by using a bootstrapping technique that we describe next.

**Two-pass diffusion bootstrapping.** In order to remove the discrepancy between the training and testing sample distributions for the denoiser $D_0$, we first use the latter to obtain ‘clean’ voxel grids from the training videos during an initial denoising phase, and then apply a diffusion process to those, finetuning $D_0$ as a result.

Our bootstrapping procedure rests on the assumption that once $L_{\text{photo}}$ is minimized, the denoisings $D_0(\bar{V}_i, t)$ of the auxiliary grids $\bar{V} \sim \mathcal{V}$ follow the clean data distribution $\mathcal{V}$, i.e., $D_0(\bar{V}_i, t) \sim \mathcal{V}$ for the optimal denoiser parameters $\theta^*$ that minimize $L_{\text{photo}}$. Simply put, the denoiser $D_0$ learns to denoise both the diffusion noise and the noise resulting from imperfect reconstructions. Note that our assumption $D_0(\bar{V}_i, t) \sim \mathcal{V}$ is reasonable since recent single-scene neural rendering methods [7, 31, 37] have demonstrated successful recovery of high-quality 3D shapes solely by optimizing the photometric loss via differentiable rendering.

Given that $D_0$, is now capable of generating clean data samples, we can expose it to the noised version of the clean samples $V$ by executing a second denoising pass in a recurrent manner. To this end, we define the bootstrapped photometric loss $L'_{\text{photo}}$:

$$L'_{\text{photo}} := \|r_{\mathcal{Z}}(D_0(\mathcal{E}_V(\hat{D}_0(\bar{V}_i, t), t'), P_j) - I_j\|^2,$$  

(9)

with $\mathcal{E}_V(Z) \sim \mathcal{N}(\sqrt{\alpha_t}Z, (1 - \alpha_t)I)$ denoting the diffusion of input grid $Z$ at time $t'$. Intuitively, eq. (9) evaluates the photometric error between the ground truth image $I$ and the rendering of the doubly-denoised grid $D_0(\mathcal{E}_V(\hat{D}_0(\bar{V}_i, t), t'))$.

### 3.4. Implementation Details

**Training details.** HOLODIFFUSION training finds the optimal model parameters $\theta, \mathcal{Z}$ by minimizing the sum of the photometric and the bootstrapped photometric losses $L_{\text{photo}} + L'_{\text{photo}}$ using the Adam optimizer with an initial learning rate $5 \cdot 10^{-5}$ (decaying ten-fold whenever the total loss plateaus) until convergence is reached.

In each training iteration, we randomly sample 10 source views $\{I_i\}$ from a randomly selected training video $s$ to form the grid of auxiliary features $V$. The auxiliary features are noised to form $\bar{V}_i$ and later denoised with $D_0(\bar{V}_i)$. Afterwards $D_0(\bar{V}_i)$ is noised and denoised again during the two-pass bootstrap procedure. To avoid two rendering passes in each training iteration (one for $L_{\text{photo}}$ and the second for $L'_{\text{photo}}$), we randomly choose to optimize either $L_{\text{photo}}$ or $L'_{\text{photo}}$ with 50-50 probability in each iteration. The photometric losses compare renders $r_{\mathcal{Z}}(\cdot, P_j)$ of the denoised voxel grid to 3 different target views (different from the source views).

**Rendering function $r_{\mathcal{Z}}$.** The differentiable rendering function $r_{\mathcal{Z}}(V, P_j)$ from eqs. (8) and (9) uses Emission-Absorption (EA) ray marching as follows. First, given the knowledge of the camera parameters $P_j$, a ray $r_u \in S^2$ is emitted from each pixel $u \in \{0, \ldots, H-1\} \times \{0, \ldots, W-1\}$ of the rendered image $I_j \in \mathbb{R}^{H \times W}$. We sample $N_S$ 3D points $(p_i)_{i=1}^{N_S}$ on each ray at regular intervals $\Delta \in \mathbb{R}$. For each point $p_i$, we sample the corresponding voxel grid feature $V[p_i] \in \mathbb{R}^{d_V'}$, where $V[\cdot]$ stands for trilinear interpolation. The feature $V[p_i]$ is then decoded by an MLP as $f_\mathcal{Z}(V[p_i], r_u) := (\sigma_i, c_i)$ with parameters $\mathcal{Z}$ to obtain the density $\sigma_i \in [0, 1]$ and the RGB color $c_i \in [0, 1]^3$ of each 3D point. The MLP $f$ is designed so that the color $c$ depends on the ray direction $r_u$ while the density $\sigma$ does not, similar to NeRF [36]. Finally, EA ray marching renders the $r_u$’s pixel color $c_i = \sum_{i=1}^{N_S} w(p_i) c_i$ as a weighted combination of the sampled colors. The weights are defined as $w(p_i) = T_{i} - T_{i+1}$ where $T_{i} = e^{-\sum_{j=1}^{i-1} \sigma_j \Delta}$.

### 4. Experiments

In this section, we evaluate our method. First we perform the quantitative evaluation followed by visualizing samples
for assessing the quality of generations.

Datasets and baselines. For our experiments, we use CO3Dv2 [46], which is currently the largest available dataset of fly-around real-life videos of object categories. The dataset contains videos of different object categories and each video makes a complete circle around the object, showing all sides of it. Furthermore, camera poses and object foreground masks are provided with the dataset (they were obtained by the authors by running off-the-shelf Structure-from-Motion and instance segmentation software, respectively).

We consider the four categories Apple, Hydrant, TeddyBear and Donut for our experiments. For each of the categories we train a single model on the 500 “train” videos (i.e. approx. 500×100 frames in total) with the highest camera cloud quality score, as defined in the CO3Dv2 annotations, in order to ensure clean ground-truth camera pose information. We note that all trainings were done on 2-to-8 V100 32GB GPUs for 2 weeks.

We consider the prior works pi-GAN [6], EG3D [5], and GET3D [11] as baselines for comparison. Pi-GAN generates radiance fields represented by MLPs and is trained us-
ing an adversarial objective. Similar to our setting, they only use 2D image supervision for training. EG3D [5] uses the feature triplane, decoded by an MLP as the underlying representation, while needing both the images and the camera poses as input to the training procedure. GET3D [11] is another GAN-based baseline, which also requires the images and camera poses for training. Apart from this, GET3D also requires the fg/bg masks for training; which we supply in form of the masks available in CO3Dv2. Since GET3D applies a Deformable Marching Tetrahedra step in the pipeline, the samples generated by them are in the form of textured meshes.

**Quantitative evaluation.** We report Frechet Inception Distance (FID) [18], and Kernel Inception Distance (KID) [4] for assessing the generative quality of our results. As shown in Table 1, our HOLODIFFUSION produces better scores than EG3D and GET3D. Although pi-GAN gets better scores than ours on some categories, we note that the 3D-agnostic training procedure of pi-GAN cannot recover the proper 3D structure of the unaligned shapes of CO3Dv2. Thus, without the 3D-view consistency, the 3D neural fields (MLPs) produced by pi-GAN essentially mimic a 2D image GAN.

**Qualitative evaluation.** Figure 4 depicts random samples generated from all the methods under comparison. HOLODIFFUSION produces the most appealing, consistent and realistic samples among all. Figure 3 further analyzes the viewpoint consistency of pi-GAN compared to ours. It is evident that, although individual views of pi-GAN samples look realistic, their appearance is inconsistent with the change of viewpoint. Please refer to the project webpage for more examples and videos of the generated samples.

**5. Conclusion**

We have presented HOLODIFFUSION, an unconditional 3D-consistent generative diffusion model that can be trained using only posed-image supervision. At the core of our method is a learnable rendering module that is trained in conjunction with the diffusion denoiser, which operates directly in the feature space. Furthermore, we use a pretrained feature encoder to decouple the cubic volumetric memory complexity from the final image rendering resolution. We
Table 1. **Quantitative evaluation.** FID and KID on 4 classes of CO3Dv2 comparing our HOLODIFFUSION with the baselines pi-GAN [6], EG3D [5], GET3D [11], and the non-bootstrapped version of our HOLODIFFUSION. The column “VP” denotes whether renders of a method are 3D view-consistent or not.

<table>
<thead>
<tr>
<th>method</th>
<th>VP</th>
<th>Apple FID ↓</th>
<th>KID ↓</th>
<th>Hydrant FID ↓</th>
<th>KID ↓</th>
<th>TeddyBear FID ↓</th>
<th>KID ↓</th>
<th>Donut FID ↓</th>
<th>KID ↓</th>
<th>Mean FID ↓</th>
<th>KID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi-GAN [6]</td>
<td>✗</td>
<td>49.3</td>
<td>0.042</td>
<td>92.1</td>
<td>0.080</td>
<td>125.8</td>
<td>0.118</td>
<td>99.4</td>
<td>0.069</td>
<td>91.7</td>
<td>0.077</td>
</tr>
<tr>
<td>EG3D [5]</td>
<td>✓</td>
<td>170.5</td>
<td>0.203</td>
<td>229.5</td>
<td>0.253</td>
<td>236.1</td>
<td>0.239</td>
<td>222.3</td>
<td>0.237</td>
<td>214.6</td>
<td>0.233</td>
</tr>
<tr>
<td>GET3D [11]</td>
<td>✓</td>
<td>179.1</td>
<td>0.190</td>
<td>303.3</td>
<td>0.380</td>
<td>244.5</td>
<td>0.280</td>
<td>209.9</td>
<td>0.230</td>
<td>234.2</td>
<td>0.270</td>
</tr>
<tr>
<td>HOLODIFFUSION (No bootstrap)</td>
<td>✓</td>
<td>342.9</td>
<td>0.400</td>
<td>277.9</td>
<td>0.305</td>
<td>222.1</td>
<td>0.217</td>
<td>272.1</td>
<td>0.199</td>
<td>278.7</td>
<td>0.280</td>
</tr>
<tr>
<td>HOLODIFFUSION</td>
<td>✓</td>
<td>94.5</td>
<td>0.095</td>
<td>100.5</td>
<td>0.079</td>
<td>109.2</td>
<td>0.106</td>
<td>115.4</td>
<td>0.085</td>
<td>104.9</td>
<td>0.091</td>
</tr>
</tbody>
</table>

![Sampling across time](image)

**Figure 5.** **Sampling across time.** Rendering of HOLODIFFUSION’s iterative sampling process for a hydrant and a teddy bear. The diffusion time decreases from left ($t = T = 1000$) to the right ($t = 0$).

demonstrate that the method can be trained on raw posed image sets, even in the few-image setting, striking a good balance between quality and diversity of results.

At present, our method requires access to camera information at training time. One possibility is to jointly train a viewpoint estimator to pose the input images, but the challenge may be to train this module from scratch as the input view distribution is unlikely to be uniform [41]. An obvious next challenge would be to test the setup for conditional generation, either based on images (i.e., single view reconstruction task) or using text guidance. Beyond generation, we would also like to support editing the generated representations, both in terms of shape and appearance, and compose them together towards scene generation. Finally, we want to explore multi-class training where diffusion models, unlike their GAN counterparts, are known to excel without suffering from mode collapse.

### 6. Societal Impact

Our method primarily contributes towards the generative modeling of 3D real-captured assets. Thus as is the case with 2D generative models, ours is also prone to misuse of generated synthetic media. In the context of synthetically generated images, our method could potentially be used to make fake 3D view-consistent GIFs or videos. Since we only train our models on the virtually harmless Co3D (Common objects in 3D) dataset, our released models could not be directly used to infer potentially malicious samples.

As diffusion models can be prone to memorizing the training data in limited data settings [54], our models can also be used to recover the original training samples. Analyzing the severity and the extent to which our models suffer from this, is an interesting future direction for exploration.

### 7. Acknowledgements

Animesh and Niloy were partially funded by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 956585. This research has been partly supported by MetaAI and the UCL AI Centre. Finally, Animesh thanks Alexia Jolicoeur-Martineau for the helpful and insightful guidance on diffusion models.

### References


Ka-Hei Hui, Ruihui Li, Jingyi Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation, 2022. 2


Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 3inGAN: Learning a 3D generative model from images of a self-similar scene. In Proc. 3D Vision (3DV), 2022. 3


Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-


[33] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. 2021. 1


