Unsupervised Point Cloud Completion and Segmentation by Generative Adversarial Autoencoding Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

Most existing point cloud completion methods assume the input partial point cloud is clean, which is not the case in practice, and are generally based on supervised learning. In this paper, we present an unsupervised generative adversarial autoencoding network, named UGAAN, which completes the partial point cloud contaminated by surroundings from real scenes and cutouts the object simultaneously, only using artificial CAD models as assistance. The generator of UGAAN learns to predict the complete point clouds on real data from both the discriminator and the autoencoding process of artificial data. The latent codes from generator are also fed to discriminator which makes encoder only extract object features rather than noises. We also devise a refiner for generating better complete cloud with a segmentation module to separate the object from background. We train our UGAAN with one real scene dataset and evaluate it with the other two. Extensive experiments and visualization demonstrate our superiority, generalization and robustness. Comparisons against the previous method show that our method achieves the state-of-the-art performance on unsupervised point cloud completion and segmentation on real data.

1 Introduction

In recent years, point clouds have gained more popularity [12] as the standard outputs of 3D scanning devices [17, 6] and the fundamental data structure to represent and process 3D data [7, 5, 18]. However, clean and complete point clouds of objects, on which the downstream applications such as reconstruction [14] significantly rely, are hard to obtain in practice due to the nature of scanners and occlusions. Point cloud completion which infers a complete object model given a partial point cloud thus has received considerable attention. Recent supervised methods [39, 37, 28, 23, 33, 31] show remarkable performance on recovering the original shapes from incomplete point clouds, but generally assume that the input point cloud is clean and does not contain any noises and outliers. Such an assumption does not always hold in practice, because the surroundings around the target object will be inevitably seen by the cameras during scanning real scenes. Though semantic or instance segmentation on point clouds has been investigated in the literature, the state-of-the-art instance segmentation methods [20, 4] only report around 68% average precision on the public ScanNet [7] and [1] datasets. That is to say, accurately cutting out the target object from the whole point cloud still remains challenging. Therefore, it is important for the downstream applications to remove noises from point clouds or directly complete the point clouds with noises.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we propose an unsupervised end-to-end network for completing the partial point cloud with noises from real scenes and outputting the object mask at the same time, dubbed UGAAN. Previous supervised point cloud completion and segmentation require the paired clean and complete point clouds, together with object-level point labels of real scenes, for training, which are hard and costly to obtain in practice [6, 21]. Most deep-learning-based methods heavily rely on such training data, which limits their applications in practice. Clearly different from previous methods, our UGAAN only needs unlabeled real-scene objects and unpaired artificial CAD models from for unsupervised training.

We employ the framework of GAN [11] which is widely used in unsupervised tasks. Traditional framework of GAN is hard to converge due to the gap between real-scene data and artificial data. To resolve this problem, we utilize an autoencoding generator to learn the basic shapes of objects from the artificial data for easier convergence. Usually, the discriminator could easily distinguish the artificial data as real data and the prediction as fake data, leading to bad performance due to the data gap between predicted point clouds and artificial data. Our discriminator, by contrast, accepts the prediction of artificial data rather than the artificial data itself as real data, which can reduce the negative effect of data gap. Our discriminator also takes latent codes extracted from real-scene data and artificial data as the criterion, by which the generator can robustly learn a better latent space for prediction. We employ a refiner to refine and upsample the first prediction result, and the final complete point clouds thus generated are more accurate and uniform. Finally, the segmentation module predicts the labels according to the predicted point cloud and input point cloud. Compared with usual point cloud completion datasets [39] whose objects are aligned by the yaw angles and scales, real-scene datasets such as ScanNet and S3DIS [1] contain objects that have different yaw angles and varied scales, making the prediction of complete point clouds challenging. To accommodate real-scene data, we randomly rotate and scale the artificial point clouds for easier learning and convergence.

We train our UGAAN on ScanNet with ShapeNet as the artificial dataset without any pre-training or fine-tuning, and then directly evaluate it on the other two real-scene datasets, including S3DIS and ScanObjectNN [25]. Comparisons with the state-of-the-art methods [21, 6] show the superiority of our method.

The main contributions of our work are as follows.

- We propose, for the first time, an unsupervised end-to-end network for completion and segmentation of point clouds with outliers by combining GAN and an autoencoding network.
- Our method completes the real-scene object with outliers from the background without the need of any pre-training.
- We conduct extensive experiments for analysis and comparisons on three different real-scene datasets, the results of which show the superiority and generalization ability of our methods.

The rest of this paper is organized as follows. Section 2 reviews briefly the works on point cloud completion and segmentation. Section 3 introduces our UGAAN in detail. Dataset generation, experiments and comparisons are shown in Section 4 and Section 5. Section 6 concludes the whole paper and highlights future work.

2 Related Work

2.1 Point cloud completion and segmentation.

In the early stage, existing successful works [24, 8] utilize voxels as representations of 3D models on which 3D convolutional neural networks can be immediately applied. PMP-Net [31] and PMP-Net++ [30] generate complete point clouds by moving input points to appropriate positions iteratively with minimum moving distance. Additional information such as images or edges of models is leveraged by [40] and [10] to complement the point cloud input. SnowflakeNet [33] utilizes transformer and point-wise feature deconvolutional modules to refine the first-stage point could multiple times. And Yu *et al.* [38] also utilize a transformer-based architecture which takes unordered

groups of points with position embeddings. For semantic segmentation, Jiang *et al.* [16] propose a multi-stage ordered convolution module to stack and encode the information from eight spaces, achieving orientation encoding and scale awareness. Liang *et al.* [20] split nodes of a pre-trained, intermediate, semantic superpoint tree for proposals of instance objects. Point Transformer [41] introduces transformer in the encoding process of point cloud to learn the representation, which can be applied to different point cloud processing such as segmentation and classification.

2.2 Unsupervised point cloud learning.

Xiao *et al.* [34] summarize the unsupervised point cloud representation learning works using DNNs. Some works [19, 35, 27] propose point cloud unsupervised pre-training approaches. PSG-Net [36] proposes an effective auto-encoder architecture and achieves competitive performance in point cloud reconstruction and unsupervised classification. Han *et al.* [13] split each point cloud into a front half and a back half with the multi-angle scenario and then train their network to predict the back half sequence from the corresponding front half sequence. Chen *et al.* [3] apply geometric sampling invariant representations to learn intrinsic feature representations of different forms of point clouds. The proposed model achieves excellent results on classification, segmentation and upsampling tasks. Jiang *et al.* [15] introduce two contrastive losses to respectively facilitate downstream classification and segmentation. Coseg [21] presents a point cloud object co-segmentation task, aiming to segment the common 3D objects in a set of point clouds, and fathom a method with co-contrastive losses to minimize feature discrepancy inside estimated object points and maximize feature separation between the object and background points, which also leads to a weak generalization. Chen *et al.* [6] propose an unpaired point cloud completion method that can be trained without requiring explicit correspondence between partial and complete point clouds by employing an adaptive transform network in GAN, however, the training process of its network is complicated with several pre-training models.

3 Method

3.1 Problem Formulation

Let $P_r \in \mathbb{M}_{n_0 \times 3}$ represent the point cloud from real-scene data with distribution p_r and P_a denote artificial data with distribution p_a, respectively. P_r contains points belonging to the object and from background (noises). $Obj(P_r)$ represents the object points on P_r. Assuming that $Obj(P_r) \sim p_o$, the goal of unsupervised point cloud completion and segmentation is to train a generator G satisfying the following two conditions:

1. $G(P_r; \theta_g) \sim p_a$,
2. $G(P_r; \theta_g) \cap P_r \sim p_o$,

where $G(P_r; \theta_g)$ represents the point cloud predicted by G with input P_r and parameters θ_g, and the operation $\cap$ denotes the intersection of two point clouds[1]. The two conditions are utilized to constrain completion and segmentation processes of point clouds, respectively. Condition 1 encourages the predicted shapes to be similar to the shapes in the artificial dataset. Condition 2 requires that the generator should predict shapes according to P_r rather than generating random shapes, and the segmentation result which is the intersection of prediction and P_r should match the object points.

3.2 Network Architecture

We employ Generative Adversarial Network (GAN) [11] to solve this unsupervised learning problem. As shown in Figure 1, our UGAAN consists of an autoencoding generator and a discriminator that plays a minimax game for unsupervised completion and segmentation of real-scene object point cloud. Furthermore, we also utilize a refiner to refine the predicted complete point cloud and a segmentation module for label prediction.

[1] A point cloud $P \in \mathbb{M}_{n \times 3}$ can also be seen as a set containing many points $\mathbf{p} \in \mathbb{M}_{1 \times 3}$.

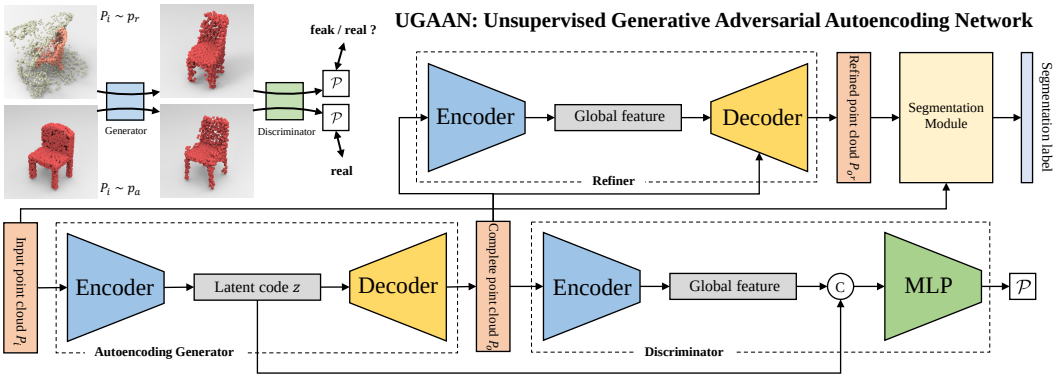


Figure 1: The overall architecture of our UGAAN. The network contains an autoencoding generator, a discriminator, a refiner and a segmentation module. The autoencoding generator predicts the complete point cloud according to the input. The discriminator takes predicted point cloud and latent code as the criterion and decides whether the input point cloud is from artificial data or not. The refiner refines the complete point cloud, and the segmentation module predicts labels.

Autoencoding Generator. We use PointNet++ [22] as the encoder E of the generator to extract the latent code $\mathbf{z}$ of input point cloud P_i, and a multi-layer perceptron (MLP) as the decoder to predict the point cloud $P_o \in \mathbb{M}_{n_1 \times 3}$. The generator takes artificial data as input to learn basic shapes of point clouds. Then, we employ a lightweight encoder-decoder refiner [39] R on P_o to get the refined and upsampled output $P_{o^r} \in \mathbb{M}_{n_2 \times 3}$. Different from [39], the upsampling module directly refines the complete point cloud P_o. Given an input point cloud P_i, we can obtain the latent code, the complete prediction and the refinement by $\mathbf{z} = E(P_i; \theta_g)$, $P_o = G(P_i; \theta_g)$ and $P_{o^r} = R(P_o; \theta_r)$, respectively, where θ_g and θ_r are the parameters of generate G and refiner R.

Discriminator. We take advantage of the encoder proposed in [39] to extract the global features $\mathbf{f}$ of P_o. Then the global feature $\mathbf{f}$ and the latent code $\mathbf{z}$ are combined and fed into a MLP for predicting the probability $\mathcal{P}$:

$$\mathcal{P}(P_i; \theta_g, \theta_d) = D(\mathbf{z}, P_o; \theta_d) = D\big(E(P_i; \theta_g), G(P_i; \theta_g); \theta_d\big), \tag{1}$$

where $P_i \sim p_a$ and θ_d is the parameters of the discriminator. To reduce the negative effect of the shape gap between the prediction and artificial data, we feed the predicted shapes rather than the artificial data into the discriminator. The discriminator also takes latent codes as the criterion by which the generator can learn better latent codes for prediction.

Segmentation Module. Finally, the segmentation results (i.e. the object points of p_r) are obtained under the guidance of the complete prediction P_o. Here, we directly calculate the intersection between P_o and P_r to get the object points, rather than predicting with another network. Due to the discrete nature of point clouds, the strict intersection between P_o and P_r is almost empty. As a consequence, we make use of general intersection between the point clouds P_1 and P_2 which is indicated by the overlapping area:

$$P_1 \cap_d P_2 = \{cd_{\mathbf{p} \to P_1} < d \wedge cd_{\mathbf{p} \to P_2} < d | \mathbf{p} \in \mathbb{M}_{1 \times 3}\}, \tag{2}$$

where $cd_{\mathbf{p} \to P_1}$ will be introduced in Equation 5 with d being the threshold. The predicted segmentation label of $\mathbf{p} \in P_r$ conditioned on P_o with threshold d can be computed as:

$$S_d(\mathbf{p}; P_r, P_O) = \begin{cases} 1 & \mathbf{p} \in P_r \cap_d P_o, \\ 0 & others, \end{cases} \tag{3}$$

Besides, the segmentation label of $P_r = [\mathbf{p}_1^T, ..., \mathbf{p}_j^T, ..., \mathbf{p}_n^T]^T$ can be obtained by $S_d(P_r; P_o) = [S_d(\mathbf{p_1}; P_r, P_O), ..., S_d(\mathbf{p_j}; P_r, P_O), ..., S_d(\mathbf{p_n}; P_r, P_O)]^T \in \mathbb{M}_{n \times 1}$.

3.3 Optimization

We modify the optimization strategy introduced by GAN [11] to train our UGAAN on the unsupervised point cloud completion and segmentation problem. Specifically, the optimization problem of

the proposed UGAAN is formulated as:

$$\min_{\theta_g}\max_{\theta_d} \mathop{\mathbb{E}}_{P_i\sim p_a}[\log \mathcal{P}(P_i;\theta_g,\theta_d)] + \mathop{\mathbb{E}}_{P_i\sim p_r}[\log(1-\mathcal{P}(P_i;\theta_g,\theta_d))]. \tag{4}$$

Due to the data gap between artificial data and real-scene data [6] and the limited fitting ability of the decoder, the terms $G(P_r)$ and P_a often exhibit huge differences as shown in the top left part of Figure1. It makes D easier to optimize than G which makes network hard to converge and will result in poor completion performance. Thus, different from the original GAN where the discriminator directly works on $G(P_r)$ and P_a, the discriminator of UGAAN predicts according to $G(P_r)$ and $G(P_a)$ to reduce the negative effects caused by these gaps for easier convergence. Besides, if we impose no other constraints on problem 4, the generator may just generate random points after the optimization process is complete. Thus, we also impose two losses on problem 4 to restrict the prediction of the generator, namely the autoencoding loss between $G(P_a)$ and P_a, and the reconstruction loss between $G(P_r)$ and P_r.

We employ the well-known Chamfer Distance(CD) [9] to measure the similarity between point clouds. For a point cloud $P=[\mathbf{p}_1^T,...,\mathbf{p}_j^T,...,\mathbf{p}_n^T]^T$, where $\mathbf{p}_j\in\mathbb{M}_{1\times 3}$, and a point $\mathbf{p}\in\mathbf{M}_{1\times 3}$, the Chamfer Distance from $\mathbf{p}$ to P is:

$$cd_{\mathbf{p}\to P}=\min_{j=1}^{n}||\mathbf{p}-\mathbf{p}_j||^2. \tag{5}$$

For two point clouds $P_1=[\mathbf{p}_1^T,...,\mathbf{p}_j^T,...,\mathbf{p}_n^T]^T$ and P_2, the Chamfer Distance from P_1 to P_2 is:

$$cd_{P_1\to P_2}=\frac{1}{n_1}\sum_{j=1}^{n} cd_{\mathbf{p}_j\to P_2}. \tag{6}$$

Besides, the Chamfer Distance between P_1 and P_2 is defined as:

$$cd_{P_1\leftrightarrow P_2}=cd_{P_1\to P_2}+cd_{P_2\to P_1}. \tag{7}$$

The point cloud P_1 will be a part of the point cloud P_2 when the term $cd_{P_1\to P_2}$ is minimized. Moreover, the point clouds P_1 and P_2 will be identical when the term $cd_{P_1\leftrightarrow P_2}$ is minimized.

We use the Chamfer Distance between $G(P_a)$ and P_a as the autoencoding loss. Beside, the Chamfer Distance from $G(P_r)$ to P_r is utilized as the reconstruction loss, which makes the predicted shape be a part of the input and forces the generator to predict according to the input. We also apply the Chamfer Distance between $R(G(P_a))$ and P_a to optimize the refiner R.

In practice, we alternately optimize G with the optimization goal:

$$\begin{aligned}\min_{\theta_g,\theta_r} \quad & \mathop{\mathbb{E}}_{P_i\sim p_r}\big[-\alpha_1\cdot\log\mathcal{P}(P_i;\theta_g,\theta_d)+\alpha_2\cdot cd_{G(P_i;\theta_g)\to P_i}\big]+\\ & \mathop{\mathbb{E}}_{P_i\sim p_a}[\alpha_3\cdot cd_{G(P_i;\theta_g)\leftrightarrow P_i}+\alpha_4\cdot cd_{R(G(P_i;\theta_g);\theta_r)\leftrightarrow P_i}],\end{aligned} \tag{8}$$

and D with optimization goal:

$$\min_{\theta_d}\mathop{\mathbb{E}}_{P_i\sim p_r}\Big[-\alpha_5\cdot\log\big(1-\mathcal{P}(P_i;\theta_g,\theta_d)\big)\Big]+\mathop{\mathbb{E}}_{P_i\sim p_a}\big[-\alpha_6\cdot\log\mathcal{P}(P_i;\theta_g,\theta_d)\big], \tag{9}$$

where $\alpha_{1\sim 6}$ are the weights used for balancing the influences between each term.

4 Experiments

4.1 Datasets

We first generate the complete point clouds by sampling CAD models of ShapeNet [2] and ModelNet [32], serving as the artificial data. We randomly rotate and scale them to simulate poses of real-scene data. Then we follow a similar process as ScanObjectNN [25] to generate the input point clouds with surrounding points (background points) by cutting out the points in axis-aligned boxes whose origins are the center of objects in ScanNet [7]. The sizes of these axis-aligned boxes are s larger than the axis-aligned bounding boxes of objects, where we set s to 50%. We also record labels of points which are 1 for object points and 0 for background points (noises) for evaluation. We select four categories to establish four datasets and randomly split the data into train, valid and test sets. The coordinates of all point clouds are normalized to $[-1,1]$. Our UGAAN is trained on the train set, and evaluated on test set.

Table 1: Comparison (mIOU) of results by our method (trained with train set of ScanNet) and previous methods on test sets of ScanNet and S3DIS.

Dataset	Setting	Model	Avg.	Bookcase	Chair	Sofa	Table
ScanNet	Full Sup.	PointTransformer [41]	0.679	0.58	0.83	0.69	0.62
	Unsup.	K-Means	0.201	0.25	0.23	0.18	0.16
		Unpairerd [6]	0.422	0.47	0.58	0.34	0.30
		Ours	**0.448**	**0.49**	**0.58**	**0.38**	**0.34**
S3DIS	Full Sup.	DGC [29]	0.722	0.34	0.96	0.69	0.9
		PN++ [22]	0.735	0.44	0.92	0.75	0.83
	Unsup.	K-Means	0.252	0.36	0.18	0.27	0.2
		AdaCoSeg [42]	0.31	0.34	0.24	0.38	0.28
		CoSeg [21]	0.465	0.36	0.51	0.50	**0.49**
		Ours	**0.486**	**0.46**	**0.51**	**0.63**	0.34

4.2 Evaluation Metrics

As we mentioned in Section 1, generating completion ground truth manually for real-scene data is much harder than labeling the segmentation ground truth. We then calculate the intersection of complete prediction and input point cloud following condition 2 in Section 3.1 to evaluate the segmentation results, by which the completion prediction is indirectly estimated. Thus we utilize the method mentioned in our segmentation module to get labels of input point cloud according to the predicted complete point cloud, and then employ IOU, following the previous work [21], to measure the difference between prediction and ground truth.

4.3 Implementation details

Our framework is implemented using PyTorch. The point numbers $n_{0\sim2}$ in Section 3 are set to 2048, 512 and 2048, and the weights $\alpha_{1\sim6}$ are set to 1, 100, 5, 100, 0.5 and 0.5. We train four categories separately with the Adam optimizer. The learning rate of the optimizer and the batch size for our network are set to 1.0×10^{-5} and 4, separately. To ensure the convergence of neural networks, we train the model for around 240 epochs. Training our UGAAN takes about 20 hours for convergence with a GTX 2080Ti GPU. We will make our dataset and codes public in the future.

4.4 Comparison

We compare our UGAAN against existing representative methods on point cloud segmentation including unsupervised methods [21, 6, 42] and supervised methods [41, 29, 22] and on unsupervised point cloud cloud completion [6] both quantitatively and qualitatively. We simply modify [6] by replacing its reconstruction loss with ours and retrain it to fit our datasets, and then utilize our segmentation module to get the segmentation results for evaluation. As shown in Table 1, we compare these methods on ScanNet and S3DIS with four categories, including the chair, table, sofa and bookshelf. Our method is trained on the training set of ScanNet and tested on the test set of these two datasets. The results are 6.16% and 4.51% higher than [6] and [21], showing the superiority of our method which can predict more accurate complete point clouds. We also test our method trained with ScanNet with more categories on the test set of ScanObjectNN, and the results are shown in the last second row of Table 2. There are only seven common categories in these two datasets. Thus, we can only conduct the comparison on these seven categories. We also train our UGAAN on 15 categories of ScanObjectNN together with ShapeNet and ModelNet to evaluate the generalization ability across multiple categories. The results are shown in the last row of Table 2. The results are 8.73% and 4.46% higher than [21] on "Avg. (part)" and "Avg. (all)". This verifies the generalization ability of our method with different categories.

Qualitative results on the three categories are shown in Figure 2. The point clouds produced by our UGAAN exhibit more accurate shapes and details, such as the legs and backrest of the chair and the

Table 2: Comparison (mIOU) of results by our and previous methods on test sets of ScanObjectNN. The last two rows show the results of our mothed trained on train set of ScanNet and ScanObjectNN, separately. The "Avg. (part)" indicates the mIOU of the first seven categories (Bookcase $\sim$ Cabinet) and "Avg. (all)" indicates the mIOU on all the categories.

Setting	Model	Avg. (part)	Bookcase	Chair	Sofa	Table	Bed	Pillow	Cabinet	Bag	Bin	Box	Desk	Display	Door	Sink	Toilet	Avg. (all)
Full Sup.	DGC [29]	0.757	0.60	0.84	0.79	0.76	0.80	0.78	0.73	0.76	0.81	0.73	0.72	0.76	0.83	0.64	0.82	0.753
	PN++ [22]	0.764	0.62	0.84	0.81	0.77	0.81	0.75	0.75	0.75	0.83	0.79	0.77	0.79	0.83	0.68	0.85	0.775
Unsup.	K-Means	0.370	0.34	0.40	0.41	0.31	0.38	0.37	0.38	0.43	0.42	0.41	0.38	0.38	0.45	0.39	0.40	0.389
	AdaCoSeg [42]	0.367	0.26	0.55	0.54	0.23	0.37	0.29	0.33	0.38	0.31	0.48	0.37	0.44	0.35	0.43	0.44	0.385
	CoSeg [21]	0.561	0.48	0.58	0.64	0.44	0.64	0.60	0.55	0.66	0.70	0.68	0.46	0.62	0.74	0.57	0.68	0.605
	Ours (ScanNet)	**0.610**	**0.53**	**0.73**	**0.64**	**0.53**	**0.66**	**0.61**	**0.56**	-	-	-	-	-	-	-	-	-
	Ours (ScanObjectNN)	**0.587**	**0.50**	**0.65**	**0.64**	**0.50**	**0.65**	**0.60**	**0.57**	**0.71**	**0.73**	**0.70**	**0.48**	**0.63**	**0.75**	**0.60**	**0.73**	**0.632**

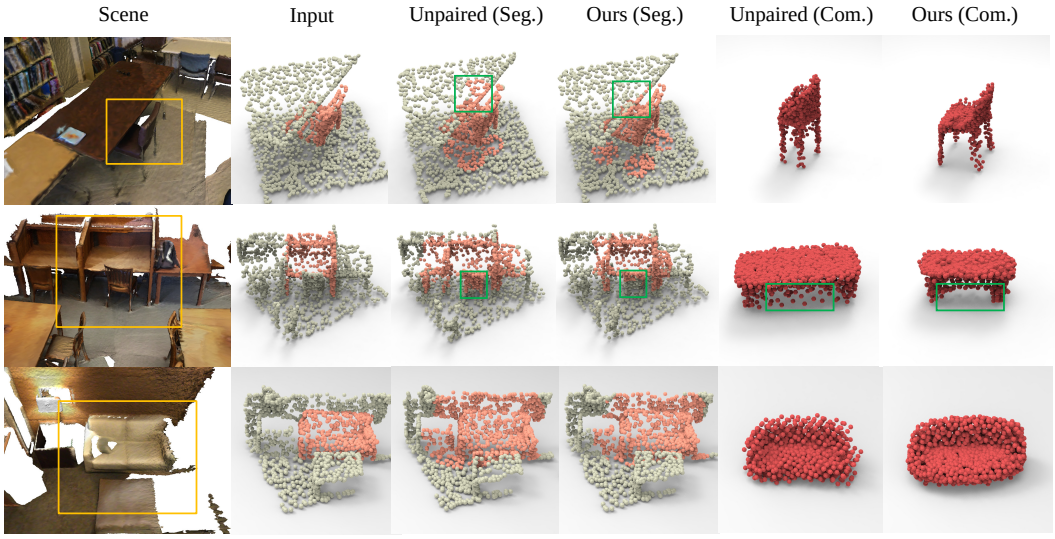


Figure 2: Point cloud completion and segmentation results by Unparied [6] and our method. The categories from top to bottom: chair, table, and soda. Red and white in the inputs denote points on the partial objects and outliers, separately. The real inputs do not have the segmentation labels. The left columns shows the real-scenes containing input point clouds.

legs of the table shown in Figure 2, leading to more accurate segmentation. And the completion results of ours present more uniform spatial distributions, such as the sofa shown in Figure 2 which contains less noise, which could be attributed to our refiner. Uniform point distribution would facilitate further applications including re-construction, simplification, and so on.

4.5 Ablation study

Ablation study on network architecture. We conduct experiments on three ablated models to evaluate the necessity of important structures and reconstruction loss of our network. The first model is the basic structure of GAN with a feature extractor and a shape generator. The generator predicts a complete point cloud according to the input point cloud, and the discriminator distinguishes between the prediction and the artificial object. The second model replaces the original generator with the autoencoding generator based on the first model. The third model adds the reconstruction loss introduced in Section 3.3 to the second model. The discriminator of the fourth model makes a distinction between the predictions with real-scene data and the artificial object as input separately. And the discrimination of our full model also takes latent codes as the criterion. As shown in Table 3, we train and test these ablated models with the ScanNet dataset on the chair category. The evaluation shows the importance of these structures and reconstruction loss. The comparison against the third and fourth models shows that there is a gap between the predictions and artificial point clouds, which

explains the reason we apply discriminator to the predictions rather than the artificial point clouds directly.

Table 3: Ablation study of the proposed method.

Model	IOU
backbone	0.122
+ autoencoding generator	0.449
+ reconstruction loss	0.474
+ discrimination on predictions	0.544
+ discrimination on latent codes (full)	**0.580**

Table 4: Ablation study of refiner.

Dataset	ScanNet	S3DIS	ScanObjectNN
Ours (no refiner)	**0.447**	0.478	0.596
Ours (full)	0.444	**0.486**	**0.610**

Refiner. We also evaluate the importance of refiner both quantitatively and qualitatively. As shown in Figure 3, the refiner can refine the prediction and make the prediction more uniform and reduce noises. The mIOU of our methods without refiner and with refiner on the test sets of the S3DIS and ScanObjNN are shown in Table 4. These methods are trained with the train set of ScanNet. The results show that even though the refiner takes no effect when the method is trained and evaluated on the same dataset, the refiner makes our method more robust when evaluated on new datasets.

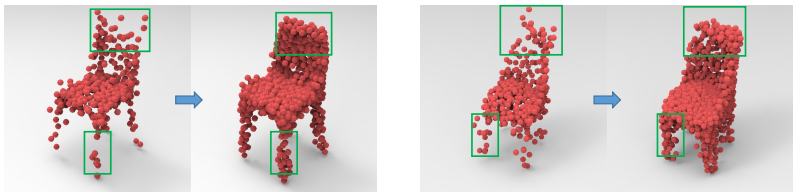


Figure 3: The point clouds before and after refining on chair category.

The noise ratio s. We also set the noise ratio s to 20%, 10%, 5% and 0%, and generate four more datasets with different noise ratios. Then we repeat the experiments above on the chair category and the IOUs are shown in Figure 4. The lower the noise ratio is, the easier the problem is, making the model perform better. The results also show our generalization performance.

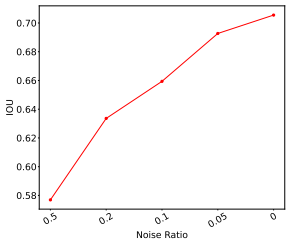
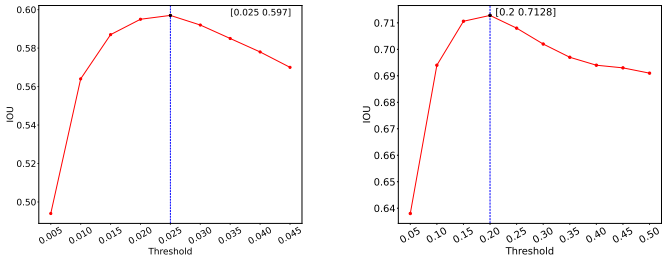


Figure 4: The IOU of our method on chair category with different noises ratio.



(a) Chair (b) Bag

Figure 5: The IOU of our method with different threshold of intersection on chair and bag categories.

The threshold d of intersection. We also evaluate the influence of different threshold d of intersection on the IOU during the evaluation. As Figure 5 shows, we apply different d to the categories chair and bag. The results show that different object has different best d, leading to best IOU according to their volume in practice. Since we normalize the coordinates of point clouds to $[-1, 1]$, the point clouds of small objects such as bags contain more details in the same space than big objects such as chairs.

Extracted features of object points and background points. We visualize the extracted features of object points and background points from the encoder for analysis. As Figure 6(a) shows, every line indicates the feature of a point. The lighter the pixel is, the larger the value of this channel is. The figure shows that object points have larger values on some channels which can be passed to the latent

code by max-pooling operation. Figure 6(b) shows the visualization of the features of artificial points clouds, whose pattern is similar to Figure 6(a), since all the points of artificial data are object points.

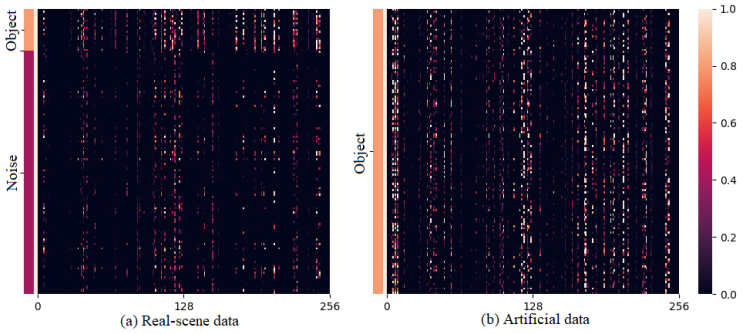


Figure 6: Visualization of the extracted features of real-scene data and artificial data. The bar on the left indicates the type (object or noise) of this line.

Visualization of latent codes. We use t-SNE [26] to visualize the latent code distribution of real-scene data and artificial data. As shown in Figure 7 (c), the blue points represent the artificial data and red points indicate the real-scene data, and Figure 7 (e) shows the detailed point clouds of an area. The real-scene data that contains the same shape as the artificial data are grouped together, which shows the robustness of our method with input contaminated by noises. The point clouds having similar yaw angles are grouped together showing that our method could learn reasonable latent codes. This could be attributed to the fact that our discriminator also takes latent codes as the criterion. Visualization of the latent codes extracted from the method whose discriminator only takes predicted point cloud as the criterion is shown in Figure 7 (d). It is obviously observed that these two kinds of data are separated clearly. We also train our method with the dataset that mix up the real-scene data with different noise ratios. As shown in Figure 7 (a) and (b), similar object with different noise ratios are grouped together, which shows the robustness of our method on real-scene data with different noise ratios.

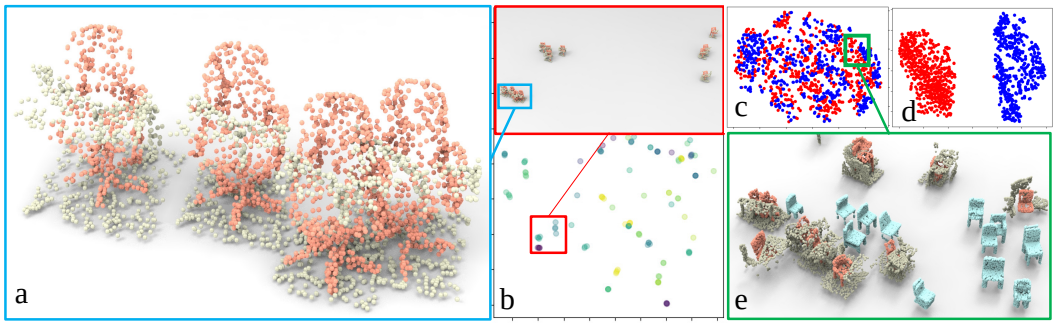


Figure 7: Visualization of t-SNE results with details. The same object with different noise ratio have the same color in Figure (b). Red and blue points are indicate the real-scene data and artificial data, respectively.

5 Conclusion, Discussion, and Future Work

We have presented UGAAN, an end-to-end network specifically designed for unsupervised completion and segmentation of point clouds contaminated by noises or containing outliers. Benefiting from our proposed autoencoding generator and latent-code-based discriminator, our method is able to generate complete point cloud accurately and segment the input real-scene point cloud in the meanwhile. Extensive experiments show the superiority and generalization of our method on different categories of different datasets. We also conduct experiments verifying the robustness of our method on noises. Our method needs artificial datasets during the unsupervised training, which limits our method to apply to those categories without artificial correspondences. The encoder and decoder of our generator can be improved for better prediction and the segmentation module can be replaced with a learning-based module, which are the future work of our method.

References

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Haolan Chen, Shitong Luo, Xiang Gao, and Wei Hu. Unsupervised learning of geometric sampling invariant representations for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 893–903, 2021.

[4] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15467–15476, October 2021.

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[6] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired point cloud completion on real scans using adversarial training. *arXiv preprint arXiv:1904.00069*, 2019.

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[8] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[9] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[10] Bingchen Gong, Yinyu Nie, Yiqun Lin, Xiaoguang Han, and Yizhou Yu. Me-pcn: Point completion conditioned on mask emptiness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12488–12497, October 2021.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[12] 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.

[13] Z. Han, X. Wang, Y. S. Liu, and M. Zwicker. Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. *IEEE*, 2019.

[14] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh. *ACM Transactions on Graphics (TOG)*, 39:126:1 – 126:12, 2020.

[15] Jincen Jiang, Xuequan Lu, Wanli Ouyang, and Meili Wang. Unsupervised representation learning for 3d point cloud data. *arXiv preprint arXiv:2110.06632*, 2021.

[16] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. 2018.

[17] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018.

[18] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[19] Z. Li, Z. Chen, A. Li, L. Fang, Q. Jiang, X. Liu, J. Jiang, B. Zhou, and H. Zhao. Simipu: Simple 2d image and 3d point cloud unsupervised pre-training for spatial-aware visual representations. 2021.

[20] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2783–2792, October 2021.

[21] Cheng-Kun Yang; Yung-Yu Chuang; Yen-Yu Lin. Unsupervised point cloud object co-segmentation by co-contrastive learning and mutual attention sampling. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *2017 Advances in Neural Information Processing Systems (NIPS)*, volume 30. Curran Associates, Inc., 2017.

[23] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019.

[24] Duc Thanh Nguyen, Binh-Son Hua, Khoi Tran, Quang-Hieu Pham, and Sai-Kit Yeung. A field model for repairing 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5676–5684, 2016.

[25] 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 (ICCV)*, 2019.

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[27] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner. Unsupervised point cloud pre-training via occlusion completion. 2020.

[28] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 790–799, 2020.

[29] 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.

[30] X. Wen, P. Xiang, Z. Han, Y. P. Cao, P. Wan, W. Zheng, and Y. S. Liu. Pmp-net++: Point cloud completion by transformer-enhanced multi-step point moving paths. 2022.

[31] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net: Point cloud completion by learning multi-step point moving paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[32] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, 2015.

[33] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[34] A. Xiao, J Huang, D. Guan, and S. Lu. Unsupervised representation learning for point clouds: A survey. 2022.

[35] S. Xie, J. Gu, D. Guo, C. R. Qi, L. J. Guibas, and O. Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. 2020.

[36] J. Yang, P. Ahn, D. Kim, H Lee, and J. Kim. Progressive seed generation auto-encoder for unsupervised point cloud learning. *arXiv e-prints*, 2021.

[37] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[38] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021.

[39] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018.

[40] Xuancheng Zhang, Yutong Feng, Siqi Li, Changqing Zou, Hai Wan, Xibin Zhao, Yandong Guo, and Yue Gao. View-guided point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15890–15899, June 2021.

[41] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[42] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Li Yi, Leonidas J. Guibas, and Hao Zhang. Adacoseg: Adaptive shape co-segmentation with group consistency loss. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8540–8549, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [No] There is no theoretical results in this paper.
   (b) Did you include complete proofs of all theoretical results? [No] There is no theoretical results in this paper.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the pseudo code of our method in our supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See our supplementary material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See the supplemental material.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See the supplemental material.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] There is no crowdsourcing or conducted research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No] There is no crowdsourcing or conducted research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] There is no crowdsourcing or conducted research with human subjects.