Geometry Distributions

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

Jing Ren
ETH Zurich
jing.ren@inf.ethz.ch

Peter Wonka
KAUST
pwonka@gmail.com

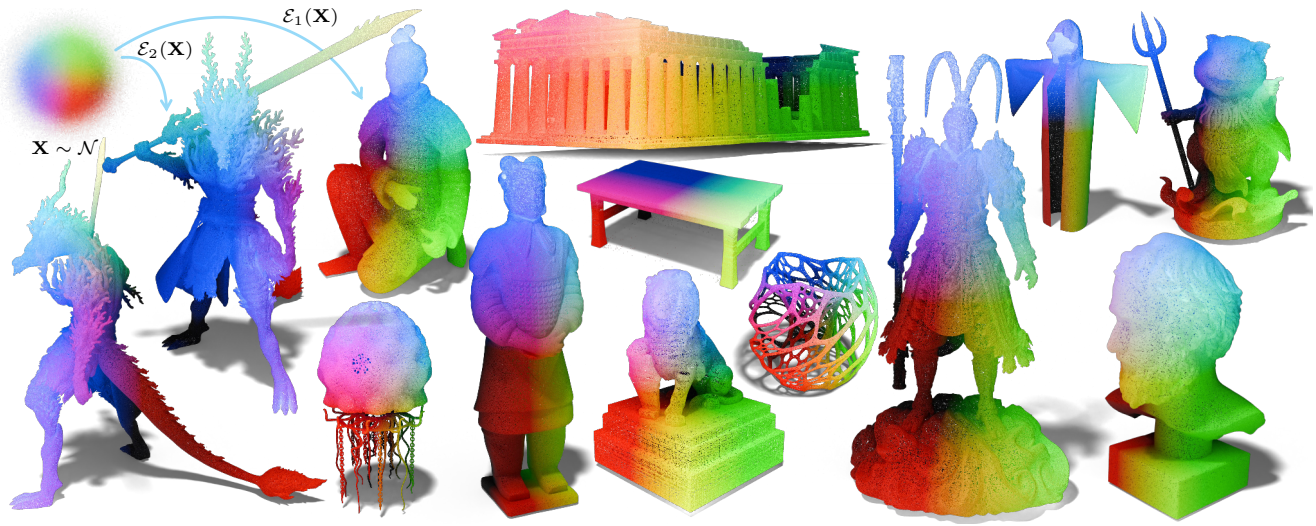


Figure 1. Our representation can handle 3D geometry with complex details, high genus, sharp features, and non-watertight surfaces: our trained diffusion networks $\mathcal{E}_i$ can transform the samples $\mathbf{X}$ from a Gaussian distribution $\mathcal{N}$ to the geometry $\mathcal{M} \subset \mathbb{R}^3$. The colors indicate the correspondence between the Gaussian noise and the surface points.

Abstract

Neural representations of 3D data have been widely adopted across various applications, particularly in recent work leveraging coordinate-based networks to model scalar or vector fields. However, these approaches face inherent challenges, such as handling thin structures and non-watertight geometries, which limit their flexibility and accuracy. In contrast, we propose a novel geometric data representation that models geometry as distributions—a powerful representation that makes no assumptions about surface genus, connectivity, or boundary conditions. Our approach uses diffusion models with a novel network architecture to learn surface point distributions, capturing fine-grained geometric details. We evaluate our representation qualitatively and quantitatively across various object types, demonstrating its effectiveness in achieving high geometric fidelity. Additionally, we explore applications using our representation, such as textured mesh representation, neural surface compression, dynamic object modeling, and rendering, highlighting its potential to advance 3D geometric learning.

1. Introduction

Geometry representations are at the heart of most 3D vision problems. With the rapid advancement of deep learning, there is growing interest in developing neural network-friendly geometric data representations. Recent advances in this field, particularly those based on coordinate networks, have shown promise in modeling 3D geometry for various applications, as their functional nature integrates well with neural networks. However, they also face challenges like limited accuracy in capturing complex geometric structures and difficulties in handling non-watertight objects.

To overcome these challenges, we propose a new geometric data representation, possessing a simple and consistent data structure capable of accommodating shapes with varying genus, boundary conditions, and connectivity—whether open, watertight, fully connected, or not (see Fig. 2). A key insight is that any surface, regardless of its topology or structural integrity, can be closely approximated by a sufficiently large number of points sampled on the surface. Recent advancements in generative models have shown that, in theory, they can sample an infinite amount of data from a distribution. Building on these

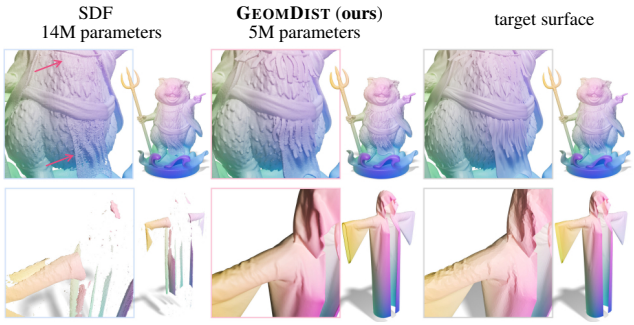
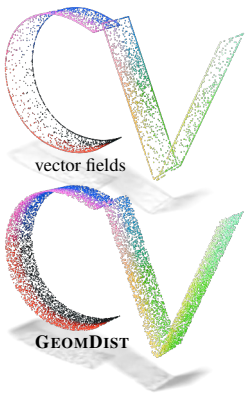


Figure 2. Compared to signed distance functions (SDFs), our **GEOMDIST** can model open and non-watertight objects using significantly *fewer* network parameters. See the Appendix for the meshing algorithm used in this figure. The SDF is fit using Instant-NGP [31], with isosurface extraction evaluated on a 512-resolution grid. We observe that SDFs struggle to represent thin structures or non-watertight geometry.

insights, we model 3D geometry as a *distribution* of surface points, encoded into a diffusion model. Unlike triangle mesh representations, which are specific discretizations of the underlying surface, or point clouds, which represent a particular sampling choice, our approach models the distribution of all possible surface points, providing a more continuous and accurate encoding of the underlying geometry.

Diffusion models, widely recognized for their effectiveness in 2D content generation, have emerged as a leading approach among generative models. However, their application to 3D geometry remains largely unexplored. We found that 3D geometry context presents unique challenges: direct adaptation often falls short in capturing geometric details and results in inaccurate geometry recovery.

In this work, we introduce **GEOMETRY DISTRIBUTIONS** (or **GEOMDIST** in short), a new representation for general geometric data. Our approach leverages a diffusion model with a novel network architecture. By solving a forward ordinary differential equation (ODE), we map spatial points, sampled from Gaussian *noise space*, to surface points in *shape space*, enabling an infinite set of points to represent the geometry. This allows us to sample on the surface uniformly comparing to existing vector fields-based formulation (see the inset and Fig. 3). Additionally, we derive the backward ODE algorithm, allowing for inverse mapping from the shape space back to noise space. Our results demonstrate the accuracy and robustness of **GEOMDIST** across a broad range of complex structures. Furthermore, our approach enables the simultaneous encoding of texture or motion information alongside geometry.



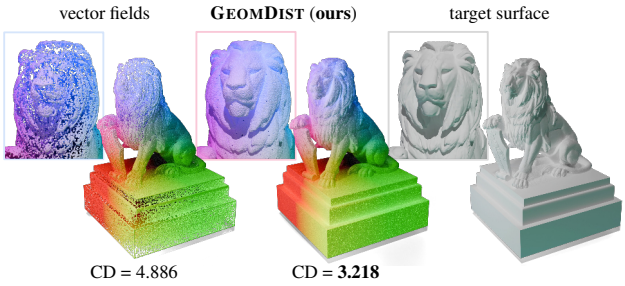


Figure 3. Compared to vector fields-based method, our **GEOMDIST** produces more uniformly distributed samples with higher fidelity. The chamfer distance ($\times 10^3$) between the samples and target surface is reported below.

To summarize, **GEOMDIST** facilitates a highly accurate yet compact neural representation of 3D geometry, demonstrating significant potential for future applications, including textured mesh representation, neural surface compression, dynamic object neural modeling, and photo-realistic rendering with Gaussian splatting.

2. Related Works

2.1. Different representations for 3D geometry

Existing geometry representations—such as meshes, voxels, point clouds, and implicit functions—each offer distinct advantages but also have inherent limitations. Triangle or polygonal meshes, which are commonly used in traditional geometry processing [4], are not ideal for geometric learning due to their inconsistent data structures when dealing with shapes that have a different number of vertices and different connectivity [5, 13].

Voxels, with their inherent grid-based structure, are ideal for learning-based tasks. However, they are memory-intensive, especially when high resolution is needed for capturing fine details [26, 42]. Point clouds, easily obtained from sensors, are widely used in geometric learning tasks [2, 17, 44]. However, they are essentially samples of the geometry, leading to potential information loss of the underlying geometry. Their expressiveness heavily depends on sampling density and uniformity, and the lack

types	*structure*	*infinity*	*surface*	*non-watertight*	*color/ textures*
point clouds	$\{\mathbf{p}_i \in \mathbb{R}^3\}_{i\in\mathcal{P}}$	✗	✗	✓	✓
meshes	$\left(\{\mathbf{v}_i \in \mathbb{R}^3\}_{i\in\mathcal{V}}, \{\mathbf{f}_j \in \mathcal{V}^3\}_{j\in\mathcal{F}}\right)$	✗	✓	✓	✓
voxels	$\mathbb{R}^{d\times h\times w\times c}$	✗	✓	✓	✓
SDFs	$\{\mathbf{p} \mid f(\mathbf{p}) = 0\}$	✓	✓	✗	✗
GEOMDIST	$\{\mathcal{E}(\mathbf{n})\}$	✓	✓	✓	✓

Table 1. Different representations for 3D geometric data.

of inherent point connectivity complicates defining surface structures, boundaries, or geodesics along surfaces. Implicit functions [30, 32] excel at generating smooth surfaces and representing complex topologies. However, they struggle with accurately modeling thin structures or non-watertight geometries (see Fig. 2 for an illustration). Additionally, integrating colors or textures with implicit functions is not straightforward.

Our goal is to design a new data representation for various 3D geometric learning tasks, featuring a network-friendly data structure that accommodates shapes with varying genus, boundary conditions, and connectivity, whether open, watertight, fully connected, or not. See Tab. 1 for a summary of different representations.

2.2. Diffusion models

Diffusion models are powerful generative models that transform data into noise through a forward diffusion process and learn to reverse this process to generate high-quality samples. Beginning with Denoising Diffusion Probabilistic Models [18], diffusion models have evolved into more efficient and flexible approaches [20, 25, 27, 40]. While our work does not contribute directly to advancements in diffusion models, we employ them as a foundation for modeling complex geometry. Our approach primarily builds upon the framework established in EDM [20].

Significant progress has been made in generating 3D geometry using diffusion models [7, 11, 19, 33, 35, 38, 45, 48, 53, 55, 57], with most approaches representing geometry via signed distance functions or occupancy fields. Fewer methods, however, focus on point clouds [28, 52, 59] or Gaussian point clouds [37, 51, 56]. We would also like to emphasize two related works [6, 46]. While both works are capable of encoding infinite-resolution point clouds using normalizing flow [36] and score-based generative models [41], they are only evaluated on toy datasets and do not demonstrate high-quality geometric details as we do. These models are trained on a dataset of 3D objects, treating each *object* as a single training sample. In contrast, our approach is fundamentally different, as we treat each *spatial point* as an individual training sample.

2.3. Coordinate-based neural representations

Signed-distance functions (SDFs) are widely used to represent 3D geometry [10, 15, 29, 31, 39, 43]. Instead of explicitly storing vertices or points, a network is trained to produce signed distances to the surface or signals indicating inside/outside for each spatial point, implicitly defining the shape's geometry. Although relatively easy to learn via neural networks, SDFs struggle to model non-watertight meshes. Follow-up works are then introduced to model open surfaces, where the outputs of the networks are unsigned distances [8, 16, 47, 58] or vectors [12, 47, 50] pointing toward the surface. These works primarily use networks to fit scalar fields or vector fields, representing 3D data through networks that map coordinates to scalar or vector values. Our approach, while distinct in methodology, shares a conceptual connection with these works: it can be interpreted as a trajectory field.

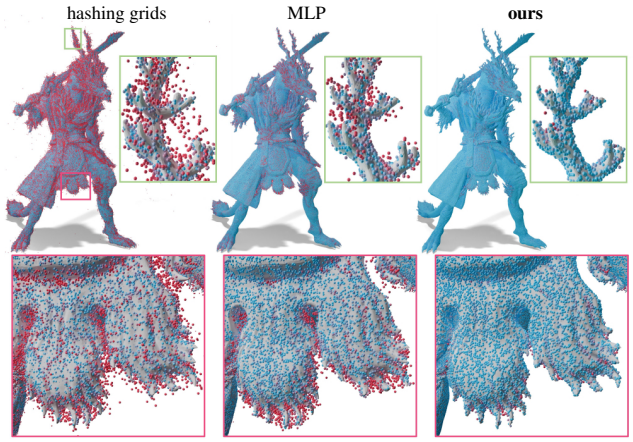


Figure 4. Heatmap of the L_2 distance from sampled points to the target surface using different network architectures.

2.4. Point-based graphics

Point-based computer graphics is an approach that represents 3D surfaces as sets of discrete points rather than traditional polygonal meshes. Unlike polygonal models that use vertices and edges to define shapes, point-based methods use individual points sampled across a surface to capture details directly. The field can be dated back to 1980s [24]. Early works [34, 61] investigated how to render with points. Until recently, works utilized point representations in differentiable rendering [22, 49]. Different from our method, these works focus on rendering with finite number of points.

3. Geometry Distributions

3.1. Problem formulation & motivations

Given a surface $\mathcal{M} \subset \mathbb{R}^3$, our goal is to model it as a probability distribution $\Phi_{\mathcal{M}}$, such that any sample $\mathbf{x} \sim \Phi_{\mathcal{M}}$ drawn from this distribution is a surface point, i.e., $\mathbf{x} \in \mathcal{M}$. In this way, the distribution $\Phi_{\mathcal{M}}$, which encodes the geometry $\mathcal{M}$, provides a flexible geometric representation-any sampling, whether dense or sparse, closely approximates the surface $\mathcal{M}$ at the target resolution. Inspired by the pioneering work "Geometry Images", which uses 2D images to represent 3D meshes [14], we name our representation as **GEOMETRY DISTRIBUTIONS**.

Numerous generative tasks have demonstrated the effectiveness of using diffusion models to learn the mapping from a Gaussian distribution to data distributions. While previous work is concerned with novel shape synthesis, we

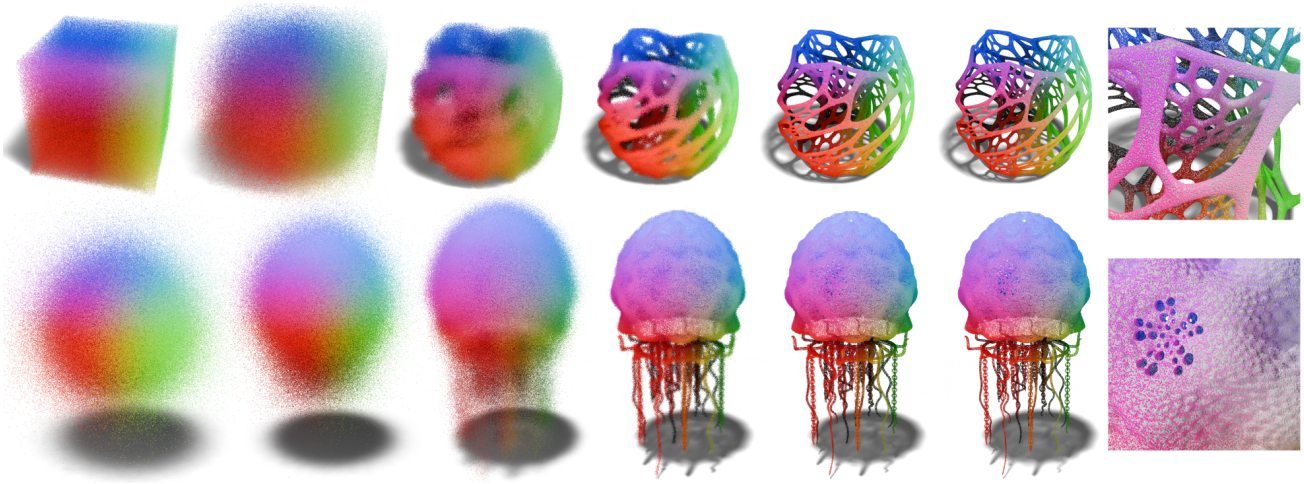


Figure 5. **Inference process** for generating 1M points on a lamp mesh (*top*) and a jellyfish mesh (*bottom*) from uniform and Gaussian distributions, respectively. Results are shown at timesteps $t = 0, 40, 48, 56, 60, 64$, with a close-up of the generated samples at $t = 64$ overlaid on the ground-truth mesh. A complete illustration is available in the accompanying video demo. Both meshes are taken from [60].

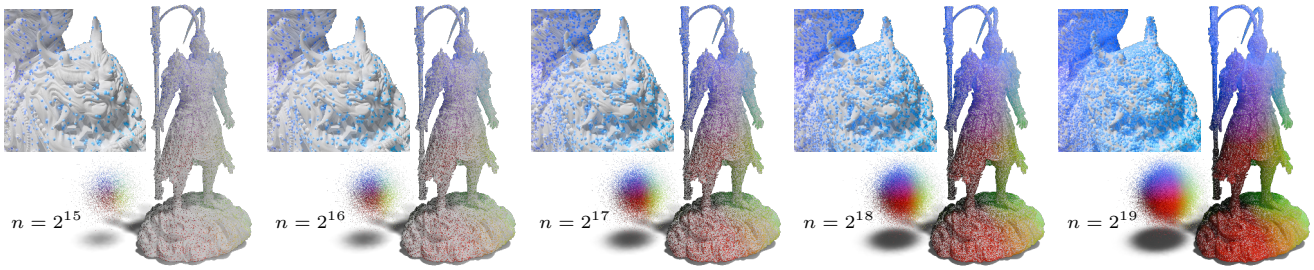


Figure 6. **Forward sampling at different resolutions** on Wukong mesh. For each example, we show the initial Gaussian samples (*bottom left*), the generated samples overlaid on the ground-truth mesh (*right*), and a zoomed-in view (*top left*).

are interested in shape representations. Different from existing work, we propose to adapt diffusion models to learn a mapping from a Gaussian distribution to the target distribution of surface points $\Phi_{\mathcal{M}}$.

Existing networks designed for diffusion models are primarily tailored for regular grids, which have a spatial structure and are high-dimensional. There is no straightforward way to adapt existing designs to our setting, *i.e.*, spatial points without regular spatial structure. A naive idea is to adapt coordinate-based networks (*e.g.*, [31, 32]), but they fail to capture detailed geometric features. For example, Fig. 4 shows the limitations of using standard MLPs and hashing grids to process sampled surface points. Our network design is inspired by [21], where the inputs and outputs of all layers are standardized to have zero mean and unit variance, resulting in improved performance. Another key design choice is to resample the training data for each epoch to simulate an infinite number of surface points, approximating the underlying geometry (see Sec. 3.3).

3.2. Inference process: forward & inverse sampling

The mapping between the Gaussian distribution and the surface points distribution is learned via a diffusion model $D_\theta(\cdot, \cdot)$ parameterized by θ. In the literature of diffusion models, $D_\theta(\cdot, \cdot)$ is often called a denoiser. We first discuss the inference process: θ is known after training, satisfying the ordinary differential equation (ODE):

$$\mathrm{d}\mathbf{x} = \frac{\mathbf{x} - D_\theta(\mathbf{x}, t)}{t}\mathrm{d}t, \tag{1}$$

where $\mathbf{x}$ is the 3D position of some sample.

Solving $\mathbf{x}(t)$ from Eq. (1) over $t \in [0, T]$ gives the *trajectory* of sample $\mathbf{x}$. This trajectory connects the Gaussian distribution and the Geometry distribution: $\mathbf{x}(0) \sim \Phi_{\mathcal{M}}$ and $\mathbf{x}(T) \sim \mathcal{N}(\mathbf{0}, T \cdot \mathbf{1})$ i.e., a Gaussian distribution with variance T, satisfying

$$\lim_{T \to \infty} \frac{\mathbf{x}(T)}{\sqrt{1 + T^2}} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}). \tag{2}$$

We refer to the sampling process from the Gaussian distribution $\mathbf{x}(T)$ to Geometry distribution $\mathbf{x}(0)$ as the *forward sampling* (denoted as $\mathcal{E}$), and the reverse process, from Geometry distribution $\mathbf{x}(0)$ to Gaussian distribution $\mathbf{x}(T)$, as the *inverse sampling* (denoted as $\mathcal{D}$). The forward and inverse sampling follow the same trajectory but in opposite

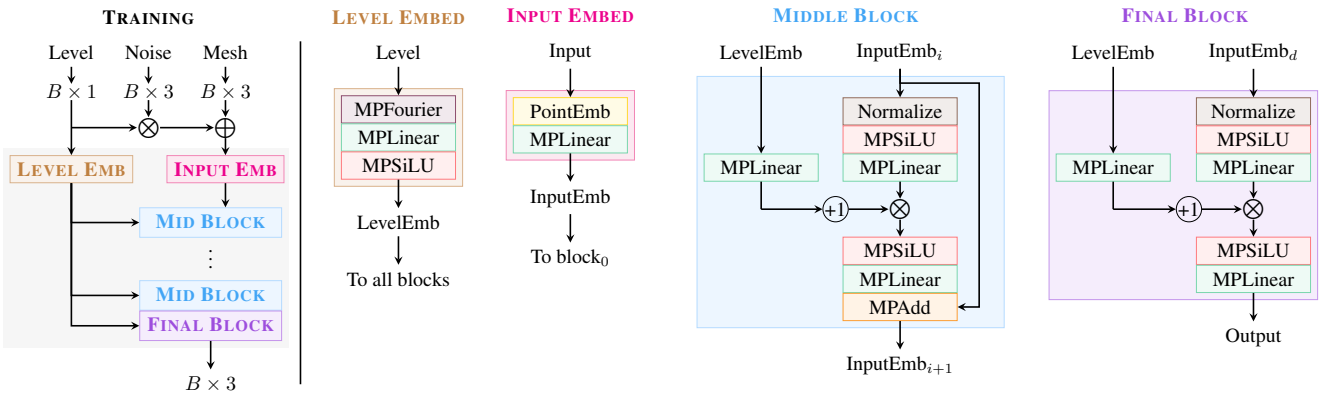


Figure 7. **Network overview**. *Left*: the training process. *Right*: detailed illustration of the modules. The magnitude-preserving (MP) layers are adpated from [21].

directions. In practice we choose discrete timesteps (noise levels) to sample on the trajectory [20], i.e., $T = t_0 > \cdots > t_i > t_{i+1} > \cdots > t_N = 0$, and denote $\mathbf{x}_i := \mathbf{x}(t_i)$.

Forward sampling $\mathcal{E}$. Starting from $\mathbf{x}_0$, a random Gaussian noise, i.e., $\mathbf{x}_0 = \mathbf{x}(t_0) = T\mathbf{n}$ where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, we iteratively compute the following steps for $i = 0, 1, \cdots, N-1$:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i) \cdot \frac{\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i)}{t_i}, \tag{3}$$

which is an Euler solver for the Eq. (1). The endpoint of the trajectory $\mathbf{x}_N$ lie on the target surface $\mathcal{M}$, i.e., $\mathbf{x}_N \sim \Phi_\mathcal{M}$. Eq. (3) has built a mapping from the *standard Gaussian* distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$ to *Geometry* distribution $\Phi_\mathcal{M}$: if we sample an infinite number of samples from the standard Gaussian distribution, the set of endpoints of their trajectories following Eq. (3) would closely approximate the surface $\mathcal{M}$. See Fig. 5 and Fig. 6 for some examples. In practice, we employ a higher-order ODE solver to accelerate the sampling process [20], but for simplicity and clarity, we only show the equations for the simplest case.

Inverse sampling $\mathcal{D}$. Starting from a random surface point $\mathbf{x}_N \in \mathcal{M}$, we reverse the trajectory, i.e., iteratively compute for $i = N, N-1, \cdots, 1$:

$$\mathbf{x}_{i-1} = \mathbf{x}_i + (t_{i-1} - t_i) \cdot \frac{\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i)}{t_i}. \tag{4}$$

The endpoint $\mathbf{x}_0$, after normalization $\mathbf{x}_0 \leftarrow {\mathbf{x}_0}/{\sqrt{1+T^2}}$ lies in the noise space according to Eq. (2). See Algorithm 1 for the full algorithm and Fig. 14 for one example of inverse sampling. In practice, the inversion process starts from $t_N = 0$, which causes the denominator in Eq. (4) to be zero. To avoid numerical issues, we instead set $t_N = 10^{-8}$.

Algorithm 1 Inverse Sampling

procedure INVERSE SAMPLING($\mathbf{x}$, $t_{i\in\{N,\ldots,0\}}$)
  $\mathbf{x}_N = \mathbf{x}$
  for $i \in \{N, N-1, \ldots, 1\}$ **do**
    $\mathbf{d}_i = (\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i))\,/t_i$
    $\mathbf{x}_{i-1} = \mathbf{x}_i + (t_{i-1} - t_i) \cdot \mathbf{d}_i$
  end for
  $\mathbf{n} = \mathbf{x}_0/\sqrt{1 + t_0^2}$
end procedure

3.3. Training process & network design

Given the input geometry $\mathcal{M}$, we first generate the training set by sampling a set of surface points $\{\mathbf{x} \in \mathcal{M}\}$. Following [20], we add noise to the data $\mathbf{y} = \mathbf{x} + \sigma\mathbf{n}$ where σ indicates the noise level, and optimize the denoiser network:

$$\arg\min_\theta \mathbb{E}_{\mathbf{x}\in\mathcal{M}} \mathbb{E}_{\mathbf{n}\sim\mathcal{N}(\mathbf{0},\mathbf{1})} \mathbb{E}_{\sigma>0} \|D_\theta(\mathbf{x} + \sigma\mathbf{n}, \sigma) - \mathbf{x}\|, \tag{5}$$

Our network is simple yet effective: the noise levels σ, standard Gaussian noise $\mathbf{n}$, and input coordinates $\{\mathbf{x}\}$ are projected to high-dimensional space following [54]. See Fig. 7 for full details of our network design and Fig. 8 for an example of the training process.

Recall that our goal is to have the learned geometry distribution to accurately approximate the target surface from an infinite number of Gaussian samples. To simulate this, we require a training dataset with an infinite number of surface points. In practice, we *resample* a set of 2^{25} surface points for training before each epoch. Over 1000 epochs, the network encounters a sufficiently large number of ground-truth surface points. This approach is fundamentally different from typical deep learning applications, where the training set is preprocessed and fixed prior to training. In our setting, however, the training datasets—*i.e.*, surface points—are intentionally varied across epochs.

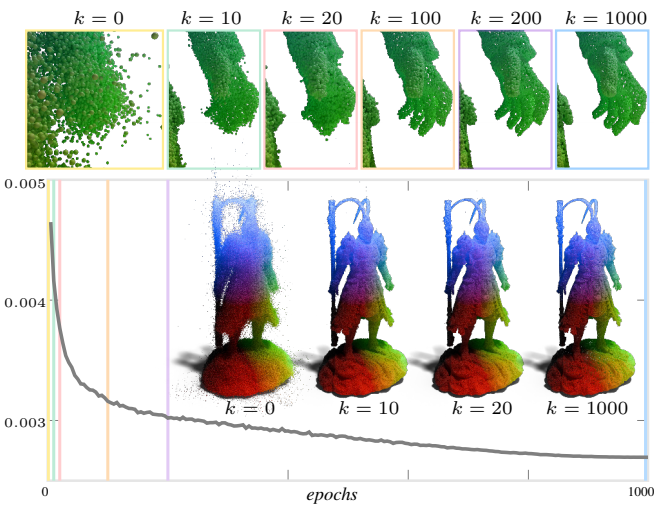


Figure 8. **Training process**. We show the Chamfer distance over epochs and highlight intermediate results (*bottom*). By the 10-th epoch, the network already captures the overall geometry, with finer details further refined in later iterations, as seen in the zoomed-in hand region (*top*).

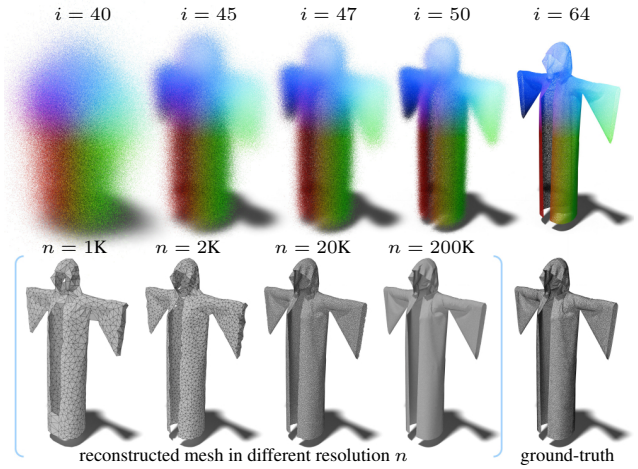


Figure 9. **Application: remeshing**. *Top*: starting from a Gaussian distribution, we show the intermediate steps at different timesteps t. *Bottom*: we use Ball Pivoting to reconstruct a mesh. The number n indicates the number of points used in the reconstruction. Since our method supports infinitely many points sampling, we show results obtained using different number n. The more points we have, the better we can approximate the original surface. The mesh is taken from [23].

4. Experiments

4.1. Implementation

The code is implemented with PyTorch. For most experiments, we use 6 blocks and $C = 512$ for all linear layers, resulting in 5.53 million parameters. One epoch (512 iterations) of training takes approximately 2.5 minutes to complete on 4 A100 GPUs. Training typically requires several hours to achieve reasonably good results. Fig. 8 shows one example of training quality over epochs.

To quantify the accuracy of our approach, we measure the distance between samples from our Geometry distribution, $\mathcal{X}_{\text{gen}}$, and the ground-truth surface $\mathcal{M}$. Specifically, we sample 1 million surface points from $\mathcal{M}$, denoted as $\mathcal{X}_{\text{ref}}$, as the reference set. We then compute the Chamfer distance between the two sets, $\mathcal{X}_{\text{gen}}$ and $\mathcal{X}_{\text{ref}}$, as our metric.

In the following we will investigate multiple applications of our novel shape representations, ablate our design choices and verify the correctness of the inversion.

4.2. Applications

We can generate a varying number of samples from the geometry distribution for surface remeshing at different resolutions. In Fig. 9, we use the Ball Pivoting algorithm [3], implemented in MeshLab [9], with default parameters, to triangulate the samples at different resolutions. Note that this example also illustrates the effectiveness of our method in representing non-watertight surfaces, where most implicit function-based methods would fail.

Geometry distributions can be further extended to incorporate additional information such as color or motion. Fig. 10 shows an example of feeding the texture color in addition to the 3D position of each surface point during training (i.e., the $\mathbf{x}$ in Eq. (5) is 6-dim). Fig. 11 shows an alternative approach: a separate color field network based on hashing grids [31] is trained, allowing the querying of color vectors for all spatial points. See more results of textured geometry distributions in the supplementary materials.

Furthermore, the sampled points can serve as inputs for Gaussian splatting [22], as shown in Fig. 12. Specifically, we sample 1 million points from the distribution to initialize the Gaussian splatting, disabling point gradients and point pruning in the original implementation during training. This optimization assigns colors, radius, and scaling to the points, and can be used for novel view synthesis.

Finally, we show an extension to dynamic geometries (4D objects), achieved by adding a temporal input to the denoiser network D_θ, making the inputs 4D. The trained network encodes the motions of the geometry distributions. See one example in Fig. 13.

4.3. Ablation studies

Using distributions to model a surface shows advantages over vector field-based methods [8, 47, 58] which usually fail to produce uniform sampling: as shown in Fig. 3, even at extremely high resolutions with 1 million samples, their samples fail to adequately cover the target surface. More results can be found in the supplementary materials.

As mentioned earlier, adapting well-established diffusion models from tasks involving regular grid data to our

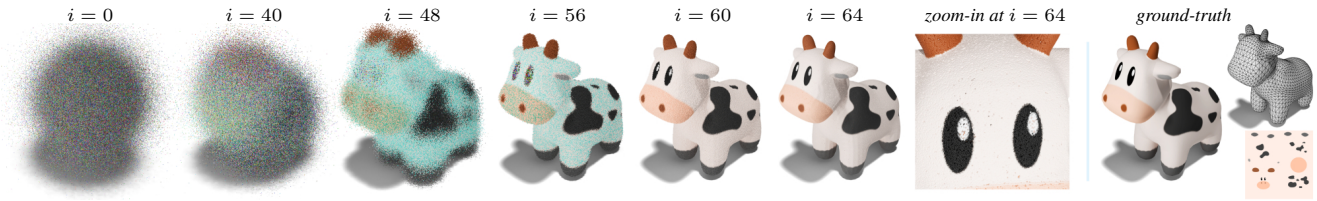


Figure 10. **Application: textured geometry.** The proposed representation can also be used for textured geometry. *Left*: 1 million points with texture (6-dimensional vectors) at different timesteps t. *Right*: the ground-truth geometry and texture.

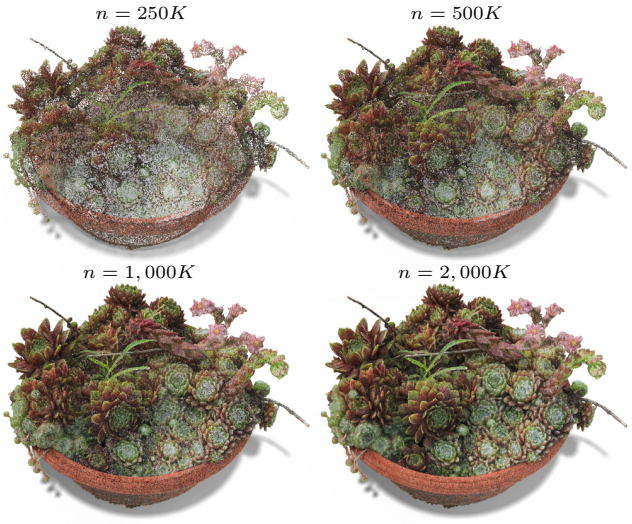


Figure 11. **Application: combination with color field network.** We show results of different numbers of points.

Figure 12. **Application: photo-realistic rendering with Gaussian splatting.** These views are not visible during training.

setting, which focuses on learning geometry distributions, may seem straightforward but proves challenging. We compare our network with two established architectures. The first is hashing grids [31], originally designed for volume rendering with 3-dimensional coordinate inputs. We adapt it to accept 4-dimensional inputs (3 for coordinates and 1 for noise level). The second is a simple MLP network based on DeepSDF [32], where we concatenate point and noise level embeddings as inputs. As shown in Tab. 2a, our proposed network significantly outperforms these straightforward adaptions. While the Chamfer distance for the MLP-baseline appears promising, the qualitative results in Fig. 4 reveal that this baseline fails to capture fine details.

Consistent with observations in other diffusion models, we find that a larger training set, more sampling steps, and

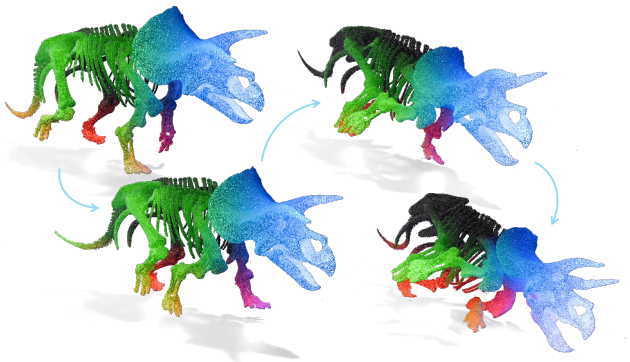


Figure 13. **Application: dynamic object modeling.** We use a single network to learn the motion of the geometry distribution. Only 4 out of 250 frames are shown here.

network arch	*Chamfer* ↓ ($\times 10^3$)
Hashing Grid	4.133
MLP	2.647
Proposed	**2.140**

(a) tested on Loong shape

dataset size	*Chamfer* ↓ ($\times 10^3$)
10^5	4.696
10^6	3.004
10^7	2.936
10^8	2.917
$\approx \infty$	**2.916**

(b) tested on jellyfish shape

# sampling steps	*Chamfer* ↓ ($\times 10^3$)
8	9.803
16	2.814
32	2.782
64	**2.780**

(c) tested on Archimedes shape

# network blocks	*Chamfer* ↓ ($\times 10^3$)
2	3.784
4	3.565
6	3.547
8	3.546
10	**3.544**

(d) tested on lamp shape

Table 2. **Ablation studies** on network architecture, dataset size, sampling steps, and network blocks, tested on shapes from Fig. 1.

deeper networks lead to higher accuracy and improved generation quality. We provide ablation studies in Tab. 2 to validate these findings. Moreover, although both types of distributions work effectively, we observe that the Gaussian distribution performs slightly better than the uniform distribution in most cases, as shown in Tab. 3.

4.4. Inversion

As discussed in Sec. 3.2, our network learns the trajectory connecting the Gaussian distribution and the Geometry dis-

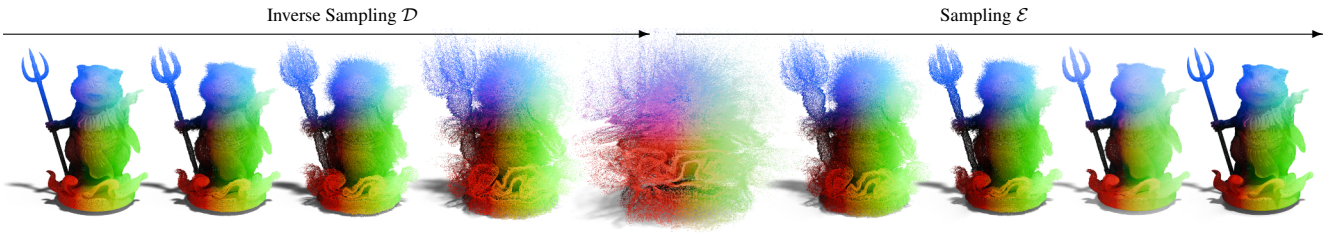


Figure 14. Inverse sampling $\mathcal{D}$ and sampling $\mathcal{E}$ for 1M points. Both inverse sampling and sampling are using $N = 64$ steps. Note that, the image in the middle is the noise space, where it does not look like a Gaussian distribution. This implies that the mapping is not bijective. Some points in the noise space will never be mapped to from the shape space.

object	Wukong	lamp	lion	Parthenon
Uniform	2.845	3.574	3.734	7.621
Gaussian	**2.715**	**3.538**	**3.246**	**7.295**

Table 3. **Ablation study** on using Uniform and Gaussian distributions as initial noise sources. We report Chamfer distance ($\times 10^3$) on different shapes from Fig. 1.

# Inversion Steps	4	8	16	32	64
MSE $\downarrow$	6.88e−1	8.52e−2	6.14e−3	1.84e−4	**1.76e−6**

Table 4. **Mean squared error** of $\|\mathbf{x} - \mathcal{E} \circ \mathcal{D}(\mathbf{x})\|_2^2$ with different inversion steps, evaluated on the mouse shape in Fig. 14.

tribution. The forward and inverse sampling follow this trajectory in opposite directions. In other words, for a surface point-*i.e.*, a sample drawn from Geometry distribution $\mathbf{x} \sim \Phi_{\mathcal{M}}$-the composition of inverse and forward sampling applied to this sample should also follow the Geometry distribution: $\mathcal{E} \circ \mathcal{D}(\mathbf{x}) \sim \Phi_{\mathcal{M}}$. To validate this, we sample 1 million surface points, denoted as $\{\mathbf{x}\}$, and apply inverse sampling, following Eq. (4), to obtain a set of Gaussian noise samples $\{\mathcal{D}(\mathbf{x})\}$. We then apply forward sampling on $\{\mathcal{D}(\mathbf{x})\}$, following Eq. (3), to obtain $\{\mathcal{E} \circ \mathcal{D}(\mathbf{x})\}$. Finally, we evaluate the mean squared error (MSE) between $\{\mathbf{x}\}$ and $\{\mathcal{E} \circ \mathcal{D}(\mathbf{x})\}$, as they are in one-to-one correspondence. In Fig. 14 we show intermediate results from the inverse and forward sampling. We can see that indeed both the initial samples (leftmost) and the results after composition (rightmost) align with the Geometry distribution. Tab. 4 reports the MSE for different choices of inversion steps. In Fig. 15, we apply inverse sampling to the original surface vertices with the ground-truth triangulation and texture coordinates, to demonstrate that our inversion is semantically meaningful. In the supplementary materials we show additional interesting results: we composite inverse sampling and forward sampling from *different* surfaces, yet still obtain expected results. This further demonstrates the validity of our method.

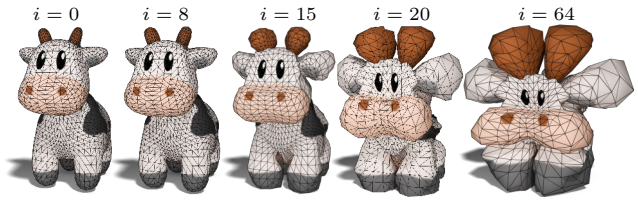


Figure 15. We use the inversion $\mathcal{D}(\cdot)$ to map the spot mesh back to the noise space. Only the original mesh vertices are mapped; textures shown here are for correspondence purposes only.

5. Conclusion

We have introduced a novel geometric data representation that addresses key limitations of traditional methods, such as watertightness and manifold constraints. Our approach models 3D surfaces as geometry distributions encoded in a diffusion model, allowing flexible and precise sampling on complex geometries. This work advances neural 3D representation techniques and establishes a foundation for further exploration and development in geometry modeling, processing, and analysis.

As a first attempt in this field, there are many exciting avenues for future research. We just highlight selected examples but hope that our initial presentation motivates others to explore this shape representation. First, the training of diffusion models builds a trajectory between the Gaussian distribution and the geometry distribution, which can be interpreted as a mapping between two distributions. We propose to investigate how to incorporate regularizers into this mapping during training, such as area/volume preservation or semantic meaningfulness. Second, we are also interested in exploring how to define neural geometry operators on geometry distributions, similar to the well-investigated geometry processing operators on triangle meshes [1, 4]. Third, we have shown some preliminary meshing results in Fig. 9. However, meshing is generally a challenging problem requiring precise algorithms to convert spatial data into graphs (vertices and faces). An interesting avenue of research is to investigate joint sampling and meshing algorithms for the proposed representation.

Acknowledgements

This work was supported by funding from King Abdullah University of Science and Technology (KAUST) — Center of Excellence for Generative AI, under award number 5940.

References

[1] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *arXiv preprint arXiv:2205.02904*, 2022. 8

[2] Saifullahi Aminu Bello, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam, and Jonathan Li. Deep learning on 3d point clouds. *Remote Sensing*, 12(11):1729, 2020. 2

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 6

[4] Mario Botsch. Polygon mesh processing. *AK Peters*, 2010. 2, 8

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[6] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 364–381. Springer, 2020. 3

[7] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 3

[8] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020. 3, 6

[9] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 6

[10] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. On the effectiveness of weight-encoded neural implicit 3d shapes. *arXiv preprint arXiv:2009.09808*, 2020. 3

[11] Yuan Dong, Qi Zuo, Xiaodong Gu, Weihao Yuan, Zhengyi Zhao, Zilong Dong, Liefeng Bo, and Qixing Huang. Gpld3d: Latent diffusion of 3d shape generative models by enforcing geometric and physical priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 56–66, 2024. 3

[12] Venkataram Edavamadathil Sivaram, Tzu-Mao Li, and Ravi Ramamoorthi. Neural geometry fields for meshes. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[13] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018. 2

[14] Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361, 2002. 3

[15] Yanran Guan, Andrei Chubarau, Ruby Rao, and Derek Nowrouzezahrai. Learning neural implicit representations with surface signal parameterizations. *Computers & Graphics*, 114:257–264, 2023. 3

[16] Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *European Conference on Computer Vision*, pages 576–592. Springer, 2022. 3

[17] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020. 2

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[19] 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. 3

[20] 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. 3, 5

[21] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proc. CVPR*, 2024. 4, 5

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 3, 6

[23] Maria Korosteleva and Sung-Hee Lee. Generating datasets of 3d garments with sewing patterns. *arXiv preprint arXiv:2109.05633*, 2021. 6

[24] Marc Levoy and Turner Whitted. The use of points as a display primitive. 1985. 3

[25] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 3

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[27] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 3

[28] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021. 3

[29] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021. 3

[30] 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

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2, 3, 4, 6, 7

[32] 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, 4, 7

[33] Dmitry Petrov, Pradyumn Goyal, Vikas Thamizharasan, Vladimir Kim, Matheus Gadelha, Melinos Averkiou, Siddhartha Chaudhuri, and Evangelos Kalogerakis. Gem3d: Generative medial abstractions for 3d shape synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[34] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000. 3

[35] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4209–4219, 2024. 3

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 3

[37] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner. L3dg: Latent 3d gaussian diffusion. *arXiv preprint arXiv:2410.13530*, 2024. 3

[38] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20875–20886, 2023. 3

[39] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 3

[40] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3

[41] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 3

[42] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469, 2022. 2

[43] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 3

[44] Aoran Xiao, Jiaxing Huang, Dayan Guan, Xiaoqin Zhang, Shijian Lu, and Ling Shao. Unsupervised point cloud representation learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11321–11339, 2023. 2

[45] Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang. Octfusion: Octree-based diffusion models for 3d shape generation. *arXiv preprint arXiv:2408.14732*, 2024. 3

[46] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019. 3

[47] Xianghui Yang, Guosheng Lin, Zhenghao Chen, and Luping Zhou. Neural vector fields: Implicit representation by explicit learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16727–16738, 2023. 3, 6

[48] Lior Yariv, Omri Puny, Oran Gafni, and Yaron Lipman. Mosaic-sdf for 3d generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4630–4639, 2024. 3

[49] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 3

[50] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187*, 2021. 3

[51] LAN Yushi, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. Gaussiananything: Interactive point cloud flow matching for 3d generation. In *The Thirteenth International Conference on Learning Representations*. 3

[52] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022. 3

[53] Biao Zhang and Peter Wonka. Functional diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4723–4732, 2024. 3

[54] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. *Advances in Neural Information Processing Systems*, 35:21871–21885, 2022. 5

[55] 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. 3

[56] Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: Structuring gaussian splatting using optimal transport for 3d generative modeling. *arXiv preprint arXiv:2403.19655*, 2024. 3

[57] 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. *ACM Transactions on Graphics (ToG)*, 42(4):1–13, 2023. 3

[58] Junsheng Zhou, Baorui Ma, Shujuan Li, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Cap-udf: Learning unsigned distance functions progressively from raw point clouds with consistency-aware field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3, 6

[59] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 3

[60] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 4

[61] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3