CarveNet: Carving Point-Block for Complex 3D Shape Completion

Qing Guo*, *Member, IEEE,* Zhijie Wang*, *Student Member, IEEE,* Lubo Wang, *Student Member, IEEE,* Haotian Dong, *Student Member, IEEE,* Felix Juefei-Xu, *Member, IEEE,* Di Lin†, *Member, IEEE,* Lei Ma, *Member, IEEE,* Wei Feng, *Member, IEEE,* Yang Liu, *Senior Member, IEEE*

***Abstract*—3D point cloud completion is very challenging because it relies on accurately understanding the complex 3D shapes (*e.g.*, high-curvature, concave/convex, and hollowed-out 3D shapes) and the unknown & diverse patterns of the partially available point clouds. In this paper, we propose a novel solution, *i.e.*, *Point-block Carving* (PC), for completing the complex 3D point cloud completion. Given the partial point cloud as the guidance, we carve a 3D block that contains the uniformly distributed 3D points, yielding the entire point cloud. We propose a new network architecture to achieve PC, *i.e.*, *CarveNet*. This network conducts the exclusive convolution on each block point, where the convolutional kernels are trained on the 3D shape data. CarveNet determines which point should be carved to recover the complete shapes' details effectively. Furthermore, we propose a sensor-aware method for data augmentation, *i.e.*, *SensorAug*, for training CarveNet on richer patterns of partial point clouds, thus enhancing the completion power of the network. The extensive evaluations on the ShapeNet, ShapNet-55/34 and KITTI datasets demonstrate the generality of our approach on the partial point clouds with diverse patterns. On these datasets, CarveNet successfully outperforms the state-of-the-art methods.**

***Index Terms*—Point Cloud Completion, Shape Completion, CarveNet, SensorAug.**

I. Introduction

3D point cloud has been regarded as one of the best representations of the 3D object [1], [2]. It is widely adopted in an array of robotic-relevant applications, such as the simultaneous localization and mapping (SLAM) [3], place recognition [4], [5], object detection [6], LiDAR processing for autonomous driving [7], *etc*. To achieve the 3D point cloud of the entire object reasonably well, it is necessary to use many range finders to capture the geometric data from different views and conduct the accurate registration among the captured data. However, the expensive, heavy, and energy-consuming finders are restricted to only a few affordable applications in practice. Yet, the sparse 3D points can often lose the geometric and semantic information, thus leading to the robotic system's performance degradation.

Qing Guo is with the Institute of High Performance Computing (IHPC) and Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore. Zhijie Wang is with the University of Alberta, Canada. Felix Juefei-Xu is with New York University, USA. Lubo Wang, Haotian Dong, Di Lin, and Wei Feng are with the School of Computer Science and Technology, College of Intelligence and Computing, Tianjin University, China. Lei Ma is with The University of Tokyo, Japan and the University of Alberta, Canada. Yang Liu is with the Nanyang Technological University, Singapore. *Qing Guo and Zhijie Wang contribute equally to this work and are co-first authors. † Di Lin is the corresponding author.

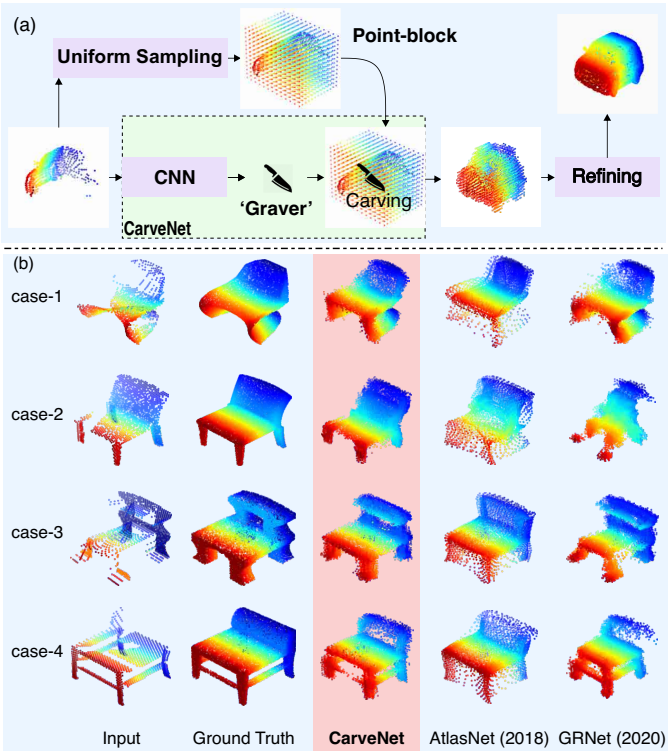


Fig. 1: (a) The intuitive idea and pipeline of our point-block carving with CarveNet. (b) The point cloud completion by AtlasNet [8], GRNet [9], and CarveNet on four cases with three complex shapes: high-curvature (case-1 and case-2), concave/convex (case-2 and case-3) and hollowed-out (case-3 and case-4).

The emergency of the point cloud completion methods alleviates the negative effect of the imperfect point cloud on the downstream applications [10]. The current methods still face challenges when dealing with complex object shapes, such as the shapes (see Fig. 1) with high curvatures (*e.g.*, 1^{st} case), concave/convex structures (*e.g.*, 2^{nd} and 3^{rd} cases), and hollowed-out structures (*e.g.*, 3^{rd} and 4^{th} cases). The methods [8], [9], [11]–[17] that focus on some kinds of complex shapes may lose the generality power for handling the completion of other shapes. For example, the recent method GRNet [9] loses some critical structures in the 2^{nd} and 4^{th} cases.

Similarly, although the advanced methods such as AtlasNet [8] employ the parameterized shape for flexible completion,

the results lack the critical visual details (*e.g.*, the concave and hollowed-out structures in 2^{nd} and 3^{rd} cases). Moreover, given the similar (or even identical) object shapes, the partial point clouds captured by different sensors may be variant. This is because the intrinsic performance of the sensors heavily affects the visual patterns of the partial point clouds. The variant information can easily mislead the completion methods, leading to inconsistent results. This post-pressing concerns especially for real-world applications and calls for better ways to reduce the impact of the variant sensors on the point cloud completion.

This paper proposes a brand-new *point-block carving* (PC). Given the partial point cloud as the input, our approach leverages a set of "gravers" to carve the point block to approximate the underlying object shape as similarly as possible (see Fig. 1). Intuitively, our carving process can be understood as the carving of a statue, where the partial shape of the statue is provided as a hint. Here, our carving process is done on the 3D block, where we add a set of uniformly distributed 3D points at the beginning. We also register the partial point cloud as the 3D block. Next, we use the *CarveNet* to generate a graver to process the 3D block and recover the object shape. This is done by using the graver to remove the redundant 3D points. Specifically, the graver is defined by the point-wise convolution whose kernels are learned from the prior knowledge of the object that captures the geometric and semantic relationships between the existing and the missing 3D points in the 3D block. The graver propagates the useful prior information from the partially-given points to the add-in uniform points, whose present/absent statuses are predicted to recover the lost part of the point cloud. We only process the add-in 3D points to reduce the carving complexity effectively. Note that the features produced by different convolutional layers of CarveNet contain rich semantic information about the objects. We use these features to regress new 3D points that are added to the block for a denser completion result.

Moreover, we propose a new *SensorAug* for augmenting the training data. SensorAug works with a similarity loss. Given a complete point cloud, we assume a LiDAR sensor is randomly placed to observe the object, leading to some invisible points. These hidden points are removed from the complete cloud, producing the partial cloud. Based on the same complete point cloud, the similarity loss of SensorAug involves more diversity of the partial point clouds. It allows CarveNet to be trained on more diverse data to enhance the completion power of complex shapes.

We evaluate our method on the ShapeNet and KITTI datasets. With CarveNet, we successfully improve the completion performance on different benchmarks, even in challenging cases where the critical points are unavailable for predicting complex object shapes. Our method helps to achieve better results than the state-of-the-art methods. Our contribution is manifold:

- We promote a novel paradigm based on the carving of the point block for point cloud completion.
- We originally propose CarveNet for point-block engraving, facilitating better completion of the complex shapes.
- We propose SensorAug for data augmentation. SensorAug enhances the generality power of the completion model and helps the model achieve state-of-the-art results on the public benchmarks.

II. Related Works

We mainly discuss the deep-learning networks with different architectures for 3D point cloud completion. We also discuss other representations of the point cloud.

A. Network Architectures for Point Cloud Completion

The deep network has been used to construct many advanced point cloud completion methods. The *folding-based* and *multilayer perceptron (MLP)-based* network are two kinds of famous architectures.

The literature on the folding-based architecture is vast. FoldingNet [11] folds 2D grid points twice into the target object's surface, with the guidance of the feature vectors extracted by PointNet [18]. However, the complex object shape significantly increases the difficulty of folding the object at the cost of expensive computation. In contrast, AtlasNet [8] separates the object's surface into small patches. Each patch is folded individually. MSN [14] employs the expansion loss to deal with overlapping patches. PCN [12] is equipped with a two-stage completion. PCN uses MLPs to predict the coarse shape, which is processed by the deformation of the 2D grids for the denser completion result. PoinTr [19] formulates point cloud completion as a set-to-set translation problem and designs a Transformer architecture to learn and generate point proxies.

There have been many works based on the MLP-based architecture. TopNet [13] involves a tree-structure decoder, where MLP is used to connect the tree nodes. MLP can be used for producing the multi-scale features [15], [20], [21] to assist the completion. Several works [20], [22]–[24] explore the generative adversarial networks (GAN) [25] to construct MLPs to produce more realistic completion results. PMP-Net [26] builds a process of point cloud gradual deformation by learning multi-step offset for each point, achieving point cloud completion based on existing points in the partial point cloud. SnowflakeNet [27] uses point splitting and offset to gradually generate new points, making the original incomplete point cloud grow progressively. VRCNet [28] designs a shared-weight dual-branch structure to achieve probabilistic modeling between incomplete and complete point clouds. SeedFormer [29] uses Patch Seeds to represent point clouds and designs an Upsample Transformer to perform coarse-to-fine point cloud completion.

In this paper, we carve the 3D block to remove the shape-irrelevant points and use CarveNet to learn from the diversity of the object shapes. Compared to the folding-based networks, CarveNet is better, especially in completing complex 3D shapes, for yielding completion results with richer details and fewer redundant points. CarveNet is equipped with MLP for refining the coarse completion results. In contrast to the current MLP-based architectures, we use different layers of MLPs, providing the semantic object information at various levels to produce denser results.

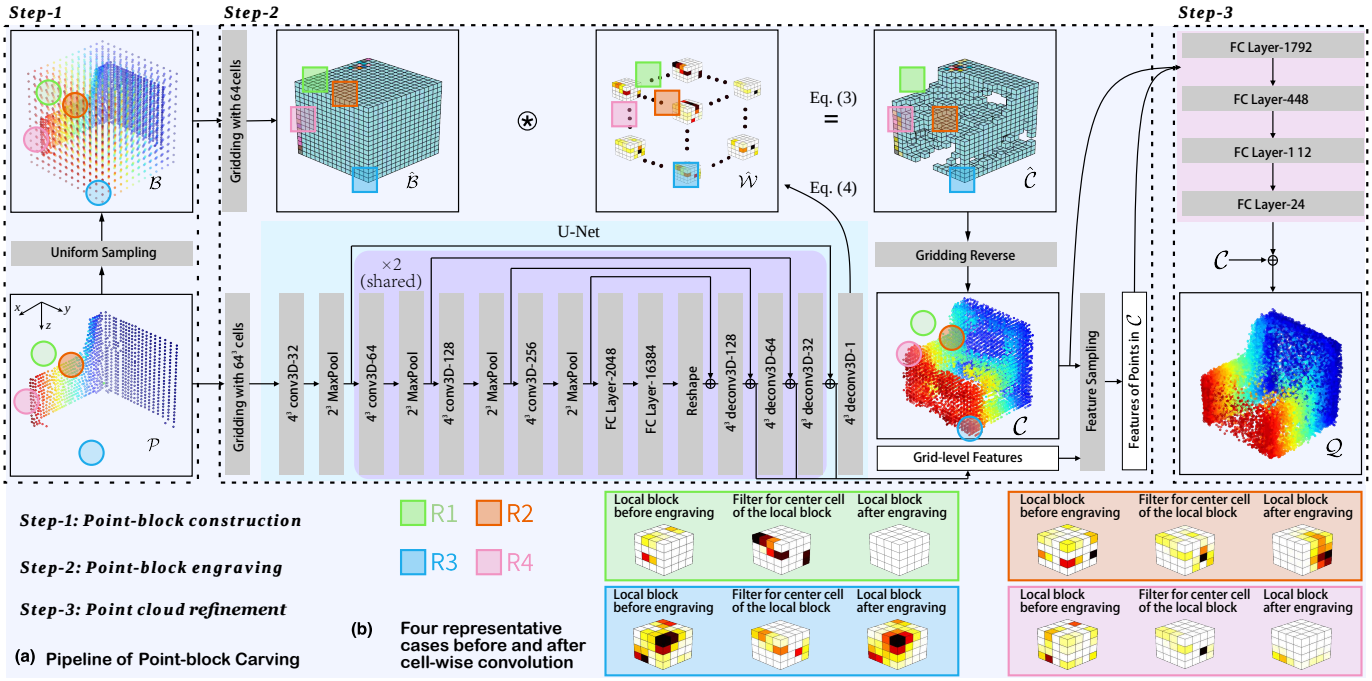


Fig. 2: (a) The point-block carving method with 3 steps, *i.e.*, point-block construction, point-block engraving via the CarveNet, and point cloud refinement. (b) Four examples of the cell-wise convolution on different cells of the point-block.

B. Point Cloud Representations

Point cloud representations play a vital role in the related tasks. A simple point cloud representation is achieved by directly voxelizing the 3D points. However, these voxelization results are sensitive to the quantization effects [30]–[35]. PointNet learns point-wise features directly from the raw data of point clouds. PointNet++ [36] group the points hierarchically to achieve flexible receptive fields in the 3D space. SO-Net [37] builds a self-organizing map to represent the spatial relationships of the unordered points. PointCNN [38] learns the $\mathcal{X}$-transformation of the input points to a latent and ordered representation. KPConv [39] uses the pseudo-grid convolution on the equally distributed spherical points. In the local feature aggregation method [40], position-pooling is used for extracting the local features efficiently.

In this work, we resort to the state-of-the-art representation, *i.e.*, 3D grid-based intermediate representation [9]. The point cloud can be quickly registered to the regular 3D grid without losing the critical object structures. CarveNet can process the 3D grid, and the output of CarveNet can be converted back to the point cloud efficiently.

III. Discussion on the Point-Block Carving

3D point cloud completion relies on the understanding of the object's shape. Intuitively, the object shape captures the object-level semantic information and the point-level spatial relationship. The conventional encoder-decoder methods (*e.g.*, AtlasNet [8] and GRNet [9]) take input as the partial point cloud. They map the points to the high-level features, which are merged to form the object-level semantic information for determining the locations of the missing points. The objects that belong to the same category likely have different structural details. It means that the spatial relationship between the 3D points is complex and hardly determined by the object-level information alone.

Our method for point-block carving takes advantage of both object-level information and point-level spatial relationships for completion. We construct the point block containing uniformly distributed 3D points and the input partial point cloud. We calculate the high-level features of the partial point cloud by the deep network. Rather than using the high-level features to predict the locations of the missing points, we use these features to learn the kernels. These kernels propagate the semantic information between the points and their neighbors in the point block, thus capturing the underlying spatial relationship between the points. With more complex spatial relationships, the redundant points can be removed from the point block by respecting the details of the object shape.

IV. Point-Block Carving for Shape Completion

A. Overview

We denote the partial point cloud as a set of 3D points, *i.e.*, $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, ..., |\mathcal{P}|\}$, where $|\mathcal{P}|$ is the number of 3D points. Based on $\mathcal{P}$, we use the point-block carving to compute a new set of points $\mathcal{Q} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, ..., |\mathcal{Q}|\}$ to represent the complete object. The set $\mathcal{Q}$ and the ground-truth set $\mathcal{G} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, ..., |\mathcal{G}|\}$ should be as similar as possible. We construct the point-block carving (PC) method to achieve the goal and schematically illustrate the whole process in Fig. 2, which consists of the point-block construction, the point-block engraving, and the point cloud refinement.

Step-1: Point-block construction. First, we compute a 3D block that contains the partial point cloud $\mathcal{P}$ (see ***Step-1***

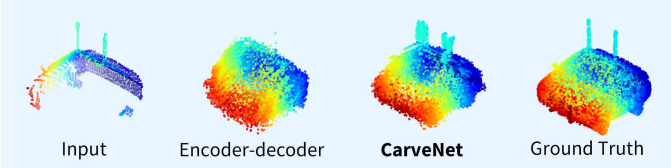


Fig. 3: Comparison between the completion results by using the state-of-the-art encoder-decoder method (GRNet) and our point-block engraving, where our method preserves better details of the object shape in the completion result.

of Figure 2). The range of the 3D block is denoted as $x \in [x_{\min}, x_{\max}]$, $y \in [y_{\min}, y_{\max}]$, and $z \in [z_{\min}, z_{\max}]$, where $x_{\min}/y_{\min}/z_{\min}$ (or $x_{\max}/y_{\max}/z_{\max}$) represents the possible minimization (or maximization) x/y/z-coordinate in the complete point cloud $\mathcal{Q}$. In practice, the range can be estimated through a 3D bounding box detection method [41]. Next, we sample T points uniformly from $x/y/z-$axis, respectively, achieving $N = T^3$ points distributed uniformly in the 3D block. The sampled 3D points are used along with the partial point cloud $\mathcal{P}$ to form the point-block $\mathcal{B} = \{\mathbf{p}_i \mid i = 1, ..., |\mathcal{P}| + N\}$.

Step-2: Point-block engraving. We use the point-block $\mathcal{B}$ to compute the coarse completion $\mathcal{C}$. This is done by using CarveNet to process the 3D grid-based intermediate representation of $\mathcal{B}$, where the irrelevant points are removed. CarveNet considers the visual and spatial properties of each 3D point, learning the point-wise graver with unique parameters to manipulate the 3D point. Thus, compared to the current encoder-decoder network like GRNet [9] that processes different 3D points by using the identical set of network parameters, CarveNet preserves better details during the carving process (see Fig. 3). The 3D grid-based intermediate representation enables the linear interpolation of the grid-level features for computing the deep features of the coarse points. It helps to refine the point cloud finally. In the process of the point-block engraving, we construct the 3D CNN for computing the graver parameters that are convoluted with the 3D grid-based intermediate representation. These details are more complicated than other steps. Thus, we expand their explanations in Sec. IV-B.

Step-3: Point cloud refinement. We refine the coarse point cloud $\mathcal{C}$ for achieving the dense point cloud $\mathcal{Q}$. Here, we use CarveNet to extract features for all points in the coarse point cloud. The feature of each 3D point is concatenated with the point's coordinate, which is passed to four fully connected layers (with 1792, 2448, 112, and 24 neurons in each layer, respectively). We use the fully connected layers to compute a set of point-wise offsets for updating the locations of the points in the coarse point cloud. The coarse point cloud and the updated counterpart are merged to form the dense point cloud $\mathcal{Q}$.

We further propose a novel data augmentation (*i.e.* SensorAug) to enhance the generalization of CarveNet across different 3D shapes and partial point clouds in Sec. IV-C and introduce the implementation details in Sec. IV-D.

B. CarveNet for Point-block Engraving

Formulation. Given the point-block $\mathcal{B}$, we construct CarveNet to compute the point-wise gravers to remove the irrelevant points. More formally, we perform the exclusive, point-wise convolution on each 3D point in $\mathcal{B}$ as:

$$\mathcal{C}(\mathbf{p}_i) = (\mathcal{B} \circledast \mathcal{W})(\mathbf{p}_i) = \sum_{\mathbf{p}_j \in \mathcal{N}(\mathbf{p}_i)} w_i(\mathbf{p}_j - \mathbf{p}_i) f_j, \quad (1)$$

where $\mathcal{B} \circledast \mathcal{W}$ denotes the point-wise convolution. $(\mathcal{B} \circledast \mathcal{W})(\mathbf{p}_i)$ is the convolutional output for the point $\mathbf{p}_i$. w_i defines the exclusive convolutional parameters for $\mathbf{p}_i$. f_j is the feature of $\mathbf{p}_j$, and it is set to 1.

The present/absent status of the point $\mathbf{p}_i$ is determined by the graver, whose parameters are represented by w_i in Eq. (1). To dynamically adjust the graver for each 3D point, we use a 3D CNN to learn the graver from the partial point cloud $\mathcal{P}$. The 3D CNN outputs a set of graver parameters $\mathcal{W} = \{w_1, ..., w_N\}$ for all of the N sampled points in $\mathcal{B}$ as:

$$\mathcal{W} = \text{3DCNN}(\mathcal{P}). \quad (2)$$

We use the 3DCNN along with the 3D grid-based intermediate representation (see Eqs. (3) and (4)) to implement the convolutional operations in Eq. (1) and (2). The alternatives, including KPConv [39] and PointConv [42], can be used instead of 3DCNN here. We compare different implementation strategies in the experiment in terms of completion accuracy.

3D grid-based intermediate representation. We choose the 3D grid-based intermediate representation [9]. This representation is based on the neighboring interpolation. It effectively avoids any quantization, thus preserving the structural information of the object. Its advantages have been evidenced in the tasks [9], [13], [18], [36], [39] where the point cloud needs to be processed efficiently.

In the intermediate representation, the *gridding layer* uses the differentiable interpolation to map a 3D point cloud to the gridding representation. Conversely, the *gridding reverse* maps a gridding to the 3D point cloud. The *feature sampling* extracts the feature of the 3D point based on the grid-level features. We construct the 3D grid-based intermediate representation for the point cloud to regularize the unordered points while explicitly preserving the structural and context information of these points. In Fig. 2, we illustrate the construction of the intermediate representation.

We use two gridding layers, which compute the gridding results of the point-block $\mathcal{B}$ and the partial $\mathcal{P}$, respectively. The gridding results are denoted as $\hat{\mathcal{B}} \in \mathbb{R}^{H \times W \times M}$ and $\hat{\mathcal{P}} \in \mathbb{R}^{H \times W \times M}$. H, W, and M denote the resolution of the grid. Here, we regard the grid as a set of $H \times W \times M$ cells.

Next, we process the $\mathbf{c}_i$ in $\hat{\mathcal{B}}$ as:

$$\hat{\mathcal{C}}(\mathbf{c}_i) = (\hat{\mathcal{B}} \circledast \hat{\mathcal{W}})(\mathbf{c}_i) = \sum_{\mathbf{c}_j \in \mathcal{N}(\mathbf{c}_i)} \hat{w}_i(\mathbf{c}_j - \mathbf{c}_i) \hat{\mathcal{B}}(\mathbf{c}_j), \quad (3)$$

where $\hat{\mathcal{W}} \in \mathbb{R}^{H \times W \times M \times K^3}$ is a set of 3D cell-wise kernels. K^3 indicates the size of a 3D kernel. We use a pre-trained 3D UNet to process the grid $\hat{\mathcal{P}}$, achieving the $\hat{\mathcal{W}}$ as:

$$\hat{\mathcal{W}} = \text{UNet}(\hat{\mathcal{P}}). \quad (4)$$

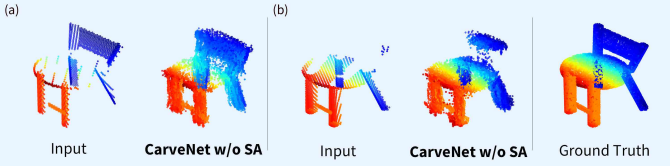


Fig. 4: (a) and (b) are different partial point clouds for the same object. The completion results of (a) and (b) with our method without the SensorAug (*i.e.*, CarveNet w/o SA) are inconsistent.

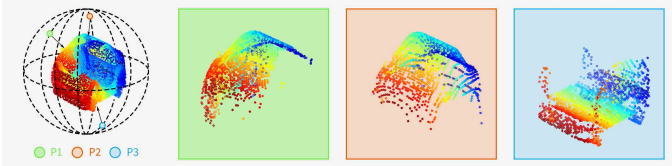


Fig. 5: The leftmost subfigure illustrates the complete point cloud and the sensors at different locations. P1, P2, and P3 are different generated partial point clouds containing the visible points from different views.

We illustrate 3D UNet in Fig. 2, whose parameters are updated during the training of CarveNet. As shown in step-2 of the updated Fig. 2, we improve the performance of CarveNet by iteratively refining the convolution features of the 3D grid-based intermediate representation in two rounds. This is done by feeding the convolution features, computed by the penultimate deconvolution layer (43 deconv3D-32), back to the second convolution layer (43 conv3D-64). In these two rounds, the convolution and deconvolution layers highlighted by the purple box share the same parameters without increasing the model complexity. The above refinement is performed on the convolution features rather than directly on the point cloud, thus saving the inference time.

We represent the gridding representation of the coarse point cloud as $\hat{\mathcal{C}} = \hat{\mathcal{B}} \circledast \hat{\mathcal{W}} \in \mathbb{R}^{H \times W \times M \times 3}$. In Fig. 2(b), we provide four examples of the cell-wise convolution on different cells in $\hat{\mathcal{B}}$. We compute the gridding reverse [9] of $\hat{\mathcal{C}}$, achieving the final coarse point cloud $\mathcal{C}$. Note that the spatial resolution $H \times W \times M$ of the coarse point cloud $\hat{\mathcal{C}}$ is the same as the point block $\hat{\mathcal{B}}$. Thus, the gridding reverse operation can use the values in the coarse point cloud $\hat{\mathcal{C}}$ to weight the eight vertices' coordinates of the same index of the cell in the point block $\hat{\mathcal{B}}$, yielding a new 3D point in the point cloud $\mathcal{C}$. This operation yields $H \times W \times M$ new points for $\mathcal{C}$.

We extract features from the UNet($\hat{\mathcal{P}}$). The features are propagated to the points in $\mathcal{C}$ by resampling w.r.t. the coordinate relationship between $\mathcal{C}$ and $\hat{\mathcal{P}}$ (see the "Feature Sampling" in Fig. 2 (a)). These features are used to refine the coarse point cloud by MLP.

C. SensorAug for Effective Training

We propose SensorAug for augmenting the training data. Rather than randomly removing 3D points from the complete point clouds, we move a virtual sensor in the 3D space lets the visible points be partial.

SensorAug works with a similarity loss function for supervising the training of CarveNet and MLP-based refinement. In Fig. 5, we illustrate SensorAug for producing the partial point clouds. We denote a complete point cloud as $\mathcal{G}$. We construct a restricted 3D space centering at $\mathcal{G}$, where the coordinates of all of the points in $\mathcal{G}$ are normalized to the range [-0.5,0.5]. To observe the object, we randomly put a virtual LiDAR sensor, whose distance to the center of $\mathcal{G}$ is 1. The visible points within the viewing frustum are used for constructing the partial point cloud. We repeat SensorAug to produce a set of partial point clouds (*e.g.*, $\{P1, P2, P3\}$ in Fig. 5) for each complete point cloud. Note that the production of the partial point clouds in the ShapeNet dataset [43] is also based on the virtual LiDAR, which is randomly placed on the surface of the 3D ball like Fig. 5. We use the virtual LiDAR with the exact spatial resolution (120×160), focal length (100), and shooting distance (1) to implement SensorAug, thus yielding partial point clouds with spatial distributions similar to those in the ShapeNet dataset.

For each complete point cloud $\mathcal{G}$, we use SensorAug to generate T partial point clouds $\{\mathcal{P}_1,...,\mathcal{P}_T\}$. These generated point clouds are used, along with $\mathcal{P}$ given in the training set, by our method to compute the coarse completion results $\{\mathcal{C}, \mathcal{C}_1, ..., \mathcal{C}_T\}$ and the refined results $\{\mathcal{Q}, \mathcal{Q}_1, ..., \mathcal{Q}_T\}$. We define the loss function below to optimize CarveNet as:

$$\mathcal{L} = \mathcal{L}_{\text{comp}} + \alpha \mathcal{L}_{\text{sim}}. \tag{5}$$

We minimize the loss function $\mathcal{L}$ to optimize CarveNet.

The loss $\mathcal{L}_{\text{comp}}$ is formulated as:

$$\mathcal{L}_{\text{comp}} = \text{CD}(\mathcal{C}, \mathcal{G}) + \text{CD}(\mathcal{Q}, \mathcal{G}), \tag{6}$$

$$\begin{aligned}\text{CD}(\mathcal{X}_1, \mathcal{X}_2) = &\frac{1}{|\mathcal{X}_1|} \sum_{\mathbf{p}_i \in \mathcal{X}_1} \min(\{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \mid \mathbf{p}_j \in \mathcal{X}_2\}) \\ &+ \frac{1}{|\mathcal{X}_2|} \sum_{\mathbf{p}_i \in \mathcal{X}_2} \min(\{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \mid \mathbf{p}_j \in \mathcal{X}_1\}).\end{aligned} \tag{7}$$

In Eq. (7), CD denotes the Chamfer distance [44] between a pair of point clouds $(\mathcal{X}_1, \mathcal{X}_2)$. We use the loss $\mathcal{L}_{\text{comp}}$ to penalize the difference between the coarse/refined completion result and the ground-truth point cloud $\mathcal{G}$.

In Eq. (5), the loss $\mathcal{L}_{\text{sim}}$ is defined as:

$$\mathcal{L}_{\text{sim}} = \sum_{i=1}^{T} \text{CD}(\mathcal{C}_i, \mathcal{C}) + \text{CD}(\mathcal{Q}_i, \mathcal{Q}). \tag{8}$$

The loss $\mathcal{L}_{\text{sim}}$ penalizes the difference between the completion results, which are computed based on different partial point clouds of the same object. It helps to boost the spatial consistency between the completions conditioned on different incomplete counterparts.

Qualitative discussion. In Fig. 2(b), we provide four examples of the cell-wise convolution on different cells in the grid $\hat{\mathcal{B}}$. We train the CarveNet and MLPs on the ShapeNet dataset [43]. We compare the partial point cloud $\mathcal{P}$ and the coarse grid $\hat{\mathcal{C}}$, where the regions of the 3D space are indicated by the circle and rectangle, respectively. CarveNet reasonably completes the

missing grids (see the blue circle and rectangle), removes the grids beyond the object (see the yellow circle and rectangle), and preserves the correct details of the partial point cloud (see the red and pink circles and rectangles).

Yet, there is still much room for improving the robustness of our method. In Fig. 4, given the different patterns of the partial point clouds of the identical object, our method produces inconsistent completion results. One of the significant reasons that led to the conflicting results is the insufficient learning from the relationship between the partial and the complete point clouds. This motivates us to propose SensorAug, which produces an arbitrary number of partial point clouds for the same object. These partial point clouds are provided with diverse patterns. With SensorAug, we can use more pairs of partial and complete point clouds to augment the training data and improve the completion power of CarveNet and MLPs.

D. Implementation Details

We use PyTorch 1.6 to implement our method. We use Adam solver for network optimization. The mini-batch size is 24/32 for CarveNet with/without SensorAug. The spatial resolution $H \times W \times M$ of the 3D grid is empirically set to 32^3, 64^3, and 80^3 in our experiment. We use four NVIDIA 2080Ti GPUs for training and testing. The initial learning rate is set to $1e-4$. It is decayed linearly by half for every 40 epochs. We set $\alpha = 0.5$ and $T = 2$. In the following experimental section, we denote our whole method as CarveNet since it is our main contribution.

V. Experiments

A. Experimental Settings

Dataset. We use the ShapeNet [43], ShapeNet-55/34 [19] and KITTI [45] datasets to evaluate the completion methods. The ShapeNet dataset consists of 30,974 3D models from 8 categories. There are 28,974/800/1,200 models in the training/validation/test set. Each model is associated with a pair of complete and partial point clouds. The complete point cloud contains 16,384 points uniformly sampled on the object surface, while the partial one contains 2,048 points. The ShapeNet-55/34 [19] are derived from the ShapeNet [43]. They include more categories to capture the diversity of 3D objects in the real world. In ShapeNet-55 with 55 categories, there are 41,952 3D models for training and 10,518 models for testing. ShapeNet-34 contains 46,765 models with 34 categories for training. It provides 3,400/2,305 models for testing on 34 seen/21 unseen categories. Each model contains 8192 points. The point clouds in KITTI [45] are captured by the LiDAR sensors. There are 2,400 partial point clouds of cars taken from 426 different timestamps. Each cloud contains 2,048 points. The ground-truth complete clouds are unavailable in KITTI.

Baselines. We compare our method with FoldingNet [11], PCN [12], AltasNet [8], TopNet [13], GRNet [9], PoinTr [19], PMP-Net [26], SnowflakeNet [27], VRCNet [28], and SeedFormer [29]. All of these methods are compared fairly with the same experimental settings. Here, we set the number of patches to 16 in AtlasNet [8]. The levels and leaves were set to 6 and 8 in TopNet [13] for generating 16,384 points.

Evaluation metrics. We use Chamfer distance, F-score and Consistency as the metrics for the evaluation. We use Chamfer distance (see Eq. (7)) to measure the similarity between the completion result $\mathcal{Q}$ and the complete point cloud $\mathcal{G}$. Besides, we follow [46] to measure the completion performance with F-score. Because the ground-truth point clouds are unavailable in the KITTI dataset, we use the consistency defined in [9], [12] to measure the quality of the completion result. We denote the completion result of i^{th} car in j^{th} frame as $\mathcal{Q}_j^i$. Suppose there are N frames for the i^{th} car. The completion results of the adjacent frames are used for computing the Chamfer distances, which are averaged to achieve the consistency as:

$$\text{Consistency} = \frac{1}{N-1}\sum_{j=1}^{N-1} \text{CD}(\mathcal{Q}_j^i, \mathcal{Q}_{j+1}^i) \tag{9}$$

A lower Chamfer distance/consistency or a higher F-score means a better result.

B. Ablation Study

We use the ShapeNet dataset for evaluating the core components of CarveNet, *i.e.*, the point-block construction, the 3D point cloud representation and SensorAug. We disable SensorAug for examining the improvements, which are solely contributed by the point-block construction and the 3D point cloud representation, respectively.

Evaluation of the point-block construction methods. In Table I, we report the average Chamfer distances for all the completion results on the ShapeNet test set. In row *w/o Construction*, we show the carving results on the point block, which only contains the partial point cloud. Because many objects have symmetric shapes, a naive solution for the point-block construction is simply merging the partial point clouds and its mirror counterpart as an entire point-block (see the results in the row *Symmetric*). For our method (*i.e.*, uniform sampling), we compare different number of points (*i.e.*, $11^3, 13^3$ and 16^3 points).

In the column *Coarse CD*, we compare different point-block construction methods in terms of the qualities of the coarse completion results. Our strategy of uniform point sampling outperforms other alternatives for point-block construction. The completion results of the compared methods are visualized

TABLE I: Comparison of using different initial points to construct the point-block. Coarse/Dense CD represents the average Chamfer distance (multiplied by 10^3) between the coarse/refined results and ground-truth point clouds.

Method	Coarse CD	Dense CD
w/o Construction	6.874	0.3376
Symmetric (2048 pts)	1.637	0.2749
Uniform (11^3 pts)	**1.443**	0.2074
Uniform (13^3 pts)	1.463	**0.2018**
Uniform (16^3 pts)	1.746	0.2021
Ground Truth Pts	1.239	0.1864

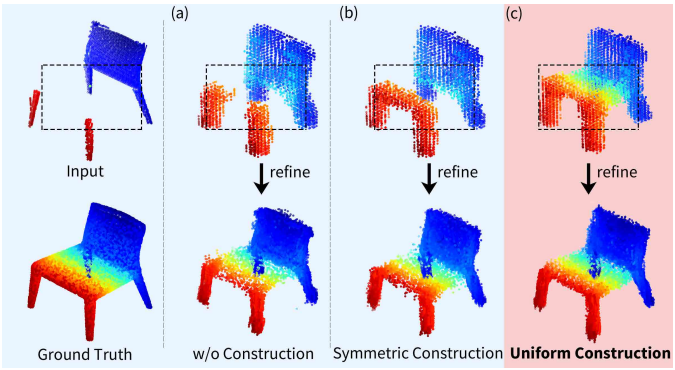


Fig. 6: The completion results achieved by different point-block construction methods. The first column shows the input and the ground truth. The first and second rows of other columns are coarse completion results (*i.e.*, after point-block engraving) and dense completion results (*i.e.*, after refinement) of each method, respectively.

in Fig. 6. By comparing Fig. 6 (a–c), we find that the point-block construction helps the point-block engraving better recover the 3D points lost in the partial point cloud. We also evaluate our point cloud refinement, which is used along with different point-block construction methods. The refinement produces the dense completion results, whose qualities are reported in the column *Dense CD*. Our refinement improves the completion qualities achieved by different methods.

In Table I, we use different numbers of the sampled points to construct the point block and compare the completion qualities. Though more sampled points (*e.g.*, 16^3) may provide more chances for recovering the object details, they significantly increase the difficulty of carving and lead to worse completion. Here, we investigate an extreme strategy where the ground-truth cloud is given for constructing the point-block. Compared to other methods, using uniform points produces competitive results. Because the prior object information is unnecessary, the uniform point sampling for making the point block can be generalized to completing objects in different categories.

TABLE II: Ablation study on the 3D representation and SensorAug. Chamfer distance (CD) is multiplied by 10^3.

Method	CD
KPConv [39]	0.5025
PointConv [42]	0.4839
3D Grids	**0.2018**

(a) Results by using different representations.

Method	ShapeNet	ShapeNet-55	ShapeNet-34	
			34 seen	21 unseen
w/o SensorAug	0.2018	0.88	0.79	1.27
Randomly Drop	0.2572	0.88	0.78	1.27
SensorAug	**0.1947**	**0.86**	**0.75**	**1.25**

(b) Results with/without SensorAug on ShapeNet and ShapeNet-55/34.

TABLE III: Performance on the ShapeNet test set with different sizes of inputs.

Grid Size	CD ($\times 10^3$)	Running time (ms)	# Parameters (M)
32^3	0.4206	30	6.9
64^3	0.2018	58	76.8
80^3	0.2083	154	178.7

Evaluation of the 3D point cloud representations. In our implementation of CarveNet, We use the 3D grid-based intermediate representation for the point-block engraving. The point-wise convolution in Eq. (1) is a general formulation for computing the convolution features for 3D points. As introduced in the **3D grid-based intermediate representation** of Sec. IV-B, we construct the 3D grid-based intermediate representation, which can be regarded as the regular grid. Every eight adjacent vertices form a normal cell, which enables the cell-wise convolution in Eq. (3). The cell-wise convolution is a specific form of the point-wise convolution in Eq. (1).

In Table II(a), we compare the 3D grid-based and different point-based representations and their performances on the ShapeNet test set. The point-based representation disables the conventional operation of the 3D convolution, which means the 3D grid-based intermediate representation is not constructed, and the cell-wise convolution is inapplicable. We resort to KPConv [39] and PointConv [42], which belong to the family of the point-based convolution to compute the 3D points' features, for the comparison with the 3D point-based representations. The 3D grid-based method produces better results than the point-based representations. In Table III, we compare the 3D grid-based method with different grid sizes. The running time is measured as the forwarding time with a batch size of 1. Considering the completion accuracy and the computation, we select the grid size 64^3 in our implementation.

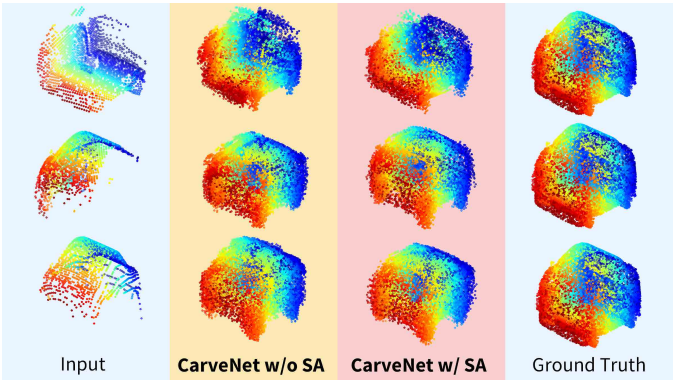


Fig. 7: Completion results without/with SensorAug for training. Here, the partial point clouds capture the same object.

Evaluation of SensorAug. We examine the effect of removing SensorAug from the network training and report the completion result in Table II(b). We successfully degrade the completion results without SensorAug (see the row *w/o SensorAug*). We also compare SensorAug with the random dropout of points (see the row *Randomly Drop*) for augmenting

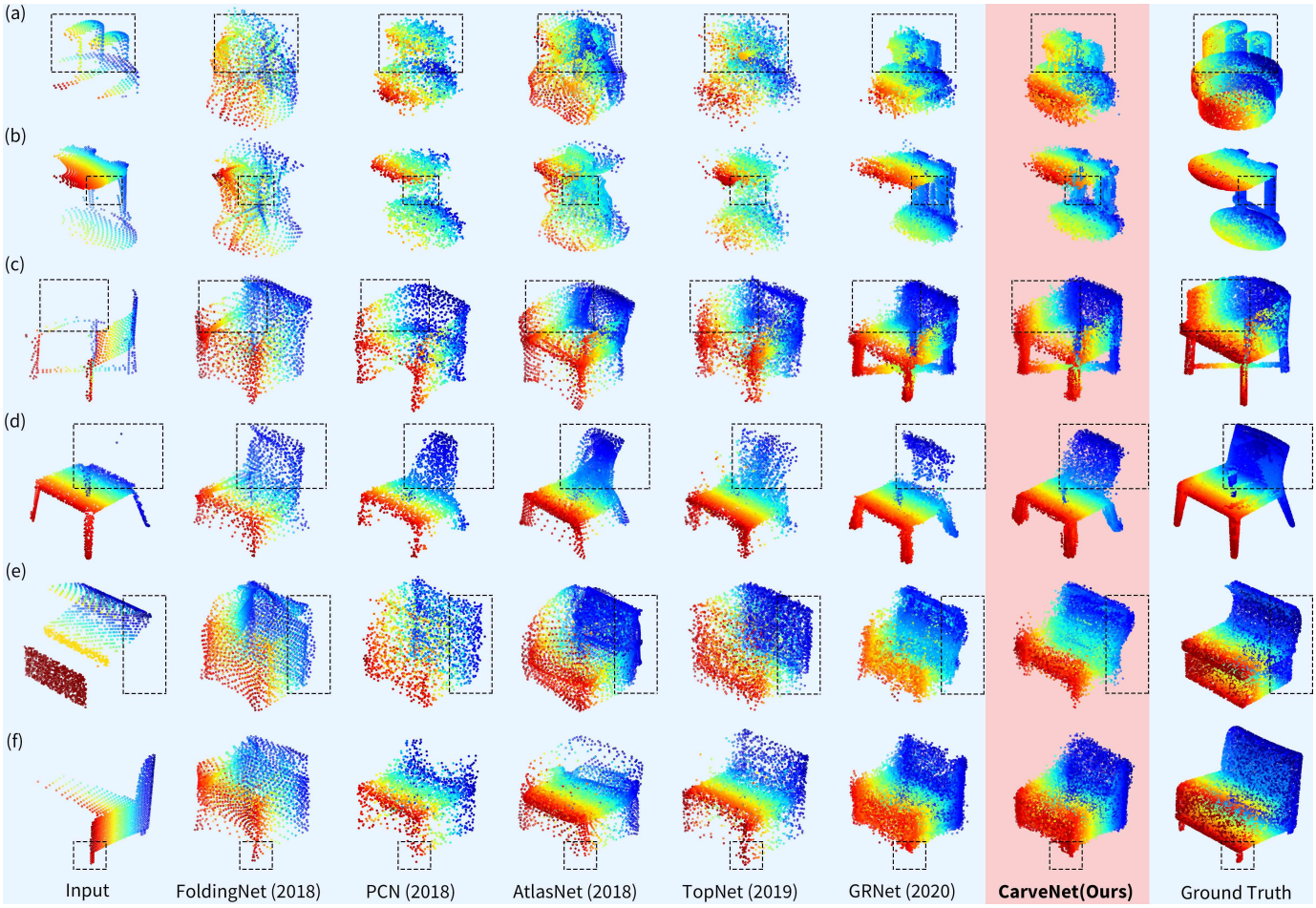


Fig. 8: Visualization of the point cloud completion results of different methods (i.e., FoldingNet, PCN, AtlasNet, TopNet, GRNet, and CarveNet) on the ShapeNet dataset.

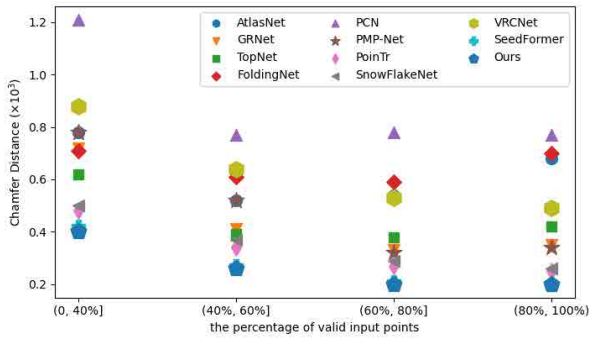


Fig. 9: Sensitivity to the percentage of valid points in the partial point clouds.

the training data. The random dropout involves inconsistent object shapes that mislead the network training, thus yielding lower completion accuracy than the method without augmentation and SensorAug. We visualize the completion results without/with SensorAug in Fig. 7. Given different partial point clouds, the network, trained with SensorAug, produces consistent completion results.

C. Comparison with State-of-the-Art Methods

Results on ShapeNet. In Table IV, we compare CarveNet with other methods. We evaluate each method regarding the average Chamfer distances on eight object categories. The distances on eight types are averaged again to measure the method's overall performance. Our method surpasses other methods in all 8 categories. We also provide examples of the completion results in Fig. 8 and Fig. 11, where the complex shapes like high-curvature and hollowed-out object parts are correctly recovered.

Results on ShapeNet-55/34. In Table V and Table VI, we compare CarveNet with other methods on ShapeNet-55/34. We evaluate each method regarding the Chamfer distances and F-score with three difficulty degrees. On ShapeNet-55, our method surpasses other methods in most categories and all difficulty degrees. On ShapeNet-34, our method surpasses other methods in 2 of 3 difficulty degrees in 34 seen categories and all difficulty degrees in 21 unseen categories.

Results on KITTI. Because the ground-truth point clouds are unavailable, we train CarveNet on the car models in the ShapeNet training set. The completion results achieved by CarveNet are evaluated on the KITTI test set. In Table VII, we report the consistencies achieved by different methods. Again,

TABLE IV: Results on the ShapeNet. Here, we compute Chamfer distance based on 16,384 points (multiplied by 10^3). The best result of each column is bold-face.

Method	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Average
FoldingNet [11]	0.4127	0.6837	0.4096	0.8226	0.8475	0.7212	0.6478	0.4778	0.6279
PCN [12]	0.3431	1.0998	0.5513	1.0955	1.1840	1.2163	1.0459	0.6917	0.9035
AtlasNet [8]	0.2503	0.6860	0.3832	0.6912	0.8178	0.8135	0.5971	0.5169	0.5945
TopNet [13]	0.1711	0.5319	0.3565	0.5947	0.5518	0.6735	0.3935	0.3710	0.4555
GRNet [9]	0.2840	0.5966	0.3347	0.5353	0.4486	0.7456	0.5066	0.3011	0.4691
PoinTr [19]	0.0820	0.3513	0.2329	0.2656	0.2104	0.3613	0.2118	0.1734	0.2361
PMP-Net [26]	0.1205	0.4203	0.2867	0.3552	0.2158	0.4231	0.2920	0.1897	0.2879
SnowflakeNet [27]	0.0934	0.3593	0.2330	0.2918	0.2306	0.3963	0.2191	0.1704	0.2492
VRCNet [28]	0.2347	0.6638	0.4366	0.5737	0.451	0.7338	0.4294	0.3586	0.4852
SeedFormer [29]	0.0663	0.3174	0.2182	0.2034	0.1476	0.3478	0.1551	0.1398	0.1995
CarveNet (Ours)	**0.0654**	**0.3127**	**0.2036**	**0.2011**	**0.1443**	**0.3383**	**0.1529**	**0.1392**	**0.1947**

TABLE V: Results on the ShapeNet-55. Here, we compute Chamfer distance based on 8192 points (multiplied by 10^3).

Method	Table	Chair	Airplane	Car	Sofa	Birdhouse	Bag	Remote	Keyboard	Rocket	CD-S	CD-M	CD-H	CD-Avg	F1
FoldingNet [11]	2.53	2.81	1.43	1.98	2.48	4.71	2.79	1.44	1.24	1.48	2.67	2.66	4.05	3.12	0.082
PCN [12]	2.13	2.29	1.02	1.85	2.06	4.50	2.86	1.33	0.89	1.32	1.94	1.96	4.08	2.66	0.133
TopNet [13]	2.21	2.53	1.14	2.18	2.36	4.83	2.93	1.49	0.95	1.32	2.26	2.16	4.3	2.91	0.126
GRNet [9]	1.63	1.88	1.02	1.64	1.72	2.97	2.06	1.09	0.89	1.03	1.35	1.71	2.85	1.97	0.238
PoinTr [19]	0.81	0.95	0.44	0.91	0.79	1.86	0.93	0.53	**0.38**	0.57	0.58	0.88	1.79	1.09	0.464
PMP-Net [26]	1.30	1.41	0.61	1.39	1.31	2.49	1.67	0.74	0.51	0.65	0.72	1.39	3.38	1.83	0.453
SnowflakeNet [27]	0.98	1.12	0.54	0.98	1.02	1.93	1.08	0.57	0.48	0.61	0.70	1.06	1.96	1.24	0.398
VRCNet [28]	1.25	1.44	0.57	1.13	1.20	2.36	1.39	0.71	0.50	0.65	0.86	1.53	2.98	1.79	0.463
SeedFormer [29]	0.72	0.81	0.40	**0.89**	0.71	1.63	0.90	0.45	0.41	0.31	0.50	0.77	1.49	0.92	0.472
CarveNet (Ours)	**0.71**	**0.78**	**0.36**	0.91	**0.68**	**1.60**	**0.87**	**0.41**	0.45	**0.29**	**0.48**	**0.74**	**1.43**	**0.88**	**0.486**

TABLE VI: Results on the ShapeNet-34. Here, we compute Chamfer distance based on 8192 points (multiplied by 10^3).

	34 seen categories					21 unseen categories				
Method	CD-S	CD-M	CD-H	CD-Avg	F1	CD-S	CD-M	CD-H	CD-Avg	F1
FoldingNet [11]	1.86	1.81	3.38	2.35	0.139	2.76	2.74	5.36	3.62	0.095
PCN [12]	1.87	1.81	2.97	2.22	0.154	3.17	3.08	5.29	3.85	0.101
TopNet [13]	1.77	1.61	3.54	2.31	0.171	2.62	2.43	5.44	3.50	0.121
GRNet [9]	1.26	1.39	2.57	1.74	0.251	1.85	2.25	4.87	2.99	0.216
PoinTr [19]	0.76	1.05	1.88	1.23	0.421	1.04	1.67	3.44	2.05	0.384
PMP-Net [26]	0.66	1.22	2.83	1.57	0.442	0.85	1.73	4.04	2.21	0.407
SnowflakeNet [27]	0.51	0.71	**1.21**	0.81	0.414	0.76	1.23	2.55	1.51	0.372
VRCNet [28]	0.69	1.12	2.14	1.32	0.433	1.13	1.96	4.20	2.43	0.397
SeedFormer [29]	0.48	0.70	1.30	0.83	0.452	0.61	1.07	2.35	1.34	0.402
CarveNet (Ours)	**0.46**	**0.67**	1.24	**0.79**	**0.465**	**0.57**	**1.00**	**2.23**	**1.27**	**0.413**

TABLE VII: Consistency on KITTI.

Method	Consistency ($\times 10^3$)
FoldingNet [11]	0.2854
PCN [12]	0.3578
AtlasNet [8]	0.3580
TopNet [13]	0.2179
GRNet [9]	0.2036
PoinTr [19]	0.1548
PMP-Net [26]	0.2074
SnowflakeNet [27]	0.1619
VRCNet [28]	0.1733
SeedFormer [29]	0.1496
CarveNet (Ours)	**0.1437**

CarveNet outperforms other methods. It also demonstrates that knowledge learned by CarveNet can be generalized to different datasets. We provide examples of the completion results on the KITTI dataset in Fig. 10.

Sensitivity to the partial point clouds. In the default setting, each partial point cloud contains 2,048 points. Generally, the fewer valid input points, the more difficult its completion can be. We conduct the below experiment to evaluate the robustness of different methods by comparing the average CD under different ratios of valid points. Here, we place each partial point cloud into $64 \times 64 \times 64$ grids, where we define the percentage of valid input points as the percentage of non-zero grids. We divide the partial point clouds with different percentages (i.e., $0 \sim 40\%$, $40 \sim 60\%$, $60 \sim 80\%$, and $80 \sim 100\%$) of valid points into the ranges plotted in Fig. 9. Even when the percentage of valid points is tiny (e.g., $0 \sim 40\%$), CarveNet produces a smaller average CD than

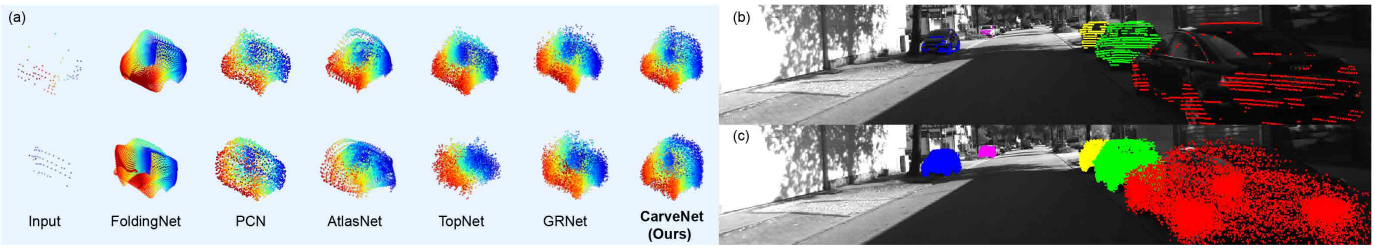


Fig. 10: Visualization of completion results on KITTI. (a) Completion results through different methods. Completion results of our method projecting on the image with (b) for partial point clouds and (c) for the completed results.

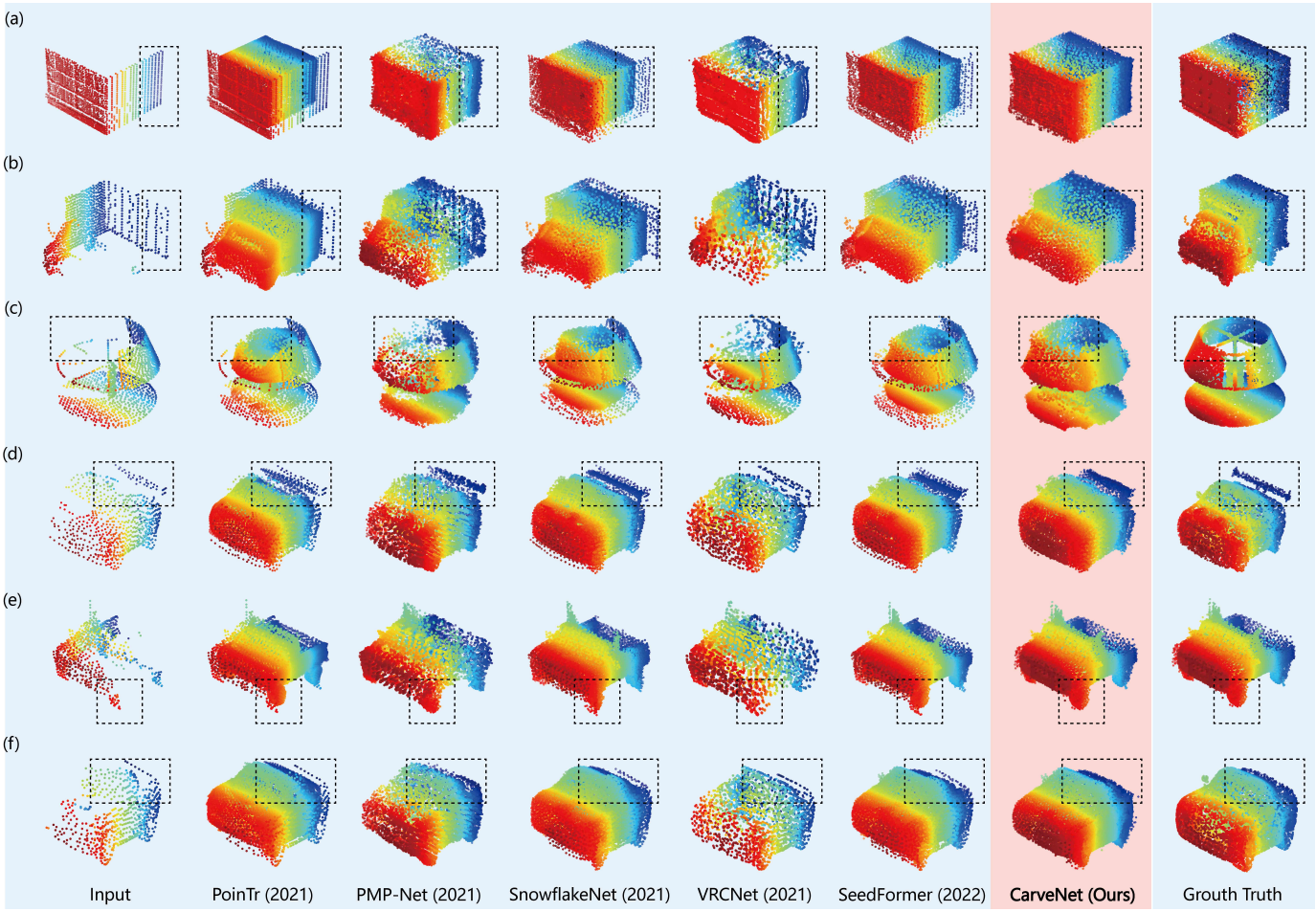


Fig. 11: Visualization of the point cloud completion results of different methods (i.e., PoinTr, PMP-Net, SnowflakeNet, VRCNet, SeedFormer, and CarveNet) on the ShapeNet dataset.

other methods, thus demonstrating its robustness.

VI. Conclusion

The complexity of 3D point cloud completion stems from the diversity of the 3D object shapes. In this paper, we have proposed an effective operation, the point-block engraving, for the completion task. We use point-block engraving to manipulate the 3D grid-based representation of the object, which is shape-agnostic, for removing the redundant points and recovering the essential details of the object. Moreover, we propose SensorAug to augment the training data, allowing the completion network to learn from more diverse object shapes. The evaluation of the public completion benchmarks demonstrates the effectiveness of our approach.

Acknowledgments

This research is supported by the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), Career Development Fund (CDF) of Agency for Science, Technology and Research (A*STAR) (No.: C233312028), and National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative (No. DTC-RGC-04). This work is also supported in

part by Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada; as well as JST-Mirai Program Grant No.JPMJMI20B8, JSPS KAKENHI Grant No.JP21H04877, No.JP23H03372, No.JP24K02920, and also with the support from TIER IV, Inc. and the Autoware Foundation.

REFERENCES

[1] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop: An explainable machine learning method for point cloud classification," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1744–1755, 2020.

[2] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "Point cloud rendering after coding: Impacts on subjective and objective quality," *IEEE Transactions on Multimedia*, vol. 23, pp. 4049–4064, 2020.

[3] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[4] M. Angelina Uy and G. Hee Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.

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

[6] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[7] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4224–4231, 2018.

[8] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224.

[9] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," *arXiv preprint arXiv:2006.03761*, 2020.

[10] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4338–4364, 2021.

[11] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

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

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

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

[15] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1939–1948.

[16] X. Wang, D. Lin, and L. Wan, "Ffnet: Frequency fusion network for semantic scene completion," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 3, 2022, pp. 2550–2557.

[17] H. Dong, E. Ma, L. Wang, M. Wang, W. Xie, Q. Guo, P. Li, L. Liang, K. Yang, and D. Lin, "Cvsformer: Cross-view synthesis transformer for semantic scene completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8874–8883.

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

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

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

[21] T. Huang, H. Zou, J. Cui, J. Zhang, X. Yang, L. Li, and Y. Liu, "Adaptive recurrent forward network for dense point cloud completion," *IEEE Transactions on Multimedia*, pp. 1–13, 2022.

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

[23] ——, "Point cloud completion by learning shape priors," *arXiv preprint arXiv:2008.00394*, 2020.

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

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[26] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Pmp-net: Point cloud completion by learning multi-step point moving paths," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7443–7452.

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

[28] L. Pan, X. Chen, Z. Cai, J. Zhang, H. Zhao, S. Yi, and Z. Liu, "Variational relational point completion network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8524–8533.

[29] H. Zhou, Y. Cao, W. Chu, J. Zhu, T. Lu, Y. Tai, and C. Wang, "Seedformer: Patch seeds based point cloud completion with upsample transformer," in *European conference on computer vision*. Springer, 2022, pp. 416–432.

[30] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[31] 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 *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 85–93.

[32] A. Sharma, O. Grau, and M. Fritz, "Vconv-dae: Deep volumetric shape learning without object labels," in *European Conference on Computer Vision*. Springer, 2016, pp. 236–250.

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

[34] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9204–9214.

[35] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive o-cnn: A patch-based deep representation of 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

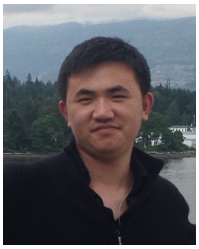
[37] J. Li, B. M. Chen, and G. Hee Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9397–9406.

[38] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in neural information processing systems*, 2018, pp. 820–830.

[39] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6411–6420.

[40] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong, "A closer look at local aggregation operators in point cloud analysis," *arXiv preprint arXiv:2007.01294*, 2020.

[41] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," *computer vision and pattern recognition*, 2018.
[42] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.
[43] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.
[44] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.
[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.
[46] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3405–3414.



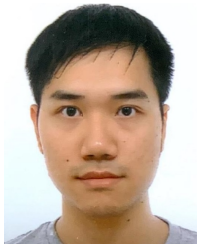
Qing Guo received his Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He is currently a senior research scientist and principal investigator (PI) at the Center for Frontier AI Research (CFAR), A*STAR in Singapore. He is also an adjunct assistant professor at the National University of Singapore (NUS), and senior PC member of AAAI. Before that, he was a Wallenberg-NTU Presidential Postdoctoral Fellow with the Nanyang Technological University, Singapore. His research interests include computer vision, AI security, and image processing. He is a member of IEEE.

Zhijie Wang is currently a Ph.D. student in Software Engineering at the University of Alberta, AB, Canada. Previously, he earned an M.E. degree in Electrical and Computer Engineering from the University of Waterloo, Ontario, Canada, and a B.E. degree in Telecommunication Engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. His research interests include quality assurance of AI systems and human-computer interaction in AI software development.

Lubo Wang is a master student with the College of Intelligence and Computing, Tianjin University, China. He received the bachelor degree in software engineering from Tianjin University in 2022. His research interest is scene analysis and understanding.

Haotian Dong is a master student with the College of Intelligence and Computing, Tianjin University, China. She received the bachelor degree in computer science and technology from Hebei University of Technology in 2021. Her research interest is scene generation.

Felix Juefei-Xu (Member, IEEE) received the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Prior to that, he received the M.S. degree in Electrical and Computer Engineering and the M.S. degree in Machine Learning from CMU, and the B.S. degree in Electronic Engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China. Currently, he is a Research Scientist with GenAI at Meta, based in New York City, where he works on robust perception and efficient learning problems in the domain of generative AI. He is also affiliated with New York University as an Adjunct Professor. Previously, he was a Research Scientist with Alibaba Group, based in Sunnyvale, CA. He was the recipient of multiple best or distinguished paper awards, including IJCB 2011, BTAS 2015 and 2016, ASE 2018, and ACCV 2018.



Di Lin is an associate professor with the College of Intelligence and Computing, Tianjin University, China. He received the bachelor degree in software engineering from Sun Yat-sen University in 2012, and the PhD degree from the Chinese University of Hong Kong in 2016. His research interests are computer vision and machine learning.

Lei Ma is currently an associate professor with The University of Tokyo, Japan and University of Alberta, Canada. He was honorably selected as a Canada CIFAR AI Chair and Fellow at Alberta Machine Intelligence Institute (Amii). Previously, he received the B.E. degree from Shanghai Jiao Tong University, China, and the M.E. and Ph.D. degrees from The University of Tokyo, Japan. His recent research centers around the interdisciplinary fields of software engineering (SE) and trustworthy artificial intelligence with a special focus on the quality, reliability, safety and security aspects of AI Systems.

Wei Feng received the PhD degree in computer science from City University of Hong Kong in 2008. From 2008 to 2010, he was a research fellow at the Chinese University of Hong Kong and City University of Hong Kong. He is now a full Professor at the School of Computer Science and Technology, College of Computing and Intelligence, Tianjin University, China. His major research interests are active robotic vision and visual intelligence, specifically including active camera relocalization and lighting recurrence, general Markov Random Fields modeling, energy minimization, active 3D scene perception, SLAM, video analysis, and generic pattern recognition. Recently, he focuses on solving preventive conservation problems of cultural heritages via computer vision and machine learning. He is the Associate Editor of Neurocomputing and Journal of Ambient Intelligence and Humanized Computing.

Yang Liu graduated in 2005 with a Bachelor of Computing (Honours) in the National University of Singapore (NUS). In 2010, he obtained his Ph.D. and started his post-doctoral work in NUS and MIT. In 2012, he joined Nanyang Technological University (NTU), and currently is a full professor and Director of the cybersecurity lab in NTU. Dr. Liu specializes in software verification, security and software engineering. His research has bridged the gap between the theory and practical usage of formal methods and program analysis to evaluate the design and implementation of software for high assurance and security. By now, he has more than 270 publications in top tier conferences and journals. He has received a number of prestigious awards including MSRA Fellowship, TRF Fellowship, Nanyang Assistant Professor, Tan Chin Tuan Fellowship, Nanyang Research Award and 8 best paper awards in top conferences like ASE, FSE and ICSE.