PointInverter: Point Cloud Reconstruction and Editing via a Generative Model with Shape Priors

Jaeyeon Kim [1] Binh-Son Hua [2] Duc Thanh Nguyen [3] Sai-Kit Yeung [1]

[1] Hong Kong University of Science and Technology [2] VinAI Research [3] Deakin University

Abstract

In this paper, we propose a new method for mapping a 3D point cloud to the latent space of a 3D generative adversarial network. Our generative model for 3D point clouds is based on SP-GAN, a state-of-the-art sphere-guided 3D point cloud generator. We derive an efficient way to encode an input 3D point cloud to the latent space of the SP-GAN. Our point cloud encoder can resolve the point ordering issue during inversion, and thus can determine the correspondences between points in the generated 3D point cloud and those in the canonical sphere used by the generator. We show that our method outperforms previous GAN inversion methods for 3D point clouds, achieving state-of-the-art results both quantitatively and qualitatively. Our code is available at `https://github.com/hkust-vgd/point_inverter`.

1. Introduction

Deep learning for 3D point clouds has been rapidly progressing in the past five years. A majority of research have been dedicated to design efficient ways for neural networks to process 3D point clouds [26, 27, 25, 35, 41]. These developments have empowered the learning ability of neural networks for a wide range of downstream tasks such as object classification, semantic segmentation, object detection on both synthetic and real-world data [36, 6, 15, 8, 33].

Compared to point cloud analysis, research on generative modeling of 3D point clouds is more scarce. A notable work, namely SP-GAN [24] demonstrates remarkable results in generating 3D point clouds from a shape prior using generative adversarial neural networks. Despite this important outcome, the generative model in the SP-GAN is unconditional, meaning that it only allows sampling of novel point clouds while leaving manipulating and editing of existing point clouds unexplored.

A common approach to perform data editing with a generative adversarial network is to follow a two-stage principle: *invert first, edit later*. This principle has been well demonstrated in the 2D domain, where state-of-the-art GAN inversion methods [29, 2, 3, 9] are used to map a real image into the latent space of the StyleGAN model [20] by using optimization-based or encoder-based techniques. Once the mapping is done, the reconstructed latent code can be manipulated to make effects on the real image (e.g., changing attributes of the image). Motivated by this principle, we develop a new approach for inverting a 3D point cloud so that the point cloud's latent code can be faithfully used to reconstruct a new point cloud via the SP-GAN model.

A key challenge in 3D deep learning for point cloud data is to design a learning mechanism that is invariant to point ordering. In point cloud analysis, this capability can be achieved by aggregating point features using a symmetric function [26], sorting the points using a grid before performing convolution operations [16]. However, point ordering has been largely ignored from point-cloud GAN inversion research [40]. The inversion is achieved by comparing reconstructed point clouds with their original point clouds using metrics such as Chamfer discrepancy or earth-mover distance, which is invariant to point ordering. However, this manner makes the correspondences between the reconstructed and original point clouds to no longer be maintained, limiting the ability of downstream applications, e.g., point cloud matching and registration.

We address this problem by resolving point orders during inversion. Our method can be summarised as follows. Given a pretrained SP-GAN model, we design a point-based encoder that takes a 3D point cloud as input, then extracts two style vectors from the input. These style vectors, together with the point cloud's features are fed into the pretrained SP-GAN. At the first stage, we reconstruct a global latent code via jointly training the encoder and refining the generator. At the second stage, we use the global latent code to resolve point ordering. We then extract local latent codes from the global latent code at the last stage. Our method achieves state-of-the-art performance in reconstructing 3D point clouds, both quantitatively and qualitatively. For applications, we show that latent codes can be applied to determine correspondences between input point clouds and

reconstructed results, enabling point cloud manipulation and editing ability.

In summary, our contributions are as follows:

- A point cloud encoder that maps a 3D point cloud to the latent space of the SP-GAN, a state-of-the-art sphere-guided point cloud generator;
- A strategy to resolve point ordering during inversion by leveraging global latent codes generated by the point cloud encoder, thus maintaining point correspondences in the reconstruction and enabling shape editing;
- A global to local latent code refinement technique that better preserves both geometric details and correspondences in the reconstruction;
- State-of-the-art results in point cloud GAN inversion, illustrated via point cloud reconstruction and editing.

The remainder of this paper is organized as follows. We review the related work in Section 2. In Section 3, we present the SP-GAN [24], which is used as a decoder reconstructing points clouds in our proposed GAN inversion method. Our method is then described in Section 4. Applications and experimental results are reported in Section 5. Section 6 concludes our work and provides remarks.

2. Related Work

3D deep learning. Deep learning in the 3D domain has progressed tremendously in the past few years, with 3D point cloud deep learning as one of the main research directions due to the universality of the point cloud representation. Several point cloud neural networks are designed for analysis tasks such as object classification, semantic segmentation, and object detection. Notable works include PointNet [26] as the pioneering method that first learns per-point features and then pools them into a global feature vector, PointNet++ [27] and DGCNN [35] as popular methods to capture local point features, and a long list of other methods for building convolutions on point clouds [25, 38, 41] and for handling downstream tasks such as semantic segmentation [17, 34, 12, 23, 14, 37]. In this work, we leverage DGCNN as a point cloud encoder in our method.

Generative modeling for 3D point clouds. Despite the rapid development of 3D deep learning [13], generative models for 3D point clouds are relatively scarce. To unconditionally generate a 3D point cloud, an early attempt is to train an autoencoder on 3D point clouds and then train a GAN in the latent space learned by the autoencoder [1]. Subsequent methods include the use of autoregressive model [32], flow-based generation [39, 21, 22, 5], and generative adversarial neural networks [28, 18, 24] with tree-based structures [31, 11]. Among those GAN-based methods, recently, SP-GAN [24] learns to generate a point cloud by transforming a prior geometry such as a canonical sphere to a 3D shape. There also exists a group of works [10, 4] that learn to predict point clouds from conditional data such as images. Inspired by the great success of generative adversarial neural networks, in this work, we focus on point clouds generated by unconditional GAN models and how these models can be leveraged to manipulate and edit 3D point cloud data by using GAN inversion pipeline.

GAN inversion. The basic principle of GAN inversion is to faithfully map an input shape to the latent space of a GAN model. Manipulation and editing can then be done in the GAN latent space, depending on the downstream task. However, reconstructing the input using a pretrained GAN is challenging. In the image domain, StyleGAN [19] has made a milestone. Typical inversion methods can be categorized as optimization-based methods or learning-based methods. Optimization-based methods include optimizing a simple projection function [19] or fine-tuning a generator for each image [30]. These methods provide high-quality reconstruction but also require heavy computation. In contrast to optimization-based methods, learning-based methods allow lighter and faster reconstruction by training an encoder to directly output the latent code [29, 2], or by training a hypernetwork for defining a refined generator to improve reconstruction quality [3, 9].

In the 3D domain, several attempts have been made for shape reconstruction using the latent space of a generative model for 3D point clouds. For example, Zhang et al. [40] proposed to use GAN inversion for 3D point clouds to address the shape completion task [7]. Their generative model was built upon the tree-GAN developed in [31], where the generator was a graph convolutional network applied to a tree data structure. Our work differs from [40]. Specifically, we focus not only on the quality of the reconstruction but also point ordering. In our work, we adopt the SP-GAN developed in [24] as our generative model. This generator leverages a geometric prior in the form of a canonical sphere to guide the shape generation process, which results in better shape quality and controllability.

3. Background

SP-GAN proposed in [24] is a neural network architecture aiming to reconstruct point clouds P from a spherical structure. Like standard unconditional GANs, the SP-GAN also includes an unconditional generator G and a discriminator D. The generator G takes input as a unit sphere and generates a point cloud P. The discriminator D takes input as a point cloud either generated by the generator G or sampled from training data, and classifies the input point cloud and all of its points into two classes: real vs fake. Both the generator G and discriminator D are trained simultaneously

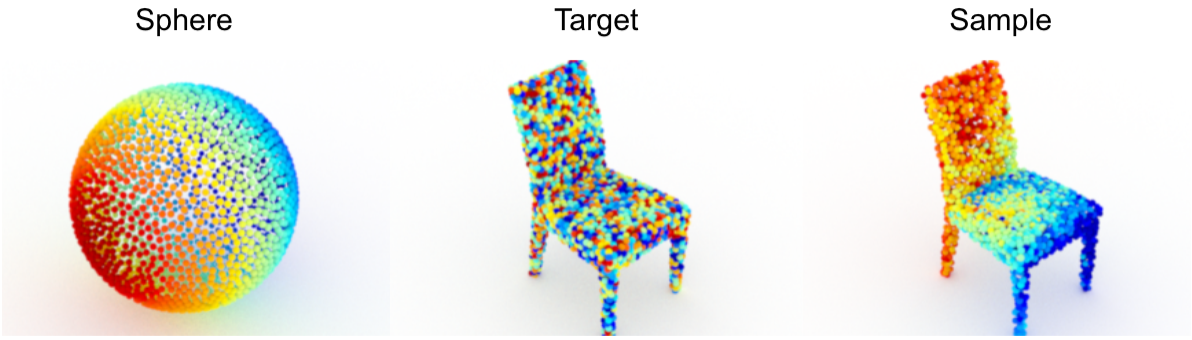

Figure 1. An advantage of point cloud GAN inversion using a generator with a shape prior (SP-GAN [24]) is that we can reconstruct both points and their correspondences to the shape prior. This figure illustrates the dense correspondence between the guided sphere, the target point cloud for inversion, and the reconstructed point cloud by our method. As can be seen, the point correspondences are random on the target point cloud before performing the inversion. Our reconstructed point cloud is geometrically similar to the target while having smooth correspondences.

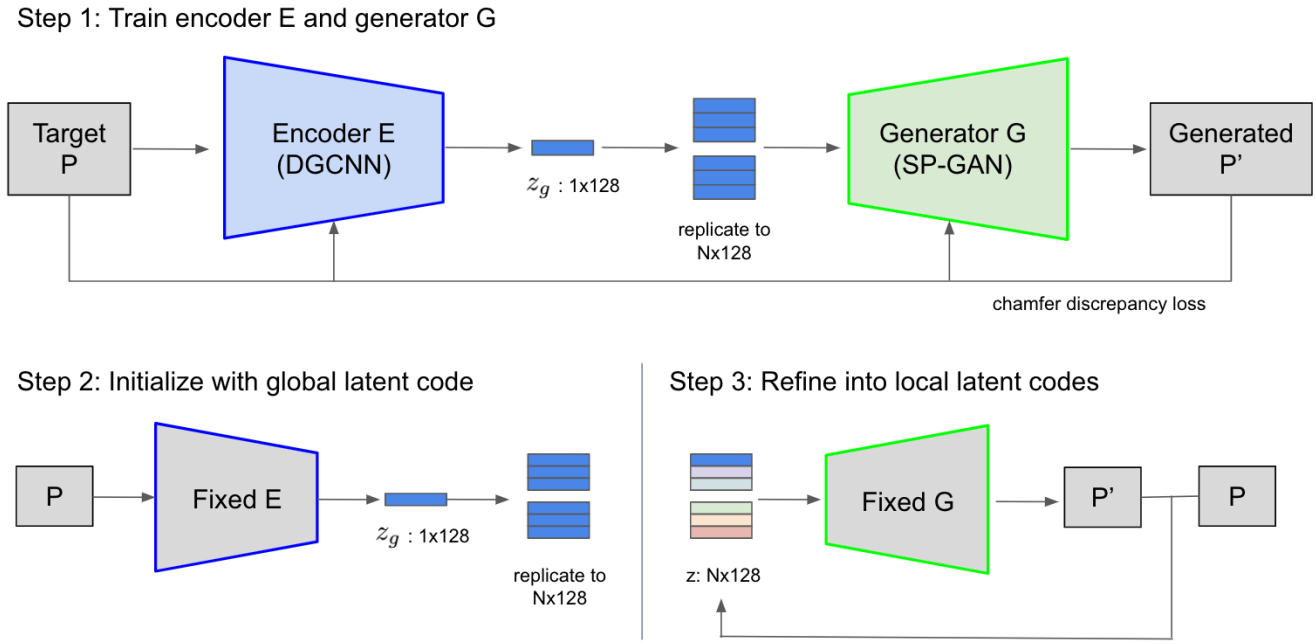

Figure 2. Our GAN inversion for point clouds. The encoder is built upon the pre-trained DGCNN and the generator is based on the pre-trained SP-GAN. In Step 1, the encoder E and generator G are trained to map a global latent code z_g to a target point cloud P. In Step 2, following SP-GAN, the global latent code z_g is replicated by the number of points N to initialize local latent codes. In Step 3, we refine the local latent codes by iteratively performing an optimization task.

in an alternating manner. We present the training and testing of the SP-GAN in detail below.

Let S be a unit sphere including N points; these points are evenly distributed on S. The spatial locations (i.e., 3D coordinates) of these N points are used to define a global prior in $\mathbb{R}^{N\times 3}$. Each point is also associated with a local prior, which is a random noise vector in $\mathbb{R}^d \sim \mathcal{N}(0,1)$, following a standard normal distribution. The sphere S is finally encoded into a prior latent code $z_S \in \mathbb{R}^{N\times(3+d)}$ by concatenating 3D coordinates and local prior for every point in S.

The prior latent code z_S is then used to train the generator G for constructing point clouds P, each of which also contains N points. While training the generator G, real point clouds P' are sampled from a training dataset and used to train the discriminator D.

The SP-GAN can be trained end-to-end by simultaneously minimizing a discriminative loss $\mathcal{L}_D$ and a generative

loss $\mathcal{L}_G$. The discriminative loss $\mathcal{L}_D$ includes two sub-losses: $\mathcal{L}_D^{\text{point cloud}}$ for an entire point cloud and $\mathcal{L}_D^{\text{point}}$ for individual points. These losses are defined as follows.

$$\mathcal{L}_D = \mathcal{L}_D^{\text{point cloud}} + \lambda \mathcal{L}_D^{\text{point}}, \tag{1}$$

$$\mathcal{L}_D^{\text{point cloud}} = \frac{1}{2}[(D(P))^2 + (D(P') - 1)^2], \tag{2}$$

$$\mathcal{L}_D^{\text{point}} = \frac{1}{2N}\sum_{i=1}^{N}[(D(p_i))^2 + (D(p_i') - 1)^2], \tag{3}$$

where λ is a user-defined parameter to balance the losses computed on an entire point cloud and on individual points; $D(P)$ and $D(P')$ are the confidence scores returned by the discriminator D when applied on generated point clouds P and sampled point clouds P', respectively; p_i and p_i' are points on P and P'.

The generative loss $\mathcal{L}_G$ is defined as,

$$\mathcal{L}_G = \frac{1}{2}(D(P) - 1)^2 + \beta\frac{1}{2N}\sum_{i=1}^{N}(D(p_i) - 1)^2, \tag{4}$$

where β is a user-defined parameter.

Once the generator G and discriminator D have been trained, the inference can be done by passing a prior latent code z_S to G to generate a point cloud $P = G(z_S)$. Figure 1 illustrates generated results of the SP-GAN.

Note that the SP-GAN can not only generate point clouds but also make point-wise correspondences between the input sphere and its generated point cloud. This enables shape editing via latent code manipulation.

4. Our Method

4.1. Overview

Given a point cloud $P \in \mathbb{R}^{N\times 3}$ where N is the number of points, we aim to learn a mapping function that maps P to the latent space of the SP-GAN [24]. Such a mapping is expected to maintain high-quality reconstruction while enabling point-wise correspondences between the input point cloud and its reconstructed point cloud.

Our GAN inversion framework consists of a point cloud encoder E, which learns a global latent code for an input 3D point cloud, and a generator G, which is the generator of the SP-GAN [24]. There are three steps in the inversion as shown in Figure 2: (1) training the encoder E and the generator G (G can also be fine-tuned); (2) resolving point ordering; and (3) refining the global latent code extracted by the encoder E into a set of local latent codes. Let us detail each step in the following sections.

4.2. Step 1: Global latent code

Our goal in this step is to train the encoder E and the generator G to learn a global latent code for each input point cloud P. The global latent code should be faithfully mapped to its target point cloud. This global latent code should also be invariant to each point in the target point cloud. Conceptually, the global latent code of a point cloud is similar to the global feature vector aggregated from pooling all per-point features in the point cloud as used in point cloud networks, e.g., PointNet [26].

Suppose that the generator G is given and fixed. To train the encoder E, we solve the following optimization problem:

$$\theta_E^* = \arg\min_{\theta_E} \sum_P L\left(G\left(E\left(P;\theta_E\right)\right), P\right), \tag{5}$$

where L denotes the distance between a target point cloud and a generated point cloud. Here the generator G is based on the pretrained SP-GAN generator [24].

To improve the reconstruction quality, we also update the parameters of G during the inversion. This can be achieved by training both E and G simultaneously, i.e., solving the following problem:

$$\theta_G^*, \theta_E^* = \arg\min_{\theta_G,\theta_E} \sum_P L\left(G\left(E\left(P;\theta_E\right);\theta_G\right), P\right)). \tag{6}$$

The parameter θ_G and θ_E are updated alternatively by using the gradient descent optimization technique.

Once the training of E and G is done, the global latent code z_g can be determined as

$$z_g = E(P;\theta_E^*). \tag{7}$$

The global latent code z_g calculated in Eq. (7) is also referred to as the prior latent code in the SP-GAN. This code is then used to generate local latent codes capturing details of the generated point cloud.

Encoder architecture. There are some design choices for the architecture of the encoder E. Similar to image-based GAN inversion [42], we can use a point cloud discriminator for the encoder. The output of the encoder is the last feature layer of the discriminator. Another option is to use a pre-trained point cloud network such as DGCNN [35], where we aggregate all local point features into a global feature vector. We empirically found that DGCNN yields better results, and hence choosing it for the encoder E.

Loss functions. Our loss function aims to enforce the learning process towards the reconstruction quality, i.e., both input and generated point clouds should have similar geometric structure, and point density. To model such similarity, we utilize the Chamfer discrepancy (CD) for our loss. In

particular, we define:

$$L_{CD}(P, P') = \max\left\{ \frac{1}{|P|} \sum_{\mathbf{p}_i \in P} \min_{\mathbf{p}'_j \in P'} \|\mathbf{p}_i - \mathbf{p}'_j\|_2, \frac{1}{|P'|} \sum_{\mathbf{p}'_j \in P'} \min_{\mathbf{p}_i \in P} \|\mathbf{p}'_j - \mathbf{p}_i\|_2 \right\}. \quad (8)$$

where P is the input point cloud and P' is the point cloud generated by the generator G.

4.3. Step 2: Point ordering

We found that the inversion with the global latent vector z_g is limited in reconstructing geometric details of the target point cloud. Despite such, a strong advantage of using the global latent code is that it is invariant to point ordering in the target point cloud. This leads to the effect that the point-wise correspondences between the generated point cloud and the input point cloud is preserved as in the original SP-GAN [24].

To make the reconstruction useful for downstream tasks, it is necessary to improve the reconstruction accuracy. A naive approach is to let the encoder learn how to output local latent codes that vary for each point in the point cloud. Unfortunately, a caveat of doing so is that the ordering of the latent codes become dependent on the point ordering of the target point cloud, which could be random. This destroys the point-wise correspondences between the generated point cloud and the input shape. We demonstrate this issue in Figure 3.

Additionally, predicting the local latent codes directly might make the encoder and generator overfitted, i.e., the latent codes can just contain the target 3D coordinates and the generator simply passes these coordinates as its output. This yields very accurate reconstruction, but the local latent codes are useless for downstream tasks.

To improve the reconstruction quality while addressing the point ordering problem, we constrain the global and local latent codes in an engaging manner. Recall that the global latent code is produced by the encoder E and is invariant to initial point ordering in a target point cloud. However, the global latent code is shown to have poor reconstruction ability. In the SP-GAN, shape prior already includes point orders, and the generator operates point-wise. However, the point orders in the shape prior can be different from those in the target shape in inversion. Therefore, to ensure consistent point ordering, we replicate the global latent code predicted by the encoder trained in Step 1 to build initial local latent codes of size $N \times (3 + d)$. This initial local latent codes are then refined in Step 3.

4.4. Step 3: Local latent codes

After resolving point ordering, we are now ready to refine the global latent code into local latent codes. In this step, we keep all the parameters of the generator G fixed, and update each entry in the latent code accordingly. The optimization hence becomes finding:

$$z^* = \arg\min_z \sum_P L\left(G\left(z; \theta_G\right), P\right), \quad (9)$$

where z is initialized by replicating the global latent code z_g. The output of this optimization is the final local latent codes of our inversion.

5. Experimental Results

5.1. Experiment Setup

Datasets. We conducted experiments for our GAN inversion method and existing ones on the ShapeNet dataset in [6]. We trained our network architecture on four object categories including chair, airplane, car, and lamp, separately. The test set was made of 10% of the total dataset. Particularly, the chair category consists of 6,101 models for training and 677 models for testing. Beyond man-made objects in the ShapeNet, we also tested our method on the Animal dataset in [43]. We uniformly sampled 2,048 points on object meshes to create point clouds. Like SP-GAN, we trained a single model on a combined category of all four-leg animal data such as dogs, big cats, hippos, and horses.

Implementation details. We adopted the SP-GAN in [24] as our pre-trained generator. Our encoder was built upon the pre-trained DGCNN in [35]. We concatenated features from four layers in the DGCNN to construct the final layer of our encoder. We used 2,000 iterations for training the encoder in Step 1, and 2,000 iterations for optimizing the latent codes in Step 3 in our method. We used the Adam optimiser with a learning rate of 0.01.

5.2. Evaluation of shape inversion

We evaluated our method based on its reproduction quality. The reproduction quality was measured via the Chamfer discrepancy between a given target point cloud and the corresponding generated point cloud.

Quantitative results. We first compared our method with existing shape inversion methods, e.g., the methods by Achlioptas et al. [1] and by Zhang et al. [40]. The method by Achlioptas et al. [1] uses latent codes in the latent space of an autoencoder for point cloud generation. Here we adopted their autoencoder for comparison of the methods in reconstruction. The method by Zhang et al. [40] is an optimization-based inversion applied to tree-GAN [31].

We report the results of this experiment in Table 1. As shown in the results, our method achieves the smallest average and per-class Chamfer discrepancy. The reconstructed

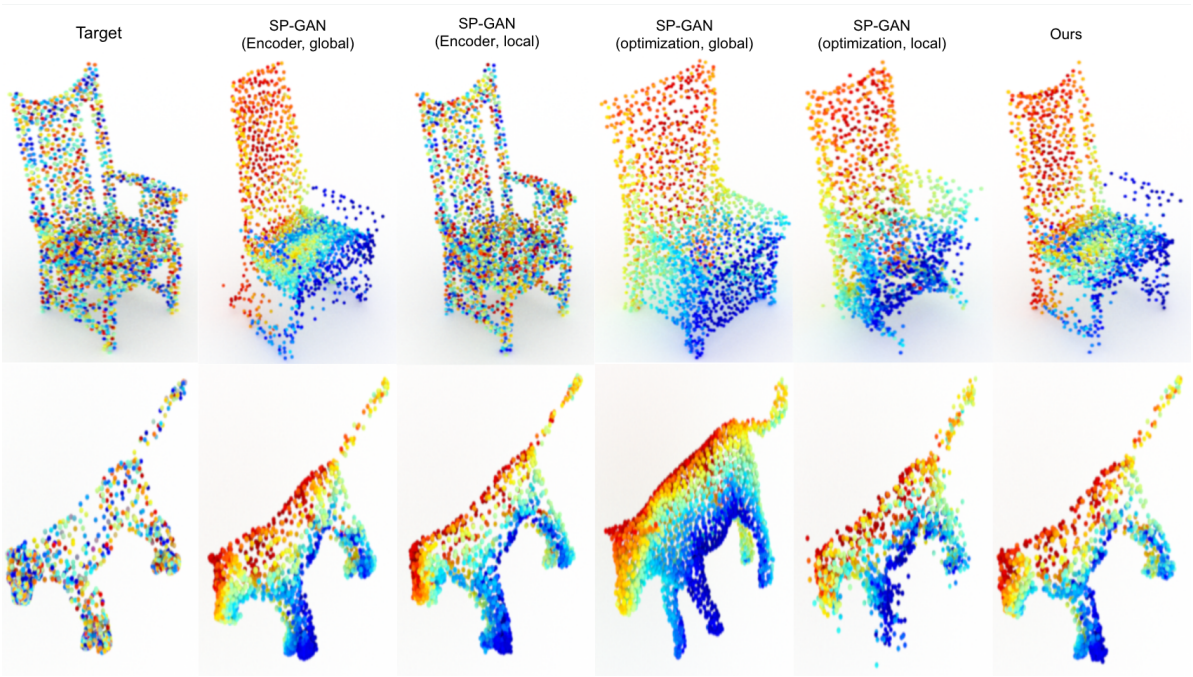

Figure 3. Design choices of the latent codes. Global latent codes are invariant to point orders and thus well preserve point correspondences given in the shape prior, but produce less faithful reconstruction details. Learning-based methods using encoders to learn local latent codes tend to over-fit, resulting accurate geometry reconstruction but wrong correspondences. Optimization-based methods however incur inaccurate reconstruction. Our inversion achieves accurate geometry while preserving point correspondences.

Table 1. Comparison of our method with existing works in point cloud reconstruction.

	avg.	chair	airplane	car	lamp
Achlioptas et al. [1]	3.46×10^{-3}	3.61×10^{-3}	1.15×10^{-3}	1.14×10^{-3}	7.95×10^{-3}
Zhang et al. [40]	2.50×10^{-3}	2.09×10^{-3}	3.59×10^{-3}	1.95×10^{-3}	2.38×10^{-3}
Ours	$\mathbf{0.54 \times 10^{-3}}$	$\mathbf{0.66 \times 10^{-3}}$	$\mathbf{0.49 \times 10^{-3}}$	$\mathbf{0.55 \times 10^{-3}}$	$\mathbf{0.49 \times 10^{-3}}$

point cloud by Achlioptas et al. [1] is not editable. Compared with the results of Zhang et al. [40], our reconstruction results are also more accurate by a large margin.

Qualitative results. We visually assess the quality of reconstructed point clouds by our method in Figure 4. As can be seen, our inversion algorithm can reconstruct target point clouds reasonably while preserving shape details better than other methods. For example, patterns on the back of chairs can be well recognized in our reconstructions.

5.3. Ablation studies

Global vs local latent codes. To further understand the effectiveness of our method, we provide an ablation study in Table 2 and Figure 3. Specifically, we built different baselines including learning-based (i.e., using learned encoders) and optimization-based baselines. For each baseline, we output global or local latent codes, which are then used by the SP-GAN for point cloud reconstruction. Table 2 shows that the learning-based baselines are generally better than the optimization-based ones. Additionally, optimizing local latent codes increases the precision of reconstruction. However, it is worth noting that this could lead to overfitting, as shown in the case of using encoders to output local codes. In this case, reconstruction achieves the best accuracy but point correspondences are significantly corrupted in reconstructed shapes (see Figure 3), making subsequent shape manipulation impossible. Our method instead has slightly lower reconstruction accuracy compared with the overfitting case, but it can keep the correspondences intact. We found that dense correspondence problem is not shown in the animal dataset. This is probably because the number of static shapes in the animal dataset is small while the shapes less diverge. The encoder can avoid overfitting, but reconstruction results are not as good as those from the ShapeNet. Note that our method guarantees the reconstruction performance and dense correspondence regardless of the datasets.

Table 2 also shows the results on the Animal dataset. As

Table 2. Ablation studies.

	avg.	chair	airplane	car	lamp	animal
Learning-based, global	2.23×10^{-3}	2.11×10^{-3}	0.94×10^{-3}	1.87×10^{-3}	4.03×10^{-3}	2.23×10^{-3}
Learning-based, local	0.62×10^{-3}	0.59×10^{-3}	0.35×10^{-3}	0.62×10^{-3}	0.31×10^{-3}	1.27×10^{-3}
Optimization-based, global	45.5×10^{-3}	13.5×10^{-3}	73.1×10^{-3}	94.9×10^{-3}	17.6×10^{-3}	28.4×10^{-3}
Optimization-based, local	21.2×10^{-3}	2.60×10^{-3}	23.4×10^{-3}	48.6×10^{-3}	2.77×10^{-3}	7.48×10^{-3}
Ours	0.63×10^{-3}	0.66×10^{-3}	0.49×10^{-3}	$\mathbf{0.55 \times 10^{-3}}$	0.49×10^{-3}	$\mathbf{0.98 \times 10^{-3}}$

Figure 4. Comparison of shape inversion examples. Our method reproduces the target more faithfully, e.g., see the back of the chairs.

Table 3. Comparison of generators in reconstruction.

	avg.
tree-GAN [31] (encoder, global)	2.64×10^{-3}
SP-GAN (encoder, global)	2.24×10^{-3}
Ours	$\mathbf{0.54 \times 10^{-3}}$

Table 4. Comparison of encoders in inversion on the chair class.

DGCNN [35]	SP-GAN's discriminator [24]
5.43×10^{-4}	6.92×10^{-4}

can be seen, our method outperforms all the baselines. Moreover, the results of all the methods on this dataset match the performance trend reported on the ShapeNet. A visualization of the results on the Animal dataset is in Figure 3.

Generator. We also experimented with our GAN inversion with different generators. We chose tree-GAN [31] and compared it with SP-GAN [24]. Comparison results are shown in Table 3. As can be seen, SP-GAN remains more effective than tree-GAN in reconstruction, even taking a simple setting including an encoder and a global latent code. SP-GAN also has better point correspondences compared to tree-GAN.

Encoder architecture. We evaluated the design choice of our encoder by comparing the inversion accuracy of two architectures: DGCNN [35] and the discriminator in the SP-GAN [24]. Note that the discriminator in the SP-GAN also

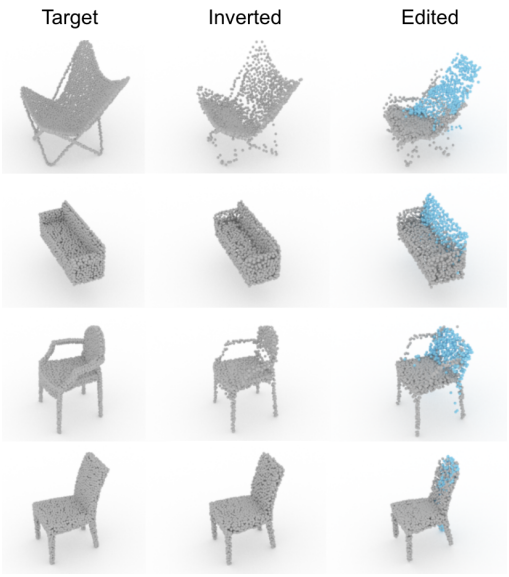

Figure 5. Shape editing of the back of the chairs. Our method allows the change of the back of a chair from a rectangular to a round shape and vice versa. The colored part is the updated geometry.

Figure 6. Shape editing of the seats and legs of the chairs. Note the changes in sizes and styles compared to the target point clouds. The colored part is the updated geometry.

makes use of convolutional layers for feature learning and thus can be used as an encoder. Results of this experiment are shown in Table 4. It can be seen that the DGCNN has a smaller Chamfer discrepancy than the SP-GAN's discriminator. Moreover, we empirically observed that the use of DGCNN makes the training of the encoder converge much faster. This consolidates our choice of the DGCNN for the encoder in our architecture.

5.4. Application: Shape editing

SP-GAN [24] is suitable for shape editing because it learns the dense correspondence implicitly. Therefore, after inversion, each (semantic) part of the generated point cloud should correspond to a region in the shape prior. As our method can maintain point-wise correspondences, we can utilize such correspondences to enable *part-aware* shape manipulation. Note that previous methods, e.g., [40], only demonstrate shape manipulation by global jittering of the latent codes. Here, we segment regions of interest in the point cloud that we would like to manipulate, then perturb the corresponding local latent codes of points in those regions to obtain new local latent codes. The final shape can be generated by passing the perturbed codes to the generator. We demonstrate this capability in Figure 5 and Figure 6.

As shown in Figure 5, we can alter the style of the back of chairs, e.g., changing a chair's back from a rectangular shape to a round shape, or making a sofa's back longer. In Figure 6, we showcase the changes in chair seat size and chair leg style.

Our shape editing is not perfect. We observed that the local latent codes might be entangled, i.e., changing a part might lead to incidental changes in other parts. Disentangling the shape latent space is left for our future work.

6. Conclusion

We proposed a new point cloud GAN inversion method that allows faithful reconstruction of 3D point clouds using a sphere-guided point cloud generator [24] while maintaining point correspondences during inversion. Our method outperforms existing GAN inversion works in terms of reconstruction quality, verified both quantitatively and qualitatively. We demonstrate the usefulness of our inversion method via a shape editing task, i.e., editing reconstructed point clouds by manipulating part-aware latent codes.

Our work is not without limitations. First, our reconstruction might still miss some small details in target point clouds. Further research in improving the reconstruction quality is thus worthwhile. Second, the latent code of the SP-GAN is not compact. Exploring GAN inversions with compact latent space would benefit a wider range of downstream applications. Finally, it is worth applying the latent codes for more downstream applications such as shape completion from real scans [7]. Extending the inversion to colored point clouds would also be an interesting research avenue.

Acknowledgment. This paper was partially supported by an internal grant from HKUST (R9429).

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative

models for 3d point clouds. *ICML*, 2018.

[2] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *ICCV*, 2021.

[3] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit H. Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing, 2021.

[4] Mohammad Samiul Arshad and William J. Beksi. A progressive conditional generative adversarial network for generating dense and colored 3D point clouds. In *International Conference on 3D Vision (3DV)*, 2020.

[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*, 2020.

[6] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[7] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired point cloud completion on real scans using adversarial training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Tan M. Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CVPR*, 2017.

[11] Rinon Gal, Amit Bermano, Hao Zhang, and Daniel Cohen-Or. MRGAN: Multi-rooted 3d shape generation with unsupervised part disentanglement. In *ICCV Workshop on Structural and Compositional Learning on 3D Data (StruCo3D)*, 2021.

[12] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[13] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[14] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[15] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, 2016.

[16] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural network. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation on point clouds. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *ECCV*, 2020.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

[21] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. SoftFlow: Probabilistic framework for normalizing flow on manifolds. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[22] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *ECCV*, 2020.

[23] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[24] 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.

[25] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[26] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[28] Sameera Ramasinghe, Salman Khan, Nick Barnes, and Stephen Gould. Spectral-GANs for high-resolution 3D point-cloud generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8169–8176, 2019.

[29] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *CVPR*, 2021.

[30] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021.

[31] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV*, 2019.

[32] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. PointGrow: Autoregressively learned point cloud

generation with self-attention. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 61–70, 2020.

[33] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision*, 2019.

[34] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[35] 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 (TOG)*, 2019.

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[37] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19, 2020.

[38] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pages 87–102, 2018.

[39] 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.

[40] Junzhe Zhang, Xinyi Chen, Zhongang Cai, Liang Pan, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. Unsupervised 3d shape completion through gan inversion. In *CVPR*, 2021.

[41] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *International Conference on Computer Vision (ICCV)*, 2019.

[42] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020.

[43] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.