Functional Diffusion

Biao Zhang
KAUST
biao.zhang@kaust.edu.sa

Peter Wonka
KAUST
pwonka@gmail.com

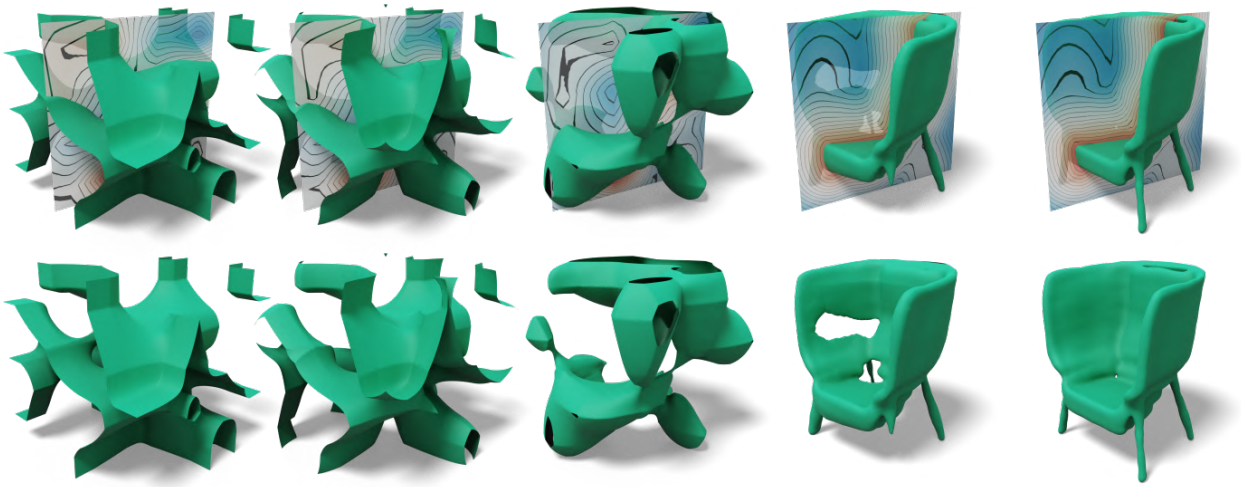

Figure 1. **Functional diffusion.** Our method is able to generate complicated functions with a continuous domain. From left to right, we show 5 steps of the generating process. This particular example shows signed distance functions and we show the zero-isosurface of the generated function in green. Furthermore, we visualize the function values on a plane, where the red colors mean larger and blue means smaller.

Abstract

We propose functional diffusion, a generative diffusion model focused on infinite-dimensional function data samples. In contrast to previous work, functional diffusion works on samples that are represented by functions with a continuous domain. Functional diffusion can be seen as an extension of classical diffusion models to an infinite-dimensional domain. Functional diffusion is very versatile as images, videos, audio, 3D shapes, deformations, etc.*, can be handled by the same framework with minimal changes. In addition, functional diffusion is especially suited for irregular data or data defined in non-standard domains. In our work, we derive the necessary foundations for functional diffusion and propose a first implementation based on the transformer architecture. We show generative results on complicated signed distance functions and deformation functions defined on 3D surfaces.*

1. Introduction

In the last two years diffusion models have become the most popular method for generative modeling of visual data, such as 2D images [34, 35], videos [13, 15], and 3D shapes [3, 5, 16, 40, 41, 46, 47]. In order to train a diffusion model, one needs to add and subtract noise from a data sample. In order to represent a sample, many methods use a direct representation, such as a 2D or 3D grid. Since diffusion can be very costly, this representation is often used in conjunction with a cascade of diffusion models [14, 35]. Alternatively, diffusion methods can represent samples in a compressed latent space [34]. A sample can be encoded and decoded to the compressed space using an autoencoder whose weights are trained in a separate pre-process.

In our work, we explore a departure from these previous approaches and set out to study diffusion in a functional space. We name the resulting method *functional diffusion*. In functional diffusion, the data samples are functions in

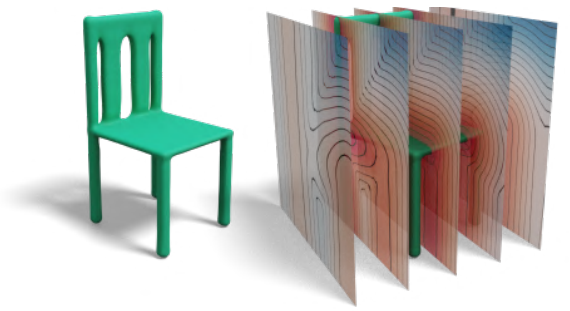

Figure 2. **Signed distance functions.** We show a 3D shape on the left and on the right, we visualize the signed distances sampled in several parallel planes.

a function space (see an example function in Fig. 2). In contrast to regular diffusion, we do not start with a noisy sample, but we need to define a noise function as a starting point. This noise function is then gradually denoised to obtain a sample from the function space. To realize our idea we need multiple different representations that are different from regular diffusion. We employ both a continuous and a sampled representation of a function. As the continuous representation of a function, we propose a set of vectors that are latent vectors of a functional denoising network. To represent a sampled function, we use a set of point samples in the domain of the function together with the corresponding function values. Both of these function representations are used during training and inference. The method is initialized by sampling a noisy continuous function that spans the complete domain. Then we evaluate this function at discrete locations to obtain a sampled representation. A training step in functional diffusion takes both the continuous and sampled representation as input and tries to predict a new continuous representation that is a denoised version of the input function. This novel form of diffusion has multiple interesting properties. First, the framework is very versatile and can be directly adapted to many different forms of input data. We can handle images, videos, audio, 3D shapes, deformations, *etc*., with the same framework. Second, we can directly handle irregular data and non-standard domains as there are few constraints on the function domain as well as the samples of the sampled function representation. For example, we can work with deformations on a surface, which constitutes an irregular domain. Third, we can decouple the representational power of the continuous and sampled function representation. Finally, we believe the idea of functional diffusion is inherently technically interesting. It is a non-trivial change and our work can lay the foundation for a new class of diffusion models with many variations.

In summary, we make the following major contributions:

- We introduce the concept of functional diffusion, explain the technical background, and derive the corresponding equations.
- We propose a technical realization and implementation of the functional diffusion concept.
- We demonstrate functional diffusion on irregular domains that are challenging to handle for existing diffusion methods.
- We demonstrate improved results on shape completion from sparse point clouds.

2. Related Work

2.1. Generative Models

Generative models have been extensively explored for image data. We have seen several popular generative models in past years such as Generative Adversarial Networks (GANs) [9], Variational Autoencoders (VAEs) [20] and Diffusion Probabilistic Models (DPMs) [12]. GANs utilize an adversarial training process. The versatility in generating high-dimensional data has been proven by numerous applications and improvements. VAEs aim to learn a representation space of the data with an autoencoder and enable the generation of new samples by sampling from the learned space. However, the quality is often lower than GANs. This idea is further improved in DPMs. Instead of decoding the representation with a one-step decoder, DPMs developed a new mechanism of progressive decoding. DPMs have demonstrated remarkable success in capturing and generating complex patterns in image data [12, 14, 34, 35].

2.2. Diffusion probabilistic models

When DPMs were invented in the beginning, they showed significant advantages in generating quality and diversity. However, the disadvantages are also obvious. For example, the sampling process is slower than other generative models. Some works [17, 23, 24, 37] are dedicated to solving the slow sampling problem. On the other hand, these works [1, 33] are proposed to solve the cases of non-Gaussian noise/degradation. However, our focus is to propose a new diffusion model for functional data. Common data forms like images can be seen as lying in a finite-dimensional space. However, a function is generally infinite-dimensional. It is not straightforward to adapt existing diffusion models for functional data. A direct solution is a two-stage training method. The first stage is to fit a network to encode functions with finite-dimensional latent space. In the later stage, a generative diffusion model is trained in the learned latent space. Many methods follow this design [5, 28, 46]. On the other hand, SSDNerf [4] combines both stages into one that jointly optimizes an autodecoder and a latent diffusion model. However, the method still trains diffusion in the latent space. The most related work to our proposed method is DPF [48]. However, DPF still works on data sampled on a discrete grid. Thus the

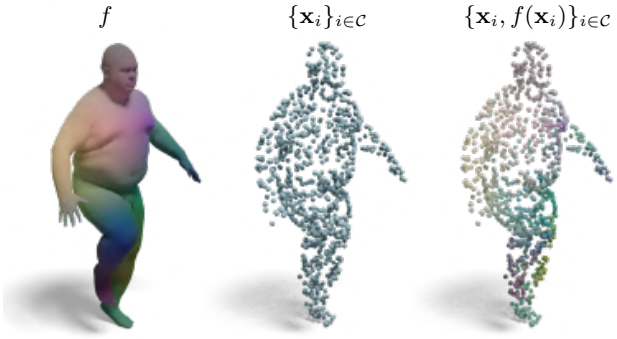

Figure 3. **Function approximation.** We illustrate how to approximate a function with its discretized state. Left: a function whose domain $\mathcal{X}$ is a manifold. Middle: sampled points in the domain. Right: the sampled points and the corresponding function values. Different from DPMs which sample on a grid of a fixed resolution, we do not have this restriction.

generated sample is still defined in a fixed resolution. Also, DPF is time-consuming when sampling data of a large resolution. Other related works [2, 7, 8, 10, 18, 22, 25] focused on building mathematical background of diffusion models in function space but did not show convincing practical results. We refer the reader to a recent survey of diffusion models in various domains besides images [31].

2.3. Neural Fields

Neural networks are often used to represent functions with a continuous domain. Here are some types of neural field applications: 1) in computer graphics and geometry processing, 3D shapes can be represented with implicit functions and thus are suitable to be modeled with neural networks [5, 21, 26, 29, 30, 38, 39, 44, 46]; 2) 3D textured objects and scenes can be rendered with radiance fields [27] which are also modeled with MLPs; 3) in physics, researchers use neural networks to represent complex functions which serve as solutions of differential equations [32]. Because of the universal approximation ability of neural networks, neural fields often provide flexibility in handling complex and high-dimensional data, and they can be trained end-to-end using gradient-based optimization techniques. Most importantly, neural fields can hold data sampled from an infinite large resolution. We refer the reader to a recent survey for more details on neural fields [43].

3. Methodology

We first introduce the definition of the functional diffusion in Sec. 3.1. Then we show how to train the denoising network in Sec. 3.2. Lastly, we show how we sample a function from the trained functional diffusion models in Sec. 3.3.

DPM	Proposed
Gaussian $\mathbf{n} \in \mathbb{R}^C$	Random function $g : \mathcal{X} \to \mathcal{Y}$
Finite-dim $\mathbf{x}_0 \in \mathbb{R}^C$	Infinite-dim $f_0 : \mathcal{X} \to \mathcal{Y}$
Noised $\mathbf{x}_t \in \mathbb{R}^C$	Noised function $f_t : \mathcal{X} \to \mathcal{Y}$
—"—	Context $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$
—"—	Queries $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{Q}}$
$D_\theta : \mathbb{R}^C \to \mathbb{R}^C$	$D_\theta : \{f : \mathcal{X} \to \mathcal{Y}\} \to \{f : \mathcal{X} \to \mathcal{Y}\}$

Table 1. **Comparison of classical DPMs and the proposed method.** For DPMs, the data samples are finite-dimensional and the denoiser is a function of the noised data $\mathbf{x}_t$. Our method deals with infinite-dimensional functions with a continuous domain. Thus the denoiser D_θ is becoming a "function of a function". This inspires us to seek a solution to find a way to process infinite-dimensional functions with neural networks. Also, note that DPM is a special case when $\mathcal{Q} = \mathcal{C}$.

3.1. Problem Definition

The training dataset $\mathcal{D}$ contains a collection of functions f_0 with continuous domains $\mathcal{X}$ and range $\mathcal{Y}$,

$$f_0 : \mathcal{X} \to \mathcal{Y}. \tag{1}$$

For example, we can represent watertight meshes as signed distance functions $f_0 : \mathbb{R}^3 \to \mathbb{R}^1$. We also define function set $\mathcal{F}$ where each element is also a function

$$g : \mathcal{X} \to \mathcal{Y}. \tag{2}$$

The function g works similarly to the noise in traditional diffusion models. However, in functional diffusion, we require the "noise" to be a function. We can obtain a "noised" version f_t given f_0 from $\mathcal{D}$ and g from $\mathcal{F}$,

$$f_t(\mathbf{x}) = \alpha_t \cdot f_0(\mathbf{x}) + \sigma_t \cdot g(\mathbf{x}), \tag{3}$$

where t is a scalar from 0 (least noisy) to 1 (most noisy). We name f_t as the *noised state* at timestep t. The terms α_t and σ_t are positive scalars. In DDPM [12], they satisfy $\alpha_t^2 + \sigma_t^2 = 1$. Thus α_t is a monotonically decreasing function of t, while σ_t is monotonically increasing. VDM [19] characterizes α_t^2/σ_t^2 as signal-to-noise ratio (SNR).

Our goal is to train a denoiser which can approximate:

$$D_\theta[f_t, t](\mathbf{x}) \approx f_0(\mathbf{x}). \tag{4}$$

This is often called x_0-prediction [19] in the literature of diffusion models. However, other loss objectives also exist, *e.g.*, ϵ-prediction [12], v-prediction [36] and f-prediction [17]. We emphasize that choosing x_0-prediction is important in the proposed functional diffusion which will be explained later.

The objective is

$$\mathbb{E}_{f_0\in\mathcal{D}, g\in\mathcal{F}, t\sim T(t)} \left[w(t) d\left(D_\theta[f_t, t], f_0\right)^2 \right], \tag{5}$$

Algorithm 1 Training

repeat
  $g \in \mathcal{F}$ ▷ noise function
  $f_0 \in \mathcal{D}$ ▷ training function
  $t \sim \mathcal{T}$ ▷ noise level
  $\alpha_t = 1/\sqrt{t^2+1}, \sigma_t = t/\sqrt{t^2+1}$ ▷ SNR
  Sample $\mathcal{C}$ ▷ context
  Evaluate $\{g(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ and $\{f_0(\mathbf{x}_i)\}_{i\in\mathcal{C}}$
  Calculate the context $\{f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ with Eq. (3)
  Sample $\mathcal{Q}$ ▷ query
  Optimize Eq. (9) ▷ denoise
until convergence

Algorithm 2 Sampling

Ensure: Sample $\mathcal{C}$ and $g \in \mathcal{F}$
Let $f_t = g$
Evaluate $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$
for $k \in \{N, N-1, \dots, 2, 1\}$ **do**
  $t_k = T(k), t_{k-1} = T(k-1)$
  $\alpha_t = 1/\sqrt{t_k^2+1}, \alpha_s = 1/\sqrt{t_{k-1}^2+1}$
  $\sigma_t = t_k/\sqrt{t_k^2+1}, \sigma_s = t_{k-1}/\sqrt{t_{k-1}^2+1}$
  Predict $\{f_s(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ with Eq. (11)
  Let $f_t \leftarrow f_s$
end for
$f_0(\mathbf{x}) = D_\theta\left(\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}, t, \mathbf{x}\right)$

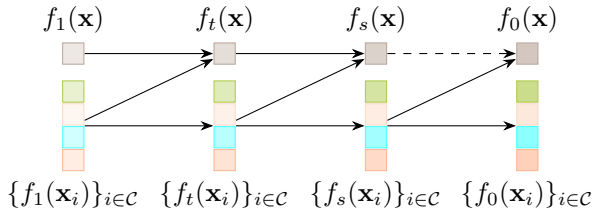

Figure 4. **Inference chain.** We show a simplified 4-steps generating process in Eq. (11). The arrows show how the data flows during inference. $\mathbf{x}$ represents an arbitrary query coordinate. $\mathcal{C}$ is the context set. The state $f_s(\mathbf{x})$ requires to know both the previous state $f_t(\mathbf{x})$ and $\{f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$. Thus it is dependent on all previous states $f_{<s}$. $f_0(\mathbf{x})$ is the only exception because $\sigma_0 = 0$. Thus $f_0(\mathbf{x})$ is fully decided by the penultimate state $\{f_s(\mathbf{x}_i)\}_{i\in\mathcal{C}}$.

where $d(\cdot,\cdot)$ is a metric defined on the function space $\{f : \mathcal{X} \to \mathcal{Y}\}$ and $w(t)$ is a weighting term. We summarize the differences between the vanilla DPMs and the proposed functional diffusion in Tab. 1.

3.2. Parameterization

Denoising network. The functional D_θ is parameterized by a neural network θ. It is impossible to feed the noised state function f_t directly to the neural network as input. In order to make the computation tractable, our idea is to represent functions with a set of coordinates together with their corresponding values. Thus we sample (discretize) a set $\{\mathbf{x}_i \in \mathcal{X}\}_{i\in\mathcal{C}}$ in the domain $\mathcal{X}$ of f_t. We feed this set to the denoising network along with the corresponding function values $\{f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ (also see Fig. 3 for an illustration),

$$D_\theta[f_t, t](\mathbf{x}) \approx D_\theta\left(\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}, t, \mathbf{x}\right). \tag{6}$$

The design of the network D_θ varies for different applications. However, we give a template design in later sections.

Function metric. For the function metric $d(\cdot,\cdot)$, we choose the l-2 metric,

$$d(D_\theta[f_t,t], f_0) = \left(\int_{\mathcal{X}} \left|D_\theta[f_t,t](\mathbf{x}) - f_0(\mathbf{x})\right|^2 \mathrm{d}\mathbf{x}\right)^{1/2} \tag{7}$$

The approximation of the metric $d(\cdot,\cdot)$ is also done by sampling (Monte-Carlo integration),

$$d\left(D_\theta[f_t,t], f_0\right) \approx \left(\sum_{i\in\mathcal{Q}} \left|D_\theta[f_t,t](\mathbf{x}_i) - f_0(\mathbf{x}_i)\right|^2\right)^{1/2}. \tag{8}$$

Thus our loss objective in Eq. (5) can be written as,

$$w(t) \sum_{i\in\mathcal{Q}} \left|D_\theta\left(\{\mathbf{x}_j, f_t(\mathbf{x}_j)\}_{j\in\mathcal{C}}, t, \mathbf{x}_i\right) - f_0(\mathbf{x}_i)\right|^2. \tag{9}$$

Pixel diffusion (DPMs trained in the pixel space) can be seen as a special case of the model by sampling $\mathcal{C}$ on a fixed regular grid and letting $\mathcal{Q} = \mathcal{C}$. DPF [48] uses the term *context* for $\mathcal{C}$ and *query* for $\mathcal{Q}$. Thus we also follow this convention. We summarized how we design $\mathcal{Q}$ and $\mathcal{C}$ for different tasks in Tab. 2.

Initial noise function. For now, we still do not know how to choose the noise function set $\mathcal{F} = \{g : \mathcal{X} \to \mathcal{Y}\}$. In DPMs, the noise is often modeled with a standard Gaussian distribution. Gaussian processes are an infinite-dimensional generalization of multivariate Gaussian distributions. Thus, it is straightforward to use Gaussian processes to model the noise functions. However, in our practical experiments, we find sampling from Gaussian processes is time-consuming during training. Thus, we choose a simplified version. In the case of Euclidean space, we sample Gaussian noise on a grid in $\mathcal{X}$. Then other values are interpolated with the values on the grid. If the domain $\mathcal{X}$ is a non-Euclidean manifold which is difficult to sample, instead we define the noise function in the ambient space of $\mathcal{X}$. In this way, we defined a way to build the function set $\mathcal{F}$. During training, in each iteration, we sample a noise function g from this set.

To sum up, the training algorithm can be found in Algorithm 1.

Task	Input	Domain Space	Output	Range Space	Condition	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{Q}\rvert$
3D Shapes	Coordinates	$\mathbb{R}^3$	SDF	$\mathbb{R}^1$	Surface Point Clouds	49152	2048
3D Deformation	Points on Manifold	$\mathcal{M}$	Vector	$\mathbb{R}^3$	Sparse Correspondence	16384	2048

Table 2. **Task designs.** We show the two main tasks used to prove the efficiency of the proposed method.

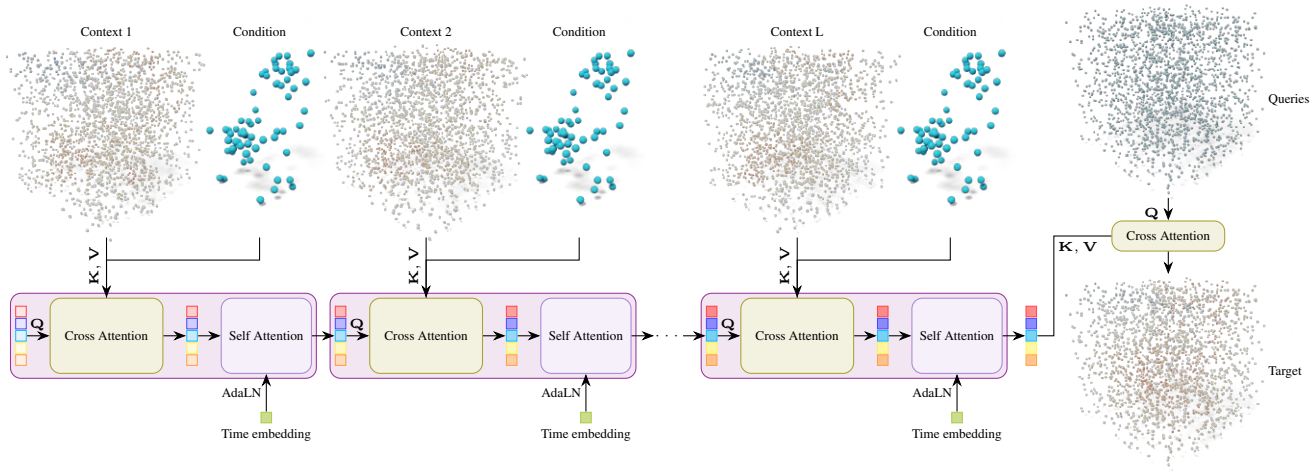

Figure 5. **The network design of the SDF diffusion model.** The context set is split into L smaller ones. They (and optionally conditions such as sparse surface point clouds) are fed into different stages of the network by using cross-attention. The time embedding is injected into the network in every self-attention layer by adaptive layer normalization. After L stages, we obtain the representation vector sets and they will be used to predict values of arbitrary queries. For SDFs, we optimize simple minimum squared errors.

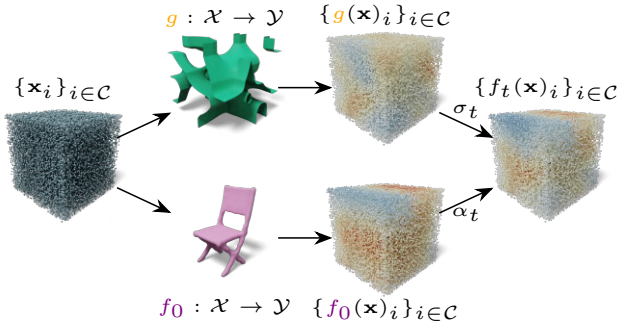

Figure 6. **Evaluation of the context $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$.** We sample a set of points $\{\mathbf{x}_i\}_{i\in\mathcal{C}}$ in the domain $\mathcal{X}$. We evaluate the values both in the noise function g and the ground-truth function f_0. This is how Eq. (3) works.

3.3. Inference

We adapt the sampling method proposed in DDIM [37] for the proposed functional diffusion. As shown in Eq. (3), the generating process is from timestep $t = 1$ (most noisy) to $t = 0$ (least noisy). We start from an initial noise function $f_1 = g \in \mathcal{F}$. Given the noised state f_t at the timestep t, we obtain the "less" noised state f_s where $0 \le s < t \le 1$,

$$f_s = \alpha_s \underbrace{D_\theta[f_t, t]}_{\text{estimated } f_0} + \sigma_s \underbrace{\left(\frac{f_t - \alpha_t D_\theta[f_t, t]}{\sigma_t}\right)}_{\text{estimated } g}. \tag{10}$$

We can also write,

$$f_s(\mathbf{x}) = \frac{\sigma_s}{\sigma_t} f_t(\mathbf{x}) + \left(\alpha_s - \sigma_s \frac{\alpha_t}{\sigma_t}\right) D_\theta\left(\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}, t, \mathbf{x}\right) \tag{11}$$

We sample a set $\mathcal{C}$ and evaluate $\{\mathbf{x}_i, f_t(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ in every denoising step. The Eq. (11) shows how the one-step denoised function f_s is obtained. We recursively apply the denoising process from f_t to f_s. In the end, we obtain the generated sample f_0. More importantly, to obtain intermediate function values $f_s(\mathbf{x})$ for an arbitrary $\mathbf{x}$, we need to know $f_t(\mathbf{x})$, and thus all previous states for $\mathbf{x}$. However, when we are denoising the last step of the generation process, $\sigma_s = 0$, which means the generated function $f_0(\mathbf{x})$ is only dependent on the penultimate state of the function $\{\mathbf{x}_i, f_s(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ (also see Fig. 4). With this observation, we can obtain the generated function values without knowing the intermediate states except the penultimate one. During inference, we only need to denoise the context set. This is a key property of the proposed method which can accelerate the generation/inference.

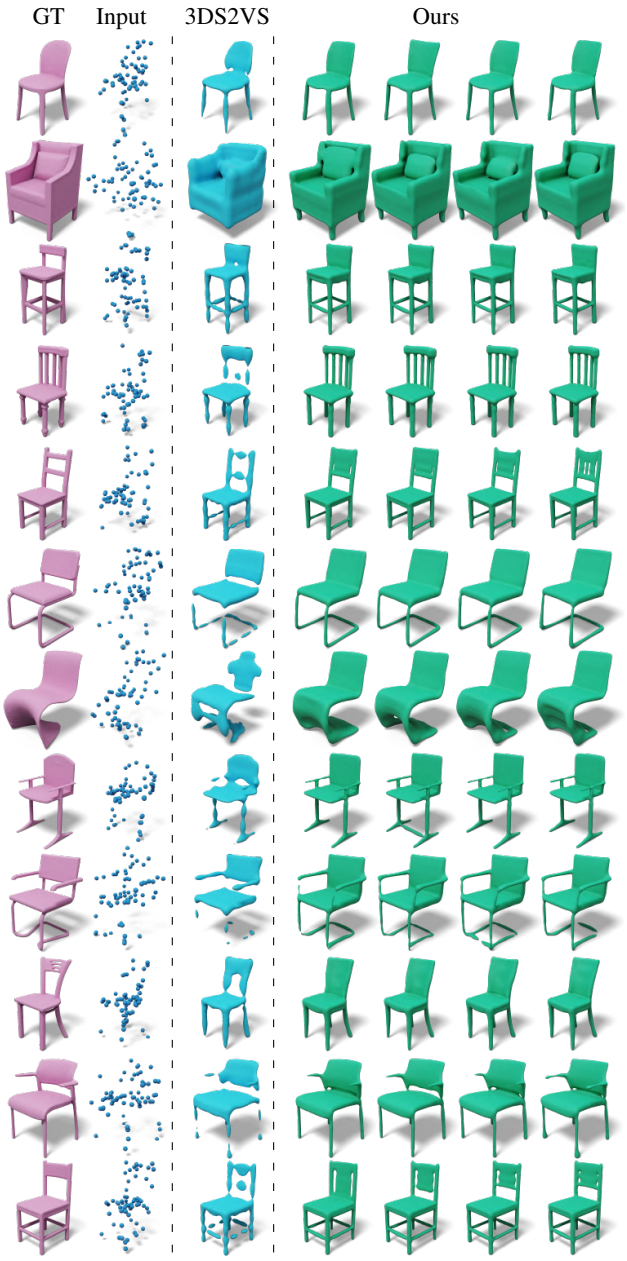

Figure 7. **SDF diffusion results.** We show ground-truth meshes and the input sparse point cloud (64 points) on the left. We compare our results with 3DS2VS. Since our model is probabilistic, we can output multiple different results given different random seeds. Our results are detailed and complete. However, the traditional method struggles to reconstruct correct objects.

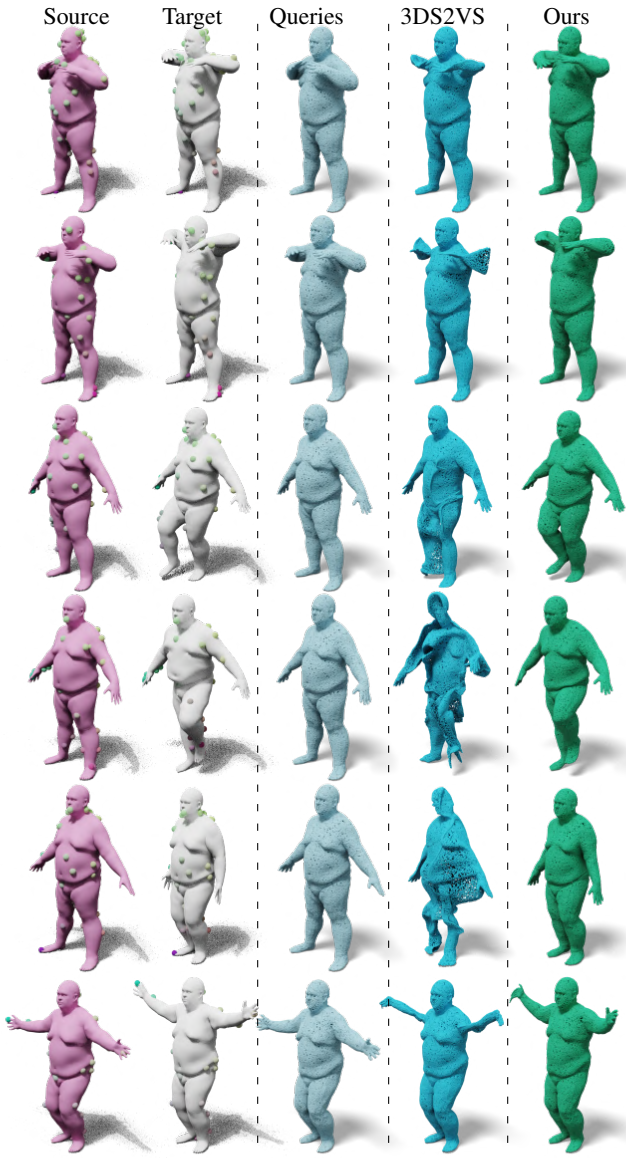

Figure 8. **Deformation diffusion results.** In the left, we show both the source and the target frame and the sparse correspondence (small spheres on the body surface).

On the contrary, DPF needs to denoise the $f_t(\mathbf{x})$ in every denoising step t. While our method only needs to denoise a small context set $\mathcal{C}$. In practice, we often sample a signed distance function with a high-resolution grid G^3 (*e.g.*, 128^3). This is complicated for DPF because the number of queries is $\underline{G^3 \times T}$ where T is the number of denoising steps ($T = 16$ in our experiments). For our method, the number of queries is $\underline{|\mathcal{C}| \times T + G^3}$ where $G^3 \gg |\mathcal{C}| = 49152$. The difference is even more significant when at higher resolution like $G = 256$ or $G = 512$. The sampling algorithm is summarized in Algorithm 2.

4. Results: 3D Shapes

In computer graphics, 3D models are often represented with a function $f : \mathbb{R}^3 \to \mathbb{R}^1$ where the input $\mathbf{x}$ is a 3D coordinate and the output y is the signed distance to the 3D boundary $\partial\Omega$, *i.e.*, $y = \text{dist}(\mathbf{x}, \partial\Omega)$ when $\mathbf{x} \in \Omega$ and $y = -\text{dist}(\mathbf{x}, \partial\Omega)$ when $\mathbf{x} \in \mathcal{X} \setminus \Omega$. The signed dis-

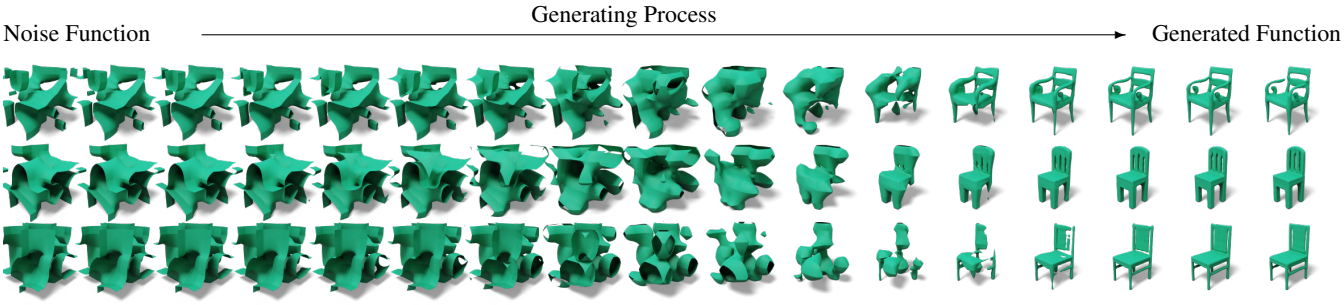

Figure 9. **Generating process of SDFs.** We show the generating process of 3 samples. In the far left, the initial noise functions are shown. In the far right, we show the generated samples. To make the visualization clear, we only show the zero-isosurface. However, the functions are actually densely defined everywhere in the space.

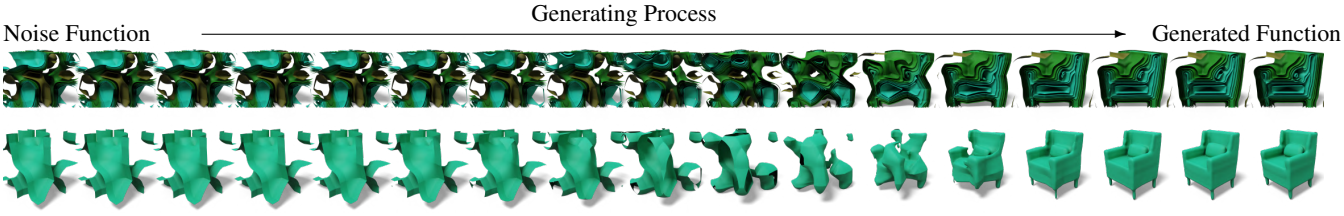

Figure 10. **Generating process of SDFs.** In the top row, we show multiple isosurfaces of each intermediate step of the generating process. They are $[-0.5, -0.2, -0.1, -0.05, -0.01, 0, 0.05, 0.1, 0.2]$ from outer to inner. They are cut with a plane to show the inner structure. In the bottom row, we show the zero-isosurface for comparison.

tance function (SDF) satisfies the partial differential equation (PDE), a.k.a., Eikonal equation,

$$\begin{aligned} |\nabla f(\mathbf{x})| &= 1, \\ f(\mathbf{x}) &= q(\mathbf{x}),\ \forall \mathbf{x} \in \partial\Omega, \end{aligned} \tag{12}$$

where $q(\mathbf{x})$ is the boundary condition. The task of predicting SDFs given a surface point cloud is equivalent to solving this PDE with a given boundary condition (the surface point cloud). This problem is solved in prior works. But most works focus on surface reconstruction only by predicting binary occupancies [26, 30, 44, 45] or truncated SDFs [29]. Thus they are not really solving this equation and cannot be used in some SDF-based applications such as sphere tracing [11]. This is a challenging task according to prior works. We choose the task to show the capability of the proposed method.

4.1. Experiment design

We choose a sparse observation of the boundary condition (surface point cloud) which only contains 64 points as the input of the model. We compare our method with OccNet [26] and the recently proposed 3DShape2VecSet [46]. As an example to show how the proposed method works, we first show how the noised state is obtained in Fig. 6. The context is then fed into the denoising network (see Fig. 5).

4.2. Results comparison

We show visual comparisons in Fig. 7. Apparently, our method shows a significant advantage over prior methods in this task. We not only output detailed and full meshes but also show the multimodality of the proposed method. However, prior works are unable to give correct reconstructions, thus also proving this task is challenging given the sparse observation.

We also show some quantitive comparison in Tab. 3. Chamfer distances and F-scores are commonly used in surface reconstruction evaluation [6, 26, 45, 46]. Furthermore, we design two new metrics. As discussed above, we are actually solving a partial differential equation. Thus, we can define the two metrics,

$$\textsc{Eikonal}(f) = \frac{1}{|\mathcal{E}_\mathcal{X}|} \sum_{i \in \mathcal{E}_\mathcal{X}} \left\| |\nabla f(\mathbf{x}_i)| - 1 \right\|^2, \tag{13}$$

$$\textsc{Boundary}(f) = \frac{1}{|\mathcal{E}_\Omega|} \sum_{i \in \mathcal{E}_\Omega} \left\| f(\mathbf{x}_i) - q(\mathbf{x}_i) \right\|^2, \tag{14}$$

where EIKONAL reflects that if the solutions satisfy the Eikonal equation and BOUNDARY shows if the solutions satisfy the boundary condition. $\mathcal{E}_\mathcal{X}$ is a set sampled in the bounding volume which contains 100k points and $\mathcal{E}_\Omega$ is a set sampled on the surface which also contains 100k points. Our method leads a large margin over existing methods in all metrics. This is also consistent with what is shown in the visual comparison.

4.3. Generating process

In Fig. 9 and Fig. 10, we show the intermediate noised function obtained during the generating process. Unlike simi-

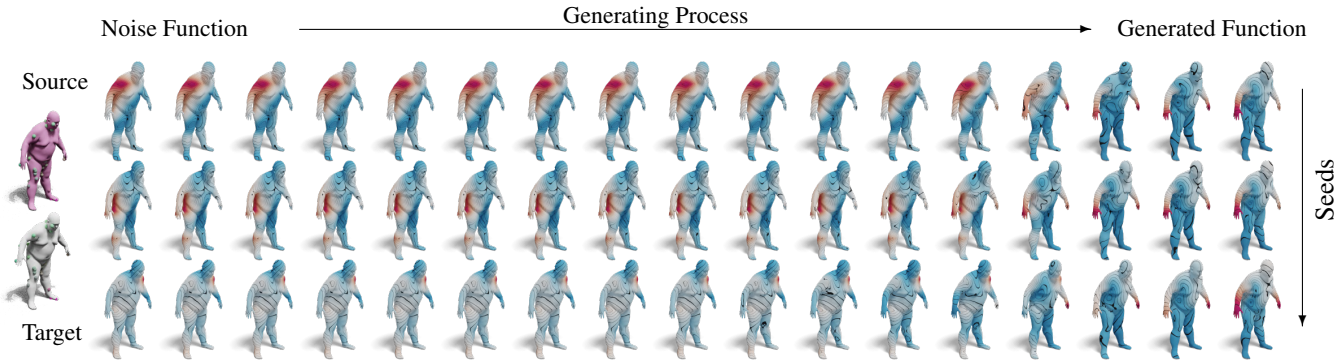

Figure 11. **Deformation Fields.** In the far left, we show the source and target frame along with the sparse correspondence. We show 3 samples generated given the same condition. In each frame, we show the deformation field on the surface. However, to simplify the visualization, the colors only indicate the magnitudes of deformation while ignoring the directions. The three samples started from different functions. But in the end, the model outputs almost the same deformation fields.

	Chamfer ↓	F-Score ↑	Boundary ↓	Eikonal ↓
OccNet	0.166	0.531	0.019	0.032
3DS2VS	0.144	0.608	0.016	0.038
Proposed	**0.101**	**0.707**	**0.012**	**0.024**

Table 3. **SDF diffusion results.** The task is SDF prediction given sparse observations on the surface. We show two commonly used metrics, Chamfer distances and F-scores. Additionally, we show the two newly proposed metrics based on the definition of partial differential equations.

lar methods proposed before which predict binary occupancies or truncated SDFs, we can generate raw SDFs directly which can be directly used in some SDF-based applications.

5. Results: 3D Deformation

The task is defined as follows: given meshes sampled in a dynamic shape sequence, and limited (32) sparse correspondence between two meshes (see Fig. 8), we want to predict a deformation field. Specifically, the deformation field takes a point on the surface of the source frame as input and outputs a deformation vector which should map the point to the target frame. The network design is similar to the Fig. 5. However, we only use 16384 points in the context set because the data is simpler than a complicated SDF. We also adapt the method 3DS2VS here to do the deformation field prediction. From the visual results in Fig. 8, we can see that our method can show vivid surface deformation, while 3DS2VS is unable to map source points to the target frame especially when the motion is large. We also show the quantitative comparisons in Tab. 4.

In Fig. 11, we show what the generated deformation fields look like. Given the same condition, three sampling processes are visualized.

	MSE ($\times 10^4$) ↓
3DS2VS	13.32
Proposed	**6.91**

Table 4. **Quantitative results in deformation field generation.** The numbers are evaluated using minimum squared error between the predicted deformation and the ground-truth.

6. Conclusions

We proposed a new class of generative diffusion models, called functional diffusion. In contrast to previous work, functional diffusion works on samples that are represented by functions. We derived the necessary foundations for functional diffusion and proposed a first implementation based on the transformer architecture.

Limitations. During our work, we identified two main limitations of our method. First, functional diffusion requires a fair amount of resources to train. However, other diffusion models also share the same issue. We would expect that significantly more GPUs would be required to train on large datasets such as Objaverse-XL. Therefore, it may be interesting to explore cascaded functional diffusion in future work. Second, our framework has an additional parameter, the sampling rate of the sampled function representation. During training, it is beneficial but also necessary to explore this hyperparameter.

Future works. In future work, we also would like to explore the application of functional diffusion to time-varying phenomena, such as deforming, growing, and 3D textured objects. Furthermore, we would like to explore functional diffusion in the field of functional data analysis (FDA) [42] which studies data varying over a continuum.

References

[1] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *arXiv preprint arXiv:2208.09392*, 2022. 2

[2] Sam Bond-Taylor and Chris G Willcocks. ∞-diff: Infinite resolution diffusion with subsampled mollified states. *arXiv preprint arXiv:2303.18242*, 2023. 3

[3] Wei Cao, Chang Luo, Biao Zhang, Matthias Nießner, and Jiapeng Tang. Motion2vecsets: 4d latent vector set diffusion for non-rigid shape reconstruction and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024. 1

[4] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. In *ICCV*, 2023. 2

[5] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023. 1, 2, 3

[6] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 7

[7] Vincent Dutordoir, Alan Saul, Zoubin Ghahramani, and Fergus Simpson. Neural diffusion processes. In *International Conference on Machine Learning*, pages 8990–9012. PMLR, 2023. 3

[8] Giulio Franzese, Simone Rossi, Dario Rossi, Markus Heinonen, Maurizio Filippone, and Pietro Michiardi. Continuous-time functional diffusion processes. *arXiv preprint arXiv:2303.00800*, 2023. 3

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27:2672–2680, 2014. 2

[10] Paul Hagemann, Lars Ruthotto, Gabriele Steidl, and Nicole Tianjiao Yang. Multilevel diffusion: Infinite dimensional score-based diffusion models for image generation. *arXiv preprint arXiv:2303.04772*, 2023. 3

[11] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996. 7

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2, 3

[13] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1

[14] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022. 1, 2

[15] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022. 1

[16] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 1

[17] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 2, 3

[18] Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. *arXiv preprint arXiv:2305.17209*, 2023. 3

[19] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 3

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[21] Tianyang Li, Xin Wen, Yu-Shen Liu, Hua Su, and Zhizhong Han. Learning deep implicit functions for 3d shapes with dynamic code clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12840–12850, 2022. 3

[22] Jae Hyun Lim, Nikola B Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, et al. Score-based diffusion models in function space. *arXiv preprint arXiv:2302.07400*, 2023. 3

[23] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 2

[24] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 2

[25] Emile Mathieu, Vincent Dutordoir, Michael J Hutchinson, Valentin De Bortoli, Yee Whye Teh, and Richard E Turner. Geometric neural diffusion processes. *arXiv preprint arXiv:2307.05431*, 2023. 3

[26] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 3, 7

[27] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[28] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias

Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023. 2

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3, 7

[30] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 3, 7

[31] Ryan Po, Wang Yifan, and Vladislav Golyanik et al. State of the art on diffusion models for visual computing. In *arxiv*, 2023. 3

[32] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017. 3

[33] Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022. 2

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 2

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 1, 2

[36] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 3

[37] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 5

[38] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6504–6513, 2021. 3

[39] Jiapeng Tang, Lev Markhasin, Bi Wang, Justus Thies, and Matthias Nießner. Neural shape deformation priors. *Advances in Neural Information Processing Systems*, 35: 17117–17132, 2022. 3

[40] Jiapeng Tang, Angela Dai, Yinyu Nie, Lev Markhasin, Justus Thies, and Matthias Niessner. Dphms: Diffusion parametric head models for depth-based tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024. 1

[41] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024. 1

[42] Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Müller. Functional data analysis. *Annual Review of Statistics and its application*, 3:257–295, 2016. 8

[43] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. 3

[44] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6239–6249, 2022. 3, 7

[45] Biao Zhang, Matthias Niessner, and Peter Wonka. 3DILG: Irregular latent grids for 3d generative modeling. In *Advances in Neural Information Processing Systems*, pages 21871–21885, 2022. 7

[46] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4), 2023. 1, 2, 3, 7

[47] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *arXiv preprint arXiv:2305.04461*, 2023. 1

[48] Peiye Zhuang, Samira Abnar, Jiatao Gu, Alex Schwing, Joshua M. Susskind, and Miguel Ángel Bautista. Diffusion probabilistic fields. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 4