RLGrid: Reinforcement Learning Controlled Grid Deformation for Coarse-to-Fine Point Could Completion

Shanshan Li, Pan Gao, Xiaoyang Tan, and Wei Xiang

***Abstract*—Many point cloud completion methods typically rely on two steps: coarse generation and 2D grid deformed fine output. However, in the fine generation, the expansion range (2D grid scale) required by each point cloud sample may be vastly different. For example, if the expansion range for a vessel shape is applied to a table shape, the final output may be blurry or sparse. To this end, we propose the RLGrid, Reinforcement Learning Controlled Grid Deformation. In detail, we firstly obtain two point cloud skeletons by two branches. One is to use an autoencoder, and the other is to convert the randomly generated normal distribution to coarse point cloud by GAN. We choose the one with smaller Chamfer Distance between coarse output and incomplete input as the input of the second stage. Then, a Reinforcement Learning (RL) agent is designed to select the appropriate expansion range based on the feature of each point cloud, and generate a 2D grid. Finally, all the features are concatenated and sent into a Multilayer Perceptron to obtain the detailed complete point cloud. Experimental results show that RLGrid achieves state-of-the-art performance on various datasets. To the best of our knowledge, RL is not widely used in point cloud completion task due to lack of custom environment, and the proposed RLGrid provides an insight on how to formulate 2D grid deformation as a sequential decision making problem. Further, it can also be plug-and-play on any 2D grid features. Code is available at https://github.com/I2-Multimedia-Lab/RLGrid.**

***Index Terms*—point clouds, 3D shape completion, reinforcement learning, 2D grid deformation.**

I. Introduction

A 3D point cloud is a collection of spatial coordinate points with disorder in a 3D system. Compared with a polygonal mesh, a point cloud does not need to store and maintain the connectivity [1] or topology consistency [2]. Due to its simple, flexible and powerful representation ability, it has been widely used in many application areas, such as cultural relic restoration, autonomous driving and robotics [3], [4]. However, due to the inherent error in the acquisition device, the point cloud collected by the sensor inevitably contains noise and outliers [5], [6]. Meanwhile, the limited resolution of the acquisition equipment, limitations of hardware and computing power, and occlusion often lead to data point missing [7], [6], which seriously affects the downstream tasks, such as surface reconstruction [8] and point cloud classification [9]. Therefore, it is necessary to complete missing parts of point clouds.

S. Li, P. Gao, and X. Tan are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China.

W. Xiang is with School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, Australia.

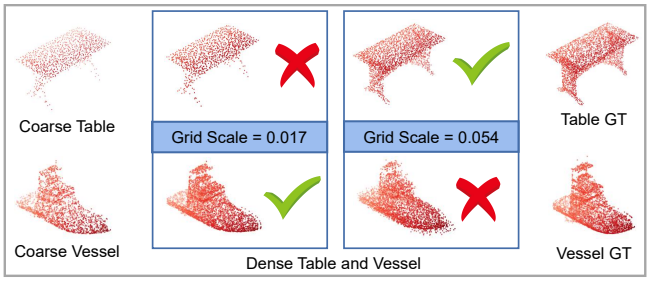


Fig. 1. Visual results of the same grid scale used on different point clouds and different scales used on the same point cloud. The points of Coarse, Dense and GT are 1024, 4096 and 4096 respectively.

In recent years, there have been many point cloud completion methods based on deep learning [10], [11], [7], [12], [13], [14], [15]. The completion process of these methods is generally divided into two stages, coarse point cloud generation and point cloud refinement. The basic framework of coarse point cloud generation is to extract features through encoder and obtain point cloud through decoder. However, only a single encoder-decoder framework may make it difficult for the network to fit a variety of shapes, and in the face of a large area of point missing, a simple encoder is incapable of extracting effective features, which is thus not conducive to decoding as a reasonable shape. More recently, SnowflakeNet [16] and PoinTr [17] have tried to use transformer on point cloud completion. SnowflakeNet models the generation of complete point clouds as the snowflake-like growth of points in 3D space while PoinTr devises a geometry-aware block that models the local geometric relationships explicitly. Although transformer demonstrates very strong feature capture capability, most works like PCN [7], ASFM-Net [14] and PoinTr [17] still need the 2D grid deformation method to turn coarse point clouds into refined point clouds. However, as shown in Fig. 1, experiments show that the best expansion scale of the table shape is 0.054 and using this scale on the vessel will result in a poor completion, which means different point clouds require different scales to control the expansion and each point cloud needs to be considered individually. Since none of the existing datasets, such as ShapeNet [18], Completion3D [19] and KITTI [20] annotate what the most suitable scale is for each point cloud, obtaining the scale of 2D grid through supervised learning does not work. A straightforward way to obtain the suitable 2D grid scale for each point cloud may be exhaustive search, which, however, leads to another huge workload problem (especially when the grid scale needs to be

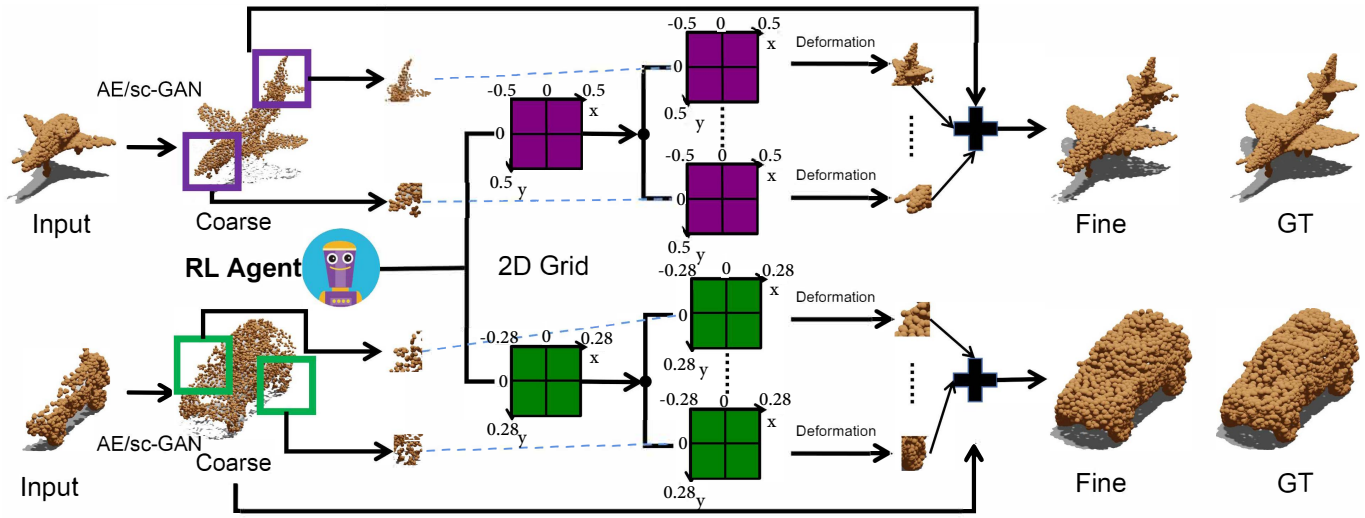


Fig. 2. The reinforcement learning is proposed to control the scale of 2D grid in coarse-to-fine completion network. Generally, different partial point clouds require different scales for 2D grid (e.g., 0.05 on plane and 0.28 on car). The agent selects different scales for different point clouds. 2D grids are generated according to customized scale and deformed into different local patches.

particularly precise such as accurate to three decimal places).

To solve the above problems, we propose a new point cloud completion network dubbed RLGrid. Specifically, we first use autoencoder and GAN to generate two coarse point clouds respectively. After evaluating and comparing the two, we select a more reasonable point cloud as the input of the refinement module. In this way, the dependence of the network on single-way point cloud generation is reduced, and the generalization performance of the network is improved. Furthermore, we incorporate refinement modules into a newly designed reinforcement learning (RL) environment and pay particular attention to the scale of the 2D grid. RL agent predicts the 2D grid scale required by each point cloud through the neural network, which not only avoids the huge workload brought by the exhaustive method, but also makes the details of the refined point cloud richer and the local density distribution more reasonable.

The main contributions of this paper are as follows:

- We propose a dual-skeleton structure for coarse output, so that the process of point cloud refinement is no longer limited by shortcomings of a single generation framework. The skeleton generated by AE is compared with the skeleton generated by GAN, and a more reasonable one is selected.
- We design a new RL environment that allows the agent to select the most appropriate scale for each point cloud based on the feature extracted by encoder, and this adaptive scale of grid used for deformation will consequently lead to a better fine shape.
- The proposed shape completion framework is robust to point clouds with different types of point missing, and it achieves state-of-the-art results on the PCN dataset, ball-hole PCN dataset, and ShapeNet55/34 dataset.

The rest of this paper is organized as follows. Related works about learning-based point cloud completion and reinforcement learning are reviewed in Section II. Section III introduces our proposed approach, including autoencoder, shape completion GAN and RL based 2D grid deformation. Comprehensive experiments have been conducted on several existing benchmarks and homemade dataset to validate the performance, which are detailed in Section IV. Conclusion is provided in Section V.

II. Related Work

The point cloud completion plays a distinct and crucial role in various computer vision applications [21]. This operation needs to meet the following three requirements [22]: (1) preserves the details of the input point cloud; (2) draws the missing parts with detailed geometry; (3) generates uniformly distributed points on the object surface. Unlike traditional two-dimensional image pixels, which have regular arrangement, point cloud data is a series of sparse three-dimensional spatial points with order invariance. With the emergence of deep learning, researchers have proposed many novel point cloud completion methods.

A. Learning-based Completion

Learning-based methods are usually based on a well-designed deep neural network model, and directly input the incomplete shape to obtain the complete one. In the early time, due to the lack of the works, such as PointNet [23] and PointCNN [24], most of the shape completion work used voxels and distance thresholds to represent 3D models, and then used 3D convolutional neural networks for feature extraction, such as methods [25], [26], [27]. Although these studies have made good performance, there are some problems, such as inevitable quantization errors and large storage space.

With the tremendous success of PointNet [23] and PointNet++ [28], direct processing of 3D coordinates has become the mainstream of point cloud based 3D analysis. Based on the network structure employed in point cloud completion and generation, at present, the common point cloud completion methods focus on two forms: point-based and generative model-based. Point-based methods typically use MLP to

model each point independently and use a symmetric function like max-pooling for global feature aggregation. On this basis, many methods [29], [30], [7], [31], [14], [32] were proposed. Yuan *et al.* [7] put forward the PCN network structure, which can directly perform feature learning in the point cloud space, which is also the first known method to directly use the deep learning method to complete the point cloud. MSN [30] introduced a morphing-based decoder and used expansion loss to avoid the problem of uneven distribution of point clouds. These methods both use PointNet-style single-scale feature extractors, and let high-dimensional features directly pass through the Max-pooling layer, resulting in considerable information loss. RLGrid does not use the simple PointNet-MLP, but draws on the multi-resolution encoder proposed by PF-Net [12] to downsample the input incomplete point cloud and perform feature extraction separately, and concatenate the features of different layers to better utilize low-level and mid-level features that contain rich local information. Differently, generative model-based methods utilize a discriminator implicitly learning to estimate the point collections provided by the generator. Due to the complexity of point cloud distribution and the difficulty of GAN training, researchers have greatly improved point cloud completion based on traditional GANs and proposed many new GAN-based methods [13], [33], [12], [34], [35], [36]. Wang *et al.* [13] proposed a point-completion network with a cascaded refinement network as the generator to synthesize these missing parts with high quality by using the details of the input. Furthermore, they design a patch discriminator that uses adversarial training to learn precise point distributions and penalizes generated shapes differently than ground truth. ShapeInversion proposed by Zhang *et al.* [35] introduced GAN inversion into point cloud completion for the first time. It first generated many full shapes via GAN and then obtained incomplete point clouds through a 3D downsampling module to select the best completion result. Although the GAN-based method can enhance the diversity of point cloud generation, it generally needs to train the generator separately for each type of shape, and cannot adapt to a variety of point cloud models. RLGrid designed shape completion GAN (sc-GAN), imitating the conditional vector proposed in Conditional GAN [37], adding one-dimensional category labels to the dataset, so that sc-GAN can use a pre-trained model to generate various shapes we want.

B. Reinforcement Learning

Reinforcement learning is one of the paradigms and methodologies of machine learning, which is used to describe and solve problems in which agents learn strategies to maximize rewards or achieve specific goals in the process of interacting with the environment. Its systems generally include four elements: policy, reward, value, and environment. Specifically, it will store all the states and actions in a two-dimensional table. Then this table is used to determine action executions and update the values of the table until convergence. Recently, RL has also been applied to image adjustment, image segmentation, point cloud registration and other vision fields [38], [39], [40], [41]. However, for RL, the input data for many practical application problems are high-dimensional and non-discrete. For such data, it is not feasible to form a state-action table by storing all the states and actions of the agent. Fortunately, with the rapid development of RL and deep learning (DL), it opens up a new field of research called deep reinforcement learning (DRL). DRL is a combination of DL and RL [42] for solving time-series decision-making problems. It uses the perception ability of deep learning to solve the modeling problem of policy and value function, and also leverages the decision-making power of RL to define problems and optimize goals [43]. There are many representative algorithms, such as DQN [44] and DDPG [45]. DQN learns directly from image pixels through an end-to-end method while DDPG modifies the Actor-Critic structure to improve the stability and convergence of RL training.

It is worth mentioning that RL-GAN-Net [46] is the first work of employing reinforcement learning for point cloud completion, which mainly learns the nonlinear mapping from the input noise of the GAN to the latent space of the point cloud. Nevertheless, RLGrid has two essential differences with RL-GAN-Net: (1) The agent of RL-GAN-Net adopts a single-step decision-making process, while ours is a sequential process, that is, the previous state is continuously replaced by the next state; (2) The agent of RL-GAN-Net chooses a seed from the latent vector for pre-trained GAN, while ours chooses the scale of 2D grid for each point cloud sample. As will be demonstrated in the experiment, our RLGrid outperforms RL-GAN-Net in terms of quantitative and qualitative results.

III. Approach

Our shape completion pipeline is composed of three fundamental modules, which are an AE, a sc-GAN and an RL agent. Each component is a deep neural network that must be trained separately. We first train AE and sc-GAN separately to ensure that they can generate plausible complete point clouds from the input incomplete point clouds. After that, the RL agent is trained along with a pre-trained AE and sc-GAN.

The overall architecture of RLGrid is illustrated in Fig. 3. The pre-trained AE and sc-GAN both generate their own coarse point cloud. The better one is sent to the RL environment, and the process of better coarse point cloud comparison and selection can be represented by (1).

$$\hat{P}_{\text{coarse}} = P_{\text{coarse-i}} \ ,$$
$$i = \begin{cases} 1 & if \ \ d_{CD}\left(P_{\text{coarse-1}}, P_{in}\right) \leq d_{CD}\left(P_{\text{coarse-2}}, P_{in}\right) \\ 2 & else \end{cases} , \tag{1}$$

where $P_{\text{coarse-i}}$ is the coarse output from AE ($i = 1$) or sc-GAN ($i = 2$). Both outputs are compared with incomplete input P_{in} using Chamfer Distance (CD) [47], [10], and the one with the smaller CD value is sent to the RL environment as the coarse point cloud $\hat{P}_{\text{coarse}}$. CD is calculated as follows:

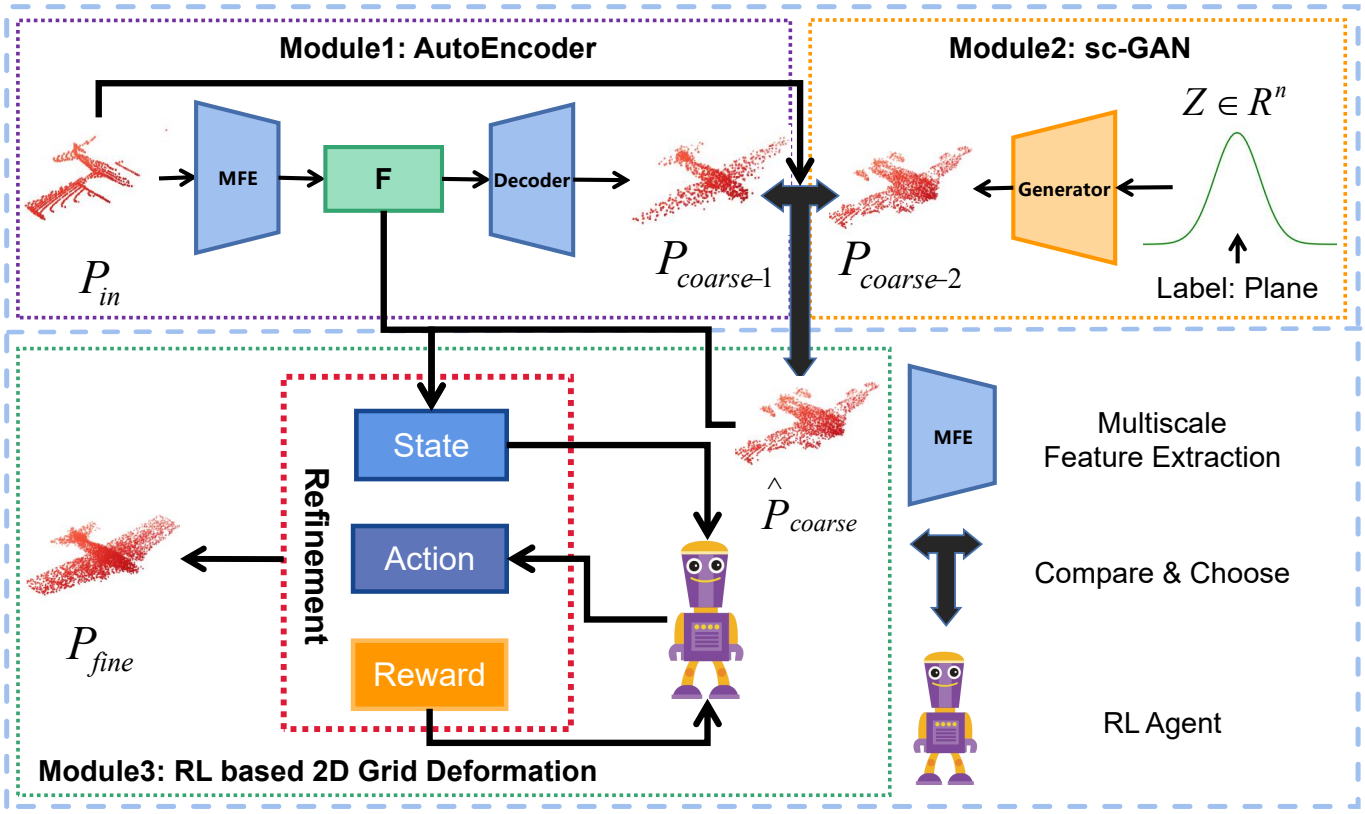


Fig. 3. The overall architecture of RLGrid. The autoencoder and shape completion GAN (sc-GAN) both produce a coarse complete point cloud. The more-plausible point cloud will be fed into the reinforcement learning (RL) environment for refinement. The RL agent selects reasonable actions from the action space to change the state according to the initialized state, and obtains rewards from the environment at the same time. After a serialization process, the final refined point cloud is given.

$$d_{\mathrm{CD}}\left(S_1, S_2\right) = \frac{1}{|S_1|}\sum_{x\in S_1}\min_{y\in S_2}\|x-y\|_2^2 + \frac{1}{|S_2|}\sum_{y\in S_2}\min_{x\in S_1}\|y-x\|_2^2 . \tag{2}$$

(2) measures the average nearest distance between predicted point cloud S_1 and the ground truth point cloud S_2.

Note that in (1), we compare to the incomplete input instead of the ground truth (GT). This is because the GT point cloud is not available for comparison during testing. Although this comparison cannot fully measure the distance between the coarse output and the GT point cloud, it can still differentiate these two outputs, since the corresponding partial part contained in the better coarse output can better comply with the incomplete input. Then, the agent selects an appropriate 2D grid scale for the deformation. After feature splicing, a Multilayer Perceptron (MLP) converts the combined feature into a fine complete point cloud. In the following subsections, we explain the three fundamental building blocks of our approach, and then describe the combined architecture.

A. Autoencoder (AE)

AE is a deep learning framework that maps the input into a latent code, and then maps the code back to the input. We refer to the intermediate representation as the global feature vector (GFV). Any off-the-shelf point cloud GFV extraction networks, such as PointNet [23], FoldingNet [11], *etc.*, can serve as the backbone of our AE. However, we experimentally find that the feature extraction part of the PFNet [12] achieves the best performance in our RLGrid.

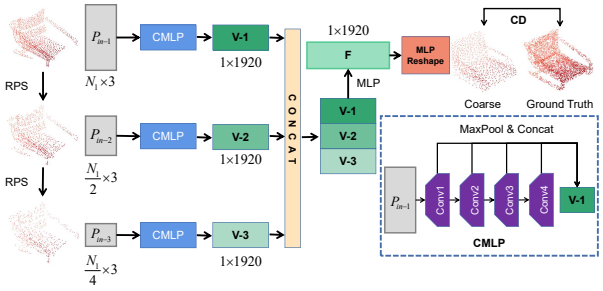


Fig. 4. The details of AE about the process of multi-scale feature extraction. CMLP means the combined multilayer perception used in PF-Net. $N_1\,(2048)$ represents the number of points in the incomplete point cloud we feed. Finally, a coarse point cloud with 1024 points is generated. This is the first stage of AE completion, *i.e.*, feature extraction and coarse point cloud generation.

Like the PCN [7] using two steps for the shape completion, we also perform coarse and fine completion steps on the incomplete point cloud. For the encoding part, we first use the multi-scale feature extraction method in PFNet [12] to downsample the input incomplete point cloud twice. For each sampling stage, we concatenate multiple dimensional latent vectors derived from the last four layers. Then, after splicing the three latent vectors, the final GFV is converted by MLP.

For the decoding part, we use a fully-connected decoder and a folding-based decoder similar to PCN. The fully-connected decoder generates a coarse complete point cloud P_{coarse} directly from the GFV. The above coarse generation process is shown in Fig. 4. And regarding the folding-based decoder, for each point q_i in P_{coarse}, a patch of points is generated by a 2D grid via the folding operation. With each point in the coarse point cloud as the center, all patches are spliced together to form a fine complete point cloud P_{fine}. After the folding-based decoder is trained, it is also used as the environmental basis for reinforcement learning. The training of AE is performed with back-propagation reducing the distance between the output and GT point cloud, and in this work, we choose CD as our completion loss since it is differentiable and efficient.

B. Shape Completion GAN (sc-GAN)

Although the architecture of AE is robust against local defects often observed in incomplete inputs, it still has some flaws. When the missing parts are concentrated and the number of missing points is large, it may generate unreasonable shapes. To solve this problem, we add another branch which adopts tree-GAN [48] to generate another coarse point cloud. Note that sc-GAN does not need to be classified for training. In order to ensure that sc-GAN can generate the shape we want, we train it with a conditional vector similar to the Conditional Generative Adversarial Nets [37]. Before training, the label data (the category the point cloud belongs to) is added to each point cloud, for example in PCN dataset (1: Plane, 2: Cabinet, 3: Car, 4: Chair, 5: Lamp, 6: Sofa, 7: Table, 8: Vessel). Since the label does not have any effect on the coordinate features and geometric information of the point cloud, and the label is only used when training sc-GAN, this also guarantees fairness when compared with other methods. For each incomplete point cloud, such as a plane, we generate 150 planes according to random normal distributions, and select the complete point cloud that best matches the current incomplete point cloud appearance from these shapes.

sc-GAN contains two parts, a generator and a discriminator. To ensure reasonable coarse point clouds are generated from the potential vector z, we give it a category condition y and use the following loss function for the generator:

$$L_G = -\mathop{\mathbb{E}}_{z\sim\mathcal{Z}}[D(G(z|y)] \ , \tag{3}$$

where G and D denote the generator and discriminator respectively, and $\mathcal{Z}$ represents the latent code distribution. As in tree-GAN, we design $\mathcal{Z}$ with a normal distribution. For the sake of preventing GAN from pattern collapse during training, we also add a gradient penalty, which is proposed in WGAN-GP [49]. So the loss function for the discriminator is shown as follows:

$$\begin{aligned} L_D = &\mathop{\mathbb{E}}_{z\sim\mathcal{Z}}[D(G(z|y))] - \mathop{\mathbb{E}}_{x\sim\mathcal{R}}[D(x|y)] \\ &+\lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right] \ . \end{aligned} \tag{4}$$

In the above loss function, x indicates the real point cloud, and $\mathcal{R}$ represents the real point cloud distribution. y represents the category information of the point cloud data. λ is a weighting parameter that we set to 10 in the experiment. $\hat{x}$ are sampled from line segments between real and fake point clouds, and the distribution satisfied by the sampling results is denoted as $\mathbb{P}_{\hat{x}}$.

C. RL Based 2D Grid Deformation

Experiments show that different point clouds require different 2D grid scales. But none of the prior methods using folding-based decoder make improvement on this important aspect. They all use a fixed 2D grid scale, which is 0.05. So next we detail the deformation process of 2D grid and how to use reinforcement learning to choose the right scale for each point cloud to generate an adaptive 2D grid.

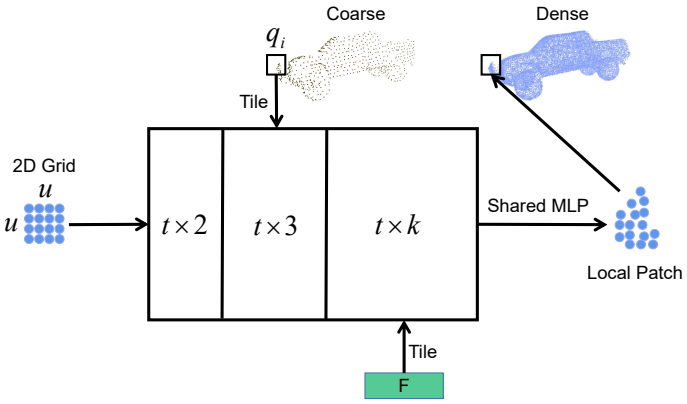


Fig. 5. The process of 2D Grid Deformation. q_i denotes each point in the coarse point cloud and k means the dimension of global feature F extracted by encoder. After a shared MLP, the 2D Grid will deform into a local patch of $t = u^2$ points.

1) 2D Grid Deformation: 2D grid deformation is a coarse-to-fine method commonly used in the field of point cloud completion. The specific process is shown in Fig. 5. firstly, we take points on a zero-centered $u \times u$ grid of side length s (s can control the expansion range of each point, that is, the diffusion ratio of the local patch) and transform it into a matrix of $t \times 2$, which contains the 2D coordinates of the 2D Grid, here $t = u^2$. Then, for each point q_i in the coarse point cloud, we tile it t times (convert it from a 1×3 matrix to a $t \times 3$ matrix). The same is true for the $1 \times k$ global feature F extracted by the encoder, which is tiled into a $t \times k$ matrix. After that, all these matrices are concatenated together to form a $t \times (k + 5)$ matrix and sent into a shared multilayer perceptron to obtain a $t \times 3$ matrix (the q_i-centric local patch).

2) RL Environment Construction: In RL, the environment model simplifies the real environment transformation model. The simplified method is to assume the Markov property of the state transformation, that is, the probability of transforming into the next state s' is only related to the previous state s, and has no correlation with the earlier states. The formula is:

$$P^a_{ss'} = \mathbb{E}\left(S_{t+1} = s' \mid S_t = s, A_t = a\right) \ , \tag{5}$$

where a represents the action at the current state s. Solving RL problems means to find an optimal policy π_* for individuals to

always get more reward than other policies. We formulate the 2D grid scale decision in shape completion as a RL framework as shown in Fig. 6.

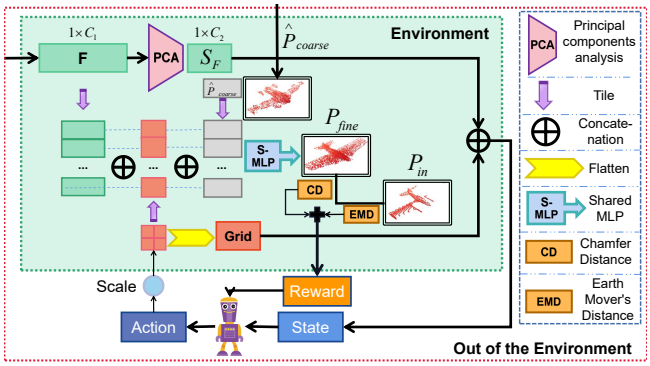


Fig. 6. Overview of environment settings of our RLGrid. The environment contains the fine point cloud generation module (folding-based decoder) of AE. The global feature F extracted by the encoder is $1 \times C_1(1920)$. Subsequently, the feature will be reduced to S_F by PCA, which is $1 \times C_2(128)$. The agent interacts with the environment. After seriatim selecting the appropriate scale, the S_F and Grid are concatenated as the state, and the corresponding rewards are obtained after completion of the complement. Through the agent, we can select the most suitable expansion range for each point cloud sample.

In our settled environment, F is extracted by AE and its dimension is 1920, and the default scale of 2D grid is $scale = 0.05$. To prevent the state space from being too large as well as to improve the inference speed of the network, we firstly use principal component analysis (PCA) to reduce F to S_F (from 1920 dim to 128 dim), and then flatten the 2D grid (suppose the 2D grid size is 2×2, it will organize the four point coordinates into a 4×2 matrix, and then reshape to a 8-dim vector). We concatenate the S_F and flattened 2D grid to be the **state**. Then according to the current state, the agent selects the appropriate **action**, *i.e.*, the scale of the output patch, and interacts with the environment. In detail, the new 2D grid is combined with F and the $\hat{P}_{\text{coarse}}$, and the fine point cloud is generated by the fine module (folding-based decoder) of AE. After the CD and Earth Mover's Distance (EMD) [50] between the fine complete point cloud and incomplete input are calculated, we use the weighted sum of these two indicators as the **reward** to return to the agent. The EMD is calculated as follows:

$$EMD\left(S_1, S_2\right) = \min_{\phi: S_1 \to S_2} \frac{1}{|S_1|} \sum_{x \in S_1} \|x - \phi(x)\|_2 \ . \tag{6}$$

(6) finds a bijection $\phi : S_1 \to S_2$ which minimizes the average distance between corresponding points. In practice, finding the optimal ϕ is too expensive. So we use an iterative $(1+\epsilon)$ approximation scheme [51].

With the above steps, the agent chooses 2D grid scale based on the current reward and repeats the process to get higher reward. When the number of steps exceeds the max_steps or the CD between the output complete point cloud and GT is less than the preset threshold, it will jump out of the current episode. As illustrated in the refinement process of the plane shape in Fig. 7, each point cloud will go through multiple scale selections in a serialized manner. The **reward** r is set as follows:

Algorithm 1 Training the Agent of RLGrid

Require:
Agent Input: State (s_t): $s_t = S_F \oplus \text{Grid}$
Reward (r_t): Calculated using Eq. (7)
Other Input: Coarse complete point cloud $\hat{P}_{coarse}$.
Ensure:
Agent Output: Action (a_t): $a_t = scale$;
Final Output: Fine complete point cloud P_{fine}.
Procedure:
Initialize environment of **DDPG**
Initialize **Critic** $Q(s, a \mid \theta^Q)$ and **Actor** $\mu(s \mid \theta^\mu)$ with weights θ^Q and θ^μ.
Initialize **Target Network** Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer **R**
for $episode < max\_episodes$ **do**
  Get incomplete point cloud P_{in}
  Use AE to create an initial state s_1
  for $t_{steps} < max\_steps$ **do**
    if $episode < start\_episodes$ **then**
      Randomly select an action a_t
    else
      Use **Actor Network** to select action a_t
    end if
    Interact with the environment based on current state s_t and action a_t, get reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in **R**
    if $R_{length} > start\_memory\_size$ **then**
      Train the **Critic and Actor** Network
      Update the **Target Network**:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    end if
  end for
end for

$$\begin{aligned} r_{CD} &= -L_{CD} \ , \\ r_{EMD} &= -L_{EMD} \ , \\ r &= \lambda_{CD} \cdot r_{CD} + \lambda_{EMD} \cdot r_{EMD} \ , \end{aligned} \tag{7}$$

where λ_{CD} and λ_{EMD} are the corresponding weights assigned to each loss function. In the experiment, we set them to 50 and 100 respectively.

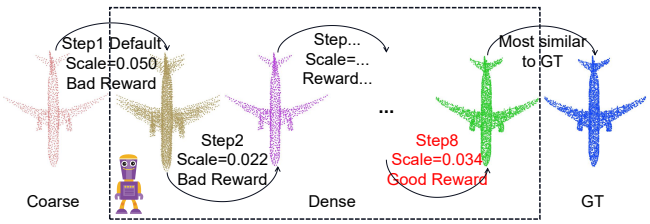


Fig. 7. An example of scale selection using RL. The points of Coarse, Dense and GT are 1024, 4096 and 4096 respectively. The agent selects the appropriate scale according to the current scale and shape in serialized form and obtains the corresponding reward.

The above settings satisfy the Markov decision process [52], and the action space is continuous. Therefore, deep deterministic policy gradient (DDPG) [45] is selected as the algorithm for training the agent. In DDPG, a parameterized actor network $\mu(s \mid \theta^{\mu})$ learns a particular policy and maps states to particular actions in a deterministic manner. The critic network $Q(s, a \mid \theta^{\mu})$ uses the Bellman equation and provides a measure of the quality of action and the state. The actor network is trained by finding the expected return of the gradient of the cost J w.r.t the actor-network parameters, which is also known as the policy gradient. It can be expressed as follows:

$$\nabla_{\theta^{\mu}} J(\theta) = \mathbb{E}_{s_t \sim \rho^{\beta}} \Big[\nabla_{\alpha} Q\left(s, a \mid \theta^{Q}\right)\big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_{\mu}} \mu\left(s \mid \theta^{\mu}\right)\big|_{s=s_t} \Big] . \tag{8}$$

While initializing the actor and critic network, we initialize the target actor μ' and target critic Q' with the same parameters as μ and Q, respectively. And we use the parameters of μ and Q to update the parameters of the corresponding target network after a certain number of times to ensure the convergence of the parameters. Besides, DDPG uses a replay buffer R to eliminate the strong correlation between input experiences. Here, R refers to a quadruple (s_t, a_t, r_t, s_{t+1}), using the replay buffer greatly improves the utilization rate of samples and the efficiency of learning.

Before training the agent, we pre-train AE and sc-GAN untill the best performance and reserve the best performing models. The initial state and reward in the environment depend on the results of the coarse point cloud network. The detailed training process is shown in Algorithm 1.

IV. Experiments

In this section, we firstly compare our method with several state-of-the-art point cloud completion schemes on a widely used benchmark: PCN [7]. To demonstrate the advancement of our method, we also transform the original PCN dataset into a homemade ball-hole PCN dataset using the missing-making approach adopted in PF-Net [12] and experiment on it. For evaluation, we adopt the L1 version of CD, which follows the same practice as previous methods. After that, we also validate our method on the KITTI [20] car dataset. To further evaluate the generalization performance of RLGrid, we train and test on ShapNet-55/34 [17], which has more varieties and is more challenging. Finally, we perform ablation experiments on each module in RLGrid to demonstrate their effectiveness.

A. Detailed Setting of RL

The network architectures of the Actor and Critic are presented in Table I and Table II. The input and output dimensions of each layer and the activation function used are given in detail.

We also list the parameter values used for the training of DDPG algorithm in Table III.

TABLE I
THE NETWORK ARCHITECTURE OF THE ACTOR.

Name	InRes	OutRes	Input	Activation
Linear-L1	128×136	128×400	input	ReLu
Linear-L2	128×400	128×400	Linear-L1	ReLu
Linear-L2-Add	128×400	128×300	Linear-L2	ReLu
Linear-L3	128×300	128×1	Linear-L2-Add	Tanh

TABLE II
THE NETWORK ARCHITECTURE OF THE CRITIC.

Name	InRes	OutRes	Input	Activation
Linear-L1	128×136	128×400	input	ReLu
Linear-L2	128×401	128×300	Linear-L1	ReLu
Linear-L2-Add	128×300	128×300	Linear-L2	-
Linear-L3	128×300	128×1	Linear-L2-Add	-

B. Evaluation on the original PCN dataset

In this phase of the experiments, we used random sampling method to sample the original 16384 points to 4096 points. Like ASFM-Net [14], the final completion results were evaluated at 4096 points. Quantitative and qualitative results are shown in Table IV and Fig. 8, respectively. Table IV shows that RLGrid achieves the lowest CD values in five categories and slightly higher than GRNet [54], PMP-Net++ [55] and PoinTr [17] in the three categories of cabinet, lamp and sofa. In order to better show the refinement process of each method, we divided the completed results into two parts, *i.e.*, a coarse point cloud (1024 points) and a refined point cloud (4096 points) for visualization, among which SnowflakeNet [16] and PoinTr [17] do not contain coarse output with 1024 points, so we only show their full point cloud with 4096 points. As we can see from Fig. 8, the RLGrid-completed point cloud retains much details, and the shape is also the closest to GT. For example, the watercraft in the last line, except for our method, GRNet and PoinTr, the remaining methods (e.g., CRN [13]) can only generate the general shape, but cannot obtain the details, and for the watercraft completed by GRNet and PoinTr, the final shape also has a certain gap with GT. In Fig. 13, we show more visualization results of RLGrid on the original PCN dataset. It also shows that our method can retain more details while completing the point cloud. We also show the error between the shapes completed by different methods and GT in Fig. 9. For example, the lamp in the second row, it can be seen that the shape predicted by RLGrid and PointTr is correct, but for PoinTr, many points have a large position offset. The points

TABLE III
THE PARAMETER VALUES USED TO TRAIN THE RL AGENT.

Parameter	Value
max episode to run environment	$1e6$
max step in one episode	20
max action	3
episode for starting training Actor and Critic	$3e3$
exploration noise	0.1
number of samples from replay-buffer	128
discount γ	0.99
noise added to policy during Critic update	0.2
range to clip target policy noise	0.005
frequency for delayed policy update	2
target network update factor τ	0.001

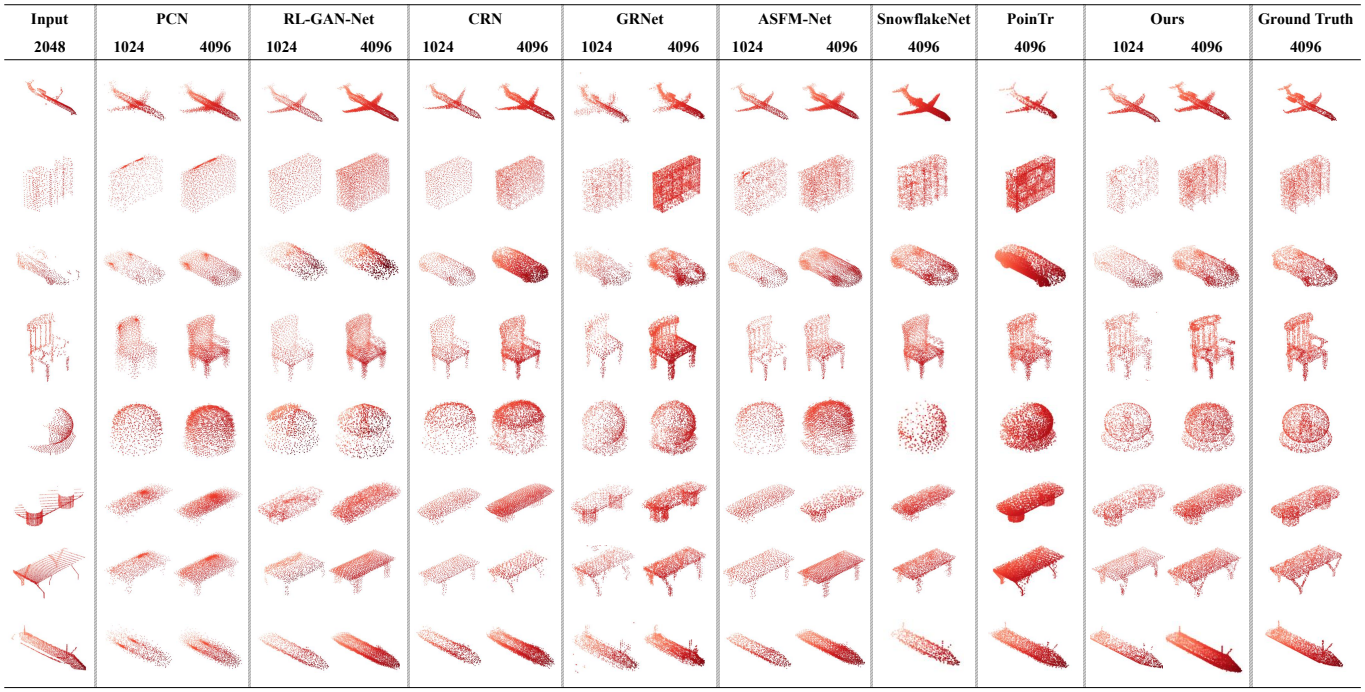


Fig. 8. Qualitative comparisons on eight categories of the original PCN dataset. The point number of input is 2048, the point number of coarse output is 1024, and the point number of fine output and GT is 4096.

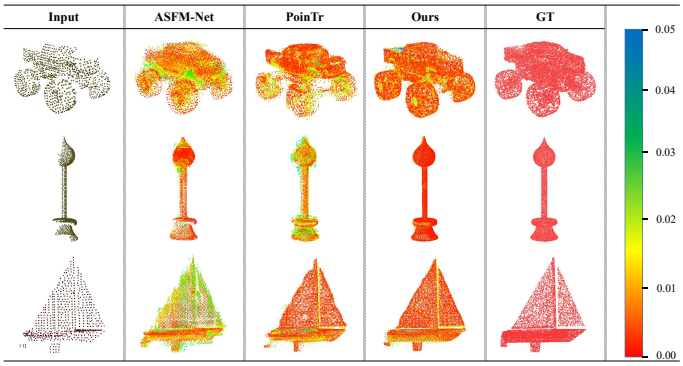


Fig. 9. Heatmap of RLGrid and other methods on the original PCN dataset. The closer the color of the predicted point is to red, the smaller the error of the point is, and the closer to blue is to indicate a larger error of the point.

predicted by our method are basically consistent with GT, which further illustrates the effectiveness of using RL agent to select 2D grid scales.

C. Evaluation on the ball-hole PCN dataset

To verify the robustness of our approach, we use the data generation method in PF-Net to randomly select a viewpoint as the center for each complete point cloud in the original PCN dataset, and perform point deletion operations with a fixed radius (In this experiment, we generated incomplete point cloud missing 30%, 40% and 50% of the original data for test). We extracted the categories of airplane, car, chair, lamp, and table as benchmarks for qualitative comparisons. As shown in Table V, Table VI and Table VII, when other methods face different level of ball-hole missing, the point cloud completion ability drops significantly, while our method does not drop significantly in overall performance. For instance, RL-GAN-Net [46], which also uses RL, in the case of 30%, 40% and 50% missing points, the average CD of the listed five categories increases by 48.93%, 60.83% and 65.85% compared to the results of the original PCN dataset, respectively. But ours only increases by 17.13%, 27.98% and 35.24%, respectively. From

TABLE IV
QUANTITATIVE COMPARISON OF THE ORIGINAL PCN DATASET. POINT RESOLUTIONS FOR THE FINE OUTPUT AND GT ARE 4096.

Methods	Avg. CD (10^{-3}), lower is better								
	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
PCN [7]	8.53	16.60	14.11	16.32	16.64	17.93	14.71	15.03	14.98
RL-GAN-Net [46]	7.97	15.44	13.89	16.01	15.45	17.44	14.33	14.64	14.40
MSN [30]	7.41	15.69	13.70	15.85	14.88	17.58	14.10	14.14	14.17
PMP-Net [53]	7.61	15.34	13.66	15.77	12.01	17.54	13.76	13.82	13.69
CRN [13]	7.34	15.10	13.40	15.65	14.37	17.06	13.15	13.26	13.67
GRNet [54]	6.96	**14.25**	13.15	14.38	12.55	16.24	12.48	11.69	12.71
ASFM-Net [14]	6.65	14.47	12.87	14.11	12.10	15.69	12.23	11.75	12.48
PMP-Net++ [55]	6.55	14.95	13.08	13.89	**11.06**	15.99	12.41	11.39	12.42
SnowflakeNet [16]	6.44	14.93	13.01	13.83	11.45	15.71	12.30	11.37	12.38
PoinTr [17]	6.57	15.10	13.21	13.97	11.14	**15.28**	12.18	11.45	12.36
Ours	**6.32**	14.33	**12.49**	**13.65**	11.98	15.82	**12.17**	**11.32**	**12.26**

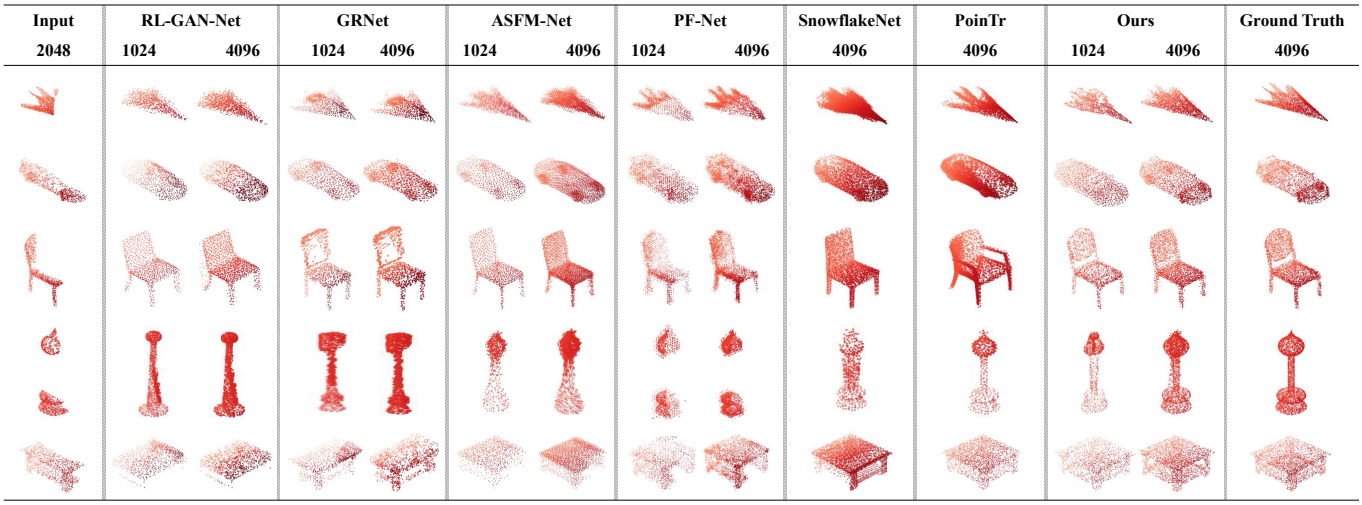


Fig. 10. Qualitative comparisons on five categories of the ball-hole PCN dataset (The point loss rate is 40%). The point number of input is 2048, the point number of coarse output is 1024, and the point number of fine output and GT is 4096.

the visualization results in Fig. 10, we can also intuitively see that our method can restore the overall shape of the point cloud well and contains a lot of details. For example, for the lamp in the fourth row, PF-Net fails to restore the overall shape, RL-GAN-Net and GRNet predicts the shape of the lamp incorrectly, and the rest of the methods do not achieve good results on the details of the base of the lamp, while our method can better restore its shape and details.

TABLE V
QUANTITATIVE COMPARISON ON THE BALL-HOLE PCN DATASET OF 30% POINT MISSING.

Methods	Avg. CD (10^{-3}), lower is better					
	Plane	Car	Chair	Lamp	Table	Avg.
RL-GAN-Net [46]	15.83	20.17	22.01	22.64	19.99	20.15
GRNet [54]	12.31	16.23	18.17	20.09	16.01	16.56
ASFM-Net [14]	11.08	16.73	17.35	18.00	17.29	16.09
PF-Net [12]	10.47	16.15	15.70	16.89	16.96	15.23
SnowflakeNet [16]	10.10	16.00	15.13	15.88	16.09	14.64
PoinTr [17]	10.01	15.96	15.00	15.59	15.98	14.51
Ours	**9.93**	**15.89**	**14.85**	**15.30**	**15.83**	**14.36**

TABLE VI
QUANTITATIVE COMPARISON ON THE BALL-HOLE PCN DATASET OF 40% POINT MISSING.

Methods	Avg. CD (10^{-3}), lower is better					
	Plane	Car	Chair	Lamp	Table	Avg.
RL-GAN-Net [46]	16.44	22.49	23.10	24.67	22.08	21.76
GRNet [54]	13.87	18.10	20.06	22.45	17.52	18.40
ASFM-Net [14]	12.63	18.79	19.35	19.98	18.60	17.87
PF-Net [12]	11.99	17.55	16.70	18.70	18.13	16.61
SnowflakeNet [16]	11.64	17.23	16.44	17.71	17.92	16.19
PoinTr [17]	11.30	17.11	16.40	17.15	17.63	15.92
Ours	**11.13**	**16.86**	**16.39**	**16.88**	**17.21**	**15.69**

TABLE VII
QUANTITATIVE COMPARISON ON THE BALL-HOLE PCN DATASET OF 50% POINT MISSING.

Methods	Avg. CD (10^{-3}), lower is better					
	Plane	Car	Chair	Lamp	Table	Avg.
RL-GAN-Net [46]	17.24	22.98	24.00	25.19	22.77	22.44
GRNet [54]	14.35	18.78	20.65	22.93	18.08	18.96
ASFM-Net [14]	13.33	19.80	20.22	20.67	19.61	18.73
PF-Net [12]	13.08	18.95	17.82	20.03	19.44	17.86
SnowflakeNet [16]	12.27	18.20	17.41	18.66	19.11	17.13
PoinTr [17]	11.87	18.00	17.25	**17.91**	18.59	16.72
Ours	**11.76**	**17.83**	**17.17**	17.93	**18.21**	**16.58**

D. *Evaluation on the ShapeNet-55/34 dataset*

We also evaluate our model on two challenging datasets, ShapeNet-55 and ShapeNet-34, proposed in PoinTr [17]. ShapeNet-55 contains point cloud data of 55 categories, and we use 41,952 models for training and 10,518 models for testing. Each complete point cloud in PoinTr contains 8192 points. ShapeNet-34 regards 34 categories in ShapNet-55 as seen categories, and the remaining 21 categories as unseen categories. In the seen category, 100 objects are randomly selected from each category to construct a test set of seen categories (3400 objects in total), and the rest are used as the training set. Meanwhile, another test set is constructed, consisting of 2305 objects from 21 unseen categories. For consistency with other experiments, we downsample the complete point cloud to 4096 points as GT on this basis. At the time of testing, like ball-hole PCN, we fixed the viewpoint and deleted 25%, 50% and 75% of the points corresponding to the three difficulties of simple, medium and hard.

We list the quantitative performance of several methods on ShapeNet-55 and ShapeNet-34 in Tables VIII and IX, respectively. It can be seen from Table VIII that RLGrid achieves the best performance on most listed categories and makes lowest CD on the three difficulty settings and exceeds PoinTr by 13.3% on average CD. What is more worth mentioning is that the experimental results in Table IX show that RLGrid achieves the lowest CD on both 34 seen categories and 21 unseen categories, surpassing PoinTr by 11.9% and 9.8% respectively. These experimental results show that RLGrid can still maintain a good effect when facing large-area missing and various types of point cloud data, and its generalization performance is also relatively good.

E. *Evaluation on the KITTI car dataset*

After the above three experiments, we test the effect of car completion on the KITTI [20] car dataset. It contains 2483 partial car point clouds, which are collected from 98 real cars under 425 different frames, and there is no GT.

TABLE VIII
RESULTS OF OUR METHODS AND STATE-OF-THE-ART METHODS ON SHAPENET-55. WE REPORT THE DETAILED RESULTS FOR EACH METHOD ON 10 CATEGORIES AND THE OVERALL RESULTS ON 55 CATEGORIES FOR THREE DIFFICULTY DEGREES. WE USE CD-S, CD-M AND CD-H TO REPRESENT THE CD RESULTS UNDER THE *Simple*, *Moderate* AND *Hard* SETTINGS, RESPECTIVELY.

	Table	Chair	Airplane	Car	Sofa	Bird House	Bag	Remote	Key Board	Rocket	CD-S	CD-M	CD-H	CD-Avg
RL-GAN-Net	6.47	5.41	3.78	6.05	5.35	9.43	7.54	5.17	4.46	4.00	6.63	7.10	7.76	7.16
ASFM-Net	6.01	6.79	5.01	6.31	6.43	6.45	5.24	4.10	4.60	4.47	4.40	5.94	7.39	5.91
GRNet	4.28	3.71	2.94	4.45	3.83	4.97	4.19	3.01	2.82	3.87	2.41	3.95	5.44	3.93
PoinTr	2.23	**0.96**	1.32	2.76	**0.62**	2.43	**0.67**	2.40	1.41	2.31	1.27	1.74	2.39	1.80
Ours	**1.76**	1.20	**1.09**	**1.58**	1.10	**1.92**	1.17	**1.18**	**0.77**	**1.89**	**1.03**	**1.50**	**2.16**	**1.56**

TABLE IX
RESULTS OF OUR METHODS AND STATE-OF-THE-ART METHODS ON SHAPENET-34. WE REPORT THE RESULTS OF 34 SEEN CATEGORIES AND 21 UNSEEN CATEGORIES IN THREE DIFFICULTY DEGREES. WE USE CD-S, CD-M AND CD-H TO REPRESENT THE CD RESULTS UNDER THE *Simple*, *Moderate* AND *Hard* SETTINGS, RESPECTIVELY.

	34 seen categories				**21 unseen categories**			
	CD-S	CD-M	CD-H	CD-Avg	CD-S	CD-M	CD-H	CD-Avg
RL-GAN-Net	6.02	6.78	7.03	6.61	6.41	7.84	9.37	7.87
ASFM-Net	4.17	5.64	6.88	5.56	5.88	7.32	8.85	7.35
GRNet	2.08	3.61	5.01	3.57	3.26	4.92	6.49	4.89
PoinTr	1.03	1.44	2.07	1.51	2.53	3.58	4.63	3.58
Ours	**0.89**	**1.23**	**1.87**	**1.33**	**2.27**	**3.31**	**4.12**	**3.23**

We use Unidirectional Chamfer Distance (UCD) and Unidirectional Hausdorff Distance (UHD) as evaluation metrics. UCD measures the squared L2 distance from the partial input shape x_p (x_A) to the complete output x_c (x_B), as shown in (9).

$$d_{UCD}\left(\mathbf{x}_A, \mathbf{x}_B\right) = \min_{\phi:\mathbf{x}_A \to \mathbf{x}_B} \frac{1}{|\mathbf{x}_A|} \sum_{p \in \mathbf{x}_A} \|p - \phi(p)\|_2 \ . \quad (9)$$

UHD, similarly, measures the single-sided Hausdorff distance:

$$d_{UHD}\left(\mathbf{x}_p, \mathbf{x}_c\right) = \max_{p \in \mathbf{x}_p} \min_{q \in \mathbf{x}_c} \|p - q\|_2^2 \ . \quad (10)$$

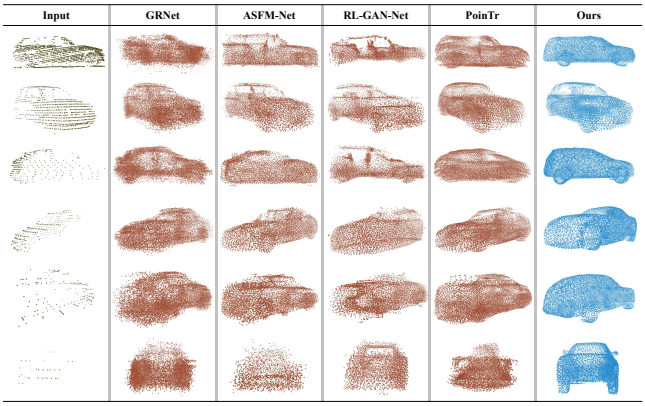


Fig. 11. Qualitative comparison on the KITTI car dataset. From left to right: raw point clouds of the same vehicle scanned in consecutive frames, shape completion based on GRNet, ASFM-Net, RL-GAN-Net, PoinTr and our method RLGrid. From top to bottom: the number of input points gradually decreases.

It can be seen from Fig. 11 that our RLGrid performs well on the KITTI car dataset. Regardless of the input point cloud containing rich or rare information, it can generate the most reasonable shape among the comparative methods. In Table X, we show the UCD and UHD [35] values of these methods on the KITTI car dataset. Compared with GRNet, ASFM-Net and PoinTr, which performed well before, our method improves UCD by at least 33.4% and UHD by at least 26.9%, which is more than sufficient to demonstrate the superiority of our method.

TABLE X
COMPLETION RESULTS EVALUATED USING UCD AND UHD, WHERE UCD IS SCALED BY 10^4 AND UHD IS SCALED BY 10^2. THESE TWO METRICS ARE CALCULATED FROM INPUT TO DENSE OUTPUT.

Methods	UCD (in →den) ↓	UHD (in →den) ↓
GRNet [54]	8.17	16.50
ASFM-Net [14]	8.58	17.11
RL-GAN-Net [46]	13.19	20.83
PoinTr [17]	6.51	15.03
Ours	**5.44**	**12.06**

F. Ablation Study

Ablation experiments are conducted to demonstrate the effectiveness of AE, sc-GAN and RL module in RLGrid. All experiments are performed on the two PCN datasets and evaluated using the L1 version of CD.

The AE Module. We tried many kinds of autoencoders on our framework, especially for the feature extraction part, we used the feature extraction module of PointNet [23], PCN [7] and PF-Net [12] for training and testing. Table XI shows the effect of different feature extraction methods on the original PCN dataset after being applied to RLGrid. We show quantitative results for five categories and Avg. CD is the average CD over eight categories.

TABLE XI
QUANTITATIVE RESULTS ON THE ORIGINAL PCN DATASET AFTER COMBING RLGRID WITH DIFFERENT FEATURE EXTRACTION MODULES.

Methods	Avg. CD (10^{-3}), lower is better					
	Plane	Car	Chair	Lamp	Table	Avg.
RLGrid + PointNet [23]	7.65	14.08	15.57	13.98	14.17	13.48
RLGrid + PCN [7]	6.65	12.73	13.89	12.44	12.51	12.59
RLGrid + PFNet [12]	**6.32**	**12.49**	**13.65**	**11.98**	**12.17**	**12.26**

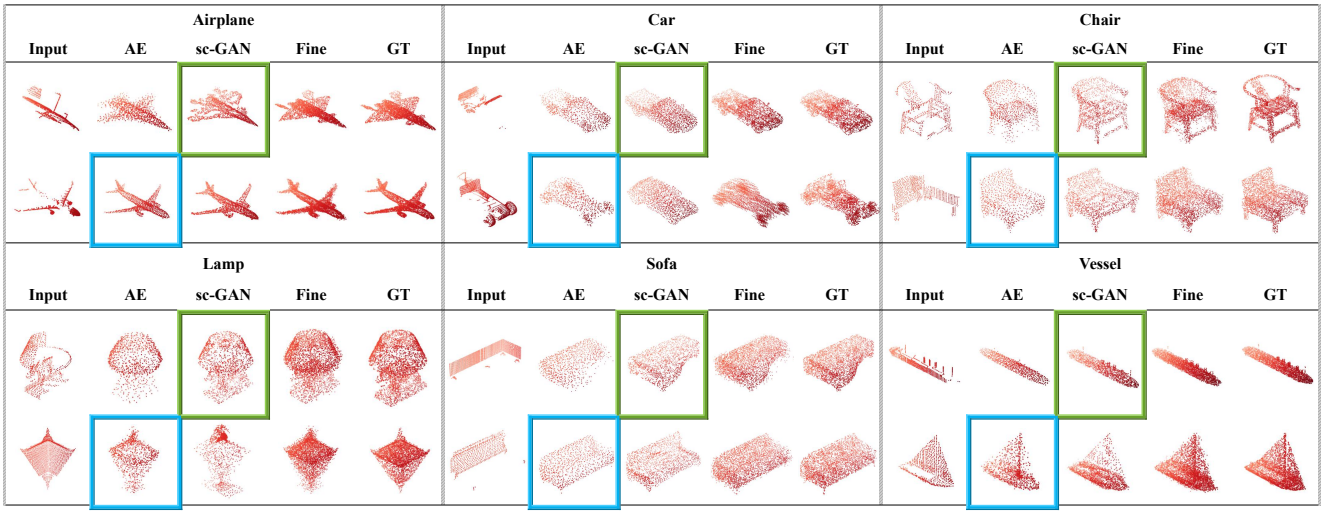


Fig. 12. Some intermediate process of RLGrid. The green box represents that the better coarse output is generated by sc-GAN, while the blue box represents that the better coarse output is generated by AE.

The sc-GAN Module. In this part of the experiment, we evaluate the effectiveness of sc-GAN. What we do is to remove it from RLGrid, which means that the coarse output in our network now only comes from the AE branch, and the final effect is shown in the lower part of Table XII. Due to the lack of the shape generated by sc-GAN, RL agent can only select the expansion range contingent on the results of AE. For some point clouds, AE cannot generate a reasonable frame from the incomplete point cloud, especially when there are too many missing points. Therefore, the CD between the final refined complete point cloud and the GT increases, which proves the effectiveness of the sc-GAN module.

In order to more clearly show the selection process of RLGrid in the coarse part, we counted how much coarse output came from AE or sc-GAN under each category on the original PCN dataset and ball-hole PCN dataset. There are 150 point cloud data in the test set for each category. We list in Table XIII the number of coarse output derived from AE and sc-GAN for each category on the two datasets. From the data in the table, we can see that AE performs better on the original PCN dataset, but when faced with ball-hole data, sc-GAN has more advantages. Fig. 12 also shows the coarse output generated by AE and sc-GAN in some categories. The blue box indicates that the result of selecting AE is sent to the RL environment, and the green box indicates that the result of selecting sc-GAN is sent to the RL environment. For example, in the category of lamps, in the first row, the lamp generated by sc-GAN retain the base information in the Input when completing the shape, while the results of AE are not particularly ideal. In the second row, since the lamps generated by sc-GAN have extra parts (there is no base information in Input, and sc-GAN still adds bases to them), therefore, AE-generated lamp have an advantage when calculating CD. This further justifies the comparison with the input without GT, and the generated shape can better preserve some of the features and details contained in the input.

The RL Module. To verify the effectiveness of our proposed RL module and its plug-and-play advantage, we add the RL module to the existing methods of FoldingNet [11], PCN [7] and ASFM-Net. We only let the agent participate in the decision-making process of 2D grid features. The training process for our RLGrid on these models is shown in Fig. 14. Taking the airplane category as an example, all of the methods are trained with 6×10^5 episodes. Fig. 14(a) is the reward obtained in the process, and Fig. 14(b) is the change trend of the CD in the process. It can be seen from the two figures,

TABLE XII
ABLATION EXPERIMENTS OF RL AND SC-GAN. THE UPPER PART SHOWS THE EFFECT OF RL MIGRATED TO SEVERAL EXISTING NETWORKS THAT HAVE USED 2D GRID, GREEN ARROWS INDICATE CD DECREASE AND RED ARROWS INDICATE CD INCREASE. THE LOWER PART SHOWS THE EFFECT OF OUR RLGRID REMOVING THE RL AND SC-GAN PARTS RESPECTIVELY.

Methods	Avg. CD (10^{-3}), lower is better								
	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
FoldingNet [11]	10.26	19.17	16.98	17.10	16.83	20.01	16.37	15.66	16.55
FoldingNet w/ RL	8.04 ↓	16.83 ↓	14.10 ↓	15.66 ↓	13.58 ↓	17.44 ↓	15.17 ↓	12.69 ↓	14.19 ↓
PCN [7]	8.53	16.60	14.11	16.32	16.64	17.93	14.71	15.03	14.98
PCN w/ RL	6.97 ↓	15.26 ↓	13.40 ↓	14.77 ↓	13.22 ↓	16.53 ↓	12.86 ↓	12.10 ↓	13.14 ↓
ASFM-Net [14]	6.65	14.47	12.87	14.11	12.10	15.69	12.23	11.75	12.48
ASFM-Net w/ RL	6.43 ↓	**14.16** ↓	12.60 ↓	14.25 ↑	11.87 ↓	15.88 ↑	12.34 ↓	11.54 ↓	12.38 ↓
PoinTr [17]	6.57	15.10	13.21	13.97	11.14	15.28	12.18	11.45	12.36
PoinTr w/ RL	**6.29** ↓	14.49 ↓	12.78 ↓	**13.40** ↓	**11.46** ↑	**15.10** ↓	12.50 ↑	**11.31** ↓	**12.17** ↓
RLGrid w/o RL	7.20	15.61	13.34	15.22	13.43	16.57	12.98	12.36	13.34
RLGrid w/o sc-GAN	6.59	14.40	12.65	14.08	12.11	15.73	12.19	11.76	12.44
RLGrid	6.32	14.33	**12.49**	13.65	11.98	15.82	**12.17**	11.32	12.26

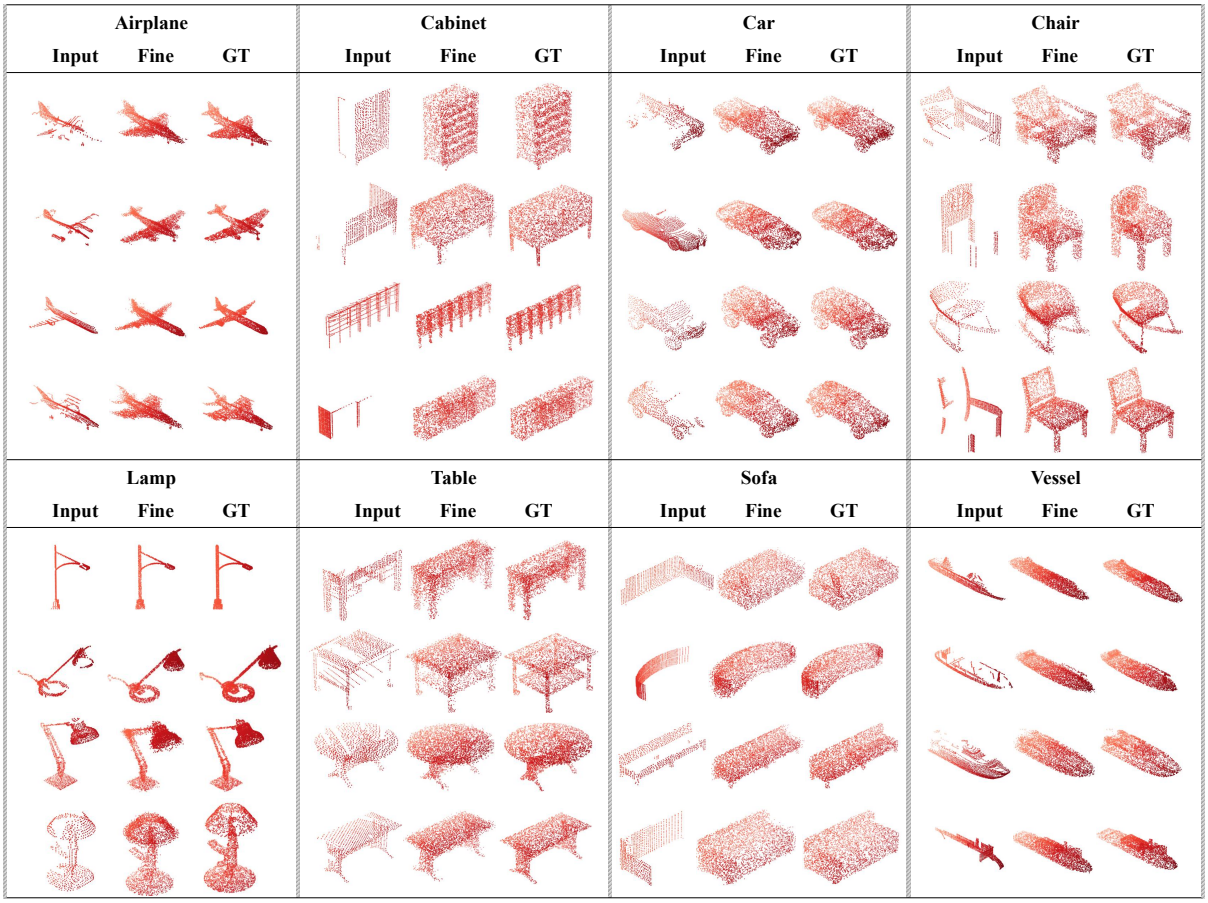


Fig. 13. More experimental results of RLGrid on original PCN dataset.

TABLE XIII
SOURCE STATISTICS FOR COARSE OUTPUT, THE UPPER PART IS THE STATISTICAL RESULTS OF THE ORIGINAL PCN DATASET, AND THE LOWER PART IS THE STATISTICAL RESULTS OF THE BALL-HOLE PCN DATASET.

Source	The number of coarse output from each category on original PCN dataset							
	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel
AE	**97**	**84**	**101**	**93**	**83**	**113**	**108**	**99**
sc-GAN	53	66	49	57	67	37	42	51

Source	The number of coarse output from each category on ball-hole PCN dataset							
	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel
AE	42	53	41	51	49	33	50	56
sc-GAN	**108**	**97**	**109**	**99**	**101**	**117**	**100**	**94**

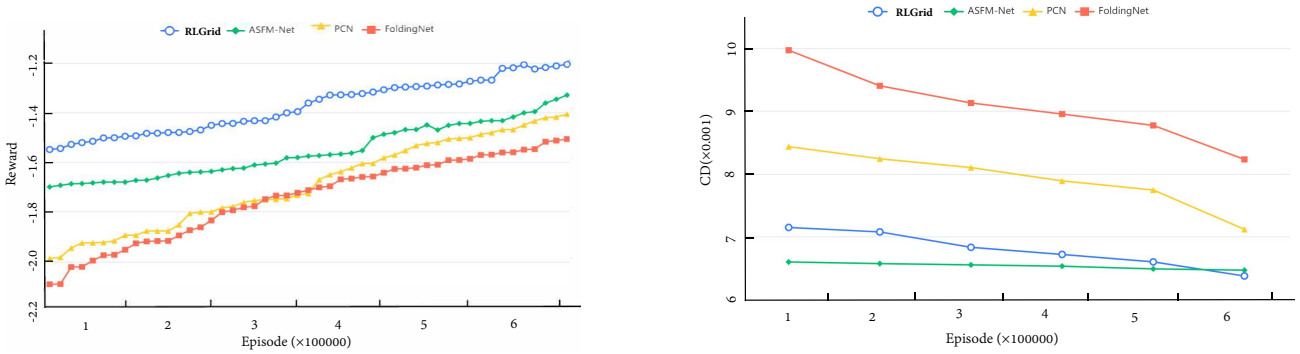
since the agent uses random actions with a certain probability, the reward rises with slight fluctuations, while the CD for each method decreases steadily. This demonstrate that the proposed RL module used for selecting the optimal grid scale do improve the performance of current methods.

TABLE XIV
ABLATION STUDY ON MAX STEP OF RL PER EPISODE.

max_step	Convergence Episode	CD On PCN
5	8.44e5	13.15
20	9.13e5	**12.26**
50	9.47e5	12.41
100	9.79e5	12.78

The quantitative results are shown in the upper part of Table XII, in which the green arrow indicates the CD decreases, *i.e.*, the completion performance is improved, and the red arrow indicates that CD increases, *i.e.*, the completion effect may decrease. Taking FoldingNet, PCN and ASFM-Net as benchmarks, we found that the average CD is decreased by 14.25% when the RL module was used on FoldingNet, and the average CD is decreased by 12.28% when used on PCN. As for ASFM-Net, although the average CD of chairs and sofas increased slightly, the overall average CD still declined. From the lower part of Table XII, we can also see that the completion ability of RLGrid drops significantly when the RL module is removed.

Additionally, we also adjust the max_step in a single episode in RL, as shown in Table XIV, a maximum step size of 20 converges quickly and performs best on the original PCN dataset.



(a) Reward (b) CD Loss

Fig. 14. RL performance analysis. We applied RL Module to ASFM-Net, PCN and FoldingNet respectively, and compared (a) Reward and (b) CD. We found that RL Module do improve the performance of each network.

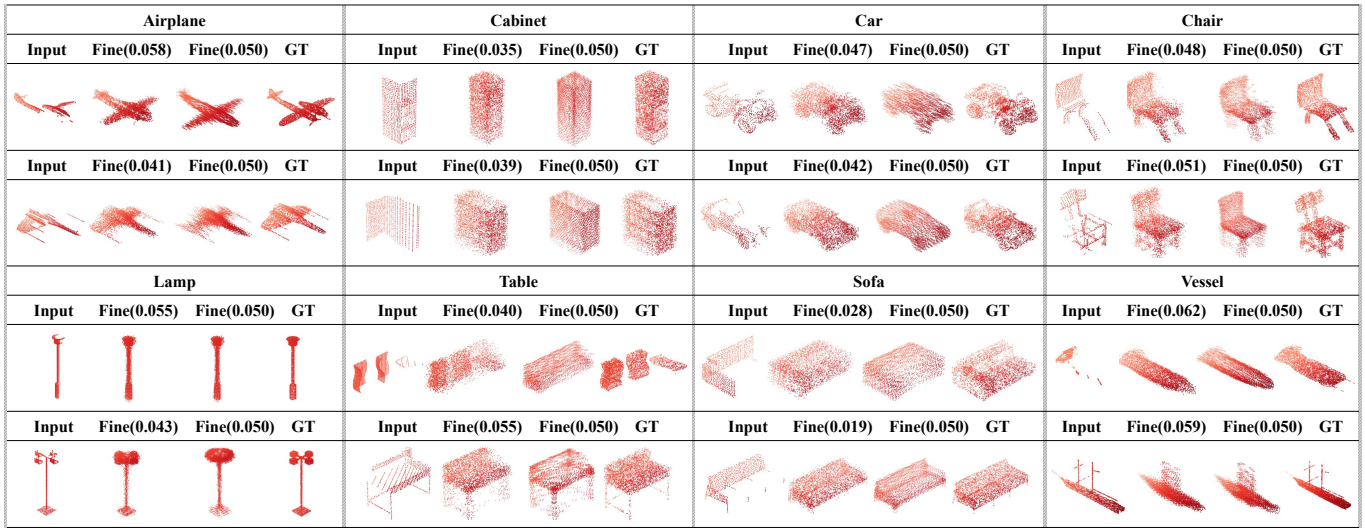


Fig. 15. Some RL Grid failure cases and their corresponding 2D grid scale. For comparison, we also list the refined point cloud generated with the default 2D grid scale (0.050).

G. Complexity Analysis

Our method achieves the best performance on PCN dataset, ShapeNet-55, ShapeNet-34 and KITTI datasets. In Table XV, we list the number of parameters (Params), theoretical computation cost (FLOPs), the average chamfer distances (CD-Avg) of our method and other three methods. From this we can see that our method has the least number of parameters and a faster inference speed, indicating that our method can balance cost and performance. The ablation of the sc-GAN module (RLGrid w/o sc-GAN) also shows that sc-GAN does not bring a lot of additional overhead, but helps the model to improve its performance in the face of datasets with large missing areas and various object types, such as ShapeNet-55 and ShapeNet-34.

TABLE XV
COMPLEXITY ANALYSIS. WE SHOW THE THE NUMBER OF PARAMETER (PARAMS) AND THEORETICAL COMPUTATION COST (FLOPS) OF OUR METHOD AND THREE EXISTING METHODS. WE ALSO PROVIDE THE AVERAGE CHAMFER DISTANCE ON PCN(CD_{pcn}), SHAPENET-55 (CD_{55}) AND SHAPENET-34 (CD_{34}).

Methods	Params	FLOPs	CD_{pcn}	CD_{55}	CD_{34}
GRNet [54]	73.15M	40.44G	12.71	3.93	3.57
ASFM-Net [14]	28.31M	25.21G	12.48	5.91	5.56
PoinTr [17]	30.90M	10.41G	12.36	1.80	1.51
RLGrid	26.42M	11.38G	**12.26**	**1.56**	**1.33**
RLGrid w/o RL	20.67M	8.08G	13.34	3.18	2.89
RLGrid w/o sc-GAN	21.89M	9.64G	12.44	2.91	2.54

H. Failure Case Analysis

Although RLGrid can complete the task of point cloud completion well, there are also some unsatisfactory results. We show some failure cases in Fig. 15. The values in brackets represent the 2D grid scale selected by RL agent and the default 2D grid scale respectively. Since RLGrid only selects one 2D grid scale for each point cloud, in some point clouds that contain both thick and thin parts such as the second failure case of Vessel, the sail part is very thin, so the required expansion range is small, while the hull part requires a larger expansion range. So, one grid scale cannot coordinate these two aspects well, resulting in an unsatisfactory final effect. But it is acceptable that even if RLGrid is not ideal in these shapes, its overall visual effect can still be closer to GT than the refined results of the default 2D grid.

V. Conclusion

In this paper, we propose a new robust network RLGrid specifically for point cloud completion. Our proposed RLGrid is built on the coarse-to-fine completion framework, where we adopted a multi-scale feature extraction AE and the sc-GAN for coarse completion, and designed a new RL environment for the fine completion network. RLGrid uses the 2D grid feature as the state and the inversion of the loss function as the reward, so that the agent can select the most suitable expansion range for each coarse point cloud and finally obtain a complete refined point cloud. Experiments on various datasets demonstrate the superiority of our method, and ablation studies also confirm that our proposed RL module is effective and transferable.

In general, this paper proposes to use reinforcement learning to select the 2D grid scale, but because the shapes of some point clouds are not uniform in thickness, some results are not satisfactory. A potential solution to this problem is to divide the point cloud into patches and give each patch a reasonable grid scale, which we will investigate in our future work.

References

[1] M. Alexa, M. Gross, M. Pauly, H. Pfister, M. Stamminger, and M. Zwicker, "Point-based computer graphics," in *ACM SIGGRAPH 2004 Course Notes*, 2004, pp. 7–es.

[2] L. Kobbelt and M. Botsch, "A survey of point-based techniques in computer graphics," *Computers & Graphics*, vol. 28, no. 6, pp. 801–814, 2004.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[4] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2442–2447.

[5] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *Eurographics 2014-State of the Art Reports*, vol. 1, no. 1, 2014, pp. 161–185.

[6] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "Pu-gcn: Point cloud upsampling using graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 683–11 692.

[7] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737.

[8] K. Li, Y. Wu, Y. Xue, and X. Qian, "Viewpoint recommendation based on object-oriented 3d scene reconstruction," *IEEE Transactions on Multimedia*, vol. 23, pp. 257–267, 2020.

[9] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Transactions on Multimedia*, vol. 24, pp. 1943–1955, 2021.

[10] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.

[11] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 206–215.

[12] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "Pf-net: Point fractal network for 3d point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7662–7670.

[13] X. Wang, M. H. Ang Jr, and G. H. Lee, "Cascaded refinement network for point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 790–799.

[14] Y. Xia, Y. Xia, W. Li, R. Song, K. Cao, and U. Stilla, "Asfm-net: Asymmetrical siamese feature matching network for point completion," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1938–1947.

[15] T. Huang, H. Zou, J. Cui, J. Zhang, X. Yang, L. Li, and Y. Liu, "Adaptive recurrent forward network for dense point cloud completion," *IEEE Transactions on Multimedia*, 2022.

[16] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, "Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5499–5509.

[17] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "Pointr: Diverse point cloud completion with geometry-aware transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 498–12 507.

[18] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[19] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 383–392.

[20] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[21] B. Fei, W. Yang, W. Chen, Z. Li, Y. Li, T. Ma, X. Hu, and L. Ma, "Comprehensive review of deep learning-based 3d point clouds completion processing and analysis," *arXiv preprint arXiv:2203.03311*, 2022.

[22] X. Chen, B. Chen, and N. J. Mitra, "Unpaired point cloud completion on real scans using adversarial training," *arXiv preprint arXiv:1904.00069*, 2019.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[24] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, 2018.

[25] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5868–5877.

[26] D. Stutz and A. Geiger, "Learning 3d shape completion from laser scan data with weak supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1955–1964.

[27] D. T. Nguyen, B.-S. Hua, K. Tran, Q.-H. Pham, and S.-K. Yeung, "A field model for repairing 3d shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5676–5684.

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[29] G. Li, Y. Chen, M. Cheng, C. Wang, and J. Li, "N-dpc: Dense 3d point cloud completion based on improved multi-stage network," in *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, 2020, pp. 274–279.

[30] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 596–11 603.

[31] Y. Nie, Y. Lin, X. Han, S. Guo, J. Chang, S. Cui, J. Zhang *et al.*, "Skeleton-bridged point completion: From global inference to local adjustment," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 119–16 130, 2020.

[32] J. Gu, W.-C. Ma, S. Manivasagam, W. Zeng, Z. Wang, Y. Xiong, H. Su, and R. Urtasun, "Weakly-supervised 3d shape completion in the wild," in *European Conference on Computer Vision*. Springer, 2020, pp. 283–299.

[33] V. Egiazarian, S. Ignatyev, A. Artemov, O. Voynov, A. Kravchenko, Y. Zheng, L. Velho, and E. Burnaev, "Latent-space laplacian pyramids for adversarial representation learning with 3d point clouds," *arXiv preprint arXiv:1912.06466*, 2019.

[34] X. Wen, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 080–13 089.

[35] J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C. K. Yeo, B. Dai, and C. C. Loy, "Unsupervised 3d shape completion through gan inversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1768–1777.

[36] C. Xie, C. Wang, B. Zhang, H. Yang, D. Chen, and F. Wen, "Style-based point generator with adversarial rendering for point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4619–4628.

[37] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[38] C. Ma, Q. Xu, X. Wang, B. Jin, X. Zhang, Y. Wang, and Y. Zhang, "Boundary-aware supervoxel-level iteratively refined interactive 3d image segmentation with multi-agent reinforcement learning," *IEEE Transactions on Medical Imaging*, vol. 40, no. 10, pp. 2563–2574, 2020.

[39] D. Bauer, T. Patten, and M. Vincze, "Reagent: Point cloud registration using imitation and reinforcement learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 586–14 594.

[40] H. G. Kim, M. Park, S. Lee, S. Kim, and Y. M. Ro, "Visual comfort aware-reinforcement learning for depth adjustment of stereoscopic 3d images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1762–1770.

[41] J. Fan and W. Li, "Dribo: Robust deep reinforcement learning via multi-view information bottleneck," in *International Conference on Machine Learning*. PMLR, 2022, pp. 6074–6102.

[42] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[43] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[46] M. Sarmad, H. J. Lee, and Y. M. Kim, "Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5898–5907.

[47] H. Fan, H. Su, and L. 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*, 2017, pp. 605–613.

[48] D. W. Shu, S. W. Park, and J. Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3859–3868.

[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[50] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[51] D. P. Bertsekas, "A distributed asynchronous relaxation algorithm for the assignment problem," in *1985 24th IEEE Conference on Decision and Control*. IEEE, 1985, pp. 1703–1704.

[52] G. Marra and O. Kuželka, "Neural markov logic networks," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 908–917.

[53] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Pmp-net: Point cloud completion by learning multi-step point moving paths," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7443–7452.

[54] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," in *European Conference on Computer Vision*. Springer, 2020, pp. 365–381.

[55] 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," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.