DOI: 10.1111/cgf.14602
Eurographics Symposium on Geometry Processing 2022
M. Campen and M. Spagnuolo
(Guest Editors)

Volume 41 (2022), Number 5

SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation

X. Zheng[†1], Y. Liu[‡2], P. Wang[2], and X. Tong[2]

[1] Tsinghua University, P. R. China, [2] Microsoft Research Asia

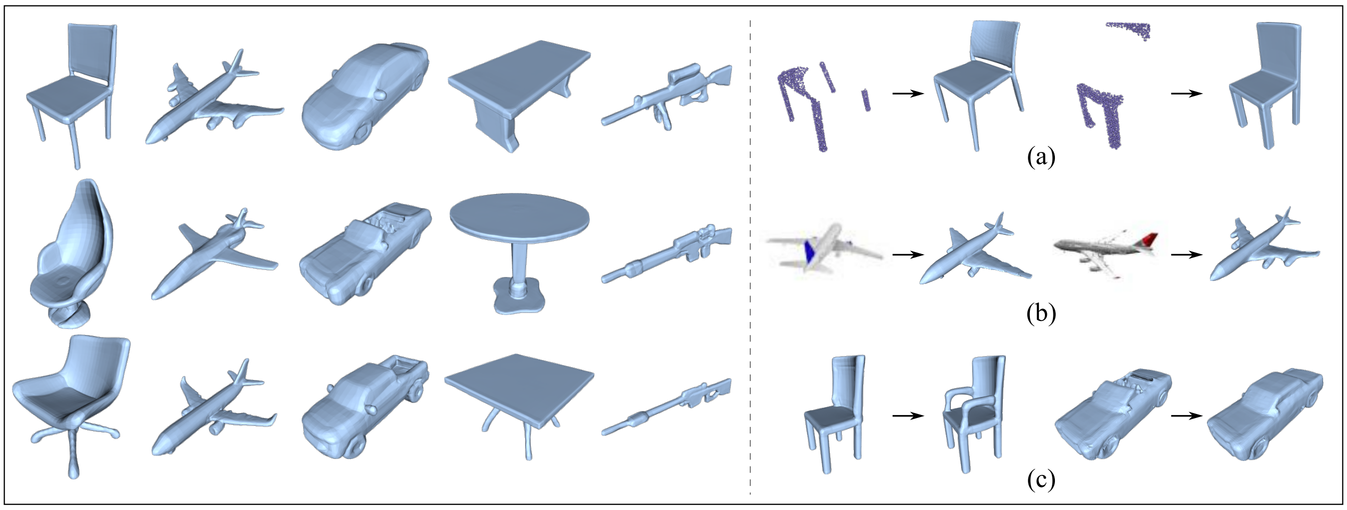


Figure 1: ***Left***: *diverse and visually plausible 3D shapes generated by SDF-StyleGAN.* ***Right***: *the applications enabled by SDF-StyleGAN. (a) Shape completion from partial point cloud inputs. (b) Shape generation from single-view images. (c) Chair arms and car roofs are generated via shape style editing.*

Abstract

We present a StyleGAN2-based deep learning approach for 3D shape generation, called SDF-StyleGAN, *with the aim of reducing visual and geometric dissimilarity between generated shapes and a shape collection. We extend StyleGAN2 to 3D generation and utilize the implicit signed distance function (SDF) as the 3D shape representation, and introduce two novel global and local shape discriminators that distinguish real and fake SDF values and gradients to significantly improve shape geometry and visual quality. We further complement the evaluation metrics of 3D generative models with the shading-image-based Fréchet inception distance (FID) scores to better assess visual quality and shape distribution of the generated shapes. Experiments on shape generation demonstrate the superior performance of SDF-StyleGAN over the state-of-the-art. We further demonstrate the efficacy of SDF-StyleGAN in various tasks based on GAN inversion, including shape reconstruction, shape completion from partial point clouds, single-view image-based shape generation, and shape style editing. Extensive ablation studies justify the efficacy of our framework design. Our code and trained models are available at* `https://github.com/Zhengxinyang/SDF-StyleGAN`.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Shape modeling; Volumetric models; Neural networks;***

1. Introduction

Automatic and controllable generation of high-quality 3D shapes is one of the key tasks of computer graphics, as it is useful to

† Work done during an internship at Microsoft Research Asia
‡ Corresponding author

enrich 3D model collections, simplify the 3D modeling and reconstruction process, and to provide a large amount of diverse 3D training data to improve shape recognition and other 3D vision tasks. Inspired by the success of generative models such as Variational AutoEncoder (VAE) [KW14], Generative Adversarial Networks (GAN) [GPAM*14] in image generation and natural language generation, the 3D shape generation paradigm has been shifted to learning-based generative models in recent years.

Previous learning-based 3D generation works can be differentiated by the chosen 3D shape representations and generative models. Point cloud representation is popular in 3D generation. However, due to its discrete nature, the generative shapes do not have continuous 3D geometry, and their conversion to 3D surfaces is also non-trivial. Mesh representation overcomes this limitation by modeling 3D shapes as a piecewise-linear surface, but the modeling capability is usually restricted by the predefined mesh template and cannot allow the change of mesh topology easily. Volumetric occupancy or signed distance function treats 3D shapes as the isosurfaces of an implicit function, and the recent development of neural implicit representation such as DeepSDF [PFS*19] offers a convenient shape representation for 3D generation. Popular generative models such as VAE, GAN, flow-based generative models [RM15], and autoregressive models [VOKK16] have been incorporated with different 3D representations.

Despite the rapid development of 3D generative models, we observed that there are clear visual and geometric differences, between generative shapes and real shape collection in existing approaches. For instance, shapes generated by many existing works often exhibit bumpy and incomplete geometry, which can be found in Fig. 6 and Fig. 7. In contrast to 3D generation, the gap between real and generative data in the image generation domain becomes much smaller by using more advanced generative models such as StyleGAN [KLA19] and its successors [KLA*20, KAL*21]. We attribute this performance gap in 3D generation to the following three aspects. First, the design of current 3D generators is relatively weaker than the state-of-the-art image generation networks; secondly, the works based on point cloud and mesh representation have limited shape representation power; thirdly, the assessment of generated shape quality that defines the discrimination loss of GAN is not sensitive to visual appearance, which is important to human perception. To this end, in the presented work, we use the StyleGAN2 network [KLA*20] to enhance the 3D shape generator, employ grid-based implicit SDF to improve shape modeling ability, and propose novel SDF discriminators that assess the SDF values and its gradient values from global and local views to improve visual and geometric similarity between the generated shapes and the training data. In particular, the use of SDF gradients greatly enhances surface quality and visual appearance, as SDF gradients determine surface normals that contribute to visual perception via rendering.

We validate the design choice of our 3D generative network — called *SDF-StyleGAN*, through extensive ablation studies, and demonstrate the superiority of *SDF-StyleGAN* to other state-of-the-art 3D shape generative works via qualitative and quantitative comparisons. To measure visual quality and shape distribution of the generated shapes, we propose shading-image-based Fréchet inception distance (FID) scores that complement the evaluation metrics for 3D generative models. We also present various applications enabled by *SDF-StyleGAN*, including shape reconstruction, shape completion from point clouds, single-view image-based shape generation, and shape style editing.

2. Related Work

2.1. 3D generative models

GAN-based models GAN models were first introduced to voxel-based 3D generation by Wu *et al.* [WZX*16] and their training was improved by adapting the Wasserstein distance [SM17, HLXT19]. Achlioptas *et al.* [ADMG18] brought GAN to point cloud generation and also invented l-GAN which first trains an autoencoder (AE) and then trains a GAN on the latent space of AE. Jiang and Marcus [JM*17] used both the low-frequency generator and the high-frequency generator to improve the quality of the generated discrete SDF grids. IMGAN [CZ19] integrated l-GAN and neural implicit representation to achieve better performance. Kleineberg *et al.* [KFW20] experimented with both implicit SDF and point cloud as shape representations in a GAN model, which uses a 3DCNN-based discriminator or a PointNet [QSMG17]-based discriminator. Hui *et al.* [HXX*20] proposed to generate multiresolution point clouds progressively and use adversarial losses in multiresolution to improve point cloud quality. Wen *et al.* [WYT21] also generated points progressively via a dual generator framework. Ibing *et al.* [ILK21] localized l-GAN on grid cells and used the implicit occupancy field as its shape representation. Li *et al.* [LDPT19], Wu *et al.* [WS20] and Luo *et al.* [LLZL21] used 2D differentiable rendering for training 3D GANs without any 3D supervision. The former two works used silhouette images, while Luo *et al.* used a designed spherical map of the extracted mesh. Shu *et al.* [MWYG20] introduced a tree-structured graph convolution network as the generator to improve feature learning. For generating part-controllable shapes, Wang *et al.* [WSH*18] introduced a global-to-local GAN in which the generator also produces segmentation labels and each local discriminator is designed for each segmented part. Shape part information was also utilized to train the network [DXA*19, LNX20] to generate semantically plausible shapes. To remove the requirement of part annotation, the StyleGAN-like architecture was adapted for part-level controllable point cloud generation [GBZCO21, LLHF21]. For better controlling the generated 3D model, Chen *et al.* [CKF*21] proposed a GAN model to refine a low-resolution coarse voxel shape into a high-resolution model with finer details. With the use of volume rendering and neural implicit 3D representations, GAN models are also used to synthesize 3D-aware images such as human faces [OELS*22, CMK*21] and achieve high quality results. In our work, we also adapt the StyleGAN architecture to generate feature vectors on grids that are mapped to implicit SDFs, and use SDF gradients to guide the global and local discriminators to improve shape quality and visual appearance.

Autoencoder-based models AtlasNet [GFK*18] encoded 3D shapes into a latent space and took a collection of parametric surface elements as shape representations for shape generation. Li *et al.* [LXC*17] created a recursive network based on variational autoencoder (VAE) to generate 3D shapes with hierarchical tree structures. Mo *et al.* [MGY*19] encoded more structural relations with hierarchical graphs and devised a Graph-CNN-based VAE for

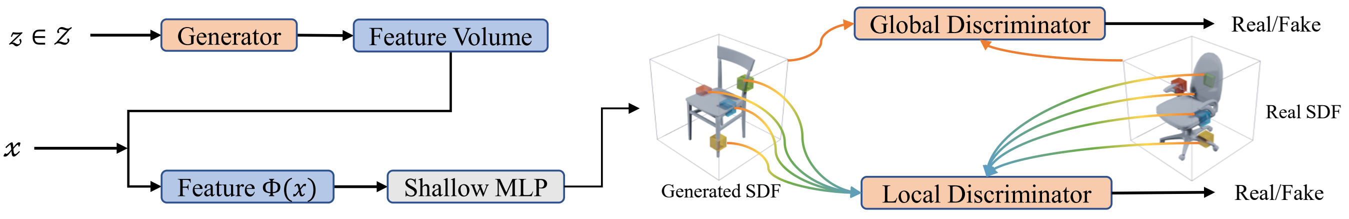


Figure 2: *Overview of SDF-StyleGAN. The original StyleGAN2 generator is extended to 3D, and it outputs the feature volume in a unit box. The feature vector $\phi(\boldsymbol{x})$ at any point $\boldsymbol{x}$ inside the volume is interpolated via trilinear interpolation and is mapped to SDF value via a shallow MLP. The global discriminator takes the SDF values and gradients sampled at the grid centers as input, and the local discriminator takes the SDF values and gradients at a local and random 3D box region near the surface as input. A few local box regions are illustrated on the above chair example, in different colors.*

shape generation. Gao *et al.* [GYW*19] proposed a two-level VAE to learn the geometry of the deformed parts and the shape structure. Jie *et al.* [YML*22] combined the advantages of the above two approaches to learn the disentangled shape structure and geometry. Wu *et al.* [WZX*20] used a Seq2Seq Autoencoder to generate 3D shapes via sequential part assembly. In these works, part information is needed for training. Li *et al.* [LLW22] introduced superquadric primitives in VAE learning to mimic shape parts for easy editing, without real part annotation.

Other generative models By transforming a complex distribution to a simple and predefined distribution, diffusion-based generative models [SE19, HJA20] have been applied to point cloud generation via learning the reverse diffusion process that maps a noise distribution to a 3D point cloud [CYAE*20, LH21, ZDW21]. By explicitly modeling a probability distribution via normalizing flow, flow-based generative models [RM15] are also popular in shape generation, especially point cloud generation [YHH*19, KBV20, KLK*20, PPMNF20, KMU21, PLS*21]. However, these works focus on point clouds and do not generate continuous shapes directly. By implicitly defining a distribution over sequences using the chain rule for conditional probability, autoregressive models achieve their success in image generation [VOKK16]. They were also used in point cloud generation [SWL*20], octree generation [IKK21], polygonal mesh generation [NGEB20], and SDF generation [MCST22]. By explicitly modeling a probability distribution in the form of the EBM [XLZW16], Xie *et al.* [XZG*20, XXZ*21] used an energy-based generative model to synthesize 3D shapes via Markov chain Monte Carlo sampling.

2.2. Evaluation metrics for 3D generation

To measure the similarity between the generated point cloud data set S_g and the referenced point cloud data set S_r, Achlioptas *et al.* [ADMG18] proposed three metrics: Coverage (COV) that is the fraction of S_r covered by S_g; Minimum Matching Distance (MMD) that measures how well the covered shapes in S_r are represented by S_g; and Jensen-Shannon Divergence (JSD) between two probability distributions over S_g and S_r. The distance between two point clouds can be either the Chamfer distance (CD) or the Earth Mover's distance (EMD). Chen *et al.* [CZ19] replaced CD and EMD with light-field-descriptor (LFD) [CTSO03] for better measuring the quality of the generated meshes. Noticing that COV, MMD and JSD do not ensure a fair model comparison, Yang *et al.* [YHH*19] proposed 1-nearest neighbor accuracy (1-NNA) to assess whether two distributions are identical. Ibing *et al.* [ILK21] also adapted a robust statistical criterion — Edge Count Difference (ECD) [CF17] to evaluate 3D generative models.

To mimic the Fréchet Inception Distance (FID) [HRU*17] that is widely used in image generation for assessing image quality, Shu *et al.* [SPK19] proposed the Fréchet Point Cloud Distance (FPD) that computes the 2-Wasserstein distance between real and fake Gaussian measures in the feature space extracted by a pretrained PointNet [QSMG17]. Li *et al.* [LLHF21] replaced PointNet with a stronger pretrained backbone – DGCNN [WSL*19] for FPD computation.

In addition, some works propose task-specific metrics. Wang *et al.* [WSH*18] proposed the 3D inception score, the symmetry score, and the distribution distance to measure the quality of the generated voxelized shapes and generated parts, Mo *et al.* [MWYG20] proposed the HierInsSeg score to measure how well the generated point clouds satisfy the part tree conditions they defined. Human preference through user studies is also used for assessing the quality of shape generation [KLK*20, LLHF21].

Other metrics based on shading images for measuring shape reconstruction quality, such as the mean square error defined over per-pixel keypoint maps [JPXZ20], have potentials to be utilized for evaluating 3D generation. In our work, we enrich the evaluation metrics by introducing Fréchet Inception Distance on shading images, to better assess the visual appearance and data distribution of 3D generative models.

3. Design of SDF-StyleGAN

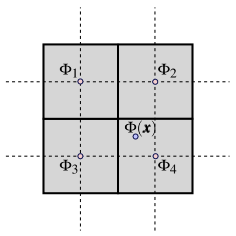
3.1. Overview

We design a StyleGAN2-based 3D shape generation architecture, as illustrated in Fig. 2. We use feature-volume-based implicit signed distance functions as shape representation to maximize implicit representation capability (Section 3.2), leverage and extend the 2D StyleGAN2 generator [KLA*20] to 3D to generate 3D feature volumes (Section 3.3), and propose global and local discriminators to distinguish *fake* SDF fields, including their gradients calculated from global and local regions of the generated field, from the sampled SDF field of the *real* shapes in the training set (Section 3.4). We propose an adaptive training scheme to gradually improve the quality of the generated shapes (Section 3.5).

3.2. Feature-volume-based implicit signed distance functions

We represent 3D shapes as feature-volume-based implicit signed distance functions, in similar spirit to the grid-based implicit occupancy function used in [JSM*20, PNM*20, ILK21].

We assume that a 3D shape $\mathcal{S}$ is generated inside a box $B : [-1,1]^3$ and the box is divided as $m \times m \times m$ grid cells with equal cell size. A dual graph is constructed on these grid cells, and on each graph node c_i, a feature vector $\Phi_i \in \mathbb{R}^l$ is stored. All feature vectors form a feature volume. For any 3D point $\boldsymbol{x} \in B$, we assign a feature vector to it, denoted as $\Phi(\boldsymbol{x})$, via trilinear interpolation in the feature volume, *i.e.*, using the feature vectors of the eight corners of the dual cell where $\boldsymbol{x}$ is located. Due to the construction above, $\Phi(\boldsymbol{x})$ is a continuous function defined in B. A shallow multilayer perceptron (MLP) is used to map $\Phi(\boldsymbol{x})$ to a signed distance value to the shape. In this way, an SDF $f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^3 \longmapsto \mathbb{R}$ is determined by the feature volume and the MLP mapping. The zero isosurface of the SDF defines the shape geometry. In the right inset, we illustrate the concept of feature volume in 2D. The gray grid is a portion of the primal grid cells and the dashed black grid is its dual graph. To calculate the feature at point $\boldsymbol{x}$, we first find the dual cell where $\boldsymbol{x}$ is located and use the feature vectors $\Phi_1, \Phi_2, \Phi_3, \Phi_4$ at the dual cell vertices to perform the interpolation.



In our implementation, $m = 32$, $l = 16$ and the MLP has two hidden layers. The feature volume is produced by the generator, and the MLP parameters are learned via training.

3.3. SDF-StyleGAN generator

We adapt the StyleGAN2 architecture [KLA*20] to our feature volume generator. In the original StyleGAN2 architecture, a latent vector $\boldsymbol{z}$ is first sampled from a high-dimensional normal distribution and mapped to an intermediate latent space $\mathcal{W}$. A learned affine transform and Gaussian noise are injected into a style block, and multiple stacked style locks are used to predict the pixel colors. We applied the following modifications to the original StyleGAN2's network structure. (i). The 2D convolution in each StyleGAN2 block is changed to 3D convolution with corresponding kernel size; (ii). The stacked blocks are upsampled from $4 \times 4 \times 4$ to $m \times m \times m$; (iii). the size of the constant tensor fed to the first style block is set to $256 \times 4 \times 4 \times 4$. The detailed structure of the network and our changes are illustrated in Fig. 3-(a,b). With these modifications, the generator output an $m \times m \times m$ feature volume with l channels.

3.4. SDF-StyleGAN discriminator

We design two discriminators to evaluate the generated SDF on the coarse scale and the fine scale: global discriminator and local discriminator. Noticing that the visual appearance of the zero isosurface is affected by the surface normal, we also propose to take both SDF values and SDF gradients on a grid region as input to discriminators because the SDF gradients at the zero isosurface are exactly shape surface normals, and the use of both zero-order and first-order SDF information on a coarse grid can approximate SDF values on a much denser grid.

3.4.1. Global discriminator

For a generated feature volume, we calculate the SDF values and gradients in a coarse grid of $[-1,1]^3$ with resolution $K_G \times K_G \times K_G$. The SDF values are evaluated directly via the network and SDF gradients are calculated via finite differentiation. These samples form a $4 \times K_G \times K_G \times K_G$ feature grid that can approximate the shape roughly. For a *real* shape in the training set, we first build a discrete SDF field with resolution $128 \times 128 \times 128$, then sample the SDF values on the same coarse grid via trilinear interpolation. The corresponding SDF gradients are approximated via finite differentiation. As SDF gradient vectors should be with unit length in theory, we normalize the gradients before constructing the feature grid. In our implementation, we set $K_G = m = 32$.

We adapt the discriminator architecture of StyleGAN2 to our global discriminator $\mathcal{D}_G$ by replacing the 2D convolution with the corresponding 3D convolution, as illustrated in Fig. 3-(c). The discriminator is used to distinguish between the synthesized shapes and the shapes from the training set.

3.4.2. Local discriminator

The global discriminator focuses on shape quality in a coarse level. Naively increasing K_G will lead to high computational and memory costs due to the use of 3D convolution, and the discrimination of *real* or *fake* SDF away from the zero isosurface has little impact on the quality of shape surface. Instead, we pay more attention to the local region close to the zero isosurface by introducing a local discriminator $\mathcal{D}_L$.

We choose a set of local regions as follows. We first evaluate the SDF values on the cell centers of the coarse grid used in the global discriminator and sort these cell centers in ascending order according to their absolute SDF values. The first N_0 cell centers are selected as the centers of candidate local regions. We then randomly select s cell centers from these candidates and construct a small box with length L_b centered at every selected center. For each small box, we divide it as a $K_L \times K_L \times K_L$ grid, sample the SDF values and calculate gradients at the cell centers of this grid to construct a $4 \times K_L \times K_L \times K_L$ feature grid. The local discriminator takes the feature grid as input, to distinguish whether it is *real* or *fake*.

Note that we did not choose the first s cell centers from the sorted center list, as this greedy selection may result in clustered local regions. Our random selection from a large candidate pool helps to distribute local regions more evenly around the zero isosurface. In our implementation, the indices of N_0 candidates are randomized directly for selection. It is possible to improve this randomization by using the furthest point sampling strategy, but it requires more computational time for training.

The network architecture of the local discriminator is similar to that of the global discriminator. During training, each *real* or *generated* shape provides s local regions to the local discriminator. In our implementation, we set $K_L = 16$, $L_b = 0.25$, $N_0 = 2048$ and $s = 16$ by default. In Fig. 2, we illustrate some local region boxes in the chair examples.

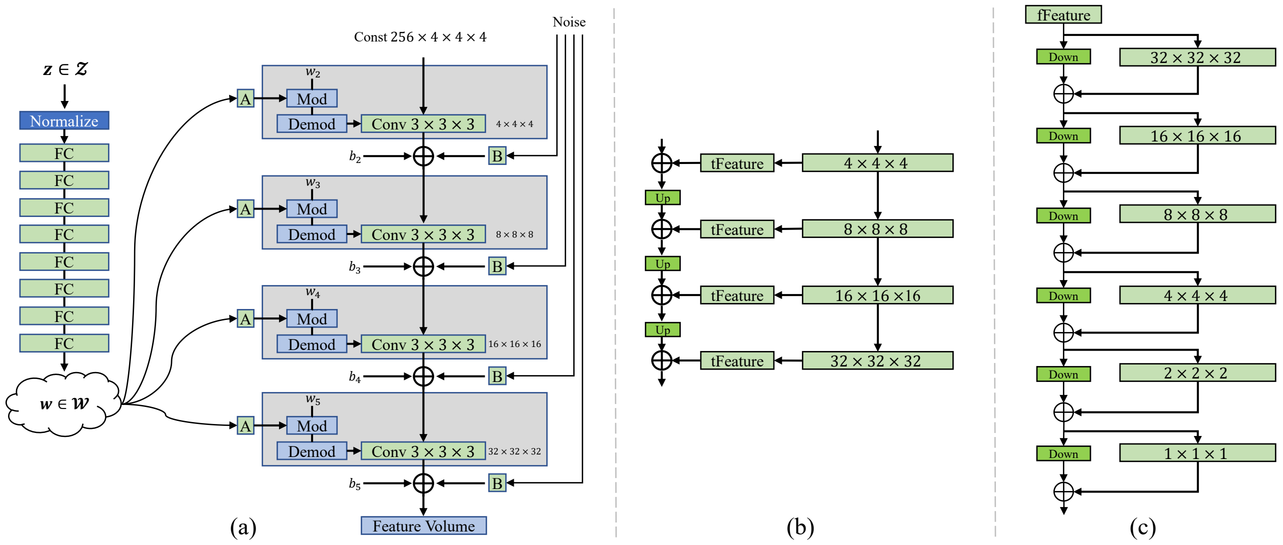


Figure 3: *(a): The revised StyleGAN2 generator for 3D feature volume generation. We used 3D convolution with kernel size 3 and four style blocks corresponding to four-level resolution, up to* $32 \times 32 \times 32$. Mod *and* Demod *are the modulation and demodulation modules adapted from StyleGAN2. (b): The skip input for the generator. (c) The discriminator architecture. The* tFeature *module and the* fFeature *module convert between the feature volume per grid cell and the high dimensional feature to/from 3D convolution.* Up *and* Down *denote the upsampling and downsampling modules. The first block in (c) is removed from the local discriminator as its input feature grid resolution is* $16 \times 16 \times 16$.

3.5. SDF-StyleGAN training

3.5.1. Loss functions

The loss functions of our discriminators and generator are adapted from StyleGAN2. The global discriminator loss $\mathcal{L}^G_{\mathcal{D}}$ has the following form:

$$\mathcal{L}^G_{\mathcal{D}} = \mathbb{E}_{z\sim\mathcal{Z}}[\zeta(\mathcal{D}_G(\boldsymbol{T}_G(\mathcal{S}(z))))] + \mathbb{E}_{\mathcal{S}\sim p_{data}}[\zeta(-\mathcal{D}_G(\boldsymbol{T}_G(\mathcal{S})))] + \mathcal{R}_{\mathcal{S}\sim p_{data}}(\mathcal{S};\Theta_G). \tag{1}$$

Here, $\mathcal{S}(z)$ and $\mathcal{S}$ denote the generated SDF and the SDF of the shape sampled from the training data set respectively. $\mathcal{D}_G(\cdot)$ is the output of the global discriminator, and $\boldsymbol{T}_G(\cdot)$ denotes the feature grid (SDF values & gradients) calculated in the coarse grid. $\zeta(x) = \log(1+e^x)$. $\mathcal{R}_{\mathcal{S}\sim p_{data}}(\mathcal{S};\Theta)$ is the R_1 regularization term adapted from [MGN18], where Θ_G is the network parameters of $\mathcal{D}_G$.

The local discriminator loss $\mathcal{L}^L_{\mathcal{D}}$ is similar to $\mathcal{L}^G_{\mathcal{D}}$.

$$\mathcal{L}^L_{\mathcal{D}} = \mathbb{E}_{z\sim\mathcal{Z}}[\zeta(\mathcal{D}_L(\boldsymbol{T}_L(\mathcal{S}(z))))] + \mathbb{E}_{\mathcal{S}\sim p_{data}}[\zeta(-\mathcal{D}_L(\boldsymbol{T}_L(\mathcal{S})))] + \mathcal{R}_{\mathcal{S}\sim p_{data}}(\mathcal{S};\Theta_L). \tag{2}$$

Here $\boldsymbol{T}_L(\cdot)$ is the feature grid calculated in the local region, and Θ_L is the network parameters of the local discriminator.

The loss function of the generator $\mathcal{L}_{\mathcal{G}}$ is defined as follows.

$$\mathcal{L}_{\mathcal{G}} = \beta\mathcal{L}_{\text{pl}} + \mathbb{E}_{z\sim\mathcal{Z}}[\zeta(-\mathcal{D}_G(\boldsymbol{T}_G(\mathcal{S}(z))))] + \alpha(t)\,\mathbb{E}_{z\sim\mathcal{Z}}[\zeta(-\mathcal{D}_L(\boldsymbol{T}_L(\mathcal{S}(z))))]. \tag{3}$$

Here, $\mathcal{L}_{\text{pl}}$ is the path length regularization term used by StyleGAN2, and $\beta = 2$. $\alpha(t)$ is the weight of the local discriminator and changed during training. We also adopt the EMA scheme [YFW*18] to stabilize the generator training.

3.5.2. Adaptive training scheme

We propose the following adaptive scheme to train SDF-StyleGAN. We first initialize the generator and discriminators with StyleGAN2's weight initialization, then update the global discriminator, the local discriminator, and the generator in sequential order. During the early training stage, we set a small α in generator training so that the global discriminator dominates, and the network focuses on generating rough SDF fields. We gradually increase α so that the local discriminator can improve the geometry details while maintaining the global shape structure. In our implementation, we set $\alpha(t) = \alpha_{\text{min}} + \alpha_{\text{max}} \times t/t_{\text{max}}$, here t is the current epoch number and t_{max} is the maximum epoch number. Our default setting is $\alpha_{\text{min}} = 0, \alpha_{\text{max}} = 8, t_{\text{max}} = 200$.

4. Experiments and Evaluation

Dataset and training We trained our SDF-StyleGAN with the five shape categories selected from ShapeNet Core V1 [CFG*15]: chair, table, airplane, car, and rifle, individually. We use the same data split of [CZ19]: 70 % data as the training set, 20 % data as the test set, and 10 % data as the validation set which is not used in our approach. For each shape with triangle mesh format in the training set, we normalize it into a $[-0.8, 0.8]^3$ box, and use the SDF computation algorithm of [XB14] to compute the discrete SDF field with resolution 128^3 in $[-1,1]^3$. This algorithm can remove nested interior mesh facets, handle non-watertight meshes and meshes with inconsistently oriented normals robustly. During training, we also ensure that the centers of the selected local regions are contained in $[-1+L_b/2, 1-L_b/2]^3$, so that all the selected local regions are strictly within $[-1,1]^3$. We conducted our experiments on a Linux server with Intel Xeon Platinum 8168 CPU (2.7 GHz) and 8 Tesla V100 GPUs (16 GB memory). The default batch size is 32. The training time (200 epochs) takes about 2 days on average. Fig. 1 illustrates a few plausible shapes generated by our approach.

Competitive methods We select the following representative works that generate continuous 3D shapes for comparison: IMGAN [CZ19] and Implicit-Grid [ILK21] that use implicit occupancy fields to represent 3D shapes, ShapeGAN [KFW20] that uses implicit SDFs to represent 3D shapes. We reused the released checkpoints of IMGAN and Implicit-Grid for evaluation. More specifically, IMGAN [CZ19]'s chair and table categories were trained on 64^3 resolution, while airplane, car and rifle categories were trained on 128^3 resolution. For Implicit Grid [ILK21], we used the checkpoints provided by the authors, trained on 256^3 resolution. As the work of [KFW20] released the pre-trained networks in three ShapeNetCore V2 shape categories only, for a fair comparison, we retrain ShapeGAN on our processed data with the same training strategy and parameters provided by the authors.

In the following subsections, we first present the evaluation metrics for 3D shape generation in Section 4.1, then provide a quantitative and qualitative evaluation of our method and the competitive methods in Section 4.2. Ablation studies on the design of SDF-StyleGAN are presented in Section 4.3.

4.1. Evaluation metrics

As briefly reviewed in Section 2.2, proper evaluation criteria for 3D generative models are still in active development. The shortcomings of commonly used COV and MMD metrics [ADMG18] were verified by [YHH*19, ILK21], and replaced by 1-NNA [YHH*19] or ECD [ILK21]. We follow their suggestions and use 1-NNA and ECD as two of our evaluation metrics and also report COV and MMD as a reference. The definitions of these metrics are at below.

Let $S_g \sim \mathbb{P}_g$ be the set of generated samples, $S_r \sim \mathbb{P}_r$ be the set of reference data, and $D(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{\geq 0}$ be the distance function.

COV [ADMG18] For any $X \in S_g$, its nearest neighbor $Y \in S_r$ is marked as a *match*, and COV measures the fraction of $Y \in S_r$ matched to any element in S_g:

$$\mathrm{COV}(S_g, S_r) = \frac{|\{\mathrm{argmin}_{Y \in S_r} D(X,Y) \mid \forall X \in S_g\}|}{|S_r|}. \tag{4}$$

MMD [ADMG18] MMD measures the average distance from any $Y \in S_r$ to its nearest neighbor $X \in S_g$:

$$\mathrm{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X,Y). \tag{5}$$

1-NNA [YHH*19] Let $S_{-X} = S_r \cup S_g - \{X\}$ and N_X be the nearest neighbor to X in S_{-X}. 1-NNA is the leave-one-out accuracy of the 1-NN classifier:

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{I}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{I}[N_Y \in S_r]}{|S_g| + |S_r|}, \tag{6}$$

where $\mathbb{I}[\cdot]$ is the indicator function.

ECD [ILK21] A k-minimum spanning tree of the neighborhood graph of $S_r \cup S_g$ is built. Three types of tree edges are defined: connected within S_r, connected within S_g, connected between S_r and S_g. ECD is the weighted difference between the number of these edges and the edge count if S_r and S_g are from the same distribution. The exact formula of ECD can be found in the Appendix of [ILK21].

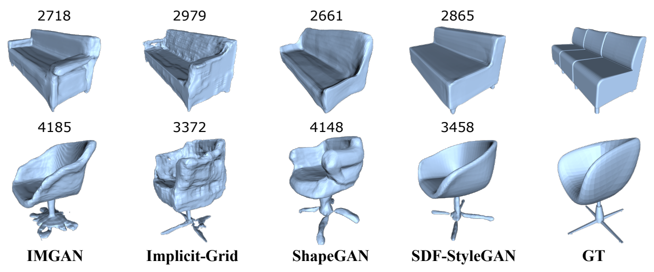


Figure 4: *Illustration of the drawback of LFD. The number above the shape is the LFD between the shape and its GT counterpart.*

For COV, MMD, LFD and ECD computations, we follow [CZ19, ILK21] to use mesh-based light-field-distance (LFD) as the distance function D, as our method and all the compared methods can extract mesh surfaces for evaluation.

Drawback of LFD Although the above metrics based on LFD is recommended by previous 3D generation works [CZ19, ILK21], a smaller LFD between shape A and shape B does not mean that their visual similarity is better than that of shape A and another shape C with a larger LFD, as LFD is based on silhouette images without considering the fidelity of the local shape geometry. We illustrate this drawback in Fig. 4 as follows. The shape in the rightmost column is sampled from ShapeNet, denoted by GT. We select the shape with the smallest LFD value to the GT shape for each method. The numbers above these shapes are the LFD values. We can see that some shapes with bumpy geometry have smaller LFDs although they have a large visual difference from their GT counterparts. This drawback indicates that the LFD-based evaluation metrics for 3D shape generative networks are not sufficient to measure the visual and geometry quality of generated shapes.

Shading-image-based FID To resolve the drawback of using LFD and take human perception into consideration, we propose to adapt Fréchet inception distance (FID) [HRU*17] on the shading images of 3D shapes, as the visual quality of 3D shapes for humans is mostly perceived from rendered view images. We name this metric by shading-image-based FID. We normalize the surface mesh into a unit sphere and render shading images from 20 uniformly distributed views, here the view selection is the same as the LFD algorithm [CTSO03] (see the illustration in Fig. 5). For the generated data set and the training set, we compute the Fréchet inception distance (FID) score based on their i-th view images, and average 20 FID scores to define the shading-image-based FID:

$$FID = \frac{1}{20}\left[\sum_{i=1}^{20} \|\mu_g^i - \mu_r^i\|^2 + \mathrm{Tr}\left(\Sigma_g^i + \Sigma_r^i - 2\left(\Sigma_r^i \Sigma_g^i\right)^{1/2}\right)\right], \tag{7}$$

where g and r denote the features of the generated data set and the training set, $\mu_\cdot^i$, $\Sigma_\cdot^i$ denote the mean and the covariance matrix of the corresponding shading images rendered from the i-th view.

Here, although the rendered images of 3D shapes are different from natural images in ImageNet [DDS*09] trained for Inception-V3 [SVI*16], we found that the shading-image-based FID is mean-

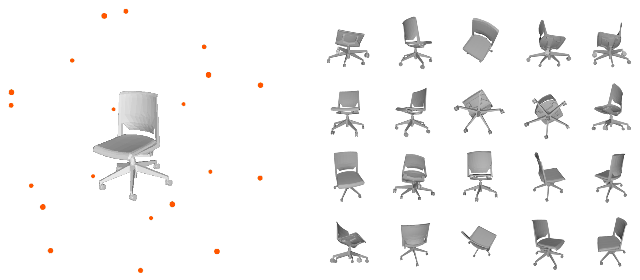


Figure 5: *Left: configuration of rendering views. 20 camera positions are illustrated as red points. Right: 20 rendered images for computing FID, the image resolution is* 299×299.

ingful, and the generated data set with a smaller FID has more plausible and visually similar shapes to the training set.

FPD [SPK19] We also adapt Fréchet point cloud distance (FPD) proposed by [SPK19] for evaluation. For any shape in S_g or S_r, we sample 2048 points on the mesh surface and pass them to a DGCNN backbone network [WSL*19, LLHF21] pre-trained with the shape classification task. The output feature vectors of S_g and S_r are used to compute FPD. Small FPD values are better.

Evaluation setup For computing COV and MMD, we follow the setting of [CZ19]: S_r is formed by the original ShapeNet meshes in the test dataset, $5|S_r|$ shapes are generated as S_g. For computing 1-NNA and ECD, S_r is not changed and $|S_r|$ shapes are sub-sampled from S_g as S'_g. 1-NNA and ECD are evaluated 10 times between S'_g and S_r, and their average numbers are reported. We note that taking the training set as S_r is more reasonable to evaluate the capability of GANs since the distribution of the test set could be very different with the training set and the number of shapes in the test set is usually small. However, here we still follow the setup of the original papers for consistency.

For computing shading-image-based FID and FPD, S_r is formed by the meshes in the training set and $|S_r|$ shapes are generated as S_g. We employ the clean-fid algorithm [PZZ21] to calculate FID. The default iso-values of the competitive methods are employed to extract surface meshes by the Marching Cube algorithm [LC87] on the 128^3 grids.

4.2. Quantitative and qualitative evaluation

Quantitative evaluation We evaluate SDF-StyleGAN, IMGAN, ShapeGAN and Implicit-Grid by using the metrics listed in Section 4.1. Quantitative results are provided in Table 1. LFD-based COV and MMD are listed for reference only as mentioned above. In the categories of airplane and table, IMGAN has better 1-NNA and ECD values. In the categories of chair and rifle, Implicit-Grid has better performance in 1-NNA and ECD. Our SDF-StyleGAN achieves the smallest 1-NNA and ECD in car shapes. However, the 1-NNA and ECD metrics do not faithfully reflect how the distribution of the generated set is similar to that of the training set, as the reference set is the test set. In terms of FID and FPD that use the training set as the reference set, our SDF-StyleGAN is significantly better than other methods, while IMGAN is the second best.

Qualitative evaluation The superiority of SDF-StyleGAN in terms

Table 1: *Quantitative evaluation of different methods. COV, MMD, 1-NNA and ECD use Light-field-distance and their reference set S_r is the test set of shape categories. FID is the abbreviation of shading-image-based FID.*

Data	Method	COV(%) ↑	MMD ↓	1-NNA↓	ECD↓	FID↓	FPD↓
Chair	IMGAN	72.57	**3326**	0.7042	1998	63.42	1.093
	Implicit-Grid	**82.23**	3447	**0.6655**	**1231**	119.5	1.456
	ShapeGAN	65.19	3726	0.7896	4171	126.7	1.177
	SDF-StyleGAN	75.07	3465	0.6690	1394	**36.48**	**1.040**
Airplane	IMGAN	76.89	**4557**	**0.7932**	**2222**	74.57	1.207
	Implicit-Grid	**81.71**	5504	0.8509	4254	145.4	2.341
	ShapeGAN	60.94	5306	0.8807	6769	162.4	2.235
	SDF-StyleGAN	74.17	4989	0.8430	3438	**65.77**	**0.942**
Car	IMGAN	54.13	2543	0.8970	12675	141.2	1.391
	Implicit-Grid	**75.13**	2549	0.8637	8670	209.3	1.416
	ShapeGAN	57.40	2625	0.9168	14400	225.2	0.787
	SDF-StyleGAN	73.60	**2517**	**0.8438**	**6653**	**97.99**	**0.767**
Table	IMGAN	83.43	**3012**	**0.6236**	**907**	51.70	1.022
	Implicit-Grid	**85.66**	3082	0.6318	1089	87.69	1.516
	ShapeGAN	76.26	3236	0.7069	1913	103.1	**0.934**
	SDF-StyleGAN	69.80	3119	0.6692	1729	**39.03**	1.061
Rifle	IMGAN	71.16	**5834**	0.6911	701	103.3	2.102
	Implicit-Grid	77.89	5921	**0.6648**	**357**	125.4	1.904
	ShapeGAN	46.74	6450	0.8446	3115	182.3	1.249
	SDF-StyleGAN	**80.63**	6091	0.7180	510	**64.86**	**0.978**

of FID and FPD metrics can be perceived directly through visual comparison. In Fig. 6, a set of chairs generated by each method is illustrated for comparison. For IMGAN, their chairs have plausible structures, but with bumpy geometry, probably because IMGAN learned the voxelization artifacts in constructing the occupancy field for the training data. For Implicit-Grid, its results contain many missing regions and are also bumpy. The visual quality of its results is the worst, although it has the best 1-NNA and ECD values among all methods. The ShapeGAN results do not have voxelization-like bump geometry, but the shapes are distorted compared to the training data. Our SDF-StyleGAN generates more geometry plausible chair: the seats are flat, the local geometry is smooth, and the overall structure is more complete. This significant improvement in geometry and visual quality by our method is correctly reflected by the FID values in Table 1. In Fig. 7, we also show the generated shapes in other shape categories, and a similar conclusion remains. In our supplemental material, we provide more randomly generated results (240 shapes for each category) without any cherry picking. These results further validate the capability and superiority of our method.

4.3. Ablation studies

We conducted a series of ablation studies to validate our framework design, using the chair category from the ShapeNet dataset.

Efficacy of local discriminator and SDF gradients Five alternative configurations on using the global discriminator $\mathcal{D}_G$ and the local discriminator $\mathcal{D}_L$ were tested:

(1) use $\mathcal{D}_G$ only, and use the SDF values as input;
(2) use $\mathcal{D}_G$ only, and use both SDF values and gradients as input;
(3) add $\mathcal{D}_L$ to (2), and use SDF values as input to $\mathcal{D}_L$;
(4) add $\mathcal{D}_L$ to (2), and use SDF gradients as input to $\mathcal{D}_L$;
(5) our default configuration that uses both $\mathcal{D}_G$ and $\mathcal{D}_L$ and feeds both SDF and SDF gradients to $\mathcal{D}_G$ and $\mathcal{D}_L$.

Table 2 shows the evaluation metrics of SDF-StyleGAN trained

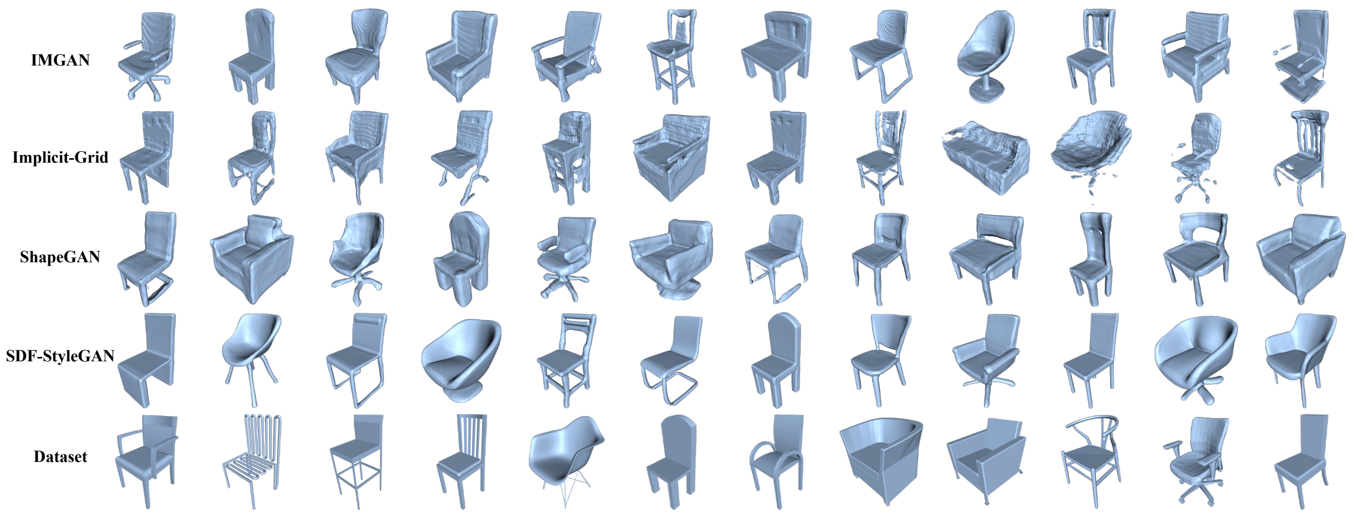


Figure 6: *Visual comparison of randomly generated chairs by different methods. The shapes in the last row are randomly sampled from the training dataset.*

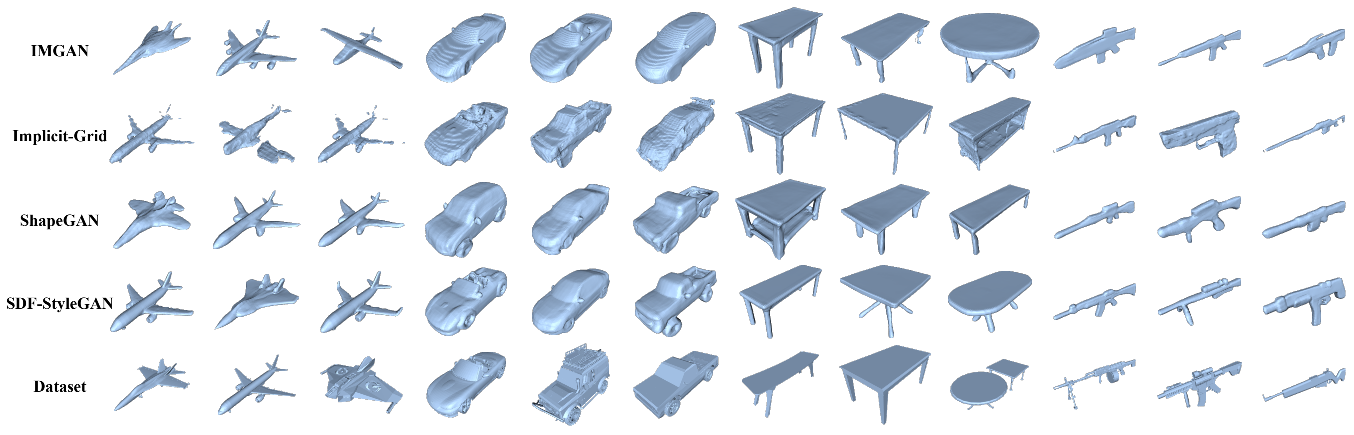


Figure 7: *Visual comparison of randomly generated shapes by different methods on airplane, car, table and rifle. The shapes in the last row are randomly sampled from the dataset.*

with these configurations. By comparing (3), (4) and (5) with (1) and (2), we can see that the addition of $\mathcal{D}_L$ increases the performance significantly; by comparing (2) with (1), and (4) with (3), we can find that the use of SDF gradients increases performance by a large margin. In Fig. 8, we visualize some shapes generated by the networks corresponding to these five configurations. We can clearly see that networks with the local discriminator produce more smooth and regular shapes, and the use of SDF gradients effectively reduces geometry distortion.

Candidate local region number N_0 Local region selection is important to our training. A smaller N_0 does not help to distribute the selected local regions more evenly, while a much larger N_0 cannot ensure that the selected local regions are around the zero isosurface. We tested three choices of N_0: 512, 2048, 8912, and found that the default value 2048 can lead to better performance, as shown in Table 3.

Feature-volume-based implicit SDF Our method benefits from feature-volume-based implicit SDF. We did an ablation study by replacing it with discrete SDFs, *i.e.*, letting the generator directly output an $N \times N \times N$ SDF grid. We use the global discriminator only for this test. We tested two kinds of resolutions: $N = 32$ and $N = 64$, and we also tested whether the additional SDF gradient input can help improve these networks. Table 4 shows the performance of these alternative networks, where the mesh extraction by the

Table 2: *Ablation study on the use of local discriminator and SDF gradients. G and L denotes the global discriminator and the local discriminator, respectively.* 0 *and* 1 *denotes the use of SDF values and the use of SDF gradients, respectively. The combinations of* G, L *and* $0, 1$ *form the five configurations.*

Config.	G0	G1	L0	L1	1-NNA↓	ECD↓	FID↓	FPD↓
(1)	✓				0.8339	6280	173.3	1.335
(2)	✓	✓			0.7642	3387	85.46	1.198
(3)	✓	✓	✓		0.7124	2436	92.35	1.111
(4)	✓	✓		✓	0.6708	1541	39.17	1.123
(5)	✓	✓	✓	✓	**0.6690**	**1394**	**36.48**	**1.040**

Table 3: *Ablation study on the number of candidate local regions.*

N_0	1-NNA↓	ECD↓	FID↓	FPD↓
512	0.8875	4249	85.90	2.597
2048	**0.6690**	**1394**	**36.48**	**1.040**
8192	0.6881	1870	45.37	1.108

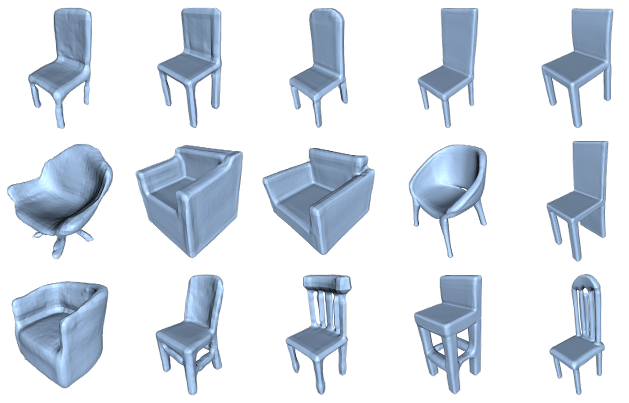


Figure 8: *Visual illustration of the generated shapes by SDF-StyleGAN trained with different configurations listed in Table 2. In the first row we intentionally pick a four-leg chair from each configuration for visual comparison. The shapes in the other rows are randomly selected.*

Marching Cube algorithm uses the same resolution of the SDF grid. We found that the use of SDF gradients can significantly improve the performance of these alternate networks, but there is still a large performance gap between them and our default design. Fig. 9 illustrates the shapes generated by these alternative networks.

5. Applications

Based on the GAN inversion technique, we employed the trained SDF-StyleGAN generator for a series of applications.

3D GAN inversion The goal of 3D GAN inversion is to embed a shape into the latent space of GAN. To this end, we first use an encoder network $\mathcal{E}_\varphi$ to encode the input sample $\boldsymbol{x}$ to a latent code $\boldsymbol{z} \in \mathcal{Z}$ or $\boldsymbol{w} \in \mathcal{W}$, then feed the resulting code to our trained SDF-StyleGAN model $\mathcal{F}_\vartheta$, and optimize the parameters φ with the following loss function:

$$\mathcal{L}_\varphi = \frac{1}{|\mathcal{V}|} \sum_{\boldsymbol{p}\in\mathcal{V}} \left|\mathcal{F}_\vartheta\left(\mathcal{E}_\varphi(\boldsymbol{x});\boldsymbol{p}\right) - \mathtt{SDF}_{gt}(\boldsymbol{p})\right|, \tag{8}$$

where $\mathcal{V}$ is a set of points sampled inside a volumetric space that contains $\boldsymbol{x}$, $\mathtt{SDF}_{gt}$ queries the ground-truth SDF values of any point with respect to $\boldsymbol{x}$, and $\mathcal{F}_\vartheta(\boldsymbol{y};\boldsymbol{p})$ returns the predicted SDF value at point $\boldsymbol{p}$ for a given latent code $\boldsymbol{y}$. After training, we can directly map $\boldsymbol{x}$ to a latent code $\boldsymbol{y}$ via network forwarding. The choice of encoders is flexible and depends on the input type of $\boldsymbol{x}$.

Shape reconstruction from point clouds We use SDF-StyleGAN to reconstruct category-wise shapes from point clouds. We first train the encoder of 3D GAN inversion on a shape category. We choose a light version of octree-based CNN [WLG*17], which is composed of 3 sparse convolution layers and 2 fully connected layers. We optimize φ with an AdamW optimizer [LH19] for 200 epochs on a shape collection while keeping ϑ fixed. We use the 3D GAN inversion to map an input point cloud to an SDF-StyleGAN latent code via the trained encoder. The latent code can be further optimized by minimizing the following loss function which forces the SDF values at $\boldsymbol{p}$ to be zero:

$$\mathcal{L}_{\boldsymbol{y}} = \frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p}\in\mathcal{P}} \left|\mathcal{F}_\vartheta(\boldsymbol{y};\boldsymbol{p})\right|, \tag{9}$$

where $\mathcal{P}$ is a set of points sampled on the surface of $\boldsymbol{x}$. In our experiment, we optimize the code obtained from the encoder with 1000 iterations for each input point cloud $\boldsymbol{x}$.

Fig. 10 shows some reconstructed shapes using our method, as well as other state-of-the-art methods designed for surface reconstruction, including Screen Poisson reconstruction (SPR) [KH13], and three learning-based methods: ConvOcc [PNM*20], DeepMLS [LGP*21], and DualOGNN [WLT22]). Each input noisy point cloud contains 3000 points, where the noisy level is the same as the noisy data used in ConvOcc, DeepMLS and DualOGNN. As our latent code is constrained by the SDF-StyleGAN space, the geometry of the reconstructed shapes respects the data distribution of the training set of SDF-StyleGAN. We can find that our approach is robust to noise, but has limitations in fitting unusual shape details, such as the bumpy surface of the backrest (Fig. 10-(d)), the additional backrest (Fig. 10-(e)), and the cabin of the fighter plane (Fig. 10-(f)).

Table 4: *Performance of an alternative network design that generates discrete SDF grids directly.*

resolution	SDF gradient	FID↓	FPD↓
32^3	✗	156.0	2.097
32^3	✓	104.6	1.532
64^3	✗	154.2	1.323
64^3	✓	80.26	1.792

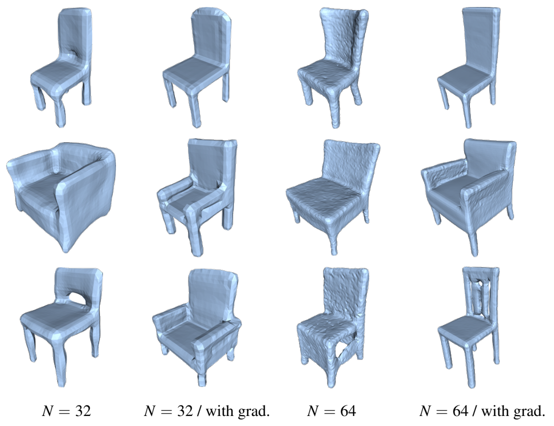


Figure 9: *Illustration of the randomly generated shapes by the alternative design of SDF-StyleGAN that generates discrete SDF grids directly. "with grad." means the SDF gradients also be fed into the discriminator.*

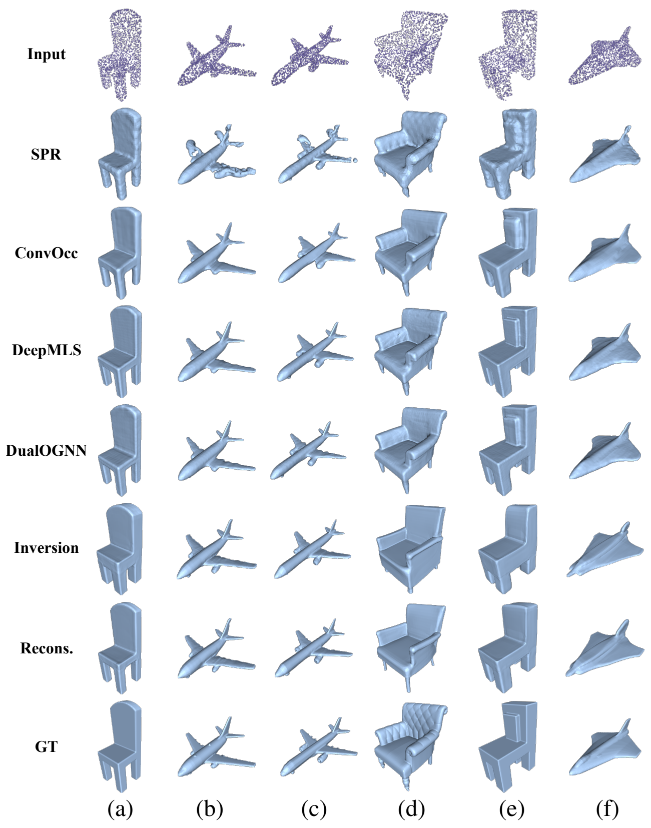


Figure 10: *Shape reconstruction. Inversion and Recons. denote the reconstructed shapes via our GAN inversion and our reconstruction, respectively. The shapes in GT row are the ground-truth shapes. Here the point normals requested by SPR [KH13] are estimated from 10 nearest points.*

Shape completion It is easy to adapt the 3D GAN inversion for the shape completion task. In this experiment, we randomly drop 75% of the input point cloud $\boldsymbol{x}$ to train the encoder while keeping the loss functions unchanged. The trained encoder maps the incomplete input to a latent code that always corresponds to a plausible shape due to the GAN training. We can further optimize the latent code via minimizing Eq. (9). In Fig. 11, we test our method on the categories of chairs and airplanes and compare it with a state-of-the-art point completion cloud approach PoinTr [YRW*21], which is based on a transformer architecture. We can see that within the shape space constrained by SDF-StyleGAN, our method has a better ability to reconstruct plausible shapes, even from a partial chair back (see Fig. 11-(a)). However, similar to the behavior on shape reconstruction, our method has worse capability in fitting fine details, such as the round joint of chair legs (see Fig. 11-(c)) and the wingtip of the airplane (see Fig. 11-(f)); while PoinTr keeps the original points to maintain the input geometry.

Shape generation from single images It is also easy to adapt the GAN inversion to predict a 3D shape from a single image input. In this experiment, the input sample $\boldsymbol{x}$ is an image, and we use a ResNet-18 network [HZRS16] to encode the input. We trained the encoder network on the airplane category, where the low-resolution training images are from [CXG*16]. Fig. 12 shows some results generated

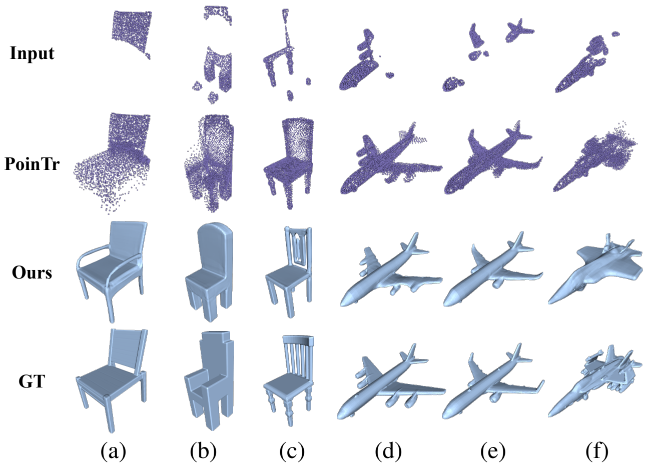


Figure 11: *Shape completion. The shapes in GT row are the ground-truth shapes of the input point clouds. Our method can recover plausible shapes from incomplete inputs.*

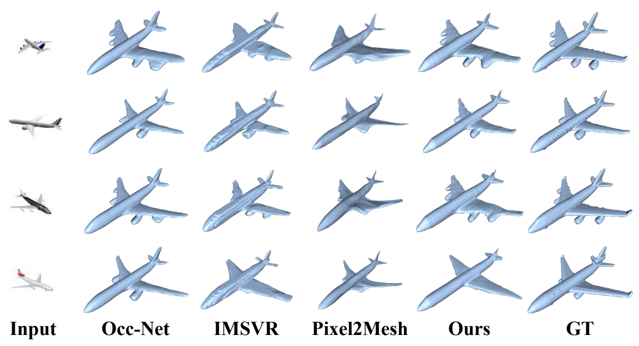


Figure 12: *Shape generation from single images. The shapes in GT column are the ground-truth shapes.*

by our method, as well as some state-of-the-art methods including IMSVR [CZ19], Pixel2Mesh [WZL*18], Occ-Net [MON*19]. Our method is able to predict plausible shapes from low-resolution images. The result in the last row shows a failure case that our method does not generate the airplane engine, as the encoder does not map the input to a good latent code.

Shape style editing To edit the attributes of facial images via GAN model, Shen *et al*. [SYTZ20] assumed that for any binary semantics, there may exist a hyperplane in the latent space serving as the separation boundary. This hyperplane in the latent space can be used for editing the image style. We adapted their approach to our SDF-StyleGAN space for 3D shape style editing. We use the shape labels provided in [MKC18] to divide the chair category into two groups: chairs with and without arms. A DGCNN network [WSL*19] was trained to infer whether an input chair contains arms or not. We then randomly generated 10000 chairs using SDF-StyleGAN and divided them into two groups using the trained classification network. As the latent codes of these generate shapes are known, the hyperplane $\boldsymbol{n} \cdot \boldsymbol{x} + d = 0$ that separates these two groups can easily be computed. For any generated shape or a shape obtained by GAN inversion, we can change its latent code $\boldsymbol{y}$ to $\boldsymbol{y} - \eta(\boldsymbol{n} \cdot \boldsymbol{y} + d)\boldsymbol{n}$, $\eta \geq 0$, to

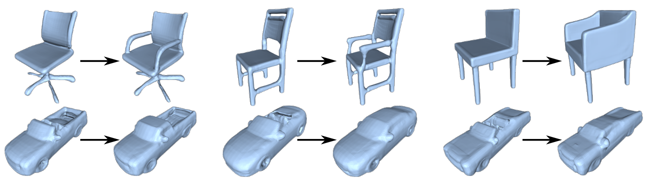


Figure 13: *Shape style editing based on the algorithm of [SYTZ20]. The style editing on chairs is to add arms, and the style editing on cars is to add car roofs.*

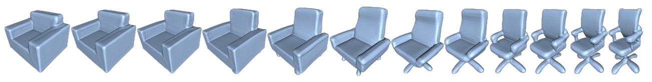


Figure 14: *An example of shape interpolation in the $\mathcal{Z}$ space.*

modify the shape style from "without arms" to "with arms", or vice versa. Similarly on the car category, we use the label "with roof" and "without roof" to find the separation plane and edit the shape style. In Fig. 13, we demonstrate this kind of shape style editing, and we find that the original geometry of the input shapes is also well-preserved.

Shape interpolation A straightforward application is to perform shape interpolation in the $\mathcal{Z}$ space, or the $\mathcal{W}$ space. Fig. 14 shows the intermediate results of the interpolation between two chairs, in the $\mathcal{Z}$ space. Due to the use of GAN model, each intermediate result is plausible, and the transition of shape geometry between adjacent frames is also smooth.

6. Conclusion

In the presented work, we offer a high-quality 3D generative model — SDF-StyleGAN for shape generation, which is capable of producing diverse and visually plausible 3D models superior to the state-of-the-art. These significant improvements are ascribed to the design of our global and local SDF discriminators, the choice of implicit SDF representation, the use of SDF gradients, and the StyleGAN2 network structure. We evaluated the learned generative model using suitable 3D GAN metrics, including our proposed FID on rendered images, and demonstrated the capability of SDF-StyleGAN on a series of applications.

Limitations A few limitations exist in our work. As there is no explicit shape part structure utilized in our framework design, we notice that a small portion of the generated shapes is not complete: tiny and thin parts could be missing, as shown in Fig. 15. Increasing the iso-value in extracting mesh surfaces can mitigate this issue, but there is no guarantee. We also found that complex geometry patterns such as various supporting beam layouts of chair backs, are not learned by our network, possibly due to unbalanced data distribution.

Future work Some research directions are left for future exploration. First, using truncated SDF would help leverage higher resolution ground-truth SDF during training to improve shape quality while reducing memory footprint and computational time. Second, our preliminary test on style mixing [KLA19] reveals that there is some relation between the styles learned from the network and the semantic structures of the shapes, but it is still difficult to make a semantically meaningful disentanglement. Furthermore, it would be very useful to extend our model to 3D scene generation.

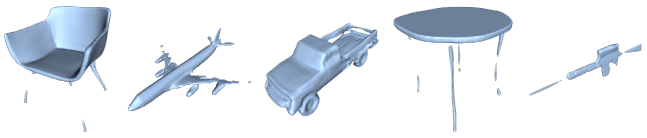


Figure 15: *Incomplete shapes generated by our method. The thin structures are not captured by the zero isosurfaces.*

References

[ADMG18] Achlioptas P., Diamanti O., Mitliagkas I., Guibas L.: Learning representations and generative models for 3D point clouds. In *ICML* (2018), PMLR, pp. 40–49. 2, 3, 6

[CF17] Chen H., Friedman J. H.: A new graph-based two-sample test for multivariate and object data. *Journal of the American statistical association 112*, 517 (2017), 397–409. 3

[CFG*15] Chang A. X., Funkhouser T., Guibas L., Hanrahan P., Huang Q., Li Z., Savarese S., Savva M., Song S., Su H., et al.: ShapeNet: An information-rich 3D model repository. arXiv:1512.03012, 2015. 5

[CKF*21] Chen Z., Kim V. G., Fisher M., Aigerman N., Zhang H., Chaudhuri S.: DECOR-GAN: 3D shape detailization by conditional refinement. In *CVPR* (2021), pp. 15740–15749. 2

[CMK*21] Chan E. R., Monteiro M., Kellnhofer P., Wu J., Wetzstein G.: pi-gan: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *CVPR* (2021), pp. 5799–5809. 2

[CTSO03] Chen D.-Y., Tian X.-P., Shen Y.-T., Ouhyoung M.: On Visual similarity based 3D model retrieval. *Computer Graphics Forum 22*, 3 (2003), 223–232. 3, 6

[CXG*16] Choy C. B., Xu D., Gwak J., Chen K., Savarese S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV* (2016). 10

[CYAE*20] Cai R., Yang G., Averbuch-Elor H., Hao Z., Belongie S., Snavely N., Hariharan B.: Learning gradient fields for shape generation. In *ECCV* (2020), Springer, pp. 364–381. 3

[CZ19] Chen Z., Zhang H.: Learning implicit fields for generative shape modeling. In *CVPR* (2019), pp. 5939–5948. 2, 3, 5, 6, 7, 10

[DDS*09] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.: Imagenet: A large-scale hierarchical image database. In *CVPR* (2009), IEEE, pp. 248–255. 6

[DXA*19] Dubrovina A., Xia F., Achlioptas P., Shalah M., Groscot R., Guibas L. J.: Composite shape modeling via latent space factorization. In *ICCV* (2019), pp. 8140–8149. 2

[GBZCO21] Gal R., Bermano A., Zhang H., Cohen-Or D.: MRGAN: Multi-rooted 3D shape representation learning with unsupervised part disentanglement. In *ICCV workshop* (2021), pp. 2039–2048. 2

[GFK*18] Groueix T., Fisher M., Kim V. G., Russell B. C., Aubry M.: AtlasNet: A Papier-Mâché approach to learning 3D surface generation. In *CVPR* (2018), pp. 216–224. 2

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. In *NeurIPS* (2014), vol. 27. 2

[GYW*19] Gao L., Yang J., Wu T., Yuan Y.-J., Fu H., Lai Y.-K., Zhang H.: SDM-NET: Deep generative network for structured deformable mesh. *ACM Trans. Graph. 38*, 6 (2019), 1–15. 3

[HJA20] Ho J., Jain A., Abbeel P.: Denoising diffusion probabilistic models. In *NeurIPS* (2020), vol. 33, pp. 6840–6851. 3

[HLXT19] HUANG W., LAI B., XU W., TU Z.: 3D volumetric modeling with introspective neural networks. In *AAAI* (2019), vol. 33, pp. 8481–8488. 2

[HRU*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: GANs trained by a two time-scale update rule converge to a local NASH equilibrium. In *NeurIPS* (2017), vol. 30. 3, 6

[HXX*20] HUI L., XU R., XIE J., QIAN J., YANG J.: Progressive point cloud deconvolution generation network. In *ECCV* (2020), Springer, pp. 397–413. 2

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *CVPR* (2016), pp. 770–778. 10

[IKK21] IBING M., KOBSIK G., KOBBELT L.: Octree Transformer: Autoregressive 3D shape generation on hierarchically structured sequences. arXiv:2111.12480, 2021. 3

[ILK21] IBING M., LIM I., KOBBELT L.: 3D shape generation with grid-based implicit functions. In *CVPR* (2021), pp. 13559–13568. 2, 3, 4, 6

[JM*17] JIANG C., MARCUS P., ET AL.: Hierarchical detail enhancing mesh-based shape generation with 3D generative adversarial network. arXiv:1709.07581, 2017. 2

[JPXZ20] JIN J., PATIL A. G., XIONG Z., ZHANG H.: DR-KFS: A differentiable visual similarity metric for 3D shape reconstruction. In *ECCV* (2020), Springer, pp. 295–311. 3

[JSM*20] JIANG C., SUD A., MAKADIA A., HUANG J., NIESSNER M., FUNKHOUSER T., ET AL.: Local implicit grid representations for 3D scenes. In *CVPR* (2020), pp. 6001–6010. 4

[KAL*21] KARRAS T., AITTALA M., LAINE S., HÄRKÖNEN E., HELLSTEN J., LEHTINEN J., AILA T.: Alias-Free generative adversarial networks. In *NeurIPS* (2021), vol. 34. 2

[KBV20] KLOKOV R., BOYER E., VERBEEK J.: Discrete point flow networks for efficient point cloud generation. In *ECCV* (2020), Springer, pp. 694–710. 3

[KFW20] KLEINEBERG M., FEY M., WEICHERT F.: Adversarial generation of continuous implicit shape representations. In *Eurographics 2020 - Short Papers* (2020), The Eurographics Association. 2, 6

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Trans. Graph. 32*, 3 (2013), 1–13. 9, 10

[KLA19] KARRAS T., LAINE S., AILA T.: A style-based generator architecture for generative adversarial networks. In *CVPR* (2019), pp. 4401–4410. 2, 11

[KLA*20] KARRAS T., LAINE S., AITTALA M., HELLSTEN J., LEHTINEN J., AILA T.: Analyzing and improving the image quality of StyleGAN. In *CVPR* (2020), pp. 8110–8119. 2, 3, 4

[KLK*20] KIM H., LEE H., KANG W. H., LEE J. Y., KIM N. S.: Softflow: Probabilistic framework for normalizing flow on manifolds. In *NeurIPS* (2020), vol. 33. 3

[KMU21] KIMURA T., MATSUBARA T., UEHARA K.: ChartPointFlow for topology-aware 3D point cloud generation. In *ACM Multimedia* (2021), pp. 1396–1404. 3

[KW14] KINGMA D. P., WELLING M.: Auto-encoding variational Bayes. In *ICLR* (2014). 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *ACM Trans. Graph.* (1987), 163–169. 7

[LDPT19] LI X., DONG Y., PEERS P., TONG X.: Synthesizing 3D shapes from silhouette image collections using multi-projection generative adversarial networks. In *CVPR* (2019), pp. 5535–5544. 2

[LGP*21] LIU S.-L., GUO H.-X., PAN H., WANG P.-S., TONG X., LIU Y.: Deep implicit moving least-squares functions for 3D reconstruction. In *CVPR* (2021), pp. 1788–1797. 9

[LH19] LOSHCHILOV I., HUTTER F.: Decoupled Weight Decay Regularization. In *ICLR* (2019). 9

[LH21] LUO S., HU W.: Diffusion probabilistic models for 3D point cloud generation. In *CVPR* (2021), pp. 2837–2845. 3

[LLHF21] LI R., LI X., HUI K.-H., FU C.-W.: SP-GAN: Sphere-guided 3D shape generation and manipulation. *ACM Trans. Graph. 40*, 4 (2021), 1–12. 2, 3, 7

[LLW22] LI S., LIU M., WALDER C.: EditVAE: Unsupervised part-aware controllable 3D point cloud shape generation. In *AAAI* (2022). 3

[LLZL21] LUO A., LI T., ZHANG W.-H., LEE T. S.: SurfGen: Adversarial 3D shape synthesis with explicit surface discriminators. In *ICCV* (2021), pp. 16238–16248. 2

[LNX20] LI J., NIU C., XU K.: Learning part generation and assembly for structure-aware shape synthesis. In *AAAI* (2020), vol. 34, pp. 11362–11369. 2

[LXC*17] LI J., XU K., CHAUDHURI S., YUMER E., ZHANG H., GUIBAS L.: GRASS: Generative recursive autoencoders for shape structures. *ACM Trans. Graph. 36*, 4 (2017), 1–14. 2

[MCST22] MITTAL P., CHENG Y.-C., SINGH M., TULSIANI S.: AutoSDF: Shape priors for 3D completion, reconstruction and generation. In *CVPR* (2022). 3

[MGN18] MESCHEDER L., GEIGER A., NOWOZIN S.: Which training methods for GANs do actually converge? In *ICML* (2018), PMLR, pp. 3481–3490. 5

[MGY*19] MO K., GUERRERO P., YI L., SU H., WONKA P., MITRA N. J., GUIBAS L. J.: StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Trans. Graph. 38*, 6 (2019), 1–19. 2

[MKC18] MURALIKRISHNAN S., KIM V. G., CHAUDHURI S.: Tags2Parts: Discovering semantic regions from shape tags. In *CVPR* (2018), pp. 2926–2935. 10

[MON*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy Networks: Learning 3D reconstruction in function space. In *CVPR* (2019), pp. 4460–4470. 10

[MWYG20] MO K., WANG H., YAN X., GUIBAS L.: PT2PC: Learning to generate 3D point cloud shapes from part tree conditions. In *ECCV* (2020), Springer, pp. 683–701. 2, 3

[NGEB20] NASH C., GANIN Y., ESLAMI S. A., BATTAGLIA P.: PolyGen: An autoregressive generative model of 3D meshes. In *ICML* (2020), PMLR, pp. 7220–7229. 3

[OELS*22] OR-EL R., LUO X., SHAN M., SHECHTMAN E., PARK J. J., KEMELMACHER-SHLIZERMAN I.: StyleSDF: High-resolution 3D-Consistent image and geometry generation. In *CVPR* (2022). 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR* (2019), pp. 165–174. 2

[PLS*21] POSTELS J., LIU M., SPEZIALETTI R., VAN GOOL L., TOMBARI F.: Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction. In *3DV* (2021), IEEE, pp. 1249–1258. 3

[PNM*20] PENG S., NIEMEYER M., MESCHEDER L., POLLEFEYS M., GEIGER A.: Convolutional occupancy networks. In *ECCV* (2020), Springer, pp. 523–540. 4, 9

[PPMNF20] PUMAROLA A., POPOV S., MORENO-NOGUER F., FERRARI V.: C-flow: Conditional generative flow models for images and 3D point clouds. In *CVPR* (2020), pp. 7949–7958. 3

[PZZ21] PARMAR G., ZHANG R., ZHU J.-Y.: On aliased resizing and surprising subtleties in GAN evaluation. arXiv:2104.11222, 2021. 7

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR* (2017), pp. 652–660. 2, 3

[RM15] REZENDE D., MOHAMED S.: Variational inference with normalizing flows. In *ICML* (2015), PMLR, pp. 1530–1538. 2, 3

[SE19] SONG Y., ERMON S.: Generative modeling by estimating gradients of the data distribution. In *NeurIPS* (2019), vol. 32. 3

[SM17] SMITH E. J., MEGER D.: Improved adversarial systems for 3D object generation and reconstruction. In *Conference on Robot Learning* (2017), PMLR, pp. 87–96. 2

[SPK19] SHU D. W., PARK S. W., KWON J.: 3D point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV* (2019), pp. 3859–3868. 3, 7

[SVI*16] SZEGEDY C., VANHOUCKE V., IOFFE S., SHLENS J., WOJNA Z.: Rethinking the inception architecture for computer vision. In *CVPR* (2016), pp. 2818–2826. 6

[SWL*20] SUN Y., WANG Y., LIU Z., SIEGEL J. E., SARMA S. E.: PointGrow: Autoregressively learned point cloud generation with self-attention. In *Winter Conference on Applications of Computer Vision* (2020). 3

[SYTZ20] SHEN Y., YANG C., TANG X., ZHOU B.: Interfacegan: Interpreting the disentangled face representation learned by GANs. *IEEE Trans. Pattern Anal. Mach. Intell.* (2020). 10, 11

[VOKK16] VAN OORD A., KALCHBRENNER N., KAVUKCUOGLU K.: Pixel recurrent neural networks. In *ICML* (2016), pp. 1747–1756. 2, 3

[WLG*17] WANG P.-S., LIU Y., GUO Y.-X., SUN C.-Y., TONG X.: O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph. 36*, 4 (2017). 9

[WLT22] WANG P.-S., LIU Y., TONG X.: Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Trans. Graph. 41*, 4 (2022). 9

[WS20] WU Y., SUN Z.: DFR: Differentiable function rendering for learning 3D generation from images. *Computer Graphics Forum 39*, 5 (2020), 241–252. 2

[WSH*18] WANG H., SCHOR N., HU R., HUANG H., COHEN-OR D., HUANG H.: Global-to-local generative model for 3D shapes. *ACM Trans. Graph. 37*, 6 (2018), 1–10. 2, 3

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph. 38*, 5 (2019), 1–12. 3, 7, 10

[WYT21] WEN C., YU B., TAO D.: Learning progressive point embeddings for 3D point cloud generation. In *CVPR* (2021), pp. 10266–10275. 2

[WZL*18] WANG N., ZHANG Y., LI Z., FU Y., LIU W., JIANG Y.-G.: Pixel2Mesh: Generating 3D mesh models from single rgb images. In *ECCV* (2018), pp. 52–67. 10

[WZX*16] WU J., ZHANG C., XUE T., FREEMAN W. T., TENENBAUM J. B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *NeurIPS* (2016), pp. 82–90. 2

[WZX*20] WU R., ZHUANG Y., XU K., ZHANG H., CHEN B.: PQ-NET: A generative part Seq2Seq network for 3D shapes. In *CVPR* (2020), pp. 829–838. 3

[XB14] XU H., BARBIČ J.: Signed distance fields for polygon soup meshes. In *Graphics Interface* (2014), p. 35–41. 5

[XLZW16] XIE J., LU Y., ZHU S.-C., WU Y.: A theory of generative convnet. In *ICML* (2016), PMLR, pp. 2635–2644. 3

[XXZ*21] XIE J., XU Y., ZHENG Z., ZHU S.-C., WU Y. N.: Generative PointNet: Deep energy-based learning on unordered point sets for 3D generation, reconstruction and classification. In *CVPR* (2021), pp. 14976–14985. 3

[XZG*20] XIE J., ZHENG Z., GAO R., WANG W., ZHU S.-C., WU Y. N.: Generative VoxelNet: Learning energy-based models for 3D shape synthesis and analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* (2020). 3

[YFW*18] YAZ Y., FOO C.-S., WINKLER S., YAP K.-H., PILIOURAS G., CHANDRASEKHAR V., ET AL.: The unusual effectiveness of averaging in GAN training. In *ICLR* (2018). 5

[YHH*19] YANG G., HUANG X., HAO Z., LIU M.-Y., BELONGIE S., HARIHARAN B.: PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV* (2019), pp. 4541–4550. 3, 6

[YML*22] YANG J., MO K., LAI Y.-K., GUIBAS L. J., GAO L.: DSG-Net: Learning Disentangled Structure and Geometry for 3D Shape Generation. *ACM Trans. Graph.* (2022). 3

[YRW*21] YU X., RAO Y., WANG Z., LIU Z., LU J., ZHOU J.: Pointr: Diverse point cloud completion with geometry-aware transformers. In *ICCV* (2021), pp. 12498–12507. 10

[ZDW21] ZHOU L., DU Y., WU J.: 3D shape generation and completion through point-voxel diffusion. In *ICCV* (2021), pp. 5826–5835. 3