GRNet: Gridding Residual Network for Dense Point Cloud Completion

Haozhe Xie[1,2,3][0000−0001−9596−5179], Hongxun Yao[1,2][0000−0003−3298−2574], Shangchen Zhou[4][0000−0001−8201−8877], Jiageng Mao[5][0000−0003−2571−8767], Shengping Zhang[2,6][0000−0001−5200−3420], and Wenxiu Sun[3][0000−0001−5026−8820]

[1]State Key Laboratory of Robotics and System, Harbin Institute of Technology
[2]Faculty of Computing, Harbin Institute of Technology
[3]SenseTime Research [4]Nanyang Technological University
[5]The Chinese University of Hong Kong [6]Peng Cheng Laboratory
https://haozhexie.com/project/grnet

Abstract. Estimating the complete 3D point cloud from an incomplete one is a key problem in many vision and robotics applications. Mainstream methods (*e.g.,* PCN and TopNet) use Multi-layer Perceptrons (MLPs) to directly process point clouds, which may cause the loss of details because the structural and context of point clouds are not fully considered. To solve this problem, we introduce 3D grids as intermediate representations to regularize unordered point clouds and propose a novel Gridding Residual Network (GRNet) for point cloud completion. In particular, we devise two novel differentiable layers, named *Gridding* and *Gridding Reverse*, to convert between point clouds and 3D grids without losing structural information. We also present the differentiable *Cubic Feature Sampling* layer to extract features of neighboring points, which preserves context information. In addition, we design a new loss function, namely *Gridding Loss*, to calculate the L1 distance between the 3D grids of the predicted and ground truth point clouds, which is helpful to recover details. Experimental results indicate that the proposed *GRNet* performs favorably against state-of-the-art methods on the ShapeNet, Completion3D, and KITTI benchmarks.

Keywords: Point cloud completion, gridding, cubic feature sampling

1 Introduction

With the rapid development of 3D acquisition technologies, 3D sensors (*e.g.*, LiDARs) are becoming increasingly available and affordable. As a commonly used format, point clouds are the preferred representation for describing the 3D shape of an object. Complete 3D shapes are required in many applications, including semantic segmentation and SLAM [2]. However, due to limited sensor resolution and occlusion, highly sparse and incomplete point clouds can be acquired, which causes loss in geometric and semantic information. Consequently, recovering the complete point clouds from partial observations, named point cloud completion, is very important for practical applications.

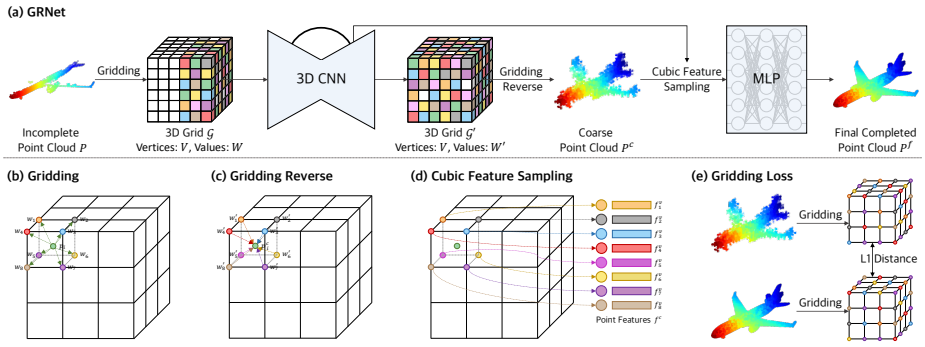

Fig. 1. Overview of the proposed **(a)** GRNet, **(b)** Gridding, **(c)** Gridding Reverse, **(d)** Cubic Feature Sampling, and **(e)** Gridding Loss.

In the recent few years, convolutional neural networks (CNNs) have been applied to 2D images and 3D voxels. Since the convolution can not be directly applied to point clouds due to their irregularity and unorderedness, most of the existing methods [3,7,34,35,29,40,26] voxelize the point cloud into binary voxels, where 3D convolutional neural networks can be applied. However, the voxelization operation leads to an irreversible loss of geometric information. Other approaches [51,27,38] use the Multi-Layer Perceptrons (MLPs) to process point clouds directly. However, these approaches use max pooling to aggregate information across points in a global or hierarchical manner, which do not fully consider the connectivity across points and the context of neighboring points. More recently, several attempts [42,41] have been made to incorporate graph convolutional networks (GCN) [14] to build local graphs in the neighborhood of each point in the point cloud. However, constructing the graph relies on the K-nearest neighbor (KNN) algorithm, which is sensitive to the point cloud density [39].

Several attempts in point cloud segmentation have been made to capture spatial relationships in point clouds through more general convolution operations. SPLATNet [36] and InterpConv [28] perform convolution on high-dimensional lattices and 3D cubes interpolated from neighboring points, respectively. However, both of them are based on a strong assumption that the 3D coordinates of the output points are the same as the input points and thus can not be used for 3D point completion.

To address the issues mentioned above, we introduce 3D grids as intermediate representations to regularize unordered point clouds, which explicitly preserves the structural and context of point clouds. Consequently, we propose a novel Gridding Residual Network (*GRNet*) for point cloud completion, as shown in Figure 1. Besides 3D CNN and MLP, we devise three differentiable layers: *Gridding*, *Gridding Reverse*, and *Cubic Feature Sampling*. In *Gridding*, for each point of the point cloud, eight vertices of the 3D grid cell that the point lies in are first weighted using an interpolation function that explicitly measures the geometric

relations of the point cloud. Then, a 3D convolutional neural network (3D CNN) with skip connections is adopted to learn context-aware and spatially-aware features, which allows the network to complete missing parts of the incomplete point cloud. Next, *Gridding Reverse* converts the output 3D grid to a coarse point cloud by replacing each 3D grid cell with a new point whose coordinate is the weighted sum of the eight vertices of the 3D grid cell. The following *Cubic Feature Sampling* extracts features for each point in the coarse point cloud by concatenating the features of the corresponding eight vertices of the 3D grid cell that the point lies in. The coarse point cloud and the features are forwarded to an MLP to obtain the final completed point cloud.

Existing methods adopt Chamfer Distance in PSGN [4] as the loss function to train the neural networks. This loss function penalizes the prediction deviating from the ground-truth. However, there is no guarantee that the predicted point clouds follow the geometric layout of objects, and the networks tend to output a mean shape that minimizes the distance [11,48]. Some recent works [11,48,12,18,21] attempt to solve the unorderness while preserving fine-grained details by projecting the 3D point cloud to an image, which is then supervised by the corresponding ground truth masks. However, the projection requires extrinsic camera parameters, which are challenging to estimate in most scenarios [31]. To solve the unorderedness of point clouds, we propose *Gridding Loss*, which calculates the L1 distance between the generated points and ground truth by representing them in regular 3D grids with the proposed *Gridding* layer.

The contributions can be summarized as follows:

- We innovatively introduce 3D grids as intermediate representations to regularize unordered point clouds, which explicitly preserve the structural and context of point clouds.
- We propose a novel Gridding Residual Network (*GRNet*) for point cloud completion. We design three differentiable layers: *Gridding*, *Gridding Reverse*, and *Cubic Feature Sampling*, as well as a new *Gridding Loss*.
- Extensive experiments are conducted on the ShapeNet, Completion3D, and KITTI benchmarks, which indicate that the proposed *GRNet* performs favorably against state-of-the-art methods.

2 Related Work

According to the network architecture used in point cloud completion and reconstruction, existing networks can be roughly categorized into MLP-based, graph-based, and convolution-based networks.

MLP-based Networks. Pioneered by PointNet [32], several works use MLP for point cloud processing [1,22] and reconstruction [51,27] because of its simplicity and strong representation ability. These methods model each point independently using several Multi-layer Perceptrons and then aggregate a global feature using a symmetric function (*e.g.*, Max Pooling). However, the geometric relationships among 3D points are not fully considered. PointNet++ [33] and TopNet

[38] incorporate a hierarchical architecture to consider the geometric structure. To relief the structure loss caused by MLP, AtlasNet [6] and MSN [23] recover the complete point cloud of an object by estimating a collection of parametric surface elements.

Graph-based Networks. By considering each point in a point cloud as a vertex of a graph, graph-based networks generate directed edges for the graph based on the neighbors of each point. In these methods, convolution is usually operated on spatial neighbors, and pooling is used to produce a new coarse graph by aggregating information from each point's neighbors. Compared with MLP-based methods, graph-based networks take local geometric structures into account. In DGCNN [42], a graph is constructed in the feature space and dynamically updated after each layer of the network. Further, LDGCNN [52] removes the transformation network and link the hierarchical features from different layers in DGCNN to improve its performance and reduce the model size. Inspired by DGCNN, Hassani and Haley [8] introduce the multi-scale graph-based network to learn point and shape features for self-supervised classification and reconstruction. DCG [41] also follows DGCNN to encode additional local connection into a feature vector and progressively evolves from coarse to fine point clouds.

Convolution-based Networks. Early works [3,7,17] usually apply 3D convolutional neural networks (CNNs) build upon the volumetric representation of 3D point clouds. However, converting point clouds into 3D volumes introduces a quantization effect that discards some details of the data [43] and is not suitable for representing fine-grained information. To the best of our knowledge, no work directly applies CNNs on irregular point clouds for shape completion. In point cloud understanding, several works [28,10,16,15,20] develop CNNs operating on discrete 3D grids that are transformed from point clouds. Hua *et al.* [10] define convolutional kernels on regular 3D grids, where the points are assigned with the same weights when falling into the same grid. PointCNN [20] achieves permutation invariance through a χ-conv transformation. Besides CNNs on discrete space, several methods [26,39,36,49,25,24,44,9] define convolutional kernels on continuous space. Thomas *et al.* [39] propose both rigid and deformable kernel point convolution (KPConv) operators for 3D point clouds using a set of learnable kernel points. Compared with graph-based networks, convolution-based networks are more efficient and robust to point cloud density [28].

3 Gridding Residual Network

3.1 Overview

The proposed GRNet aims to recover the complete point cloud from an incomplete one in a coarse-to-fine fashion. It consists of five components, including *Gridding* (Section 3.2), 3D Convolutional Neural Network (Section 3.3), *Gridding Reverse* (Section 3.4), *Cubic Feature Sampling* (Section 3.5), and Multi-layer Perceptron (Section 3.6), as shown in Figure 1. Given an incomplete point cloud P as input, *Gridding* is first used to obtain a 3D grid $\mathcal{G} =< V, W >$, where V and W are the vertex set and value set of $\mathcal{G}$, respectively. Then, W is fed to a

3D CNN, whose output is W'. Next, *Gridding Reverse* produces a coarse point cloud P^c from the 3D grid $\mathcal{G}' =< V, W' >$. Subsequently, *Cubic Feature Sampling* generates features F^c for the coarse point cloud P^c. Finally, MLP takes the coarse point cloud P^c and the corresponding features F^c as input to produce the final completed point cloud P^f.

3.2 Gridding

2D and 3D convolutions have been developed to process regularly arranged data such as images and voxel grids. However, it is challenging to directly apply standard 2D and 3D convolutions to unordered and irregular point clouds. Several methods [3,7,26,17] convert point clouds into 3D voxels and then apply 3D convolutions to them. However, the voxelization process leads to an irreversible loss of geometric information. Recent methods [51,38] adopt Multi-layer Perceptrons (MLPs) to directly operate on point clouds and aggregate information across points with max pooling. However, MLP-based methods may lose local context information because the connectivity and layouts of points are not fully considered. Recent studies also indicate that simply applying MLPs to point clouds cannot always work in practice [28,49].

In this paper, we introduce 3D grids as intermediate representations to regularize point clouds and further propose a differentiable *Gridding* layer, which converts an unordered and irregular point cloud $P = \{p_i\}_{i=1}^{n}$ into a regular 3D grid $\mathcal{G} =< V, W >$ while preserving spatial layouts of the point cloud, where $p_i \in \mathbb{R}^3$, $V = \{v_i\}_{i=1}^{N^3}$, $W = \{w_i\}_{i=1}^{N^3}$, $v_i \in \{(-\frac{N}{2}, -\frac{N}{2}, -\frac{N}{2}), \ldots, (\frac{N}{2}-1, \frac{N}{2}-1, \frac{N}{2}-1)\}$, $w_i \in \mathbb{R}$, n is the number of points in P, and N is the resolution of the 3D grid $\mathcal{G}$. As shown in Figure 1 (b), we define a cell as a cubic consisting of eight vertices. For each vertex $v_i = (x_i^v, y_i^v, z_i^v)$ of the 3D grid cell $\mathcal{G}$, we define the neighboring points $\mathcal{N}(v_i)$ as points that lie in the adjacent 8 cells of this vertex. The point $p = (x, y, z) \in \mathcal{N}(v_i)$ is defined as a neighboring point of vertex v_i by satisfying $p \in P$, $x_i^v - 1 < x < x_i^v + 1$, $y_i^v - 1 < y < y_i^v + 1$, and $z_i^v - 1 < z < z_i^v + 1$, respectively. In standard voxelization, value w_i at the vertex v_i is computed as

$$w_i = \begin{cases} 0 & \forall p \notin \mathcal{N}(v_i) \\ 1 & \exists p \in \mathcal{N}(v_i) \end{cases} \tag{1}$$

However, this voxelization process introduces a quantization effect that discards some details of an object. In addition, voxelization is not differentiable and thus can not be applied to point cloud reconstruction. As illustrated in Figure 1 (b), given a vertex v_i and its neighboring points $p \in \mathcal{N}(v_i)$, the proposed *Gridding* layer computes the corresponding value w_i of this vertex v_i as

$$w_i = \sum_{p \in \mathcal{N}(v_i)} \frac{w(v_i, p)}{|\mathcal{N}(v_i)|} \tag{2}$$

where $|\mathcal{N}(v_i)|$ is the number of neighboring points of v_i. Specially, we define $w_i = 0$ if $|\mathcal{N}(v_i)| = 0$. The interpolation function $w(v_i, p)$ is defined as

$$w(v_i, p) = (1 - |x_i^v - x|)(1 - |y_i^v - y|)(1 - |z_i^v - z|) \tag{3}$$

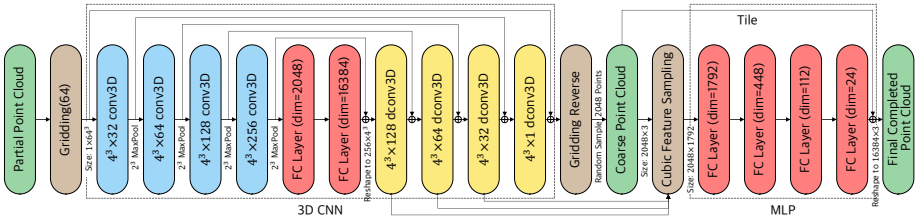

Fig. 2. The network architecture of GRNet. $\bigoplus$ denotes the sum operation. Tile creates a new tensor of size 16384×3 by replicating the "Coarse Point Cloud" 8 times.

3.3 3D Convolutional Neural Network

The 3D Convolutional Neural Network (3D CNN) with skip connections aims to complete the missing parts of the incomplete point cloud. It follows the idea of a 3D encoder-decoder with U-net connections [46,47]. Given W as input, the 3D CNN can be formulated as

$$W' = \text{3DCNN}(W) \tag{4}$$

where $W' = \{w'_i\}_{i=1}^{N^3}$ and $w'_i \in \mathbb{R}$.

As shown in Figure 2, the encoder of the 3D CNN has four 3D convolutional layers, each of which has a bank of 4^3 filters with padding of 2, followed by batch normalization, leaky ReLU activation, and a max pooling layer with a kernel size of 2^3. The numbers of output channels of convolutional layers are 32, 64, 128, 256, respectively. The encoder is finally followed by two fully connected layers with dimensions of 2048 and 16384. The decoder consists of four transposed convolutional layers, each of which has a bank of 4^3 filters with padding of 2 and stride of 1, followed by a batch normalization layer and a ReLU activation.

3.4 Gridding Reverse

As illustrated in Figure 1 (c), we propose *Gridding Reverse* to generate the coarse point cloud $P^c = \{p_i^c\}_{i=1}^m$ from the 3D grid $\mathcal{G}' =< V, W' >$, where $p_i^c \in \mathbb{R}^3$ and m is the number of points in the coarse point cloud P^c. Let $\Theta^i = \{\theta_j^i\}_{j=1}^8$ be the index set of vertices of the $i-$th 3D grid cell. *Gridding Reverse* generates one point coordinate p_i^c for this grid cell by a weighted combination of eight vertices coordinates $\{v_\theta | \theta \in \Theta^i\}$ and the corresponding values $\{w'_\theta | \theta \in \Theta^i\}$ in this cell, which is computed as

$$p_i^c = \frac{\sum_{\theta \in \Theta^i} w'_\theta v_\theta}{\sum_{\theta \in \Theta^i} w'_\theta} \tag{5}$$

Specially, we ignore the point p_i^c for this cell if $\sum_{\theta \in \Theta^i} w'_\theta = 0$.

3.5 Cubic Feature Sampling

MLP-based methods (*e.g.*, PCN) are unable to take the context of neighboring points into account due to no local spatial connectivity across points. These

methods use max-pooling to aggregate information globally, which may lose local context information.

To overcome this issue, we present *Cubic Feature Sampling* to aggregate features $F^c = \{f^c\}_{i=1}^m$ for the coarse point cloud P^c, which is helpful for the following MLP to recover the details of point clouds, as shown in Figure 1 (d). Let $\mathcal{F} = \{f_1^v, f_2^v, \ldots, f_{t^3}^v\}$ be the feature map of 3D CNN, where $f_i^v \in \mathbb{R}^c$ and t^3 is the size of the feature map. For a point p_i^c of the coarse point cloud P^c, its features f_i^c are computed as

$$f_i^c = [f_{\theta_1^i}^v, f_{\theta_2^i}^v, \ldots, f_{\theta_8^i}^v] \tag{6}$$

where $[\cdot]$ is the concatenation operation. $\{f_{\theta_j^i}^v\}_{j=1}^8$ denotes the features of eight vertices of the i-th 3D gird cell where p_i^c lies in.

In *GRNet*, *Cubic Feature Sampling* extracts the point features from feature maps generated by the first three transposed convolutional layers in 3D CNN. To reduce the redundancy of these features and generate a fixed number of points, we randomly sample $2,048$ points from the coarse point cloud P^c. Consequently, it produces a feature map of size 2048×1792.

3.6 Multi-layer Perceptron

The Multi-layer Perceptron (MLP) is used to recover the details from the coarse point cloud by learning residual offsets between the coordinates of points in the coarse and final completed point cloud. It takes the coarse point cloud P^c and the corresponding features F^c as input, and outputs the final completed point cloud $P^f = \{p_i^f\}_{i=1}^k$ as

$$P^f = \mathrm{MLP}(F^c) + \mathrm{Tile}(P^c, r) \tag{7}$$

where $p_i^f \in \mathbb{R}^3$ and k is the number of points in the final completed point cloud P^f. Tile creates a new tensor of size $rm \times 3$ by replicating P^c r times.

In *GRNet*, r is set to 8. The MLP consists of four fully connected layers with dimensions of 1792, 448, 112, and 24, respectively. The output of MLP is reshaped to 16384×3, which corresponds to the offsets of the coordinates of $16,384$ points.

3.7 Gridding Loss

Existing methods adopt Chamfer Distance [4] as the loss function to train the neural networks. This loss function penalizes the prediction deviating from the ground-truth. However, it can not guarantee that the predicted points follow the geometric layout of the object. Therefore the networks tend to output a mean shape that minimizes the distance, which causes the loss of the object's details [11,48].

Due to the unorderedness of point clouds, it is difficult to directly apply binary cross-entropy like voxels or L1/L2 loss like images. With the proposed

Gridding, we can convert unordered point clouds into regular 3D grids (Figure 1 (e)). Therefore, we design a new loss function based on *Gridding*, namely *Gridding Loss*, which is defined as the L1 distance between value sets of the two 3D grids. Let $\mathcal{G}_{pred} =< V^{pred}, W^{pred} >$ and $\mathcal{G}_{gt} =< V^{gt}, W^{gt} >$ be the 3D grids obtained by *Gridding* the predicted and ground truth point clouds, respectively, where $W^{pred} \in \mathbb{R}^{N_G^3}$, $W^{gt} \in \mathbb{R}^{N_G^3}$, and N_G is the resolution of the two 3D grids. The *Gridding Loss* can be defined as

$$\mathcal{L}_{Gridding}(W^{pred}, W^{gt}) = \frac{1}{N^3} \sum ||W^{pred} - W^{gt}|| \tag{8}$$

4 Experiments

4.1 Datasets

ShapeNet. The ShapeNet dataset [45] for point cloud completion is derived from PCN [51], which consists of 30,974 3D models from 8 categories. The ground truth point clouds containing 16,384 points are uniformly sampled on mesh surfaces. The partial point clouds are generated by back-projecting 2.5D depth maps into 3D. For a fair comparison, we use the same train/val/test splits as PCN.

Completion3D. The Completion3D benchmark [38] is composed of 28,974 and 800 samples for training and validation, respectively. Different from the ShapeNet dataset generated by PCN, there are only 2,048 points in the ground truth point clouds.

KITTI. The KITTI dataset [5] is composed of a sequence of real-world Velodyne LiDAR scans, also derived from PCN [51]. For each frame, the car objects are extracted according to the 3D bounding boxes, which results in 2,401 partial point clouds. The partial point clouds in KITTI are highly sparse and do not have complete point clouds as ground truth.

4.2 Evaluation Metrics

Let $\mathcal{T} = \{(x_i, y_i, z_i)\}_{i=1}^{n_\mathcal{T}}$ be the ground truth and $\mathcal{R} = \{(x_i, y_i, z_i)\}_{i=1}^{n_\mathcal{R}}$ be a reconstructed point set being evaluated, where $n_\mathcal{T}$ and $n_\mathcal{R}$ are the numbers of points of $\mathcal{T}$ and $\mathcal{R}$, respectively. In our experiments, we use both Chamfer Distance and F-Score as quantitative evaluation metrics.

Chamfer Distance. Follow PSGN [4] and TopNet [38], the distance between $\mathcal{T}$ and $\mathcal{R}$ are defined as

$$\text{CD} = \frac{1}{n_\mathcal{T}} \sum_{t \in \mathcal{T}} \min_{r \in \mathcal{R}} ||t - r||_2^2 + \frac{1}{n_\mathcal{R}} \sum_{r \in \mathcal{R}} \min_{t \in \mathcal{T}} ||t - r||_2^2 \tag{9}$$

F-Score. As pointed out in [37], Chamfer Distance may sometimes be misleading. As suggested in [37], we take F-Score as an extra metric to evaluate the performance of point completion results, which can be defined as following

$$\text{F-Score}(d) = \frac{2P(d)R(d)}{P(d) + R(d)} \tag{10}$$

Table 1. Point completion results on ShapeNet compared using Chamfer Distance (CD) with L2 norm computed on 16,384 points and multiplied by 10^4. The best results are highlighted in bold.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Overall
AtlasNet [6]	1.753	5.101	3.237	5.226	6.342	5.990	4.359	4.177	4.523
PCN [51]	**1.400**	4.450	**2.445**	4.838	6.238	5.129	3.569	4.062	4.016
FoldingNet [50]	3.151	7.943	4.676	9.225	9.234	8.895	6.691	7.325	7.142
TopNet [38]	2.152	5.623	3.513	6.346	7.502	6.949	4.784	4.359	5.154
MSN [23]	1.543	7.249	4.711	4.539	6.479	5.894	3.797	3.853	4.758
GRNet	1.531	**3.620**	2.752	**2.945**	**2.649**	**3.613**	**2.552**	**2.122**	**2.723**

Table 2. Point completion results on ShapeNet compared using F-Score@1%. Note that the F-Score@1% is computed on 16,384 points. The best results are highlighted in bold.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Overall
AtlasNet [6]	0.845	0.552	0.630	0.552	0.565	0.500	0.660	0.624	0.616
PCN [51]	0.881	**0.651**	**0.725**	0.625	0.638	0.581	0.765	0.697	0.695
FoldingNet [50]	0.642	0.237	0.382	0.236	0.219	0.197	0.361	0.299	0.322
TopNet [38]	0.771	0.404	0.544	0.413	0.408	0.350	0.572	0.560	0.503
MSN [23]	**0.885**	0.644	0.665	0.657	0.699	0.604	**0.782**	0.708	0.705
GRNet	0.843	0.618	0.682	**0.673**	**0.761**	**0.605**	0.751	**0.750**	**0.708**

where $P(d)$ and $R(d)$ denote the precision and recall for a distance threshold d, respectively.

$$P(d) = \frac{1}{n_{\mathcal{R}}} \sum_{r \in \mathcal{R}} \left[\min_{t \in \mathcal{T}} ||t - r|| < d \right] \tag{11}$$

$$R(d) = \frac{1}{n_{\mathcal{T}}} \sum_{t \in \mathcal{T}} \left[\min_{r \in \mathcal{R}} ||t - r|| < d \right] \tag{12}$$

4.3 Implementation Details

We implement our network using PyTorch [30] and CUDA[1]. All models are optimized with an Adam optimizer [13] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train the network with a batch size of 32 on two NVIDIA TITAN Xp GPUs. The initial learning rate is set to $1e-4$ and decayed by 2 after 50 epochs. The optimization is set to stop after 150 epochs.

4.4 Shape Completion on ShapeNet

To compare the performance of *GRNet* with other state-of-the-art methods, we conduct experiments on the ShapeNet dataset. **AtlasNet** [6] generates a point

[1] The source code is available at https://github.com/hzxie/GRNet.

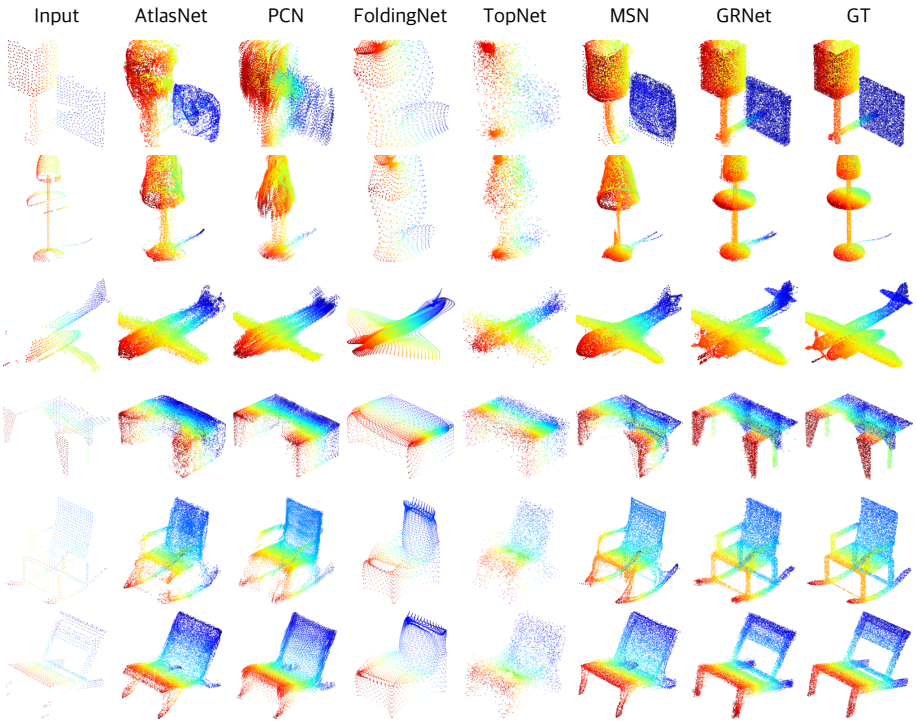

Fig. 3. Qualitative completion results on the ShapeNet testing set. GT stands for the ground truth of the 3D object.

cloud with a set of parametric surface elements. To compare with other methods fairly, we sample 16,384 points from the generated primitive surface elements. **PCN** [51] completes the partial point cloud with a stacked version of PointNet [32], which directly outputs the coordinates of 16,384 points. **FoldingNet** [50] is a baseline method adopted in PCN [51], which deforms a 128×128 2D grid into 3D point cloud. **TopNet** [38] incorporates a decoder following a hierarchical rooted tree structure to consider the topology of point clouds. Due to the scalable architecture of TopNet, it can easily generate 16,384 points by setting the number of nodes and the size of feature embedding. A very recent method **MSN** [23] generates dense point cloud containing 8,192 points in a coarse-to-fine fashion. To generate 16,384 points, we combine the generated points of 2 times forward propagation.

Quantitative results in Tables 2 and 1 indicate that *GRNet* outperforms all competitive methods in terms of Chamfer Distance and F-Score@1%. Figure 3 shows the qualitative results for point completion on ShapeNet, which indicates that the proposed method recovers better details of objects (*e.g.*, chairs and lamps) than the other methods.

Table 3. Point completion results on Completion3D compared using Chamfer Distance (CD) with L2 norm. Note that the CD is computed on 2,048 points and multiplied by 10^4. The best results are highlighted in bold.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Overall
AtlasNet [6]	10.36	23.40	13.40	24.16	20.24	20.82	17.52	11.62	17.77
FoldingNet [50]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
PCN [51]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet [38]	7.32	18.77	12.88	19.82	14.60	16.29	14.89	8.82	14.25
GRNet	**6.13**	**16.90**	**8.27**	**12.23**	**10.22**	**14.93**	**10.08**	**5.86**	**10.64**

4.5 Shape Completion on Completion3D

Using the model with the lowest Chamfer Distance (CD) on the validation set, we recover the complete point clouds for 1,184 objects in the Completion3D testing set. Then, random subsampling is applied to the generated point clouds to obtain 2,048 points for benchmark evaluation. According to the online leaderboard [2], as shown in Table 3, the overall CD for the proposed *GRNet* is 10.64, which remarkably outperforms all state-of-the-art methods and ranks first on this benchmark.

4.6 Shape Completion on KITTI

To evaluate the performance of the proposed method on real-world LiDAR scans, we test *GRNet* on the KITTI dataset for completing sparse point clouds of cars. Unlike ShapeNet generated by back-projected from 2.5D images, point clouds from LiDAR scans can be highly sparse, which are much sparser than those in ShapeNet.

We fine-tuned all competitive methods on ShapeNetCars (the cars from ShapeNet) except PCN that directly uses released output for evaluation. During testing, each point cloud is transformed into the bounding box's coordinates and transformed back to the world frame after completion. The models trained specifically on cars are able to incorporate prior knowledge of the object class.

Since there are no complete ground truth point clouds for KITTI, we use Consistency and Uniformity to evaluate the performance of all competitive methods. Consistency in PCN [51] is the average CD between the output of the same car instance in n_f consecutive frames. Let $\mathcal{R}^j_{t_i}$ be the output for the j-th car instance at time t_i. The Consistency for the j-th car can be calculated as

$$\text{Consistency} = \frac{1}{n_f - 1} \sum_{i=2}^{n_f} \text{CD}(\mathcal{R}^j_{t_{i-1}}, \mathcal{R}^j_{t_i}) \tag{13}$$

[2] https://completion3d.stanford.edu/results

Table 4. Point completion results on LiDAR scans from KITTI compared using Consistency and Uniformity. The best results are highlighted in bold.

Methods	Consistency ($\times 10^{-3}$)	Uniformity for different p				
		0.4%	0.6%	0.8%	1.0%	1.2%
AtlasNet [6]	0.700	1.146	1.005	0.874	0.761	0.686
PCN [51]	1.557	3.662	5.812	7.710	9.331	10.823
FoldingNet [50]	1.053	1.245	1.303	1.262	1.162	1.063
TopNet [38]	0.568	1.353	1.326	1.219	1.073	0.950
MSN [23]	1.951	0.822	0.675	0.523	0.462	0.383
GRNet	**0.313**	**0.632**	**0.572**	**0.489**	**0.410**	**0.352**

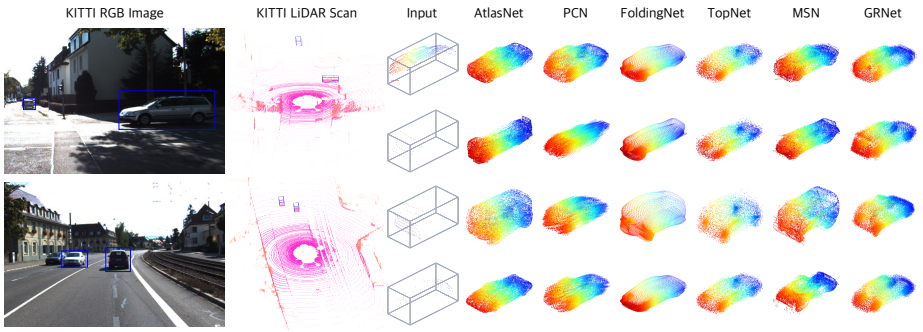

Fig. 4. Qualitative completion results on the LiDAR scans from KITTI. The incomplete input point cloud is extracted and normalized from the scene according to its 3D bounding box.

Following PU-GAN [19], we adopt Uniformity to evaluate the distribution uniformity of the completed point clouds, which can be formulated as

$$\text{Uniformity}(p) = \frac{1}{M}\sum_{i=1}^{M} \text{U}_{\text{imbalance}}(S_i)\text{U}_{\text{clutter}}(S_i) \tag{14}$$

where $S_i (i = 1, 2, \ldots, M)$ is a point subset cropped from a patch of the output $\mathcal{R}$ using the farthest sampling and ball query of radius $\sqrt{p}$. The term $\text{U}_{\text{imbalance}}$ and $\text{U}_{\text{clutter}}$ account for the global and local distribution uniformity, respectively.

$$\text{U}_{\text{imbalance}}(S_i) = \frac{(|S_i| - \hat{n})^2}{\hat{n}} \tag{15}$$

where $\hat{n} = p|\mathcal{R}|$ is the expected number of points in S_i.

$$\text{U}_{\text{clutter}}(S_i) = \frac{1}{|S_i|}\sum_{j=1}^{|S_i|} \frac{(d_{i,j} - \hat{d})^2}{\hat{d}} \tag{16}$$

Table 5. The Chamfer Distance (CD), F-Score@1%, numbers of parameters, and backward time on ShapeNet with different resolutions of 3D grids generated by *Gridding*. The backward time is measured on an NVIDIA TITAN Xp GPU with batch size of 1.

Resolutions	CD ($\times 10^{-4}$)		F-Score@1%		# Parameters	Backward Time
	Coarse	Complete	Coarse	Complete	(M)	(ms)
32^3	23.339	5.943	0.329	0.549	69.54	64
64^3	**11.259**	**2.723**	0.340	0.708	76.70	100
128^3	12.383	2.732	**0.366**	**0.712**	76.77	302

Table 6. The Chamfer Distance (CD), F-Score@1%, and numbers of parameters of MLPs on ShapeNet with different features maps feeding into *Cubic Feature Sampling*. The backward time is measured on an NVIDIA TITAN Xp GPU with batch size of 1.

The Size of Feature Maps			CD	F-Score	# Parameters	Backward Time
128×8^3	64×16^3	32×32^3	($\times 10^{-4}$)	@1%	(M)	(ms)
			11.375	0.343	0	72
		✓	2.922	0.640	0.11	80
	✓	✓	2.805	0.686	0.96	88
✓	✓	✓	**2.723**	**0.708**	4.07	100

where $d_{i,j}$ represents the distance to the nearest neighbor for the j-th point in S_i, and $\hat{d}$ is roughly $\sqrt{\frac{2\pi p}{|S_i|\sqrt{3}}}$ if S_i has a uniform distribution [19].

Table 4 shows the completion results for cars in the LiDAR scans from the KITTI dataset. Experimental results indicate that *GRNet* outperforms other competitive methods in terms of Consistency and Uniformity. Benefited from *Gridding* and *Gridding Reverse*, *GRNet* is more sensitive to the spatial structure of the input points, which leads to better consistency between the two consecutive frames. As shown in Figure 4, the cars are barely recognizable due to incompleteness of the input data. In contrast, the completed point clouds provide more geometric information. In addition, the qualitative results also demonstrate the proposed method generates more reasonable shape completion.

4.7 Ablation Study

The performance improvement of *GRNet* should be attributed to three key components, including *Gridding*, *Cubic Feature Sampling*, and *Gridding Loss*. To demonstrate the effectiveness of each component in the proposed method, we evaluate the performance with different parameters.

Gridding. Table 5 shows the results of different resolutions of 3D grids generated by *Gridding*. The F-Score of final completed point clouds increases with the 3D grids' resolutions. However, the numbers of parameters and the backward time

Table 7. The Chamfer Distance (CD) and F-Score@1% on ShapeNet with different resolutions of 3D grids generated by *Gridding Loss*. The backward time is measured on an NVIDIA TITAN Xp GPU with batch size of 1.

Resolutions	CD ($\times 10^{-4}$)		F-Score@1%		Backward Time (ms)
	Coarse	Complete	Coarse	Complete	
Not Used	11.259	4.460	0.340	0.624	86
64^3	10.275	3.427	0.364	0.672	92
128^3	**9.324**	**2.723**	**0.386**	**0.708**	100

also increases. To archive a balance between effect and efficiency, we choose the resolution of size 64^3 for *Gridding* in *GRNet*.

Cubic Feature Sampling. To quantitatively evaluate the effect of *Cubic Feature Sampling*, we compare the performance without *Cubic Feature Sampling* and with different feature maps fed into it. The experimental results presented in Table 6 indicate that *Cubic Feature Sampling* improves the point cloud completion results significantly. In addition, with more feature maps are fed, the completion quality becomes better without a significant increase in the numbers of parameters and backward time.

Gridding Loss. We further validate the effects of *Gridding Loss*, as shown in Table 7. There is a decrease in terms of both CD and F-Score when removing *Gridding Loss*. When increasing the resolution of 3D grids from 64^3 to 128^3, there are 25.9% and 5.4% improvements in CD and F-Score, respectively.

5 Conclusion

In this paper, we study how to recover the complete 3D point cloud from an incomplete one. The main motivation of this work is to enable the convolutions on 3D point clouds while preserving their structural and context information. To this aim, we introduce 3D grids as intermediate representations to regularize unordered point clouds. We then propose a novel Gridding Residual Network (GRNet) for point cloud completion, which contains three novel differentiable layers: *Gridding*, *Gridding Reverse*, and *Cubic Feature Sampling*, as well as a new *Gridding Loss*. Extensive comparisons are conducted on the ShapeNet, Completion3D, and KITTI benchmarks, which indicate that the proposed *GRNet* performs favorably against state-of-the-art methods.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Nos. 61772158, 61702136 and 61872112), National Key Research and Development Program of China (Nos. 2018YFC0806802 and 2018YFC0832105), and Self-Planned Task (No. SKLRS202002D) of State Key Laboratory of Robotics and System (HIT).

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J.: Learning representations and generative models for 3D point clouds. In: ICML 2018 (2018) 3
2. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I.D., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016) 1
3. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In: CVPR 2017 (2017) 2, 4, 5
4. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3D object reconstruction from a single image. In: CVPR 2017 (2017) 3, 7, 8
5. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal Robotics Research (IJRR) **32**(11), 1231–1237 (2013) 8
6. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3D surface generation. In: CVPR 2018 (2018) 4, 9, 11, 12
7. Han, X., Li, Z., Huang, H., Kalogerakis, E., Yu, Y.: High-resolution shape completion using deep neural networks for global structure and local geometry inference. In: ICCV 2017 (2017) 2, 4, 5
8. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: ICCV 2019 (2019) 4
9. Hermosilla, P., Ritschel, T., Vázquez, P., Vinacua, A., Ropinski, T.: Monte carlo convolution for learning on non-uniformly sampled point clouds. ACM Transactions on Graphics **37**(6), 235:1–235:12 (2018) 4
10. Hua, B., Tran, M., Yeung, S.: Pointwise convolutional neural networks. In: CVPR 2018 (2018) 4
11. Jiang, L., Shi, S., Qi, X., Jia, J.: GAL: geometric adversarial loss for single-view 3D-object reconstruction. In: ECCV 2018 (2018) 3, 7
12. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. In: NIPS 2017 (2017) 3
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR 2015 (2015) 9
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR 2017 (2017) 2
15. Lan, S., Yu, R., Yu, G., Davis, L.S.: Modeling local geometric structure of 3D point clouds using Geo-CNN. In: CVPR 2019 (2019) 4
16. Lei, H., Akhtar, N., Mian, A.: Octree guided CNN with spherical kernels for 3D point clouds. In: CVPR 2019 (2019) 4
17. Li, D., Shao, T., Wu, H., Zhou, K.: Shape completion from a single RGBD image. IEEE Transactions on Visualization and Computer Graphics **23**(7), 1809–1822 (2017) 4, 5
18. Li, K., Pham, T., Zhan, H., Reid, I.D.: Efficient dense point cloud object reconstruction using deformation vector fields. In: ECCV 2018 (2018) 3
19. Li, R., Li, X., Fu, C., Cohen-Or, D., Heng, P.: PU-GAN: a point cloud upsampling adversarial network. In: ICCV 2019 (2019) 12, 13
20. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on x-transformed points. In: NeurIPS 2018 (2018) 4
21. Lin, C., Kong, C., Lucey, S.: Learning efficient point cloud generation for dense 3D object reconstruction. In: AAAI 2018 (2018) 3

22. Lin, H., Xiao, Z., Tan, Y., Chao, H., Ding, S.: Justlookup: One millisecond deep feature extraction for point clouds by lookup tables. In: ICME 2019 (2019) 3
23. Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S.M.: Morphing and sampling network for dense point cloud completion. In: AAAI 2020 (2020) 4, 9, 10, 12
24. Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., Pan, C.: DensePoint: Learning densely contextual representation for efficient point cloud processing. In: ICCV 2019 (2019) 4
25. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: CVPR 2019 (2019) 4
26. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel CNN for efficient 3D deep learning. In: NeurIPS 2019 (2019) 2, 4, 5
27. Mandikal, P., Radhakrishnan, V.B.: Dense 3D point cloud reconstruction using a deep pyramid nxetwork. In: WACV 2019 (2019) 2, 3
28. Mao, J., Wang, X., Li, H.: Interpolated convolutional networks for 3D point cloud understanding. In: ICCV 2019 (2019) 2, 4, 5
29. Nguyen, D.T., Hua, B., Tran, M., Pham, Q., Yeung, S.: A field model for repairing 3D shapes. In: CVPR 2016 (2016) 2
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.a.: PyTorch: An imperative style, high-performance deep learning library. In: NeurIPS 2019 (2019) 9
31. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6dof pose estimation. In: CVPR 2019 (2019) 3
32. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: CVPR 2017 (2017) 3, 10
33. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NIPS 2017 (2017) 3
34. Sharma, A., Grau, O., Fritz, M.: VConv-DAE: Deep volumetric shape learning without object labels. In: ECCV 2016 Workshops (2016) 2
35. Stutz, D., Geiger, A.: Learning 3D shape completion from laser scan data with weak supervision. In: CVPR 2018 (2018) 2
36. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: CVPR 2018 (2018) 2, 4
37. Tatarchenko, M., Richter, S.R., Ranftl, R., Li, Z., Koltun, V., Brox, T.: What do single-view 3D reconstruction networks learn? In: CVPR 2019 (2019) 8
38. Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I.D., Savarese, S.: TopNet: Structural point cloud decoder. In: CVPR 2019 (2019) 2, 4, 5, 8, 9, 10, 11, 12
39. Thomas, H., Qi, C.R., Deschaud, J., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV 2019 (2019) 2, 4
40. Varley, J., DeChant, C., Richardson, A., Ruales, J., Allen, P.K.: Shape completion enabled robotic grasping. In: IROS 2017 (2017) 2
41. Wang, K., Chen, K., Jia, K.: Deep cascade generation on point sets. In: IJCAI 2019 (2019) 2, 4
42. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics **38**(5), 146:1–146:12 (2019) 2, 4

43. Wang, Z., Lu, F.: VoxSegNet: Volumetric CNNs for semantic part segmentation of 3D shapes. IEEE Transactions on Visualization and Computer Graphics p. DOI: 10.1109/TVCG.2019.2896310 (2019) 4
44. Wu, W., Qi, Z., Li, F.: PointConv: Deep convolutional networks on 3D point clouds. In: CVPR 2019 (2019) 4
45. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: CVPR 2015 (2015) 8
46. Xie, H., Yao, H., Sun, X., Zhou, S., Zhang, S.: Pix2Vox: Context-aware 3D reconstruction from single and multi-view images. In: ICCV 2019 (2019) 6
47. Xie, H., Yao, H., Zhang, S., Zhou, S., Sun, W.: Pix2Vox++: Multi-scale context-aware 3D object reconstruction from single and multiple images. IJCV (2020). https://doi.org/10.1007/s11263-020-01347-6 6
48. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: DISN: deep implicit surface network for high-quality single-view 3D reconstruction. In: NeurIPS 2019 (2019) 3, 7
49. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In: ECCV 2018 (2018) 4, 5
50. Yang, Y., Feng, C., Shen, Y., Tian, D.: FoldingNet: Point cloud auto-encoder via deep grid deformation. In: CVPR 2018 (2018) 9, 10, 11, 12
51. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: PCN: point completion network. In: 3DV 2018 (2018) 2, 3, 5, 8, 9, 10, 11, 12
52. Zhang, K., Hao, M., Wang, J., de Silva, C.W., Fu, C.: Linked Dynamic Graph CNN: learning on point cloud via linking hierarchical features. arXiv 1904.10014 (2019) 4