WarpingGAN: Warping Multiple Uniform Priors for Adversarial 3D Point Cloud Generation

Yingzhi Tang[1*] Yue Qian[1*] Qijian Zhang[1] Yiming Zeng[1] Junhui Hou[1] Xuefei Zhe[2]
[1]City University of Hong Kong [2]Tencent AI lab
{yztang4-c, yueqian4-c, qijizhang3-c, ym.zeng}@my.cityu.edu.hk, jh.hou@cityu.edu.hk

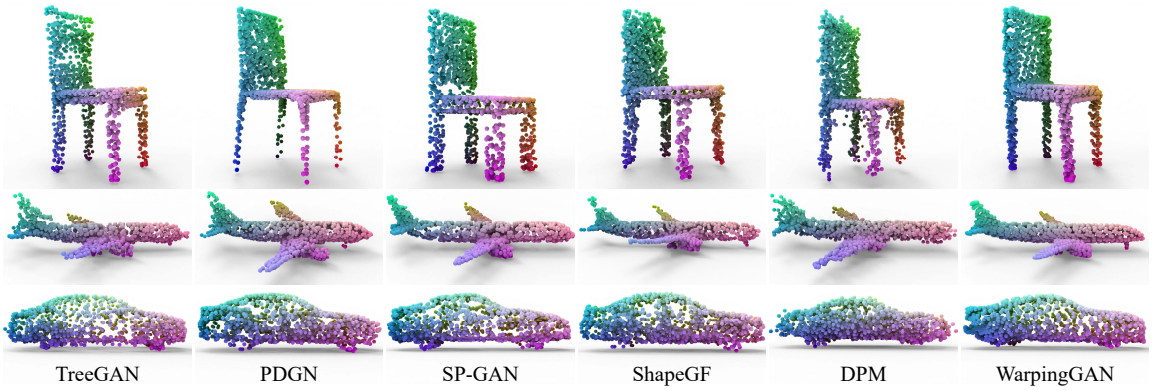


Figure 1. Visual comparisons of generated shapes with state-of-the-art 3D point cloud generation methods. TreeGAN [20], PDGN [10] and SP-GAN [15] are GAN-based, while ShapeGF [5] and DPM [17] are probabilistic-based.

Abstract

We propose WarpingGAN, an effective and efficient 3D point cloud generation network. Unlike existing methods that generate point clouds by directly learning the mapping functions between latent codes and 3D shapes, WarpingGAN learns a unified local-warping function to warp multiple identical pre-defined priors (i.e., sets of points uniformly distributed on regular 3D grids) into 3D shapes driven by local structure-aware semantics. In addition, we also ingeniously utilize the principle of the discriminator and tailor a stitching loss to eliminate the gaps between different partitions of a generated shape corresponding to different priors for boosting quality. Owing to the novel generating mechanism, WarpingGAN, a single lightweight network after one-time training, is capable of efficiently generating uniformly distributed 3D point clouds with various resolutions. Extensive experimental results demonstrate the superiority of our WarpingGAN over state-of-the-art methods in terms of quantitative metrics, visual quality, and efficiency. The source code is publicly available at https://github.com/yztang4/WarpingGAN.git.

1. Introduction

3D point clouds have been employed in various applications, such as computer-aided design [11, 13], augmented/virtual reality [16], animation [9, 27], and immersive telepresence [21]. However, obtaining 3D point cloud data is still costly and time-consuming in realistic scenarios, especially the shapes with complex geometry and topology. Besides, the acquired point clouds with 3D sensing devices are usually incomplete and sparse due to occlusions, distances, and surface materials. The great success of generative adversarial network (GAN)-based 2D image generation [4, 6, 31] makes synthesizing realistic-looking 3D point clouds promising, i.e., generating point clouds whose statistical distribution is similar to real point clouds. However, the essentially different data modality as well as the unique characteristics of 3D point clouds, i.e., the irregular structure and unorderness, makes it non-trivial to extend GAN-based methods for generating 2D images to 3D point cloud generation.

Recently, several works on 3D point cloud generation have been proposed [1, 10, 12, 15, 20, 22, 26, 28]. For example, GAN-based methods [1, 10, 20, 22, 26] usually use multi-layer perceptrons (MLPs) as generators to directly learn mapping functions between latent codes and 3D point clouds, which require a large number of parameters to fit. Moreover, as the adversarial learning mechanism cannot impose strong constraints on global shapes and local geometric details, these approaches tend to generate non-uniformly distributed point clouds, as illustrated in Fig. 2. Yang *et al.* [28] and Luo *et al.* [17] considered 3D point cloud generation as probabilistic problems, which first sample points from a Gaussian space and then move them to the

This work was supported by the HK RGC Grant CityU 11202320 and 11218121. Corresponding author: J. Hou.
*Equal Contributions

(a) TreeGAN (b) PDGN (c) SP-GAN (d) WarpingGAN

Figure 2. Visual comparisons of the point clouds generated by different GAN-based methods.

target position by learning the distribution transformation. However, these methods generate blurry point clouds without clear global shapes and local details, since they tend to estimate the average distribution of training data. In addition to the limited quality, existing methods also suffer from low efficiency because time-consuming k nearest neighbor (kNN) search is adopted in PDGN [10] and SP-GAN [15], a progressive generation process is utilized in TreeGAN [20] and PDGN [10], and a two-stage training strategy is required in ShapeGF [5], which also prohibits the end-to-end optimization.

To address the above-mentioned issues, we propose WarpingGAN, which introduces a novel mechanism for GAN-based 3D point cloud generation. By taking advantage of multiple 3D uniform priors, i.e., sets of points uniformly located on a unit 3D cube, WarpingGAN formulates the generation process as the learning of a function that warps multiple 3D priors into different local regions of a 3D shape under the guidance of local structure-aware semantics, which is fundamentally different from existing methods that directly learn the process of producing a fixed number of points from the latent code. Meanwhile, we tailor a stitching loss, which minimizes the local difference between generated and real point clouds, to shrink the gaps between different partitions. Such a new mechanism makes WarpingGAN featured with compactness and high efficiency. Also, it enables WarpingGAN to generate point clouds with various numbers of points after one-time training. Besides, the uniformity of the 3D priors can implicitly regularize WarpingGAN to some extent to promote the generation of uniformly distributed point clouds, as shown in Fig. 2.

In summary, we make the following contributions:

- we investigate the GAN-based 3D point cloud generation from the new perspective of *unified local-warping*, leading to WarpingGAN featured with lightweight, high efficiency, and flexible output; and
- by taking advantage of the inherent design of the discriminator without introducing additional complex operations, we propose a stitching loss tailored to WarpingGAN to boost the generator; and
- we conduct extensive experiments and analysis to demonstrate the superiority of WarpingGAN over state-of-the-art methods.

2. Related Work

Existing 3D point cloud generation methods can be roughly classified into two categories, i.e., GAN-based methods and Probabilistic-based.

GAN-based approaches. As the first work, Achlioptas *et al.* [1] proposed rGAN, whose generator consists of several fully connected layers. However, both the generator and discriminator of rGAN cannot well utilize the local information and tends to generate defective parts. Valsesia *et al.* [22] utilized a graph convolution network to learn local dependencies between a point and its neighbors by designing a localized operation. Shu *et al*. proposed [20] TreeGAN, where a tree structure is introduced to preserve ancestor information instead of neighbor information to generate new points. Hui *et al.* [10] proposed a progressive learning strategy to generate multi-resolution point clouds, and a learning-based bilateral interpolation is utilized to exploit local geometric structure. The above methods employ MLPs on the global feature to directly generate point clouds, which can be hard to optimize and inflexible in terms of the number of points. To simplify the learning process, TreeGAN and PDGN adopt the inefficient progressive architecture in the generator.

Recently, Li *et al.* [15] proposed SP-GAN for point cloud generation and manipulation. They introduce a pre-defined sphere to perform deformation. By contrast, our method adopts multiple uniform 3D priors to warp each shape partition, enabling higher-quality point clouds. Furthermore, unlike SP-GAN, our architecture does not employ the time-consuming kNN operation in the generator. The experiment findings show that our proposed WarpingGAN is more effective and efficient than SP-GAN.

It is worth noting that Wang *et al.* [24] experimentally found that the current GAN-based frameworks can only adopt PointNet as the discriminator. Other more advanced point cloud frameworks such as PointNet++ [19] and DGCNN [25] are unable to be optimized as discriminators. However, PointNet learns point-wise features and uses the max-pooling symmetric function to select critical points to determine the global shape feature. Thus, only a small portion of critical points guide the global shape, and local geometric information is lost. Therefore, a well-designed generator is crucial for the GAN-based method.

Probabilistic-based approaches regard point clouds as samples from a distribution, and then move the sampled points to the target positions during the generative phase. PointFlow [28] uses the continuous normalizing flow framework to transform the parameters of the distributions of shapes and the distribution of points given a shape. ShapeGF [5] generates point clouds by learning the gradient field of its log-density and moves points gradually in the gradient direction. DPM [17] simulates the generation process as non-equilibrium thermodynamics by converting the

noise distribution into the shape distribution with a Markov chain. Some of them [17,28] can accomplish flexible generation because they treat each point independently. However, due to the absence of association between the points, they cannot properly settle the non-uniform and noisy problems for generated shapes. Moreover, many probabilistic-based methods [5] adopt a two-stage training process, which requires additional training for auto-encoders. It's worth noting that ShapeGF uses the GAN structure as well, not only the auto-encoder.

Point cloud auto-encoders aim to reconstruct the input point cloud with a narrow bottleneck layer. For example, FoldingNet [29] introduces a folding-style operation to reconstruct 3D shapes by learning a mapping function from a 2D grid to a 3D point cloud. AtlasNet [7] utilizes a series of 2D grids via multiple independent MLPs to reconstruct surfaces patch-wisely. Bednarik *et al.* [2] addressed patch collapse and overlap problems of AtlasNet by computing the differential properties of reconstructed surface with first and second derivatives.

Remark. We argue that such folding-style decoders are potentially suitable for the GAN-based point cloud generation task, which have been ignored in previous frameworks. In this work, we take a step forward in this direction by investigating a more powerful point cloud generator. Note that directly adopting the decoders of FoldingNet and AtlasNet as the generator of a GAN-base point cloud generation framework fail to generate satisfied point clouds. See the analysis in Section 3.1 and experimental demonstration in Section 4.3.

3. Proposed Method

3.1. Problem Analysis and Formulation

Given a latent code $\mathbf{z} \in \mathbb{R}^C$ following a Gaussian distribution, the generator denoted as $\mathcal{F}(\cdot)$ attempts to produce a point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$ which shares the same statistical distribution as a real dataset $\{\widetilde{\mathbf{P}}\}$. Most of existing GAN-based methods directly learn the mapping function between $\mathbf{z}$ and $\mathbf{P}$, i.e., $\mathcal{F}(\mathbf{z};\mathbf{\Theta}) = \mathbf{P}$ with $\mathbf{\Theta}$ being the network parameters to be learned. However, due to the weak supervision ability of the discriminator, it is difficult to learn $\mathcal{F}(\cdot)$, especially for the dataset with complex shapes, thus limiting the quality of generated point clouds. More specifically, Wang *et al.* [24] analyzed that as the only feasible discriminator, the PointNet architecture [18], which applies the max-pooling operator to sample the feature space, can only perceive the distribution of a small number of critical points rather than the whole point cloud.

Motivated by the the folding-style design mentioned earlier, we consider warping a uniform 3D prior (i.e., a set of 3D points uniformly located in a regular 3D grid) denoted as $\mathbf{U} \in [0,1]^{N\times 3}$ into a 3D shape. Accordingly, we reformulate the generation process point-wisely as $\mathcal{F}(\mathbf{u}_i, \mathbf{z}; \mathbf{\Theta}) = \mathbf{p}_i \ \forall i \in [1, N]$, where $\mathbf{u}_i$ and $\mathbf{p}_i$ are the i-th point of $\mathbf{U}$ and $\mathbf{P}$, respectively. We expect that the pre-defined uniformity of the 3D prior could regularize the distribution of all of the generated points to mitigate the limitation of the max-pooling-based discriminator mentioned earlier. However, a single prior may fail to generate complex shapes well due to the essential difference between the topology of the prior and the 3D objects. Thus, we plan to use multiple uniform 3D priors denoted as $\left\{\mathbf{U}^j \in [0,1]^{n\times 3}\right\}_{j=1}^{M}$ and warp each of them to capture the local region (i.e., $\mathbf{P}^j \in \mathbb{R}^{n\times 3}$) of a generated point cloud $\mathbf{P} = \bigcup_{j=1}^{M} \mathbf{P}^j$, where $N = M \times n$. To achieve this, like AtlasNet [7], one intuitive way is to learn a warping process for each pair of $\mathbf{U}^j$ and $\mathbf{P}^j$ independently, i.e., $\mathbf{p}_i^j = \mathcal{F}(\mathbf{u}_i^j, \mathbf{z}; \mathbf{\Theta}^j)$, where $\mathbf{u}_i^j$ and $\mathbf{p}_i^j$ are the i-th points of $\mathbf{U}^j$ and $\mathbf{P}^j$, respectively. However, this manner significantly increases network parameters $\{\mathbf{\Theta}^j\}_{j=1}^{M}$, making the network difficult to train.

To achieve the generation in an efficient and effective manner, we finally formulate it as a unified local-warping process. That is, we use M different global-correlated local codes $\{\mathbf{z}^j\}_{j=1}^{M}$ (see Section 3.2 for details), each of which is expected to embed the semantic of a typical local region. Thus, a local code can drive the warping of a prior to the corresponding structure via an *identical* MLPs. Accordingly, the process is generally written as

$$\mathbf{p}_i^j = \mathcal{F}(\mathbf{u}_i^j, \mathbf{z}^j; \mathbf{\Theta}), \forall j \in [1, M] \text{ and } \forall i \in [1, n]. \quad (1)$$

Compared with the AtlasNet-based idea, our warping process only requires unified parameters $\mathbf{\Theta}$ to be optimized, which is more compact and easier. Moreover, although we expect to use multiple priors rather than a single one improve generation quality, the weak supervision ability of the discriminator may not realize our objective, i.e., the supervision is insufficient to drive the local regions warped from different priors to tightly fit to each other, resulting in gaps between them. To handle this issue, we further tailor a simple yet effective stitching loss to supervise the training of the generator.

3.2. Warping-based Generator

Fig. 3 illustrates the overall architecture of the generator of our WarpingGAN, which consists of two modules, i.e., code enhancement and unified local warping. Specifically, taking a latent code $\mathbf{z}$ as input, the code enhancement module first enhances its representation ability to 3D shapes by changing its distribution. Conditioned on the enhanced code, the unified local-warping module is then successively performed twice to warp multiple pre-defined uniform 3D priors to different local regions, which are finally assembled into a 3D shape. Owing to the warping-style mechanism, WarpingGAN is featured with highly compact and efficient. Besides, WarpingGAN is able to generate point

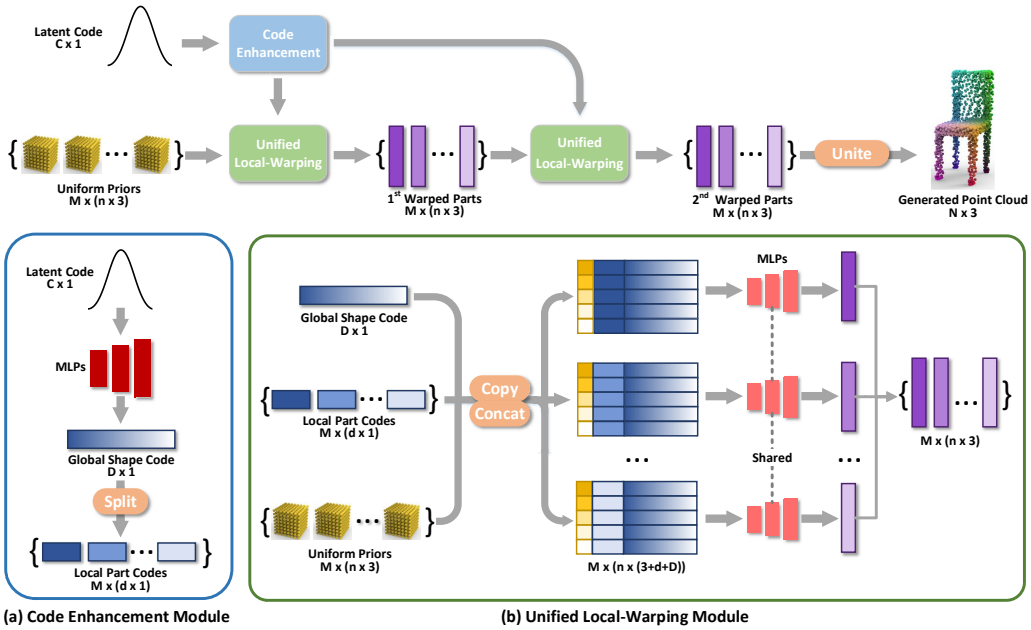


Figure 3. Illustration of the architecture of the ***generator*** of WarpingGAN, which consists of a code enhancement module and a unified local-warping module. During training, it takes a latent code and M pre-defined uniform 3D priors as input. The unified local-warping module is performed twice to generate a 3D shape.

clouds with various numbers of points by changing the size of $\mathbf{U}^j$ (i.e., the value of n), after one-time training.

Code enhancement. As aforementioned, WarpingGAN aims to learn a warping function under the guidance of the latent code $\mathbf{z}$. However, $\mathbf{z}$ is randomly drawn from a Gaussian distribution and thus lacks the implicit semantic information to represent the shape faithfully. To fill this knowledge gap, we propose a code enhancement module shown in Fig. 3(a), composed of five fully-connected layers to transform $\mathbf{z}$ to $\widetilde{\mathbf{z}} \in \mathbb{R}^D$ of a higher dimension ($D > C$). In Section 4.3, we illustrate that after the data-driven training process, such a module can transform the Gaussian distribution to a distribution that is comparable to that of the features extracted from real datasets that encode shape semantics.

Unified local-warping. Guided by the enhanced latent code $\widetilde{\mathbf{z}}$, the unified local-warping module in Fig. 3(b) can generate from multiple 3D priors various local regions that are further assembled into a point cloud with complex topology. Specifically, we first partition $\widetilde{\mathbf{z}}$ into M local codes of an equal length denoted as $\{\widetilde{\mathbf{z}}^j \in \mathbb{R}^{D/M}\}_{j=1}^{M}$ and then concatenate each of the local codes with the global shape information, leading to $\mathbf{z}^j = [\widetilde{\mathbf{z}}^j\ \widetilde{\mathbf{z}}] \in \mathbb{R}^{D(M+1)/M}$. Finally, we concatenate $\mathbf{z}^j$ with each coordinate of $\mathbf{U}^j$, which is then fed into an MLPs to regress the points of the j-th local region in point-wise. Moreover, we perform such a warping process twice in a row, Therefore, such a unified local-warping process is written as

$$\mathbf{p}_i^j = \mathcal{F}\left(\mathcal{F}(\mathbf{u}_i^j, \mathbf{z}^j; \mathbf{\Theta}_1), \mathbf{z}^j; \mathbf{\Theta}_2\right), \quad (2)$$
$$\forall j \in [1, M] \text{ and } \forall i \in [1, n],$$

where $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ are the network parameters of the two consecutive warping processes. Note that it is crucial to concatenate $\widetilde{\mathbf{z}}$ in the local codes since $\widetilde{\mathbf{z}}$ provides the essential global shape information to coordinate different local codes (see the demonstration in Section 4.3).

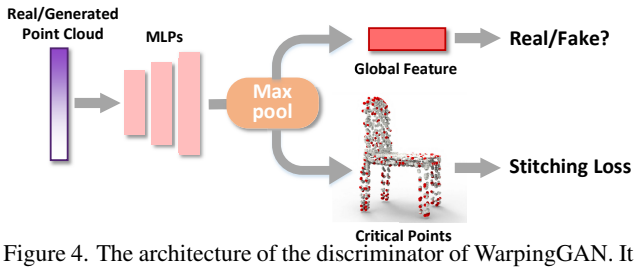


Figure 4. The architecture of the discriminator of WarpingGAN. It takes real point clouds or generated point clouds as input and produces the confidence value and the critical points (i.e., red points) that are used by the stitching loss.

3.3. Training Objectives

Discriminator. Following previous works [10, 20], we adopt the PointNet as the discriminator $\mathcal{D}(\cdot)$ for training the generator. As shown in Fig. 4, it takes $\{\widetilde{\mathbf{P}}\}$ as input to perform binary classification. More specifically, it first learns point-wise features via MLPs and then adopts max-pooling to obtain a global shape feature that is utilized to determine the confidence value. Meanwhile, owing to the max-pooling operation, we can retrieve a small number of *critical points* from the input cloud, which depict the skeleton of the input shape [18].

Stitching loss. As analyzed in Section 3.1, the inher-

ent design of the discriminator results in a limited supervision ability, which is inadequate to force the generated local regions to fit each other tightly. To tackle this issue, by taking advantage of critical points without introducing additional complex operations, we propose a stitching loss, which minimizes the local difference between $\mathbf{P}$ and $\widetilde{\mathbf{P}}$.

Let $\{\mathbf{q}_i\}_{i=1}^{Q} \subset \mathbf{P}$ and $\{\widetilde{\mathbf{q}}_i\}_{i=1}^{\widetilde{Q}} \subset \widetilde{\mathbf{P}}$ denote the critical points retrieved from the input point clouds based on the max-pooling operation of the discriminator. We first find out the K nearest neighbors $\mathcal{N}(\mathbf{q}_i) = \{\mathbf{p}_i^k\}_{k=1}^{K} \subset \mathbf{P}$ for each of $\{\mathbf{q}_i\}_{i=1}^{Q}$ and compute the pairwise distance $d_i^k = \|\mathbf{q}_i - \mathbf{p}_i^k\|_2$. We also conduct the same operation for $\{\widetilde{\mathbf{q}}_i\}_{i=1}^{\widetilde{Q}}$. With this information, we define the stitching loss as

$$\mathcal{L}_s = \left(\frac{1}{Q} \sum_{i=1}^{Q} \text{var}(\mathbf{q}_i, \mathcal{N}(\mathbf{q}_i)) - \frac{1}{\widetilde{Q}} \sum_{i=1}^{\widetilde{Q}} \text{var}(\widetilde{\mathbf{q}}_i, \mathcal{N}(\widetilde{\mathbf{q}}_i)) \right)^2, \tag{3}$$

where $\text{var}(\mathbf{q}_i, \mathcal{N}(\mathbf{q}_i)) = \sum_{k=1}^{K} \frac{(d_i^k - \bar{d}_i)^2}{K}$ and $\bar{d}_i$ is the mean value of $\{d_i^k\}_{i=1}^{Q}$.

Note that the stitching loss requires the kNN operation over the very small number of critical points only during training, and thus, the high efficiency of the proposed method is still retained during both training and testing. See Section 4.2.

Joint optimization. To train the proposed WarpingGAN, we adopt the improved WGAN loss [8], which consists of the loss $\mathcal{L}_g(\cdot)$ for the generator and $\mathcal{L}_d(\cdot)$ with the Lipschitz constraint for the discriminator. More precisely, $\mathcal{L}_g(\cdot)$ is written as

$$\mathcal{L}_g = -\mathbb{E}_{\mathbf{P} \sim \mathbb{P}_\mathbf{P}}[\mathcal{D}(\mathbf{P})] + \lambda_s \mathcal{L}_s, \tag{4}$$

where $\mathbb{P}_\mathbf{P}$ is the distribution of generated shape $\mathbf{P}$ and $\mathcal{L}_s$ is the proposed stitching loss which is balanced by the weight $\lambda_s > 0$. $\mathcal{L}_d(\cdot)$ is formulated as

$$\begin{aligned} \mathcal{L}_d = \mathbb{E}_{\mathbf{P} \sim \mathbb{P}_\mathbf{P}}[\mathcal{D}(\mathbf{P})] - \mathbb{E}_{\mathbf{P} \sim \mathbb{P}_{\widetilde{\mathbf{P}}}}[\mathcal{D}(\widetilde{\mathbf{P}})] \\ + \lambda_{gp} \mathbb{E}_{\widehat{\mathbf{p}} \sim \mathbb{P}_{\widehat{\mathbf{P}}}}[(\|\nabla_{\widehat{\mathbf{p}}} \mathcal{D}(\widehat{\mathbf{p}})\|_2 - 1)^2], \end{aligned} \tag{5}$$

where $\mathbb{P}_{\widetilde{\mathbf{P}}}$ is the distribution of the real point cloud $\widetilde{\mathbf{P}}$, and $\widehat{\mathbf{p}}$ is uniformly sampled by interpolating pairs of shapes sampled from $\mathbb{P}_\mathbf{P}$ and $\mathbb{P}_{\widetilde{\mathbf{P}}}$ to satisfy the 1-Lipschitz constraint [8], and $\lambda_{gp} > 0$ the weight to balance the gradient penalty term.

4. Experiments

4.1. Experiment Settings

Dataset. Following the settings in [20], we selected three categories of ShapeNet, i.e., *Chair*, *Airplane* and *Car shapes*, to train and evaluate WarpingGAN. Each point cloud contains 2048 points.

Implementation details. WarpingGAN samples latent codes of dimension $C = 128$ following a Gaussian distribution as input to generate point clouds each with $N = 2048$ points. We set the number of priors M to 16 for all shapes, $K = 40$ for computing the stitching loss, and $\lambda_s = 0.05$ and $\lambda_{gp} = 10$ during the training phase. We adopted LeakyReLU with a negative slope equal to 0.2 in each layer. We utilized Adam with the learning rate $r = 0.0001$, $\beta_1 = 0$ and $\beta_2 = 0.99$ as the optimizer to optimize both the generator and discriminator, and set the batch size to 32. We implemented the whole network with PyTorch and trained it on Nvidia RTX 2080ti GPU with Intel(R) Xeon(R) CPU.

Table 1. Quantitative comparison of WarpingGAN with five state-of-the-art methods over two categories. The listed MMD and COV values were obtained by multiplying the original values with 10^3 and 10^2, respectively. ↑ (resp. ↓) means the higher (resp. lower), the better.

Method \ Metric	Chair			Airplane		
	MMD↓	COV↑	Uniform↓	MMD↓	COV↑	Uniform↓
TreeGAN [20]	9.6	45.00	0.88	3.8	42.50	0.45
PDGN [10]	9.3	51.25	0.85	3.4	41.25	0.21
SP-GAN [15]	11.5	41.25	0.34	3.5	46.25	0.05
ShapeGF [5]	9.6	50.00	0.64	3.5	47.50	0.09
DPM [17]	9.4	37.50	1.45	3.4	33.75	0.35
WarpingGAN	**8.7**	**53.75**	**0.29**	**3.3**	**48.75**	**0.02**

4.2. Comparison with State-of-the-Art Methods

We compared the proposed WarpingGAN with five state-of-the-art point cloud generation frameworks, including three GAN-based methods, i.e., TreeGAN [20], PDGN [10] and SP-GAN [15], and two probablistic-based methods, i.e., ShapeGF [5] and DPM [17].

Quantitative comparison. Following the settings of SP-GAN [15], we utilized Minimal Matching Distance (MMD) and Coverage (COV) to quantitatively evaluate the quality of generated point clouds by different methods. Besides, we also adopted the uniformity loss* in [14] to quantitatively measure the uniformity of generated point clouds.

As listed in Table 1, our WarpingGAN outperforms all the other methods in terms of all metrics. Specifically, the lower MMD values imply that WarpingGAN can generate shapes with high fidelity to point clouds in the real dataset, the high COV values demonstrate that the generated shapes of WarpingGAN match well with the real shapes in terms of fraction, and the lower Uniform values indicate our WarpingGAN can generate point clouds with more uniformly distributed points. Moreover, we want to point out that metrics MMD and COV do not necessarily and reliably correlate to the quality of generated data, which has also been discussed in [28, 30]. Thus, we refer readers to examine visual quality of generated point clouds provided as follows and in *Supplementary Material*.

Visual comparison. We visualized the generated 3D

*We measured the normalized point clouds with various percentages of points, i.e., $p \in \{0.002, 0.004, 0.006, 0.008, 0.012, 0.015\}$. We reported the average uniformity over all p.

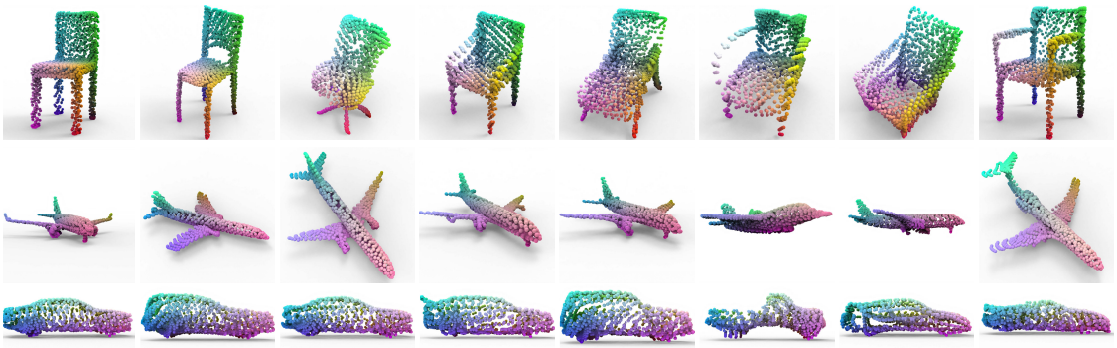


Figure 5. Visual illustration of the generated *Chair*, *Airplane* and *Car* shapes by our WarpingGAN. These shapes have fine global structures and present a variety of geometric typology. See *Supplementary Material* for more visual results.

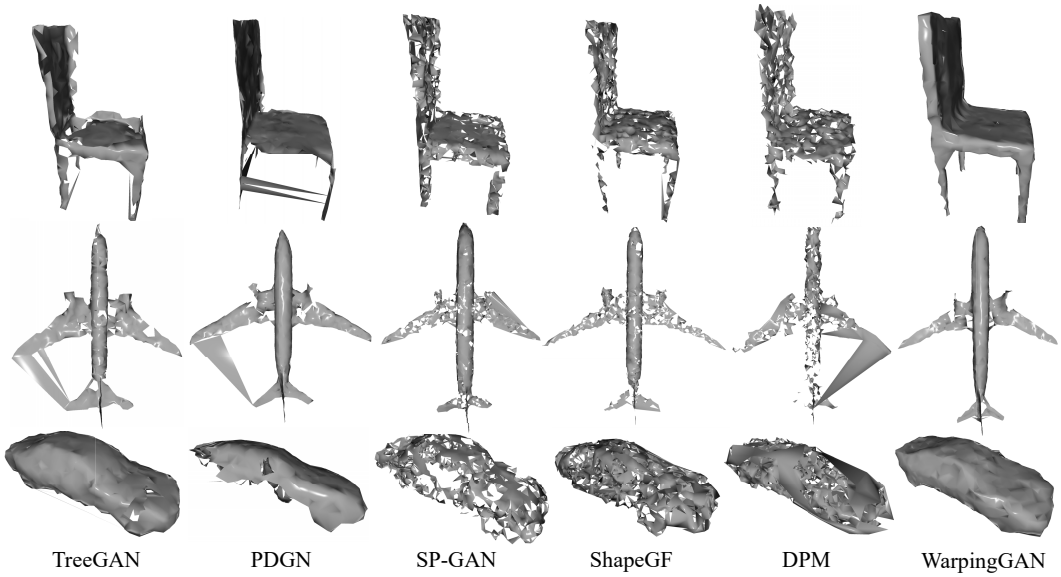


Figure 6. Visual comparison of the reconstructed 3D surfaces from the point clouds (shown in 1) generated by different methods via the ball pivoting algorithm.

point clouds by different methods. Besides, we also reconstructed 3D meshes from them via the ball pivoting[†] algorithm [3]. As illustrated in Fig. 1, WarpingGAN generates shapes with finer global shapes and local details than the other methods. Particularly, the points of the generated data by our WarpingGAN are uniformly distributed, thus avoiding "holes" which appear in the results by compared methods. Besides, WarpingGAN does not generate outlier points. From Fig. 6, it can be observed that the reconstructed 3D meshes from the point clouds generated by our WarpingGAN have much better quality than those by other methods, which also validates the higher quality of the point clouds generated by our WarpingGAN. See *Supplementary Material* for more visual results.

Efficiency comparison. We also compared the training time, inference time and parameter sizes of different methods. We only reported the inference time of ShapeGF and DPM. As listed in Table 2, during interface, WarpingGAN achieves the most compact model size and $4 \sim 300$ faster than state-of-the-art methods, owing to the exclusion of the time-consuming kNN and progressive design. Although the stitching loss of WarpingGAN requires calculating kNN for subsets with a small size during training, which slightly increases the training time, the training process of WarpingGAN is still much more efficient than PDGN and SP-GAN.

Table 2. Comparison of the training time, inference time and parameter size of different methods. The training time refers to the average time of one iteration.

Method	Training (s)	Inference (s)	Params (M)
TreeGAN [20]	**0.04**	0.014	40.69
PDGN [10]	0.63	0.077	12.71
SP-GAN [15]	0.29	0.031	0.59
ShapeGF [5]	-	2.660	4.40
DPM [17]	-	0.641	1.58
WarpingGAN	0.08	**0.008**	**0.58**

Feature comparison. We also employed t-SNE [23] to visually compare the *Airplane* shapes generated by different methods in the feature space. Specifically, we adopted the

[†]For a fair comparison, the same hyper-parameters were applied to the generated point clouds by different methods.

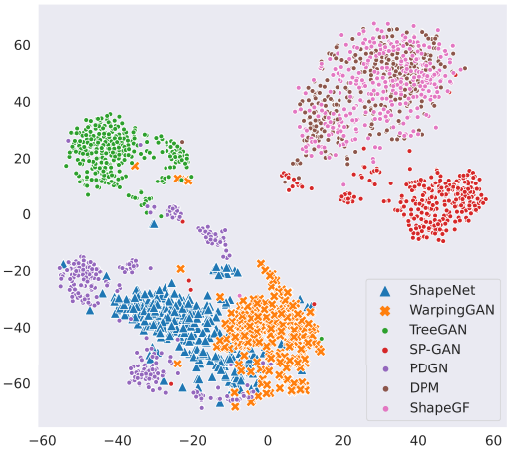
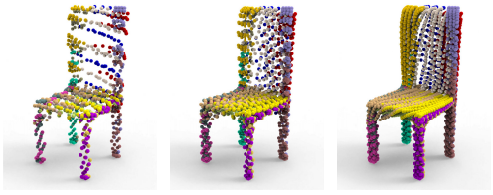


Figure 7. Visual illustration of the t-SNE feature clustering of the real point clouds of ShapeNet and generated point clouds by different methods.

pre-trained DGCNN with ModelNet40 to extract features of generated point clouds by different methods and the corresponding *Airplane* category of ShapeNet. As shown in Fig. 7, compared with other methods, the feature distribution of WarpingGAN is *more compact* and *closer* to that of the real point cloud set, indicating that the generated point clouds by our WarpingGAN are more realistic.



(a) 1024 points (b) 2048 points (c) 4096 points

Figure 8. Visual illustration of the flexibility of our WarpingGAN in generating point clouds with various number of points after one-time training. Points with the same color correspond to the same prior.

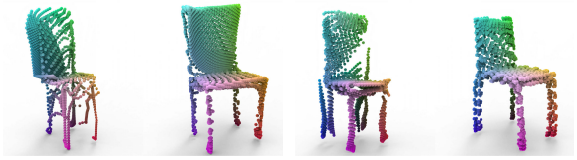
Flexibility illustration. To demonstrate the flexibility of our WarpingGAN, we fixed the WarpingGAN trained with $M = 16$ and $N = 2,048$ and modified the size of the 3D priors in order to generate point clouds with $N = 1,024, 2048$ and $4,096$ from an identical latent code. As shown in Fig. 8, the three generated point clouds correspond to the same shape, the warping manners of corresponded priors in different point clouds are the same, and the point cloud quality potentially improves with the number of points increasing. However, the compared methods that directly generate points from the latent code cannot achieve such a flexibility.

4.3. Ablation Study

The effectiveness of the code enhancement is demonstrated via the quantitatively and qualitatively in Table 3 and Fig. 9, respectively. As listed in Exps. #1 and #2 of Table 3, it can be seen that this module can effectively improve the generated point cloud quality in terms of all metrics. Fig. 9 (a) visually demonstrates that the quality of the generated point cloud degrades dramatically without using this module. Besides, We also validated the advantage of integrating the global code $\widetilde{\mathbf{z}}$ into the local code $\mathbf{z}^j$ in Fig. 9 (b). Moreover, we also investigated the distribution of the enhanced latent code to understand this module better. As shown in Fig. 10, it can be seen that this module is able to transform the initial Gaussian distribution (Fig. 10 (a)) to a distribution (Fig. 10 (c)) which is very close to the distribution of semantic features extracted from real datasets by using FoldingNet (Fig. 10 (b)).

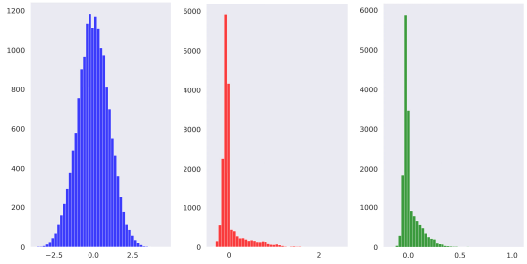
Table 3. Ablation studies conducted on the *Chair* category. Exps. #1 - #8 are the results of our WarpingGAN with various settings. The result of our final model is highlighted in red. Exps. #9 and #10 are the results of FoldingNet-based and AtlasNet-based GANs, respectively. "-" means the stitching loss is not applicable, "U" and "NU" mean uniform and non-uniform priors, respectively.

Exp. Num.	Code Enhance.	Prior Type	Prior Num.	Stitching Loss	MMD↓	COV↑	Uniform↓
1	✗	3D+U	1	-	13.4	25.00	1.25
2	✓	3D+U	1	-	11.0	43.75	1.47
3	✓	3D+U	16	✗	9.9	45.00	0.43
4	✓	2D+U	16	✓	10.0	46.25	0.32
5	✓	3D+NU	16	✓	10.2	50.00	0.49
6	✓	3D+U	4	✓	9.6	51.25	0.78
7	✓	3D+U	16	✓	8.7	53.75	0.29
8	✓	3D+U	64	✓	8.3	46.25	0.34
9	✓	2D+U	1	-	14.4	27.50	1.57
10	✓	2D+U	16	✗	10.5	36.25	0.90



(a) wo/w code enhancement (b) wo/w global shape code

Figure 9. Visual illustration of the effectiveness of (a) the code enhancement module and (b) the global shape code, where wo and w denote "without" and "with", respectively.

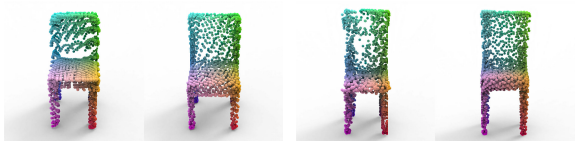


(a) latent code (b) FoldingNet θ (c) enhanced code

Figure 10. Visual comparison of the distribution of (a) the input latent code, (b) the latent semantic features of FoldingNet, and (c) the enhanced code of WarpingGAN ($M = 1$).

Figure 11. Visual comparison of our WarpingGAN trained (top row) without and (bottom row) with the stitching loss.

The effectiveness of the stitching loss is quantitatively validated by comparing the results of Exps. #3 and #7 listed in Table 3, where it can be seen that the results of all metrics improve when adopting the loss. Besides, as shown in Fig. 11, the point clouds generated by WarpingGAN trained without the stitching loss suffer from significant gaps between different partitions, while WarpingGAN trained with the stitching loss can greatly alleviate the gaps and increase the visual quality.



(a) 2D vs. 3D priors (b) non-uni vs. uni priors

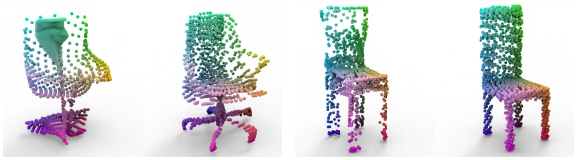
Figure 12. Visual comparison of our WarpingGAN equipped with (a) 2D priors and 3D priors, and (b) 3D non-uniform priors and 3D uniform priors.

The effect of different prior settings. First, we substituted the 3D uniform priors with 2D uniform priors, while keeping the remaining settings unchanged. By comparing the results of Exps. #4 and #7 in Table 3, we can conclude the advantage of 3D priors over 2D priors. From Fig. 12(a), it can be seen that the generated point cloud by 3D uniform priors retain local details better. Besides, to demonstrate the necessity of the uniformity of the prior, we replaced the 3D uniform priors with 3D non-uniform priors and kept the remaining settings unchanged. As shown in Fig. 12 (b), the non-uniform priors cannot lead to uniformly distributed point clouds, which is consistent with the quantitative results in Exps. #5 and #7 of Table 3.

The effect of the number of 3D priors. We trained three WarpingGAN models with 4, 16 and 64 priors, respectively. The quantitative comparisons are listed in Exps. #6, #7, and #8 of Table 3, where it can be seen that WarpingGAN with 16 priors produces the generally best performance, which is consistent with the visual comparison shown in Fig. 13. The reason is that a limited number of priors cannot fit 3D shapes well, while too many priors make it hard to optimize the stitching loss. In this paper, we set $M = 16$.

(a) 4 priors (b) 16 priors (c) 64 priors

Figure 13. Visual comparison of our WarpingGAN equipped with different numbers of 3D priors. Note that the total size of priors under different settings are equal for generating point clouds each with 2,048 points.



(a) Folding-based vs. WarpingGAN (b) Atlas-based vs. WarpingGAN

Figure 14. Visual comparison of WarpingGAN with (a) FoldingNet-based GAN and (b) AtlasNet-based GAN.

Comparison with FoldingNet-based and AtlasNet-based GANs. In this experiment, we retained the code enhancement module and substituted the prior warping module of the proposed generator with the decoders of FoldingNet and AtlasNet. Exps. #9 and #10 of Table 3 provide the quantitative performance of these two baselines, which rev eal the limited performance of the direct extensions of these auto-encoder frameworks. Besides, as shown in Fig. 14, FoldingNet-based GAN fails to generate a chair with a complex structure correctly and AtlasNet-based GAN loses much local details even when generating a chair with a simple structure, while our WarpingGAN can generate chairs with much better quality.

5. Conclusion

We presented WarpingGAN, a novel point cloud generation framework that is capable of generating high-quality point clouds in an effective and efficient manner. In contrast to existing approaches that usually produce point clouds by learning the direct mapping between the random latent codes and 3D shapes, we designed WarpingGAN by investigating a unified local-warping mechanism, in which multiple pre-defined 3D priors uniformly distributed in the 3D Euclidean space are *conditionally* warped to various local regions of a shape. Meanwhile, by examining the principle of the discriminator, we customized a stitching loss to eliminate the gaps between different regions. Such a new mechanism makes WarpingGAN compact, efficient, and flexible. We conducted extensive experiments to demonstrate the significant advantages of WarpingGAN over state-of-the-art methods in terms of quantitative metrics, visual quality, and efficiency.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, pages 40–49, 2018. 1, 2

[2] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. In *CVPR*, pages 4716–4725, 2020. 3

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio 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] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2018. 1

[5] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, pages 364–381, 2020. 1, 2, 3, 5, 6

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 27, 2014. 1

[7] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, pages 216–224, 2018. 3

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 5

[9] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans. In *CVPR*, pages 3093–3102, 2020. 1

[10] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *ECCV*, pages 397–413, 2020. 1, 2, 4, 5, 6

[11] Hamid Izadinia and Steven M Seitz. Scene recomposition by learning-based icp. In *CVPR*, pages 930–939, 2020. 1

[12] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. *NeurIPS*, pages 16388–16397, 2020. 1

[13] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, pages 9601–9611, 2019. 1

[14] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *ICCV*, pages 7203–7212, 2019. 5

[15] Ruihui Li, Xianzhi Li, Ke-Hei Hui, and Chi-Wing Fu. SP-GAN:sphere-guided 3d shape generation and manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 40(4), 2021. 1, 2, 5, 6

[16] Daquan Liu, Chengjiang Long, Hongpan Zhang, Hanning Yu, Xinzhi Dong, and Chunxia Xiao. Arshadowgan: Shadow generative adversarial network for augmented reality in single light scenes. In *CVPR*, pages 8139–8148, 2020. 1

[17] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, pages 2837–2845, 2021. 1, 2, 3, 5, 6

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 3, 4

[19] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 2

[20] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV*, pages 3859–3868, 2019. 1, 2, 4, 5, 6

[21] Xin Suo, Yuheng Jiang, Pei Lin, Yingliang Zhang, Minye Wu, Kaiwen Guo, and Lan Xu. Neuralhumanfvv: Real-time neural volumetric human performance rendering using rgb cameras. In *CVPR*, pages 6226–6237, 2021. 1

[22] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. In *ICLR*, 2018. 1, 2

[23] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. 6

[24] He Wang, Zetian Jiang, Li Yi, Kaichun Mo, Hao Su, and Leonidas J Guibas. Rethinking sampling in 3d point cloud generative adversarial networks. *CVPR Workshop "Learning to generate 3D Shapes and Scenes"*, 2021. 2, 3

[25] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics*, 38(5):1–12, 2019. 2

[26] Cheng Wen, Baosheng Yu, and Dacheng Tao. Learning progressive point embeddings for 3d point cloud generation. In *CVPR*, pages 10266–10275, 2021. 1

[27] Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. Photo wake-up: 3d character animation from a single photo. In *CVPR*, pages 5908–5917, 2019. 1

[28] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, pages 4541–4550, 2019. 1, 2, 3, 5

[29] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, pages 206–215, 2018. 3

[30] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, pages 5826–5835, 2021. 5

[31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 1