# Adaptive Recurrent Forward Network for Dense Point Cloud Completion

Tianxin Huang, Hao Zou, Jinhao Cui, Jiangning Zhang, *Member, IEEE*, Xuemeng Yang, Lin Li, *Graduate Student Member, IEEE*, and Yong Liu

***Abstract*—Point cloud completion is an interesting and challenging task in 3D vision, which aims to recover complete shapes from sparse and incomplete point clouds. Existing completion networks often require a vast number of parameters and substantial computational costs to achieve a high performance level, which may limit their practical application. In this work, we propose a novel Adaptive efficient Recurrent Forward Network (ARFNet), which is composed of three parts: Recurrent Feature Extraction (RFE), Forward Dense Completion (FDC) and Raw Shape Protection (RSP). In an RFE, multiple short global features are extracted from incomplete point clouds, while a dense quantity of completed results are generated in a coarse-to-fine pipeline in the FDC. Finally, we propose the Adamerge module to preserve the details from the original models by merging the generated results with the original incomplete point clouds in the RSP. In addition, we introduce the Sampling Chamfer Distance to better capture the shapes of the models and the balanced expansion constraint to restrict the expansion distances from coarse to fine. According to the experiments on ShapeNet and KITTI, our network can achieve state-of-the-art completion performances on dense point clouds with fewer parameters, smaller model sizes, lower memory costs and a faster convergence.**

***Index Terms*—3D point clouds, recurrent structure, highly efficient completion.**

## I. Introduction

With the rapid development of real-time 3D sensors such as LiDAR and depth cameras, 3D data have attracted increasing attention in the computer vision and robotics fields. As an appropriate representation for 3D spatial positions, 3D point clouds have been widely used in applications such as SLAM [1] and object detection [2], [3], [4]. However, point clouds acquired from sensors are often incomplete and sparse due to their resolution and occlusion limitations. As a consequence, recovering complete and dense models from incomplete inputs has been an important and challenging task, known as point cloud completion. An appropriate completion can improve the perception performances on downstream tasks such as object detection [5], [6], tracking [7] and scene understanding [8], [9].

Since the work of PCN [10], many deep learning-based methods have been proposed for the 3D point cloud completion task. Some of them are based on 3D grids and 3D convolutional neural networks (CNNs), such as GRNet [11], while others are built based on PointNet [12] and PointNet++ [13], such as TopNet [14] and SANet [15]. These networks are based on high-dimensional global features or multiple local features to acquire enough shape information from the inputs. Most of them have numerous parameters and use a large amount of memory to achieve good performance. To overcome these problems, we propose a novel well-performing recurrent forward point cloud completion framework that shares parameters in layers and extracts multiple short global features to greatly reduce the parameters and memory cost. In addition, most of the above works pay little attention to preserving the details of the original incomplete point clouds, which will cause a distortion of the outputs. Large distortions will lead to meaningless completion results. In these cases, we merge the original shapes from the incomplete models with the outputs of different resolutions to prevent our completion results from having large distortions.

In this paper, we propose a novel Adaptive Recurrent Forward Network (ARFNet). As shown in Fig. 1, it is organized in a "forward" framework, different from a "backward" framework, such as U-Net [16], which has been proven to work well on segmentation [17], [18] and point cloud completion [15]. However, U-Net is computationally expensive in searching and aggregating the local features of different resolutions, especially for dense point clouds. In addition, the intermediate coarse completed point clouds are not well considered because the features are only extracted from the incomplete model. In our framework, the operations are organized into multiple recurrent levels. Modules are parameter-shared to reduce the parameters and model size, and we only extract multiple short global features in different recurrent levels to decrease the computational cost. ARFNet is composed of three parts: Recurrent Feature Extraction (RFE), Forward Dense Completion (FDC) and Raw Shape Protection (RSP). The output model from the former level is concatenated with the incomplete model and fed to the latter level as a "partial incomplete model". In RFE, we design an Encode Cell and Recover Cell to extract features for the different recurrent levels

Manuscript received 31 December 2021; revised 1 June 2022 and 23 July 2022; accepted 17 August 2022. Date of publication 22 August 2022; date of current version 1 November 2023. This work was supported by the Key Research and Development Project of Zhejiang Province under Grant 2021C01035. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr. Lu Fang. *(Corresponding author: Yong Liu.)*

Tianxin Huang, Hao Zou, Jinhao Cui, Jiangning Zhang, Xuemeng Yang, and Lin Li are with the Institute of Cyber-System and Control, Zhejiang University, Hangzhou, Zhejiang 310058, China (e-mail: 21725129@zju.edu.cn; zouhao@zju.edu.cn; 21932011@zju.edu.cn; 186368@zju.edu.cn; xuemengyang@zju.edu.cn; 22032043@zju.edu.cn).

Yong Liu is with the Institute of Cyber-System and Control, Zhejiang University, Hangzhou, Zhejiang 310058, China, and also with Huzhou Institute, Zhejiang University, Hangzhou, Zhejiang 310058, China (e-mail: yongliu@iipc.zju.edu.cn).

Digital Object Identifier 10.1109/TMM.2022.3200851

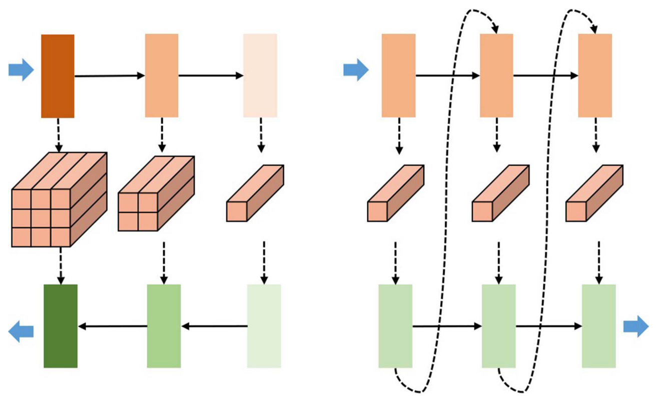

Fig. 1. Comparison of U-Net (left) and our framework (right). U-Net is based on multiresolution local features, while our framework works on multilevel short global features and considers intermediate coarse completion results for feature extraction to achieve comparable performances with relatively low computational costs.

from an intermediate "partial incomplete model". In FDC, we propose an initial cell to create an initial sparse model, which is lifted to higher resolutions with Decode cells. A larger lifting ratio than most previous methods [19], [20] is adopted for the Decode cell to generate dense point clouds with fewer recurrent levels. We share the parameters of the Encode cells and Decode Cells to reduce the parameters and abstract a generic pattern between the different resolutions. Finally, we introduce the Adamerge module to preserve the details in the original incomplete models by driving the generated completion results from the FDC toward their nearest neighbors in the original point clouds in RSP. The Adamerge module adopts a small network to predict a driving distance for each generated point. To better capture the shapes and improve the uniformity of the results, we apply a Chamfer Distance in the randomly divided subsets of the dense point clouds, called a Sampling Chamfer Distance. In addition, we improve the generation continuity in the FDC by constraining the expansion distances by balancing them with their estimated expectation.

Our research contributions can be summarized as follows:

- We propose a novel recurrent forward point cloud completion network by cyclically completing models with features from coarse completed results;
- We propose an Adamerge module to adaptively preserve the original shapes in a learnable way;
- We propose the Sampling Chamfer Distance to better capture the shape differences between the models and the balanced expansion constraint to restrict the expansion distances from coarse to fine;
- The experiments on ShapeNet and KITTI demonstrate that our network outperforms the existing methods on the 3D completion task. In addition to the improvements in completion performance, the model size and parameters are greatly reduced due to this structure.

## II. Related Work

*Point Cloud Learning:* Early works [21], [22], [23] usually applied 3D CNNs based on voxel representations of 3D point clouds. However, the 3D voxels cannot be directly acquired. Converting point clouds to voxels is expensive, which leads to quantization errors, which are caused by ignoring some of the details of the original data. Although some works [24], [25] process 3D shapes through corresponding multiview images, projections from 3D models to images may take extra computation resources and lose structural information. Qi et al. first introduced a point-based point cloud learning network named PointNet [12]. It processes point clouds directly with multilayer perceptrons (MLPs) and aggregates the features with symmetric functions. PointNet++ [13] captures local features by recurrently applying PointNet in the local regions acquired by ball queries around the sampled points. Many works have been proposed based on PointNet and PointNet++ such as point cloud analysis [26], [27], [28], [29], [30], semantic segmentation [31], [32], reconstruction [33], [34], [35] and compression [36], [37] or upsampling [38], [39]. The latter works [40], [41], [42] enhance the performances of learned point cloud representations by introducing graph convolutions, while some methods, such as PointConv [43] and KPconv [44], use MLPs to assign different weights for the points according to their coordinates and design specific convolution methods.

*Point Cloud Completion:* Early works [21], [22] concentrated on voxel-based model completions with 3D convolutions. However, voxel-based models incur inevitable quantization errors from the collected point clouds, which limit their further application. PCN [10] is the first point-based model for point cloud completion. It generates a complete model in a two-stage process that first generates a coarse result by a fully connected network and refines it to a higher resolution with a folding-based network. PFNet [45] completes models by generating the missing parts with the proposed point pyramid decoder (PPD), which is interesting but may have difficulty in generating missing regions with unknown point numbers. TopNet [14] explores the hierarchical rooted tree structure as a decoder to generate an arbitrary grouping of points in the completion task. SANet [15] adopts a commonly used U-Net structure with a skip-connection and self-attention modules to complete the missing features. It performs well on datasets with sparse points. The cascaded refinement network (CRN) [19] adds a cascaded refinement module to achieve a transformation from coarse to fine by multiple lifting with a small upsampling ratio. A mirroring operation on the $xy$-plane and downsampling are used to initialize the coarse outputs and introduce a shape prior. CRN performs better than PCN and TopNet on ShapeNet [46], while it requires shape priors to complete the models. The morphing and sampling network (MSN) [47] improves the completion performance by assembling incomplete models with outputs through minimum density sampling. It also proposes an approximation for the Earth Mover Distance to train the network. GRNet [11] and NSFA [20] achieve completion with quite different ideas. GRNet transforms the incomplete models into 3D grid representations and adopts a 3D CNN to learn the features and to complete the models. NSFA treats point cloud completion as upsampling. The hierarchical feature learning architecture in PU-Net [38] is adopted to extract the local features. Local features of different resolutions are used to construct the points of the preserving or missing parts.

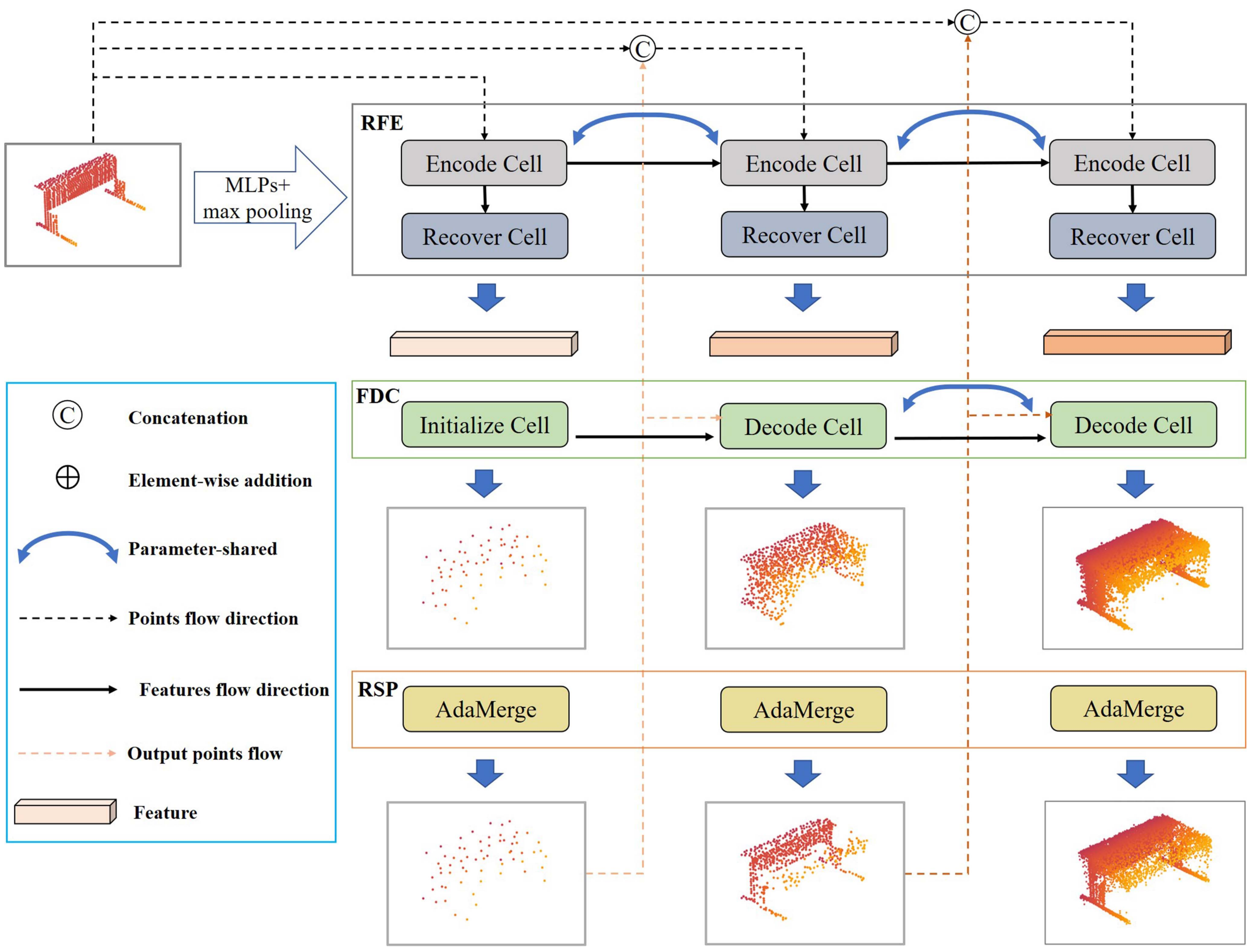

Fig. 2. Structure of ARFNet. It is organized into three recurrent levels, which can also be divided into three parts: RFE, FDC and RSP. The output from the former level is concatenated with the incomplete model and fed to the latter level as an intermediate "partial completed model".

CON [48] and IFNet [49] are two common surface completion methods that concentrate on completing full surfaces from the point clouds with little shape loss and relatively complete overall shapes, while point cloud completion focuses on completing shapes from the point clouds with large occlusions and incomplete overall shapes. In addition, the outputs of CON and IFNet are actually surfaces instead of point coordinates, which makes them distinct from the point cloud completion methods.

## III. Methodology

The goal of our work is the completion of incomplete point clouds to dense shapes with fewer parameters and less cost. As indicated in Fig. 2, the structure is organized into multiple recurrent levels. The output model from the former level is concatenated with the incomplete model and fed to the latter level as a "partial incomplete model". The whole network is composed of three parts, that include the Recurrent Feature Extraction (RFE), Forward Dense Completion (FDC) and Raw Shape Protection (RSP). The RFE extracts features for completion in different recurrent levels, while the FDC creates models of different resolutions based on the output features. Subsequently, shape details from the incomplete model are added to the model by RSP. The output of the last recurrent level is taken as the final output.

### A. Recurrent Feature Extraction

In the Recurrent Feature Extraction module (RFE), we propose the Encode Cell and Recover Cell to extract the global features for subsequent completion operations. An Encode cell extracts the initial features for completion, and the state features for the Encode cell in the next recurrent level. The Recover Cells recover the initial features by further aggregating the information from the incomplete models and return the final features for the FDC. The designations of the Encode Cell and Recover Cell are presented in Fig. 3.

*Encode Cell:* We design the Encode cell to extract an initial feature $F_e$ from the input points, which is parameter-shared between the different levels to reduce the parameters. State features $F_s$ are extracted to record the current recurrent level and help the Encode cell adaptively focus on different regions

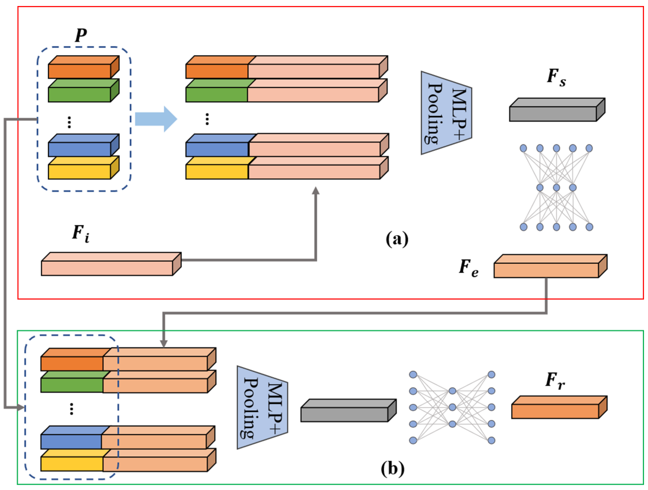

Fig. 3. (a) and (b) denote the structures of the Encode Cell and Recover Cell, respectively. $P$ and $F_i$ are the input point set and input state feature, while $F_s$ and $F_r$ are the state feature for the next level and the recovered feature that generates the completion results. $F_e$ is the initial feature extracted by the Encode Cell.

of the input point clouds. In an Encode cell, the state features acquired from the former Encode cell are concatenated with points and fed into MLPs to produce the state features $F_s$ for the next Encode cell and the output features $F_e$ for completion.

*Recover Cell:* The features $F_e$ directly extracted by the Encode Cells are not sufficient for completion due to the lack of information in the original model. Under this condition, Recover Cells are adopted to further aggregate the information from the incomplete model to complete the features extracted by the Encode Cells. In a Recover Cell, input features $F_e$ from the Encode Cell are concatenated with the input points again and fed to the MLPs to obtain the recovered features $F_r$.

## B. Forward Dense Completion

Forward Dense Completion (FDC) includes an Initialize Cell and Decode cells, which generate completed point clouds based on the extracted features from the RFE. The Initialize Cell generates an initial model in the first level, while Decode cells lift the model to higher resolutions in later levels.

*Initialize Cell:* We design the Initialize Cell to create a basic structure of the completion result. Generating points directly by a fully connected network or FoldingNet [50] may be a commonly used alternative. It is flexible but greatly limited by the effectiveness of a network, it has difficulty in constructing complex shapes.

The structure of the Initialize Cell is presented in Fig. 4. To acquire more accurate basic shapes, we introduce the contour of the original incomplete model by adding sampled points from the incomplete model. Half of the contour points $P_o$ are acquired from the sampled points $P_s$, and the others are directly generated with networks. Direct sampling seeds may be not stable enough to provide complete contours. Therefore, we introduce input features $F_i$ to predict the offsets for each sampled point

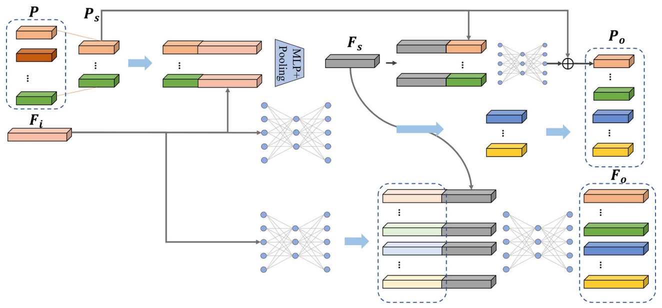

Fig. 4. Structure of the Initialize Cell. $P$ and $F_i$ are the input partial point clouds and the extracted feature from RFE, respectively. $P_s$ is sampled from $P$, which is adjusted and combined into the initialized completed points $P_o$ with generated results from $F_i$. $F_o$ includes state features for $P_o$.

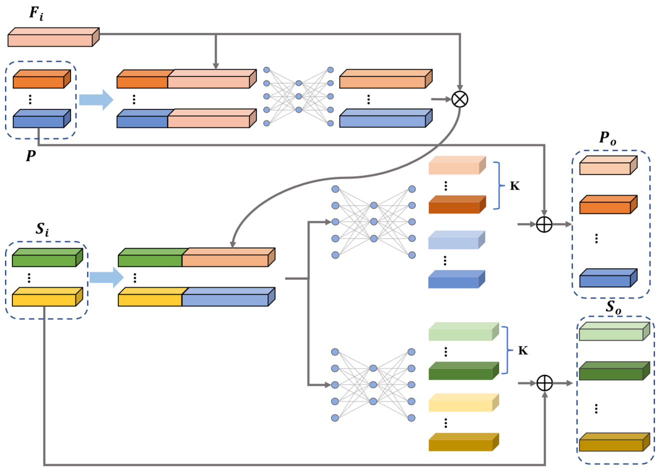

Fig. 5. Structure of the Decode cell. Input points $P$ and state features $S$ are lifted to K times larger output points $P_o$ and state features $S_o$ based on the guidance of the extracted feature $F_i$.

in $P_s$ by the MLPs to refine their positions. Contour points produced from the sampled points are restricted by the shape of the original incomplete model, which means that other points need to be generated to fill up missing parts. The fully connected network has a satisfactory performance, generating missing parts not covered by the incomplete model. In this work, we fuse the outputs grown from the sampled points and fully connected networks to produce an initial completed contour $P_o$. State features $F_o$ are generated with fully connected networks to hold the structural information in the current recurrent level.

*Decode cell:* A Decode cell is designed to learn a parameter-shared transformation from lower resolution completion results $P$ to K times larger $P_o$, as shown in Fig. 5, which is achieved by predicting K offsets for each point from the last level. As we use multiple parameter-shared Decode cells in the whole pipeline, as shown in Fig. 2, we need the state features to record the state and level of the current Decode cell. Input features $F_i$ extracted from the Encode and Recover Cell are adopted to introduce the information from the original point clouds. K parameter-separate MLPs are used for the prediction of the output state features $S_o$, while a fully connected network is adopted to predict the offsets for output points $P_o$. K is set as 16 for this work.

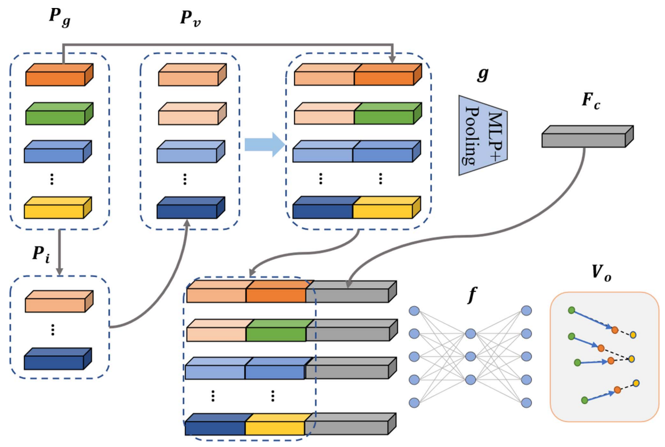

Fig. 6. Structure of the Adamerge module. The neighbors of the generated results $P_g$ in the partial point clouds $P_i$ are denoted as $P_v$. $P_g$, $P_i$ and $P_v$ are used together to predict a displacement field $V_o$, including an offset for each generated point in $P_g$ with network-based operations $g(\cdot)$ and $f(\cdot)$.

### C. Raw Shape Protection

Raw shape protection (RSP) is responsible for the preservation of the original details, which is achieved with multiple Adamerge modules by adaptively merging the generated completion results from the FDC with partial inputs.

*Adamerge module:* The Adamerge module is proposed to introduce the details from the original partial inputs to the completion results generated by the FDC, which works by driving the points toward their nearest neighbors in the incomplete input point clouds. The driving distance is adaptively controlled with a network. The operation of the Adamerge module can be described as follows:

$$dis = \min_{x \in P_i, \forall y \in P} \|x - y\|_2, \tag{1}$$

$$P_v = \{x | \min \|x - y\|_2, \forall y \in P, x \in P_i\} \tag{2}$$

$$\sigma = f(g(P_g, P_v), P_g, P_v) \tag{3}$$

$$V_o = e^{-\frac{dis}{\sigma}}(P_v - P_g) \tag{4}$$

$$\hat{P} = P + V_o, \tag{5}$$

$\sigma$ is predicted by the networks according to the generated results $P_g$ and partial input $P_i$ to adjust the displacement field $V_o$. Each point in $P_g$ will adaptively acquire a separate value for the driving distance. The specific structure of Adamerge is presented in Fig. 6.

### D. Loss Function

*Basic multilevel loss:* We add the losses of the different outputs together to acquire the basic multilevel loss. There are two commonly used loss functions to measure the differences between two point clouds: Chamfer Distance (CD) and Earth Mover Distance (EMD) [51]. Their basic forms are shown as follows:

$$\mathcal{L}_{CD}(S_1, S_2) = \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \frac{1}{|S_2|}\sum_{x \in S_2} \min_{y \in S_1} \|x - y\|_2\right), \tag{6}$$

$$\mathcal{L}_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \frac{1}{|S_1|}\sum_{x \in S_1} \|x - \phi(x)\|_2, \tag{7}$$

where $S_1$ and $S_2$ are two point sets. $\phi$ is a bijection between $S_1$ and $S_2$. CD works mainly on the contours of the models, which may lead to a nonuniform result. We propose the Sampling Chamfer distance (SCD) to improve uniformity. It can be described as

$$\mathcal{L}_{SCD}(S_1, S_2) = \frac{1}{N}\sum_{S_1^i \in D_1, S_2^i \in D_2} \mathcal{L}_{CD}(S_1^i, S_2^i), \tag{8}$$

while $D_1 = RD(S_1, N), D_2 = RD(S_2, N)$. $RD(S, N)$ means randomly dividing the point set $S$ into $N$ isometric sets. $l$ is defined as the number of recurrent levels with low resolutions less than 10000 points, while $h$ is the number of recurrent levels. $l$ and $h$ are set as 2 and 3 in this work. In our work, we apply EMD to the outputs of the first level. Due to the high calculation cost of EMD on dense point clouds, we use SCD and CD for our high-resolution outputs in the second and third levels. Finally, the basic multilevel loss can be described as

$$\mathcal{L}_{BM} = \sum_{i=1}^{l} \mathcal{L}_{EMD_i} + \sum_{j=l}^{h}(\mathcal{L}_{SCD_j} + \mathcal{L}_{CD_j}). \tag{9}$$

*Balanced Expansion Constraint:* A balanced expansion constraint is used to prevent the points predicted by the Decode Cells from going too far from the centers. It will ensure a Decode cell generates continuous local shapes instead of discrete points in the 3D space. However, constraining distances from the centers directly is too ambiguous because the gradient is zero only when the generated points coincide with the corresponding centers. This will inevitably impact the network performance. In this circumstance, we propose the balanced expansion constraint, which can be described as

$$\mathcal{L}_{EC} = \frac{1}{|\hat{S}|}\sum_{x \in \hat{S}, f: S \to \hat{S}} \|x - f^{-1}(x)\|_2, \tag{10}$$

$$E(\mathcal{L}_{EC}) \approx \frac{1}{|\hat{S}_0|}\sum_{x \in \hat{S}_0} \min_{y \in S_0} \|x - y\|_2, \tag{11}$$

$$\mathcal{L}_{BEC} = ReLU(\mathcal{L}_{EC} - \epsilon * E(\mathcal{L}_{EC})), \tag{12}$$

where $S$ and $\hat{S}$ are the inputs and outputs of the Decode Cell, respectively, and $S_0$ and $\hat{S}_0$ are their ground truths. We consider that the expansion distances are related to the differences between the two different resolution models. We estimate the expectation of the expansion distances by Equation 11 while using it as an additional item to balance the expansion constraint. In this way, the gradient can be zero when the expansion distances are small enough, eliminating the disturbance to the network. The influence of an additional item can be adjusted by $\epsilon$.

*Merge range constraint:* The merge range constraint is used to restrain the search radii of Adamerge by constraining $\sigma$ in Eq. (3), Section III-C. Smaller merge ranges will increase the weights of the generated points in the completion results, while bigger ranges increase the weights of the original partial shapes. $\xi$ is a parameter guided by the annealing strategy. It will decrease gradually as the iterations increase. In this way, the merge range is small, to pay more attention to the generated results at the beginning of the iterations, and large, to introduce more information from the original incomplete models at the later period of iterations. The merge range constraint can be formulated as

$$\mathcal{L}_{MR} = \xi * \|\sigma\|_2^2. \tag{13}$$

*Overall loss:* With a balanced expansion constraint and a merge range constraint for each recurrent level, the overall loss is the weighted sum of the mentioned losses as follows:

$$\mathcal{L} = w_1 \mathcal{L}_{BM} + w_2 \sum_{i=1}^{n-1} \mathcal{L}_{BEC_i} + w_3 \sum_{j=1}^{n} \mathcal{L}_{MR_j}, \tag{14}$$

where $n$ is defined as the number of recurrent levels and $w_1$, $w_2$ and $w_3$ are the weights for different constraints.

## IV. Experiments

### A. Dataset and Implementation Details

*ShapeNet:* ShapeNet [46] for completion contains 30974 models from 8 categories, which is provided by PCN [10]. Ground truth models contain 16384 points uniformly sampled on surfaces of mesh models. Partial point clouds are generated by back-projecting 2.5D depth images into 3D. For fair comparisons, we follow same train/val/test splits as PCN [10].

*KITTI:* To further test our network, we evaluate it on the real-world scans from KITTI [52]. Cars are acquired with the ground truth object bounding boxes from each frame. The test set includes 2401 partial point clouds labeled as cars.

*Implementation details:* We adopt the train split of ShapeNet to conduct end-to-end training for ARFNet. $w_1$, $w_2$ and $w_3$ are set as 1.0, 0.05 and 1.0, respectively. $\xi$ is set as 0.01 before 20 epochs and 0.001 after, which limits the merge range more at the beginning of the iterations to pay more attention to the generated results, and less at the later period to introduce more information from the original incomplete models. We train our models using the Adam optimizer [53] with an initial learning rate $5e^{-4}$ (decayed by 0.5 every 7 epochs) and a batch size of 32. It will converge after approximately 33 epochs.

*Metrics:* In our work, we adopt the Chamfer Distance (CD) mentioned in Section III-D as a global metric for completion performance. However, the output models may be changed considerably and lose the details of the incomplete models during completion while maintaining relatively small global errors. An example is shown in Fig. 8. The CD metric of the distorted result on the left is even smaller than the well-performing result on the right, while its fidelity error (FD) [10] is approximately 3 times larger. In terms of this problem, we use the FD as a supplementary evaluation for the distortions. It is defined as the average distance from each point in the input to its nearest neighbor in the output, which can be shown as

$$FD(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2, \tag{15}$$

where $S_1$ and $S_2$ are input and output point clouds, respectively. As there is no complete ground truth model for KITTI, the FD and minimal matching distance (MMD) are used together to evaluate the completion performance. MMD is defined as the minimum CD between the output and all the car point clouds from ShapeNet. It measures how much the completed output resembles a typical car.

### B. Comparison on ShapeNet

In this section, we qualitatively and quantitatively compare our work on ShapeNet with the state-of-the-art point cloud completion methods. The test data include two parts: data containing 8 known categories of models that are the same as the training data, and data containing 8 novel categories of models that are different from the training data. The quantitative comparisons are presented in Tables I and III. The bold and underlined values are the best and second-best values, respectively.

We can see that our RFNet achieves the best performances on both the known and novel category models, while our proposed approach, ARFNet, can further improve the completion performances. Although CRN also performs well on known categories of ShapeNet, it needs a mean shape prior feature from a pretrained network for each category, which is slightly difficult because the categories of the models cannot always be known before completion.

In addition, our network improves considerably on FD, which means our work can make fewer distortions during the completion process and preserve the original shapes better than the other methods.

To intuitively compare the completion results, we choose some models from the test data to make the qualitative comparison. As shown in Fig. 7, FC, Folding, PCN and TopNet create good global shapes while losing most of the details from the incomplete model. MSN, GRNet and CRN can preserve the details to some extent, while they still suffer from obvious distortions. Although NSFA can keep the details much better than the other methods, its local feature aggregation operations greatly increase the computational cost. In addition, it may mistake some discontinuous regions as details and have difficulty completing the models with large and concentrated missing parts, such as the airplane wings and sofa body in the second and fourth rows. Our work can preserve details with fewer distortions and clearer textures, and also has low computational costs, as discussed in Section IV-H. As presented in the third row of Fig. 7, our RFNet may overfocus on the shape defects and produce incorrect completion details. The reason is that RFNet controls the merge range between the generated point clouds and the partial inputs with a group of learned parameters, which means that the merge range would be the same for all the points and models at the inference phase. Hence, some points might be overly merged to the defect areas when the merge range is too large for the model. In this work, ARFNet overcomes this problem by adaptively predicting the different merge ranges according to the model shapes with the use of networks, as illustrated in Section III-C.

TABLE I
QUANTITATIVE COMPARISONS ON KNOWN CATEGORIES OF SHAPENET WITH THE METRICS MULTIPLIED BY $10^3$

| Method | Metric | airplane | cabinet | car | chair | lamp | sofa | table | vessel | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| FC [10] | CD | 5.69 | 11.02 | 8.77 | 10.98 | 11.13 | 11.75 | 9.32 | 9.72 | 9.79 |
| | FD | 5.49 | 9.27 | 10.10 | 9.86 | 10.39 | 8.89 | 9.64 | 8.53 | 9.02 |
| Folding [10] | CD | 5.96 | 10.83 | 9.27 | 11.24 | 12.17 | 11.63 | 9.45 | 10.02 | 10.07 |
| | FD | 6.60 | 8.89 | 11.42 | 10.43 | 11.98 | 9.25 | 10.16 | 10.02 | 9.85 |
| PCN [10] | CD | 5.50 | 10.63 | 8.69 | 10.99 | 11.33 | 11.67 | 8.59 | 9.66 | 9.63 |
| | FD | 5.14 | 7.28 | 9.47 | 7.99 | 8.75 | 7.27 | 8.05 | 7.44 | 7.67 |
| TopNet [14] | CD | 5.85 | 10.78 | 8.84 | 10.80 | 11.15 | 11.41 | 8.79 | 9.17 | 9.60 |
| | FD | 7.97 | 12.44 | 10.76 | 13.50 | 13.94 | 12.32 | 12.15 | 10.63 | 11.71 |
| MSN [47] | CD | 5.60 | 11.90 | 10.70 | 10.60 | 10.70 | 11.80 | 8.71 | 9.48 | 9.96 |
| | FD | 3.22 | 6.42 | 6.19 | 4.96 | 3.65 | 6.04 | 5.38 | 4.57 | 5.05 |
| CRN [19] | CD | **4.79** | **9.97** | **8.31** | 9.49 | 8.94 | 10.69 | 7.81 | 8.05 | 8.51 |
| | FD | 2.80 | 4.89 | 7.20 | 4.06 | 4.15 | 3.83 | 4.05 | 3.63 | 4.33 |
| GRNet [11] | CD | 6.44 | 10.39 | 9.45 | 9.41 | 7.96 | 10.50 | 8.44 | 8.04 | 8.83 |
| | FD | 3.70 | 6.55 | 7.77 | 5.30 | 4.50 | 4.90 | 5.88 | 3.93 | 5.32 |
| NSFA [20] | CD | 5.22 | 10.51 | 9.00 | 9.33 | 8.26 | 10.74 | 7.78 | 7.66 | 8.55 |
| | FD | 3.37 | 4.94 | 7.42 | 4.11 | 3.57 | 4.17 | 3.99 | 3.44 | 4.38 |
| RFNet [54] | CD | 4.91 | 9.98 | 8.66 | 9.14 | 7.16 | 10.45 | 7.45 | **7.28** | 8.13 |
| | FD | 1.98 | 3.49 | 6.96 | 2.83 | 3.02 | 2.95 | 2.86 | 2.75 | 3.35 |
| Ours | CD | 4.86 | 10.02 | 8.75 | **9.02** | **6.93** | **10.26** | **7.26** | 7.35 | **8.10** |
| | FD | **1.86** | **3.28** | **5.78** | **2.19** | **2.28** | **2.23** | **2.73** | **2.23** | **2.82** |

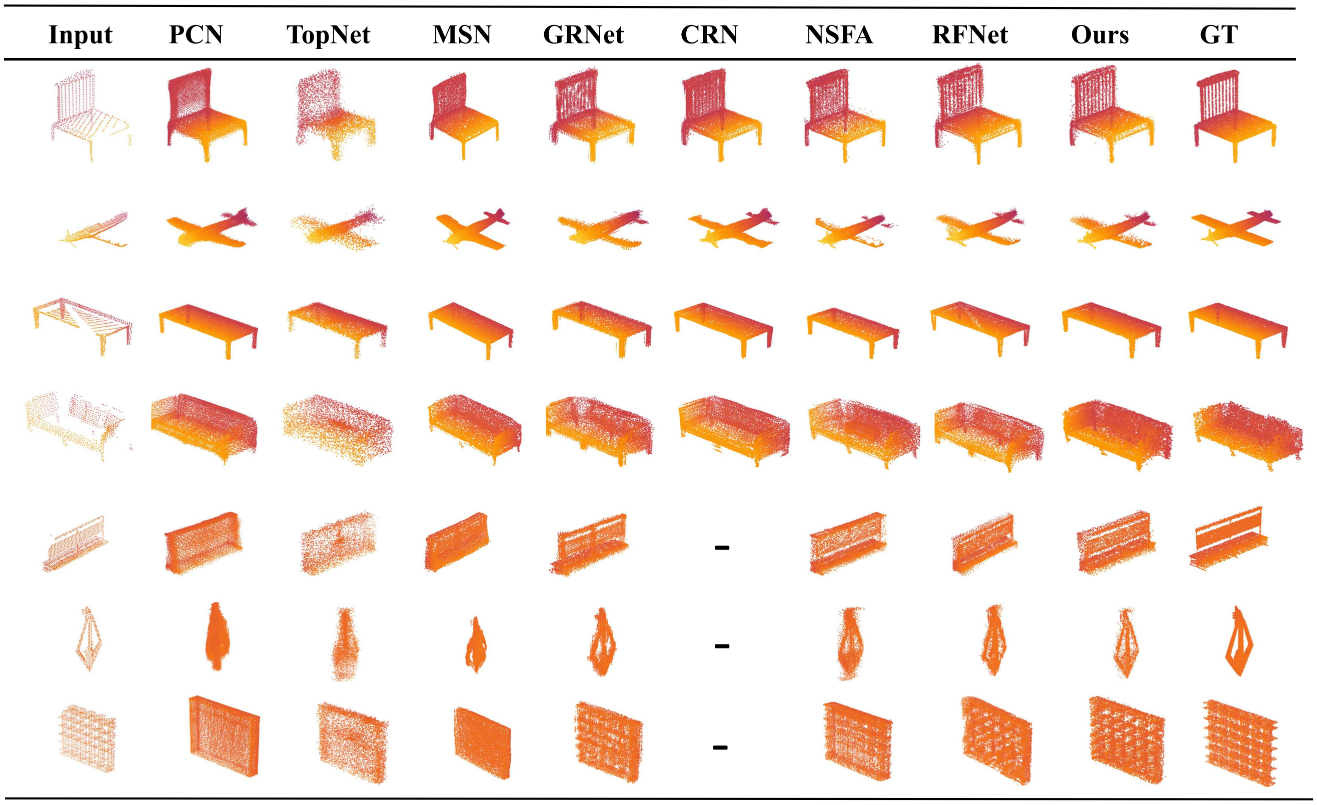

Fig. 7. Qualitative comparisons with state-of-the-art methods on ShapeNet. There is no result for CRN on novel category models because there is no known shape prior feature for them.

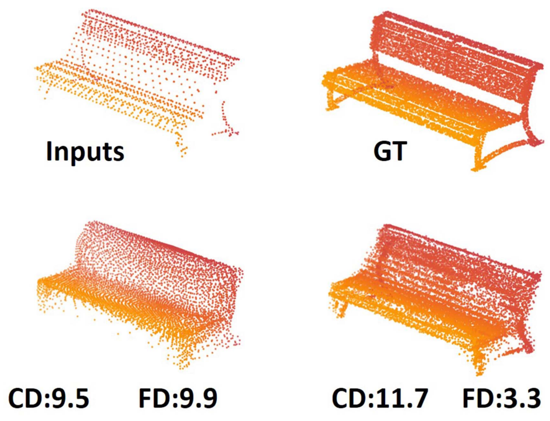

Fig. 8. Comparisons of CD and FD multiplied by $10^3$.

TABLE II
QUANTITATIVE COMPARISONS ON CAR CATEGORY OF KITTI

| Method | FC | Folding | PCN | TopNet | GRNet | RFNet | Ours |
|---|---|---|---|---|---|---|---|
| FD | 0.0331 | 0.0361 | 0.0308 | 0.0335 | 0.0192 | 0.0258 | **0.0184** |
| MMD | 0.0148 | 0.0146 | 0.0158 | 0.0151 | 0.0374 | **0.0146** | 0.0149 |
| FD+MMD | 0.0479 | 0.0507 | 0.0466 | 0.0486 | 0.0557 | 0.0404 | **0.0333** |

## C. Comparison on Kitti

We evaluate our network with the car category of real-world scans from KITTI. Our network is trained on ShapeNet for approximately 0.08M iterations (10 epochs) in this section, without any fine-tuning on other datasets. The quantitative and qualitative results are presented in Table II and Fig. 9, respectively. We can see that our network (ARFNet) achieves the lowest FD and a comparable MMD, which can recover fine car shapes from quite sparse and incomplete point clouds, as illustrated in Fig. 9. Although GRNet also obtains a relatively low FD, it also has the largest MMD on KITTI, which means that GRNet pays more attention to reconstructing the incomplete models instead of completing them, as shown in Fig. 9. To make a trade-off between FD and MMD, we add them together to conduct an overall evaluation. Our ARFNet performs much better on the overall evaluation, which means our work can outperform former methods to complete the shapes while keeping more details. This confirms the great robustness of ARFNet on unseen real scans.

## D. Visualization of the Entire Working Pipeline

The max pooling operation used in an Encode Cell or Recover Cell is actually a selection of key points that achieve the maximum value in multiple feature dimensions. As shown in Fig. 10, we visualize the key points selected by the Encode cells and Recover Cell in the RFE to observe the regions focused at each recurrence level. The coarse completion results are concatenated with the original inputs as "partial completed models" for the next levels. We can see that points selected by the Encode Cells gradually move to the missing parts as the recurrent level increases, which proves that our network is capable of extracting features to adaptively complete the missing parts. In addition, the key points selected by the Recover Cells are around those generated by the Encode Cells, which indicates that the Recover Cell learns to aggregate information from input point clouds based on the output features of the Encode Cells. We can also find that the FDC generates completion results with attractive overall contours, while the Adamerge module introduces more accurate local details and textures from the original incomplete models, as illustrated by the circled regions.

## E. Points Grown From Seeds in the FDC

In the FDC, an Initialize Cell generates an initial sparse model by the combination of the fully connected network output and refined sampled points, which is lifted to higher resolutions with Decode cells. In this section, we visualize the points grown from these two parts at different levels to observe the generation process of the FDC. As demonstrated in Fig. 11, points grown from sampled points form a contour for the original incomplete parts, while points grown from the fully connected network output fill up the missing parts of the models. These two parts are expanded and combined together to make up the final output.

## F. Robustness for Occlusion

In real-world applications, missing points, also known as occlusions, may introduce extra noise to the data and harm completion. To further study the robustness of our method against missing points, we conduct experiments by occluding inputs with a $\%p$ occlusion following PCN [10] and CRN [19], as demonstrated in Fig. 12. We can see that our method performs best both in CD and FD under multiple occlusion ratios, which confirms that our network is more robust against occlusions than the former methods. Our ARFNet has obvious improvements over our RFNet, which confirms that controlling the merge ranges adaptively can truly improve the completion performance and robustness, as discussed in Section IV-B.

## G. Discussion of the Adamerge Module.

Adamerge actually works by learning to drive the points generated by the Decode Cell to original incomplete point clouds. It is not only a module that forces output models close to input but also a derivable method to fuse partial and output models. In this section, we discuss the adoption of Adamerge.

*Differences between mix and Adamerge:* Mix is an easy method to merge the incomplete input and generated output points. We conduct a comparison in Fig. 13 to observe the different performances between the direct mix and the merge. We can see that the direct mix operation may cover the details in the original model, while Adamerge can preserve details by appropriately driving output points to the input shape.

*Differences between constraints and Adamerge:* Adding FD loss defined in Eq. (15) to the loss function can also force the generated points from the completion network toward the input of incomplete point clouds. To distinguish the effects of adding constraints and using Adamerge, we conduct a comparison in Fig. 14. We can see that the constraint enforcing the output to input cannot preserve the details as well as Adamerge.

TABLE III
QUANTITATIVE COMPARISONS ON NOVEL CATEGORIES OF SHAPENET WITH METRICS MULTIPLIED BY $10^3$

| Method | Metric | Similiar | | | | | Dissimiliar | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | bus | bed | bookshelf | bench | Average | guitar | motor | skateboard | pistol | Average |
| FC [10] | CD | 9.82 | 21.23 | 15.12 | 10.81 | 14.20 | 9.92 | 14.56 | 12.00 | 14.97 | 12.90 |
| | FD | 7.87 | 13.54 | 10.53 | 8.88 | 10.20 | 9.26 | 11.97 | 7.77 | 13.86 | 10.72 |
| Folding [10] | CD | 10.58 | 19.08 | 14.88 | 10.55 | 13.80 | 9.06 | 15.56 | 11.91 | 13.13 | 12.40 |
| | FD | 8.14 | 13.32 | 10.39 | 9.43 | 10.32 | 9.30 | 14.49 | 7.49 | 12.96 | 11.06 |
| PCN [10] | CD | 9.46 | 21.63 | 14.79 | 11.02 | 14.20 | 10.40 | 14.75 | 12.04 | 14.23 | 12.90 |
| | FD | 6.41 | 10.63 | 8.52 | 7.58 | 8.28 | 8.61 | 11.48 | 6.56 | 10.70 | 9.34 |
| TopNet [14] | CD | 9.31 | 20.38 | 14.12 | 10.16 | 13.40 | 9.88 | 14.30 | 9.26 | 12.86 | 11.50 |
| | FD | 9.93 | 15.37 | 12.69 | 11.08 | 12.27 | 10.11 | 14.52 | 9.63 | 15.42 | 12.42 |
| MSN [47] | CD | 11.60 | 24.10 | 16.20 | 10.80 | 15.67 | 10.40 | 15.50 | 11.70 | 14.20 | 13.95 |
| | FD | 5.40 | 6.27 | 6.45 | 5.00 | 5.78 | 2.40 | 4.39 | 4.00 | 2.87 | 3.42 |
| GRNet [11] | CD | 11.50 | 22.42 | 14.91 | 11.47 | 15.08 | 8.88 | 11.83 | 11.30 | 13.27 | 11.32 |
| | FD | 4.92 | 5.97 | 6.22 | 5.38 | 5.62 | 4.04 | 4.51 | 3.73 | 3.55 | 6.79 |
| NSFA [20] | CD | 9.24 | **17.30** | 12.63 | 9.76 | 12.23 | 8.72 | **10.56** | 8.68 | 11.03 | 9.75 |
| | FD | 3.86 | 4.84 | 4.78 | 3.82 | 4.33 | 3.30 | 3.80 | 2.88 | 3.54 | 3.38 |
| RFNet [54] | CD | **8.98** | 19.20 | 12.91 | 9.79 | 12.72 | 7.59 | 10.88 | **8.66** | 9.74 | 9.22 |
| | FD | 2.42 | 4.35 | 3.79 | 2.84 | 3.35 | 1.89 | 4.99 | 1.48 | 2.90 | 2.82 |
| Ours | CD | 9.09 | 17.54 | **12.40** | **9.56** | **12.15** | **6.81** | 10.65 | 9.27 | **9.74** | **9.12** |
| | FD | **2.09** | **3.25** | **3.28** | **2.49** | **2.78** | **0.78** | **3.53** | **1.29** | **1.98** | **1.88** |

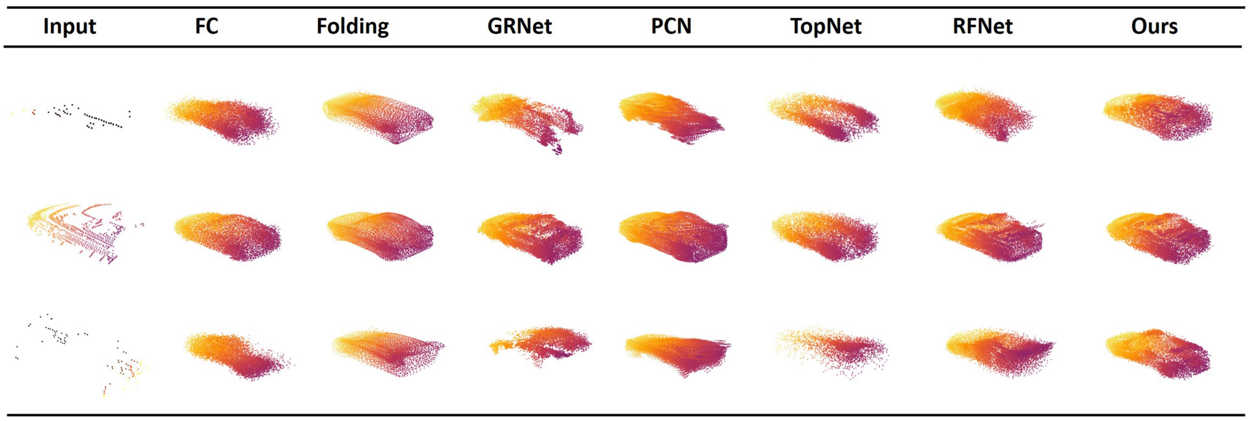

Fig. 9. Qualitative comparisons with other methods on Kitti. We can see that our method has strong ability to recover car shapes from quite incomplete scans.

*Differences between postprocessing and Adamerge:* To explore the necessity of the learning-based operation, we also compare the performances between the learning-based Adamerge and the manually controlling driving strategy in this section. As the Adamerge module works by learning to drive points to their nearest neighbor in the partial point cloud, manually controlling the driving procedure seems to be another alternative. Let $N$ and $D$ denote the number of generated points and their distances to the nearest neighbors in the partial inputs. We choose the $p_1 \times N$ points closest to the partial input and move them $p_2 \times D$ towards their nearest neighbors in the partial point cloud to observe the feasibility of manual control. As shown in Table IV, the horizontal and vertical axes denote $p_1$ and $p_2$, respectively. The network without a merge or any processing obtains 8.96/6.44, as shown in the fifth row of Table VII. We can see that manually controlling the results under multiple settings is quite inferior to Adamerge. It demonstrates that it is difficult to manually find exact appropriate settings, while Adamerge can avoid the manual setting by learning the merge ranges to decide which points need to be merged and how far they should be driven.

TABLE IV
COMPARISONS WITH POST PROCESSING ON CD/FD

| How far ($p_2$) | 0.3 | 0.5 | 0.7 |
|---|---|---|---|
| How many ($p_1$) | CD/FD | CD/FD | CD/FD |
| 0.3 | 10.74/6.00 | 10.62/4.57 | 10.63/3.38 |
| 0.5 | 10.42/5.68 | 10.24/4.07 | 10.29/2.70 |
| 0.7 | 10.29/5.58 | 10.21/3.89 | 10.52/2.45 |

## *H. Comparison of Network Efficiency*

In this section, we compare the model size, memory cost, time cost and training requirements. The time and memory costs are

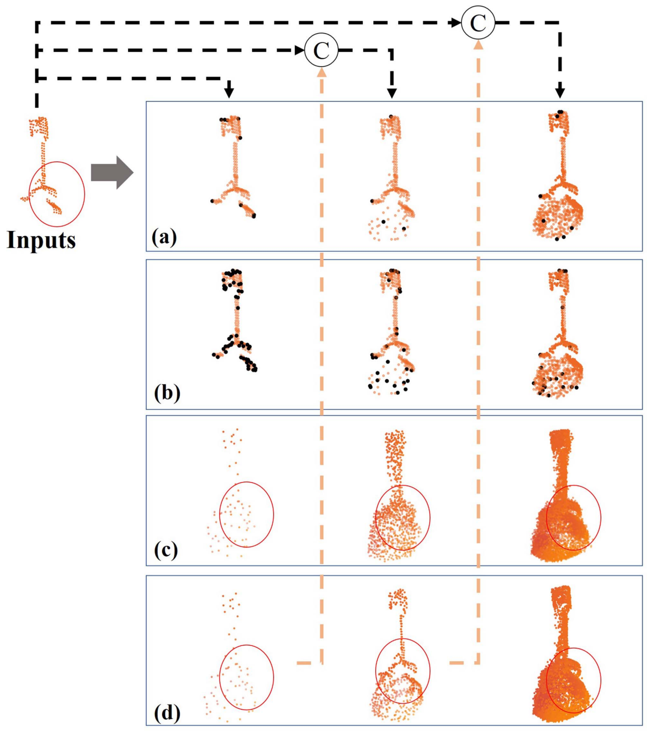

Fig. 10. (a) and (b) show the areas of the Encode cell and Recover Cell in RFE, where the black points denote the selected key points. (c) generated completion results by the Initialize Cell and Decode cell in FDC. (d) merged completion results by the Adamerge module in RSP.

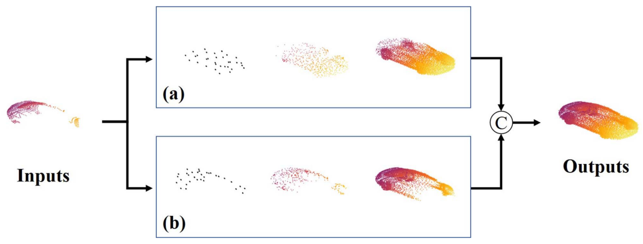

Fig. 11. Visualization of the FDC. (a) and (b) denote points grown from the generated and sampled points in the different levels, respectively.

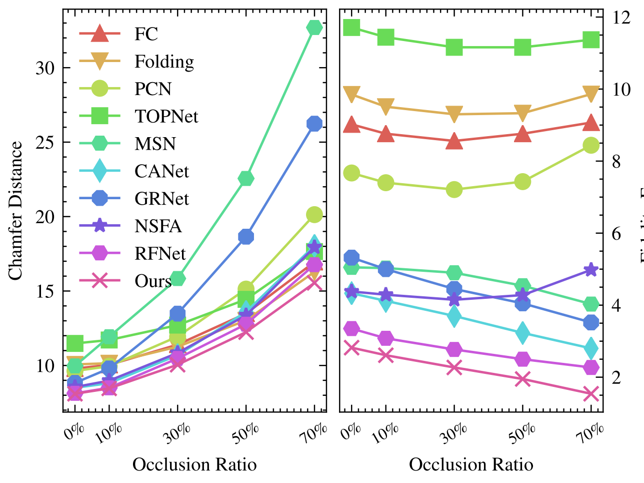

Fig. 12. Quantitative comparison for the occluded point clouds under different occlusion ratios.

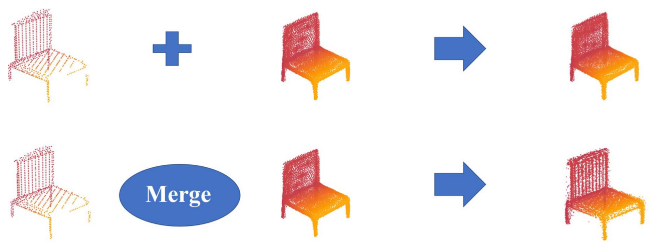

Fig. 13. Quantitative comparison between mix and Adamerge.

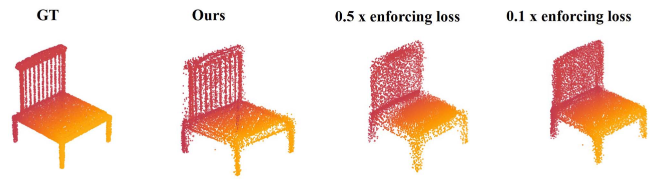

Fig. 14. Qualitative comparison between constraints and Adamerge.

TABLE V
MODEL SIZE COMPARISON

| Method | PCN | TopNet | MSN | GRNet | CRN | NSFA | RFNet | Ours |
|---|---|---|---|---|---|---|---|---|
| Parameters (M) | 6.85 | 9.96 | 29.00 | 76.77 | 5.14 | 5.60 | <u>3.82</u> | **3.43** |
| Model Size (MB) | 82.3 | 79.8 | 116.0 | 292.6 | 61.9 | 66.0 | <u>50.1</u> | **46.0** |

TABLE VI
MODEL EFFICIENCY COMPARISON

| Method | Inference | | Training requirements | | |
|---|---|---|---|---|---|
| | Time(ms) | Memory(MB) | Batch | Iter (M) | Memory(GB) |
| PCN | 6.68 | 973 | 32 | 0.3 | 11 |
| TopNet | **5.09** | 732 | 32 | 0.23 | 11 |
| MSN | 20.16 | 1417 | 160 | 0.23 | 8x11 |
| GRNet | <u>5.92</u> | 1719 | 32 | 1.09 | 2x11 |
| CRN | 9.22 | 973 | 32 | 0.27 | 11 |
| NSFA | 104.80 | 973 | 8 | 0.67 | 11 |
| RFNet | 9.00 | <u>710</u> | 32 | <u>0.23</u> | 11 |
| Ours | 7.96 | **689** | 32 | **0.21** | 11 |

TABLE VII
ABLATION STUDY FOR THE PROPOSED MODULES

| Enc | SCD | BEC | Rec | Mer | CD | FD | CD* |
|---|---|---|---|---|---|---|---|
| ✓ | - | - | - | - | 9.35 | 6.86 | 9.69 |
| ✓ | ✓ | - | - | - | 9.27 | 6.58 | 9.81 |
| ✓ | ✓ | - | - | ✓ | 8.23 | 2.75 | 8.70 |
| ✓ | ✓ | ✓ | - | - | 9.26 | 6.59 | 9.72 |
| ✓ | ✓ | ✓ | ✓ | - | 8.96 | 6.44 | 9.45 |
| ✓ | ✓ | ✓ | ✓ | ✓ | **8.10** | **2.68** | **8.45** |

evaluated on an Nvidia 2080ti GPU with a 2.9 GHz i5-9400 CPU. As illustrated in Table V, we can see that our network has the fewest parameters and the smallest model size since we share the parameters between some relatively complex modules. As presented in Table VI, our network also has comparable time costs and the lowest memory costs. Although GRNet is faster, it requires more than 2 times more memory than ours. This confirms that the basic recurrent forward structure is an effective lightweight framework. Additional works can be further developed based on our work.

TABLE VIII
ABLATION STUDY FOR PARAMETER-SHARED LINKAGES

| Enc* | Dec* | Rec | Raw | CD | FD | Para (M) |
|---|---|---|---|---|---|---|
| ✓ | - | - | - | 8.15 | 2.84 | 4.41 |
| - | ✓ | - | - | 8.12 | 3.36 | 4.81 |
| - | - | ✓ | - | 8.12 | **2.78** | **3.04** |
| - | - | - | ✓ | **8.10** | 2.82 | 3.43 |

Moreover, our work has a relatively low training requirement. Although hardly any other work pays attention to this, it is actually important for model applications in different scenarios. The comparison of training requirements with several completion networks is reported in Table VI. Note that our network only needs 11 GB to train. It will take only approximately 0.08M iterations to converge to a result better than the former networks, as discussed in Section IV-I, and 0.21M iterations to the best result in 1 d, which is much faster than the other methods.

## I. Ablation Study

*Effects of the proposed modules:* In this section, we evaluate the effects of different modules. The experiments are conducted on known category models of ShapeNet by removing the modules and retraining the network. We use CD and FD to evaluate the completion results, as illustrated in Table VII. Enc, SCD, BEC, Rec and Mer denote the Encode cell, Sampling Chamfer Distance, balanced expansion constraint, Recover Cell and Adamerge, respectively. CD* means CD measured at 0.08M iterations to compare the convergence efficiency. We can see that the full network with all modules works the best. Removing any component decreases the performance, which indicates that each component makes a contribution. Adamerge contributes the most to reducing the error. Although other modules help relatively less, they can improve the final performance and accelerate convergence, which helps achieve a much smaller CD*.

*Effection of parameter-shared operations:* To further confirm the rationality of the parameter-shared linkages, we conduct an ablation study on them by adding or removing the linkages between modules. The results are presented in Table VIII. * and underline denote removing and adding parameter-shared linkages between modules, respectively. We can see that our method with no changing linkages is mostly the best. Although sharing parameters between the Recover Cell reduces FD and parameters, it creates relatively weak overall shapes and higher CD. It is an interesting phenomenon, indicating that appropriate parameter-shared linkages to learn a common pattern, such as transformation from lower to higher resolutions, can improve the network performance.

*Influence of the recurrence level:* In this section, we explore the influence of the recurrence level on the completion performance. By adjusting the number of output points in the Initialize Cell, we attempt to complete the original model with 1, 2 and 3 levels. Note that we do not test a recurrent level of more than 3 points because the resolution of the Initialize Cell needs to be 4 points or less under this condition, which cannot provide an available initial shape. The results are presented in Table IX. We

TABLE IX
INFLUENCE OF THE RECURRENT LEVEL

| Level | 1 | 2 | 3 |
|---|---|---|---|
| CD | 9.13 | 8.15 | **8.10** |
| FD | **1.87** | 2.41 | 2.82 |

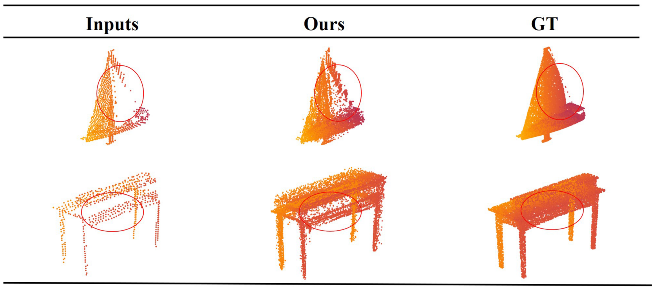

Fig. 15. Failure cases of ARFNet.

can see that the CD increases and FD decreases as the recurrence level increases, which means that the network focuses more on the overall performance and weakens the fidelity. We adopt 3 recurrent levels in this work.

## J. Limitation and Failure Cases

In this section, we discuss the limitation of ARFNet. A few failure cases are presented in Fig. 15. We can see that some textures from the missing regions of the partial input are also introduced to the completed results. This condition may come from the limitations of the Adamerge module. Although the Adamerge module can adaptively adjust the merge range for each point to avoid a lot of noise following Section III-C, it may sometimes regard noise as a part of the original shape details. More reasonable network structures can be designed in the Adamerge module to help it better distinguish the details from the noise in future works.

## V. Conclusion

In this paper, we propose an adaptive recurrent forward network for dense point cloud completion (ARFNet), which is organized into multiple recurrent levels. The output model from the former level will be concatenated with the incomplete model and fed to the latter level as a “partial incomplete model”. The work consists of three parts: RFE, FDC and RSP. The RFE extracts multiple global features for completion at different resolutions, and the FDC generates completed point clouds from coarse to fine. The RSP is used to introduce details from the original incomplete models to the generated outputs. Specifically, we propose the Adamerge module to adaptively drive points toward the original points. In addition, we share parameters between some complex modules to greatly reduce the parameter quantities and model size. We also propose a Sampling Chamfer Distance and a balanced expansion constraint to better capture the shape differences and improve the completion performances in the multilevel structure. Experiments on ShapeNet and KITTI indicate that ARFNet can achieve state-of-the-art performances with a lower cost than the existing methods. The limitations of

ARFNet lies in the operation of the Adamerge module, which may sometimes regard noise from partial inputs as shape details and introduce them to the completed results. We will attempt to overcome this problem by improving the network structures in the Adamerge module in future works. In addition, completing large-scale scene data might also be an interesting task, which is a current challenge for the existing completion networks. We will also further explore the application of ARFNet to scenes.

## Acknowledgment

The authors would like to thank the excellent and inspiring effort of the authors, reviewers and editor to improve the quality of this article.

## References

[1] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[2] N. Dinesh Reddy, M. Vo, and S. G. Narasimhan, "Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1906–1915.

[3] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 918–927.

[4] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.

[5] A. H. Lang et al., "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12697–12705.

[6] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 641–656.

[7] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3D siamese tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1359–1368.

[8] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D semantic instance segmentation of RGB-D scans," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4421–4430.

[9] A. Dai et al., "Scancomplete: Large-scale scene completion and semantic segmentation for 3D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4578–4587.

[10] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 728–737.

[11] H. Xie et al., "GRNet: Gridding residual network for dense point cloud completion," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 365–381.

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet : Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[14] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Top-Net: Structural point cloud decoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 383–392.

[15] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1939–1948.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2015, pp. 234–241.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[19] X. Wang, M. H. Ang Jr, and G. H. Lee, "Cascaded refinement network for point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 790–799.

[20] W. Zhang, Q. Yan, and C. Xiao, "Detail preserved point cloud completion via separated feature aggregation," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 512–528.

[21] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3D-encoder-predictor CNNs and shape synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5868–5877.

[22] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 85–93.

[23] D. Li, T. Shao, H. Wu, and K. Zhou, "Shape completion from a single RGBD image," *IEEE Trans. Visual. Comput. Graph.*, vol. 23, no. 7, pp. 1809–1822, Jul. 2017.

[24] S. Bai et al., "GIFT: Towards scalable 3D shape retrieval," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1257–1271, Jun. 2017.

[25] J. Huang, W. Yan, T. H. Li, S. Liu, and G. Li, "Learning the global descriptor for 3D object recognition based on multiple views decomposition," *IEEE Trans. Multimedia*, vol. 24, pp. 188–201, 2022.

[26] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 984–993.

[27] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9204–9214.

[28] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9775–9784.

[29] H. Su et al., "Splatnet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2530–2539.

[30] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Trans. Multimedia*, vol. 24, pp. 1943–1955, 2022.

[31] H. Liu, Y. Guo, Y. Ma, Y. Lei, and G. Wen, "Semantic context encoding for accurate 3D point cloud segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 2045–2055, 2020.

[32] C. Chen, S. Qian, Q. Fang, and C. Xu, "HAPGN: Hierarchical attentive pooling graph network for point cloud segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 2335–2346, 2021.

[33] 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," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 10441–10 450.

[34] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 52–66.

[35] G. Yang et al., "Pointflow: 3D point cloud generation with continuous normalizing flows," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 4541–4550.

[36] X. Sheng et al., "Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Trans. Multimedia*, vol. 24, pp. 2617–2632, 2021.

[37] T. Huang and Y. Liu, "3D point cloud geometry compression on deep learning," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 890–898.

[38] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-Net: Point cloud upsampling network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.

[39] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7203–7212.

[40] Y. Wang et al., "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[41] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized representations of point clouds with graph-convolutional generative adversarial networks," *IEEE Trans. Multimedia*, vol. 23, pp. 402–414, 2020.

[42] Z. Fu and W. Hu, "Dynamic point cloud inpainting via spatial-temporal graph learning," *IEEE Trans. Multimedia*, vol. 23, pp. 3022–3034, 2021.

[43] W. Wu, Z. Qi, and L. Fuxin, "PointCONV: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.

[44] H. Thomas et al., "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.

[45] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, “PF-Net: Point fractal network for 3D point cloud completion,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7662–7670.

[46] A. X. Chang et al., “ShapeNet: An information-rich 3D model repository,” 2015, *arXiv:1512.03012*.

[47] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, “Morphing and sampling network for dense point cloud completion,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11596–11603.

[48] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, “Convolutional occupancy networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 523–540.

[49] J. Chibane, T. Alldieck, and G. Pons-Moll, “Implicit functions in feature space for 3D shape reconstruction and completion,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6970–6981.

[50] Y. Yang, C. Feng, Y. Shen, and D. Tian, “FoldingNet: Point cloud auto-encoder via deep grid deformation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 206–215.

[51] H. Fan, H. Su, and L. J. Guibas, “A point set generation network for 3D object reconstruction from a single image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 605–613.

[52] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[53] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, *arXiv:1412.6980*.

[54] T. Huang et al., “RFNet: Recurrent forward network for dense point cloud completion,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12508–12517.