EINet: Point Cloud Completion via Extrapolation and Interpolation

Pingping Cai, Canyu Zhang, Lingjia Shi*, Lili Wang*, Nasrin Imanpour*, and Song Wang

University of South Carolina, Columbia, South Carolina, USA
{pcai, canyu, lingjia, liliw, imanpour}@email.sc.edu
songwang@cec.sc.edu

Abstract. Scanned point clouds are often sparse and incomplete due to the limited field of view of sensing devices, significantly impeding the performance of downstream applications. Therefore, the task of point cloud completion is introduced to obtain a dense and complete point cloud from the incomplete input. The fundamental challenges in tackling this task involve accurately inferring the missing shapes and upsampling them to higher densities. In this paper, we propose a novel approach to address this task, which formulates the completion task as a dual problem: a feature-wise extrapolation problem where the shape features of the partial point cloud are extrapolated to outlier regions for the recovery of missing portions, and a feature-wise interpolation problem to achieve point cloud upsampling. Based on these, we propose the EINet, a new point cloud completion paradigm with a novel Extrapolation module that can predict the missing shapes for the partial point cloud and a newly designed Interpolation module to upsample the point cloud. Extensive evaluation results demonstrate that EINet achieves compelling performance compared to previous state-of-the-art methods. The code is open-sourced at https://github.com/corecai163/EINet.

1 Introduction

Due to their simplicity in depicting high-resolution 3D objects, point clouds have been widely used across various 3D applications, including autonomous driving [14, 38], robotics [21], and 3D object classification [11, 20, 26, 34]. However, acquired point clouds are usually sparse and incomplete, lacking crucial details due to limited scanning viewpoints and restricted fields of view of the sensing devices. This incompleteness can substantially impede the performance of various downstream applications [15, 22, 39, 43, 45]. To address this issue, the task of point cloud completion has been introduced, which aims to reconstruct the missing shape and generate a dense and complete output. However, this task is notably challenging given the fact that the shape of the missing portion is usually unknown.

* L. Shi, L. Wang, and N. Imanpour assert joint third authorship.

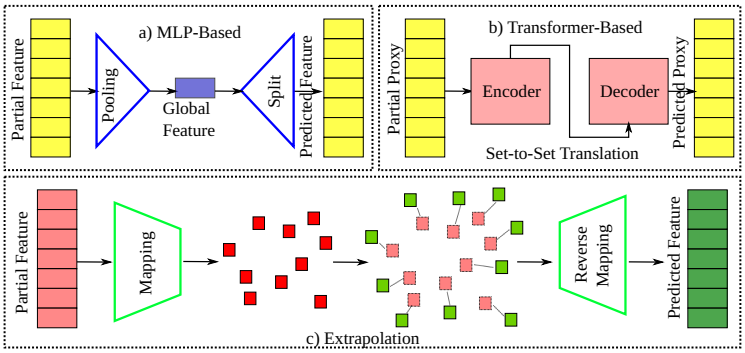


Fig. 1: Distinguishing between approaches for predicting the missing shape features, we illustrate three categories: a) MLP-Based, b) Transformer-Based, and c) The proposed Extrapolation-Based method.

The overarching idea to address this task involves leveraging the partially observed point cloud to infer the missing parts, resulting in the creation of a coarse but complete seed point cloud, which is subsequently upsampled to a denser and complete output. As a result, many distinct foundational models have been introduced in line with this idea [8,24,25,36,37,45]. These models can be broadly classified into three categories based on their approaches to seed point cloud generation: 3D Convolution-Based, MLP-Based, and Transformer-Based.

3D Convolution-Based methods [5,7,31,45] initially transform the input point cloud into voxels and employ robust 3D convolution techniques to propagate features to the empty voxels. However, 3D convolution operations are computationally intensive, and these methods are often limited by the number of voxels used, resulting in the loss of detailed geometric structure during the voxelization process. Conversely, MLP-Based methods process the points directly by first extracting a global shape feature from the incomplete point cloud and subsequently employing MLP layers to split this global shape feature into multiple seed points [37, 40] or features [1, 24, 30] to cover the missing portion (Figure 1a). Nevertheless, the extracted global feature contains only partial information representing an incomplete shape. Thus, these methods struggle to construct intricate details of the missing portion and tend to produce ambiguous seed points. In contrast, recent Transformer-Based methods [12, 25, 35, 36, 44] have achieved promising completion quality. They reframe the point cloud completion task as a set-to-set translation problem, where they first extract point proxies from the input partial point cloud and then employ a transformer encoder-decoder architecture to predict the proxies of the missing area, which is then used to reconstruct the output partial point cloud (Figure 1b).

Different from previous works, our innovation to solve this challenging problem comes from a deep insight into inferring the missing shapes given partial observations. Since the partial observations and the missing shapes together constitute the complete point clouds, they should inherently be highly correlated.

Instead of utilizing the transformer to capture the relation as in a set-to-set translation task, we present a unique viewpoint to tackle this problem: although it is evident that the partial shape is not linearly correlated with the missing shape in the coordinate space due to the complex shapes of point clouds, it is possible to map them into a high-dimensional feature space that can establish linear correlation. This insight led us to believe that the completion process can be achieved through a simple extrapolation operation in the feature space. Thus, we present a novel point cloud completion paradigm by reformulating the shape completion process as a feature space extrapolation problem, where we linearly extrapolate existing features extracted from the partial point cloud into new regions in the feature space to obtain the features of the missing portion, as illustrated in Figure 1c. Furthermore, to increase the density of the point cloud, we reframe the upsampling process as a feature space interpolation problem, where existing features are adaptively interpolated into inner areas based on their surrounding features to generate new points. But, the success of the proposed extrapolation or interpolation operation hinges upon a fundamental prerequisite – the existence of features within a linear-addable feature space. This condition enables straightforward feature-wise addition and subtraction operations, facilitating the seamless application of these techniques. To ensure this condition, we introduce the Mapping Constraints, a paired set of mapping and reverse mapping functions that transform the shape features into a linear-addable feature space and subsequently restore them to their original space.

Following this novel paradigm, we introduce EINet, a straightforward yet effective point cloud completion network. EINet comprises newly designed Predictive Linear Extrapolation modules, Seed Feature Propagation modules, and Adaptive Interpolation modules. To validate the performance of the proposed EINet, we evaluate it on three standard datasets: PCN [37], ShapeNet-55/34 [36], and KITTI [6]. The experimental results demonstrate the effectiveness of EINet by achieving promising performance even comparable to previous state-of-the-art (SOTA) methods. In summary, our main contributions are as follows:

1. We introduce a novel paradigm for solving the point cloud completion task, where the extrapolation operation is used to reconstruct the missing shape and the interpolation operation is used to upsample the coarse point cloud.
2. We propose EINet, with carefully designed Predictive Linear Extrapolation module and Adaptive Interpolation modules to achieve point cloud completion.
3. We evaluate the proposed method on multiple datasets and show that the proposed EINet achieves competitive performance compared to previous foundation methods.

2 Related Work

Traditional approaches to 3D shape completion rely on geometric rules [9, 16, 18, 42] or template matching [10, 13, 17, 23] for inferring missing components. These methods typically assume the presence of a smooth 3D shape surface or

rely on comprehensive shape datasets, making them vulnerable to novel objects and environmental noise. Early deep learning methods [5, 7, 31, 45] adopt voxel grids to represent 3D objects. These methods utilize the strong capabilities of 3D convolutions to propagate and infer missing information, achieving remarkable performance in 3D shape completion. Nevertheless, voxelization has its limitations, as it sacrifices finer details in the 3D shape and substantially escalates memory usage when increasing the voxel grid's resolution.

Recent advancements in deep neural networks have led to methods that directly manipulate raw point cloud data [2, 19, 20, 41]. Based on these techniques, Yuan *et al*. introduced PCN [37], a novel coarse-to-fine point cloud completion network. It first extracts a global shape code to describe the coarse structure of the input points. Subsequently, it generates an initial coarse but complete point cloud from the global shape code, followed by processing it through a folding-based upsampling block to create a denser point cloud. But folding-based upsampling blocks may not effectively reconstruct the upsampled point cloud, potentially leading to the loss of crucial geometric details. Recognizing this limitation, methods like SA-Net [27] have extended the upsampling process into multiple stages by introducing hierarchical folding blocks. This has triggered a series of following works aimed at designing increasingly proficient upsampling blocks. For instance, Xiang *et al*. devised SnowFlakeNet [30], with well-crafted Snowflake Point Deconvolution blocks designed to better preserve local geometric details. Additionally, FBNet [32] and SeedFormer [44] have made notable contributions by introducing the Feedback-Aware Completion block and Upsample Transformers, respectively, to refine and upsample coarse point clouds.

Different from previous methods, PoinTr [36] explores a new approach for point cloud completion via reformulating this task as a set-to-set translation problem, where they use a Set Transformer encoder-decoder architecture for generating the proxies of missing regions based on the proxies of existing regions. Following this idea, ProxyFormer [12] introduces the missing part sensitive transformer, which converts random normal distribution into reasonable position information and uses proxy alignment to refine the missing proxies. SDNet [4] proposed two sub-networks to refine both the partial inputs and the partial regions reconstructed by the Set Transformer. AdaPoinTr [35] introduced an adaptive query generation mechanism to deal with diverse situations and an auxiliary denoising task is designed to effectively make the optimization more stable and efficient.

3 Proposed Method

We follow the previous coarse-to-fine pipeline [30, 44] to achieve point cloud completion. Specifically, when provided with an incomplete point cloud $P \in \mathbb{R}^{N\times 3}$, our initial objective is to infer the missing shapes and generate a seed point cloud $S \in \mathbb{R}^{N_2\times 3}$, along with seed features $SF \in \mathbb{R}^{N_2\times C}$. Then, we progressively upsample the seed point cloud to the target resolution, resulting in a complete and densely populated output point cloud $O \in \mathbb{R}^{N_3\times 3}$. To realize these ideas,

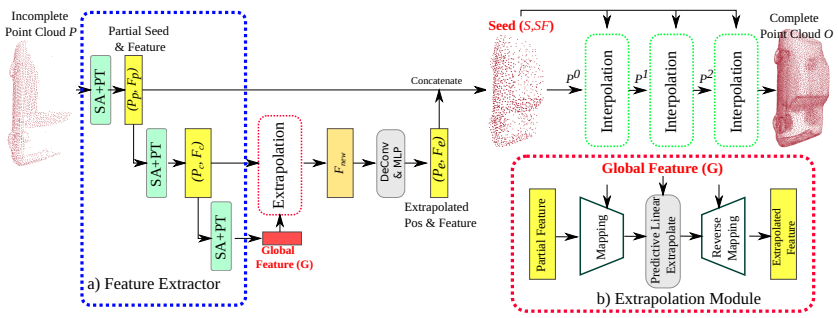


Fig. 2: The overall pipeline of the proposed network. a) The architecture for the feature extractor. b) The architecture for the extrapolation module. We use the extrapolation module to generate the missing points and features, which are then merged with the input partial point cloud to obtain a coarse but complete seed point cloud.

our approach, EINet, comprises three key components: 1) the feature extractor, 2) the extrapolation module, and 3) the interpolation module. Figure 2 provides an overall illustration of the network architecture.

3.1 Feature Extractor

The feature extractor aims to subsample a set of partial seed points $P_p \in \mathbb{R}^{N_0 \times 3}$ and their corresponding features $F_p \in \mathbb{R}^{N_0 \times C}$. We follow the previous work [30, 44] and leverage the capabilities of both the Set Abstraction (SA) layer [20], and the Point Transformer (PT) [41] to extract multi-resolution partial point clouds and features, as illustrated in Figure 2a. The Set Abstraction layer employs FPS layers to downsample a cluster of points and utilizes MLPs with max pooling layers to extract permutation-invariant features. Meanwhile, the Point Transformer layer enhances the local shape context [30]. Multiple layers of Set Abstraction and Point Transformer are sequentially stacked together to capture the partial features. They are designed to represent the local structures of the partial point cloud. The feature extractor also extracts the global shape feature denoted as $G \in \mathbb{R}^{1 \times C1}$, which serves as additional guidance in the subsequent Extrapolation Module.

3.2 Extrapolation Module

The extrapolation module is introduced to infer the features of missing regions based on the previously extracted partial features. As visualized in Figure 2b, the key idea of this extrapolation module is quite straightforward. Given a set of partial features $F_c \in \mathbb{R}^{N_1 \times 2C}$ obtained from an incomplete point cloud, a **predictive linear extrapolation** is employed to extend the existing features into the new region of the feature space.

Nonetheless, it is important to emphasize that the success of the linear extrapolation method hinges on the additivity of the features. In order to fulfill this

prerequisite, we have designed the alignment **mapping** layer, which maps the input partial features into a feature space so that mapped features are amenable to linear addition. Specifically, it is an encoder $\mathbb{EN}$ with a three-layer MLP to transform the partial features into the feature space, denoted as $F'_c = \mathbb{EN}(F_c, G)$, where $F'_c \in \mathbb{R}^{N_1 \times \frac{C}{2}}$. Next, we execute the extrapolation operation to derive the new shape feature in the feature space as follows:

$$\begin{aligned} w_{scale}, W_{center} &= \mathbf{MLP}(G), \\ F'_{new} &= F'_c + w_{scale}(F'_c - W_{center}) - Avg(F'_c), \end{aligned} \tag{1}$$

where $w_{scale} \in \mathbb{R}$ and $W_{center} \in \mathbb{R}^{1 \times \frac{C}{2}}$ are predicted extrapolation parameters, and $Avg(F'_c) \in \mathbb{R}^{1 \times \frac{C}{2}}$ calculates the mean value of input features. This equation means that we extrapolate the current features to obtain new features based on the predictive centers and scales. Then, a **reverse mapping** layer $\mathbb{DE}$, which is another three-layer MLP decoder, is used to map the extrapolated features back into the original feature space, leading to $F_{new} = \mathbb{DE}(F'_{new}, G)$.

Once the features of the missing portion are obtained, we employ a Deconvolution layer to upsample the feature and obtain $F_e \in \mathbb{R}^{N_0 \times C}$ and then use an MLP to predict the coordinates: $P_e = \mathbf{MLP}(F_e)$. Finally, the extrapolated features and coordinates are concatenated with the partial points, resulting in a coarse but complete **seed point cloud** S with associated seed features SF.

3.3 Interpolation Module

Given complete seed point clouds, our next objective is to gradually upsample them to the target resolution. We formulate this process as an interpolation problem, wherein we aim to predict a set of unique interpolation coefficients. These coefficients are employed to combine the point features within a local region and generate new point features.

However, unlike the extrapolation module, the upsampling task places a stronger emphasis on local details to represent the underlying surface accurately and avoid outlier points. To achieve this, we introduce the Seed Feature Propagation module, which integrates shape features from the seed points. Then, guided by these propagated features, the Adaptive Interpolation module has been devised to predict the interpolation coefficients for generating new features. Figure 3a shows the architecture of the Interpolation Module.

Seed Feature Propagation Given an input coarse point cloud P^l and corresponding feature set F^l, where l indicates the upsampling step (*e.g.*, $l = 0$ means seed point cloud), the Seed Feature Propagation module aims to propagate shape features from seed points into the input coarse point cloud. Previous feature propagation methods [20, 44] use only the coordinates differences as weights to merge nearby seed features, which overlooks the local feature context. Thus, we proposed to use both the coordinates differences and the feature differences to enrich the local context and offer regional insights.

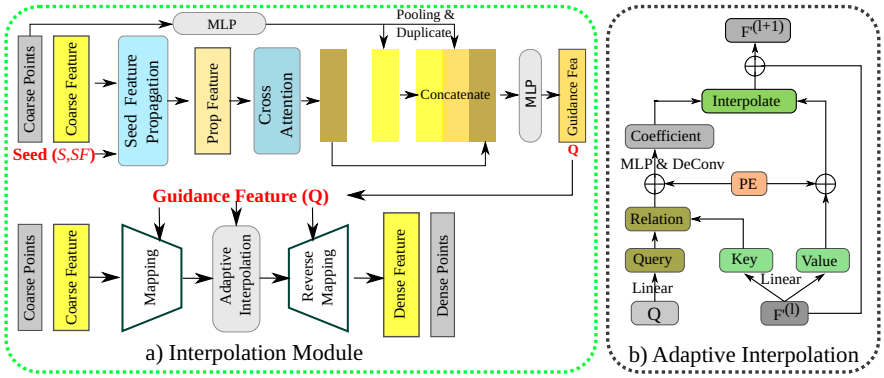


Fig. 3: The network architecture for the adaptive interpolation module. PE means the Point Proxy [36] Embedding.

Especially, for each point $p_i \in P^l$, we first calculate the coordinate differences d_{ij} between its k nearest seed points $s_j \in S$. Here $d_{ij} = Concat(ps_{ij}, ||ps_{ij}||^2)$, with $ps_{ij} = p_i - s_j$, where $j \in NS_k(i)$ and $NS_k(i)$ represents the index of k nearest points of p_i in seed points S (with a default value of $k = 8$). Next, an MLP $\mathbb{M}$ is employed to map the coordinate difference into the feature space and serve as the positional embedding.

Similarly, we calculate the feature differences via subtraction relation within the local neighbor features of the seed points and apply the channel-wise cross attention to predict the propagation weights:

$$\begin{aligned} q_i = Linear(f_i); \quad k_j = Linear(sf_j); \\ w_{ij} = \sigma(\mathbf{MLP}(q_i - k_j + \mathbb{M}(d_{ij})) \end{aligned} \tag{2}$$

where $f_i \in F^l, sf_j \in SF$, $j \in NS_k(i)$, d_{ij} is the coordinate differences, and σ is the softmax function so that $\sum_j w_{ij} = 1$. The propagated feature PF_i^l is then derived as follows:

$$v_j = Linear(sf_j); \quad PF_i^l = F_i^l + \sum_j w_{ij}(v_j + \mathbb{M}(d_{ij})) \tag{3}$$

Then a cross attention layer is applied to merge the propagated feature with the coarse input feature: $PF^l = CrossAttention(PF^l, F^l, F^l)$.

Finally, inspired by SnowFlake [30] we employ an MLP to extract the low-level coordinate features. These features are then grouped using max pooling and combined with the propagated features as depicted in Figure 3a. An additional MLP is employed to generate the guidance feature, denoted as Q, which serves as local shape guidance in the interpolation module.

Adaptive Interpolation The adaptive interpolation block achieves upsampling via interpolating the coarse feature F^l by adaptively predicting a set of

interpolation coefficients. Similar to the extrapolation module, the initial step is to map the coarse point features into a linearly additive feature space using the local shape guidance provided by Q. This mapping is achieved as follows: $F'^l = \mathbf{EN}(F^l, Q)$.

Then, to predict the interpolation coefficients, we also borrow ideas from point transformer [41] via applying channel-wise cross attention on F'^l. However, it has been observed that relying solely on these mapped features does not suffice to generate well-distributed points. This is because a uniform feature distribution within the linearly additive feature space may become non-uniform once these features are mapped back to the original feature space.

To this end, an additional guidance strategy is required to enhance the proposed feature space interpolation approach. Especially, we incorporate the point proxies [36] in the original feature space as additional interpolation guidance denoted as **Proxy Embedding**. Note that the proxy embedding PE is constructed by incorporating positional information into PF^l [36]. Figure 3b illustrates the architecture design of the proposed adaptive interpolation module.

Given local shape guidance $q_i \in Q$ and the mapped feature $f'_i \in F'^l$ for each point $p_i \in P^l$, we first calculate its feature differences df_{ij} via subtraction relation within its k neighbor points:

$$q_i = Linear(q_i);\ k_j = Linear(f'_j);\ df_{ij} = q_i - k_j \tag{4}$$

where $j \in NN_k(i)$ and $NN_k(i)$ represents the k nearest points of p_i in P^l. Then similar to the feature difference we calculate the proxy differences pe_{ij} as follows: $pe_{ij} = pe_i - pe_j$, where $j \in NN_k(i)$.

The feature differences and the point proxy differences are combined together to estimate the interpolation coefficient: $w_{ij} = \sigma(\mathbf{DeConv}(\mathbf{MLP}(df_{ij} + pe_{ij}))$ where σ is the softmax function so that $\sum_j w_{ij} = 1; j \in NN_k(i)$ and DeConv is the 1D deconvolution layer to generate multiple sets of coefficients.

Then, we proceed to interpolate the neighboring features of p_i based on the predicted coefficients w_{ij}. Each interpolated feature F'^{l+1} is calculated as follows:

$$v_j = Linear(f'_j); \quad F_i'^{l+1} = F_i'^l + \sum_{j=0}^{K} w_{ij}(v_j + pe_{ij}) \tag{5}$$

Finally, the interpolated feature is mapped back to the original feature space through the decoder $F^{l+1} = \mathbf{DE}(F'^{l+1}, Q)$.

3.4 Upsampling via Interpolation Modules

After obtaining the newly interpolated features, we then employ another MLP to predict the point coordinates: $P^{l+1} = P^l + \mathbf{MLP}(F'^{l+1})$. It's worth noting that these interpolated features, along with the coordinates, can serve as input for the following interpolation modules. Thus, we can stack multiple adaptive interpolation modules together to effectively achieve varying upsampling ratios and generate a complete and densely populated output point cloud O.

3.5 Loss Function

Following previous methods, we use Chamfer Distance (CD) as loss functions to supervise the coarse-to-fine completion steps and train our network. However, another concern lies in accurately mapping the extrapolated features back into the original feature space through the proposed reverse mapping function. To address this, we introduce a **Mapping Constraint** as an additional loss function. For instance, given the partial features F_p, our target is to have F_p be identical to $\mathbb{DE}(\mathbb{EN}(F_p))$. Thus, the mapping constraint loss is devised as follows:

$$L_{MC} = ||F_p - \mathbb{DE}(\mathbb{EN}(F_p))|| \tag{6}$$

To sum up, the total loss function used in training is defined as follows:

$$L = L_{seed} + \alpha L_{upsample} + \beta \sum L_{MC} \tag{7}$$

where L_{seed} calculates the chamfer loss between the seed point cloud and ground truth, $L_{upsample}$ calculates the chamfer loss for each upsampling step, and $\sum L_{MC}$ indicates the Mapping Constraints across all Extrapolation and Interpolation modules. The hyperparameters α and β are designed for balancing the weights of these components.

4 Experiments

4.1 Datasets and Evaluation Metrics

The **PCN** dataset was first introduced in [37]. It contains 30,974 distinct shapes collected from the ShapeNet dataset [3]. The complete point clouds are created by uniformly sampling 16,384 points from original 3D meshes, while the partial point clouds are generated by back-projecting 2.5D depth images into the 3D space from various viewpoints. In summary, it consists of 28,974 training samples, 800 validation samples, and 1,200 testing samples. Furthermore, since incomplete point clouds may have different numbers of points, we follow previous methods by resampling them to a fixed number of 2,048 points. The **ShapeNet-55/34** datasets, introduced in PoinTr [36], are also derived from the ShapeNet dataset [3] by uniformly sampling 8,192 points from original meshes. The ShapeNet-55 subset contains 55 categories with 41,952 training shapes and 10,518 testing shapes. Conversely, ShapeNet-34 subset comprises 46,765 shapes across 34 categories for training, and the testing set contains 5,705 shapes, which is further divided into two segments: one with 3,400 shapes from 34 seen classes and another with 2,305 shapes from 21 unseen classes. In accordance with previous approaches, the models are evaluated on point clouds with varying missing ratios, e.g., 25%, 50%, and 75%, corresponding to three levels of completion task difficulty: simple (S), moderate (M), and hard (H), respectively. The **KITTI** dataset [6] is also included in the experiment to evaluate the robustness of the proposed method on real-collected point clouds. This dataset is gathered from an

Table 1: Point cloud completion results on the PCN dataset compared to previous algorithms (CD-L1 $\times 10^{-3}$). Bolded text means the best performance and the underlined text means the second-best.

Methods	Average	Plane	Cabinet	Car	Chair	Lamp	Couch	Table	Watercraft
FoldingNet [33]	14.31	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99
TopNet [24]	12.15	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12
PCN [37]	9.64	5.50	22.70	10.63	8.70	11.0	11.34	11.68	8.59
GR-Net [31]	8.83	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04
PMP [28]	8.73	5.65	11.24	9.64	9.51	6.95	10.83	8.72	7.25
NSFA [29]	8.06	4.76	10.18	8.63	8.53	7.03	10.53	7.35	7.48
SnowFlake [30]	7.21	4.29	9.16	8.08	7.89	6.07	9.23	6.55	6.40
PoinTr [36]	7.26	4.05	9.34	7.97	7.92	6.40	9.29	6.66	6.47
AdaPoinTr [35]	**6.53**	**3.68**	<u>8.82</u>	**7.47**	**6.85**	5.47	**8.35**	**5.80**	<u>5.76</u>
FBNet [32]	6.94	3.99	9.05	7.90	7.38	5.82	8.85	6.35	6.18
ProxyFormer [12]	6.77	4.01	9.01	7.88	7.11	5.35	<u>8.77</u>	<u>6.03</u>	5.98
SeedFormer [44]	6.74	3.85	9.05	8.06	7.06	<u>5.21</u>	8.85	6.05	5.85
Ours	<u>6.63</u>	<u>3.96</u>	**8.81**	<u>7.74</u>	<u>6.93</u>	**5.03**	8.80	6.15	**5.57**

autonomous driving platform, and due to the limited viewpoints of the LiDAR scanner, the collected point clouds exhibit natural incompleteness and sparsity. Following previous approaches, a sequence of Velodyne scans is extracted from the KITTI dataset, with the focus solely on point clouds within the object bounding boxes labeled as cars. In total, the KITTI dataset comprises 2,483 partial point clouds without ground truth.

For PCN and ShapeNet datasets, we follow previous methods by employing three widely used metrics: **CD-L1**, **CD-L2**, and **F1-Score@1%** to obtain a thorough assessment of the performance of the proposed method. Note that smaller values of the CD metric indicate better performance, while for the F1-Score, larger values are indicative of better results. For the KITTI dataset, we employ Fidelity and Minimal Matching Distance (MMD) as evaluation metrics. Since there is no ground truth available, it is important to note that these two metrics serve as general indicators, and a lower/higher score does not necessarily equate to better performance.

4.2 Implementation Details

The feature extractor implemented in this paper contains two layers of set abstraction with point transformers, subsampling $N_0 = 512$ and $N_1 = 128$ partial points, followed by one layer of global max-pooling to extract the global shape feature G with a channel $C_1 = 512$. Subsequently, the extrapolation module generates an extrapolated point cloud comprising $N_0 = 512$ points, which will be merged with the partial seeds to generate an initial seed point cloud S with $N_2 = 1024$ points. Then the interpolation modules progressively upsample the coarse point cloud $P^0 = S$, yielding dense point clouds P^1,P^2,O, where O represents the final predicted point cloud with the required resolution. The number of

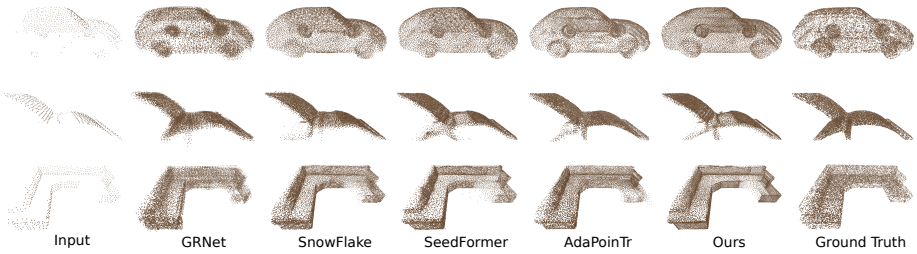


Fig. 4: The point cloud completion results of different methods on the PCN dataset. Notably, we see that our method is able to reconstruct the missing component with better shapes, *e.g.* couch boundary and chair legs. Please zoom in for more details.

Table 2: Quantitative results on Seen ShapeNet-34 test set and Unseen ShapeNet-21 test set (CD-L2 $\times 10^{-3}$).

	Seen ShapeNet-34					Unseen ShapeNet-21				
Method	CD-S	CD-M	CD-H	CD-avg	F1	CD-S	CD-M	CD-H	CD-avg	F1
FoldingNet [33]	1.86	1.81	3.38	2.35	0.139	2.76	2.74	5.36	3.62	0.095
PCN [37]	1.87	1.81	2.97	2.22	0.154	3.17	3.08	5.29	3.85	0.101
TopNet [24]	1.77	1.61	3.54	2.31	0.171	2.62	2.43	5.44	3.5	0.121
GRNet [31]	1.26	1.39	2.57	1.74	0.251	1.85	2.25	4.87	2.99	0.216
PoinTr [36]	0.76	1.05	1.88	1.23	0.421	1.04	1.67	3.44	2.05	0.384
AdaPoinTr [35]	0.48	**0.63**	**1.07**	**0.73**	0.469	0.61	**0.96**	**2.11**	**1.23**	0.416
SnowFlake [30]	0.6	0.86	1.5	0.99	0.422	0.88	1.46	2.92	1.75	0.388
ProxyFormer [12]	**0.44**	0.67	1.33	0.81	0.466	0.60	1.13	2.54	1.42	0.415
SDNet [4]	0.49	0.67	1.27	0.81	0.535	0.64	1.02	2.32	1.33	0.503
SeedFormer [44]	0.48	0.7	1.3	0.83	0.452	0.61	1.07	2.35	1.34	0.402
Ours	0.46	0.68	1.24	0.79	0.446	**0.59**	1.01	2.19	1.26	0.413

feature channels C remains fixed at 128 for all extrapolation and interpolation modules. Besides, throughout our experiments, the hyperparameters α and β consistently maintain values of 1 and 0.01.

Our network is implemented using PyTorch with Nvidia CUDAs and is trained from scratch in an end-to-end manner. Specifically, we set the total number of training epochs to 400, employing a batch size of 64. We utilize the AdamW optimization algorithm to train the network, with an initial learning rate of 0.0016, which is gradually reduced by a factor of 0.2 every 100 epochs. The training process is executed on four Nvidia V100 32G GPUs and costs approximately 2 days for PCN and ShapeNet-55/34 dataset.

4.3 Experiment Results

We first evaluate the performance of the proposed EINet on the PCN dataset and compare it with previous SOTA methods. Table 1 presents the results of different methods. Notably, our method achieves an outstanding average CD-L1 score of 6.63×10^{-3}, showcasing a significant improvement over the performance of prior foundation methods like PoinTr [36], PCN [37], TopNet [24]. Such performance is

Table 3: Quantitative results of different methods on the ShapeNet-55 benchmark dataset (CD-L2 $\times 10^{-3}$).

Method	Table	Chair	Air-plane	Car	Sofa	Bird house	Bag	Rem-ote	Key-board	Roc-ket	CD S	CD M	CD H	CD Avg	F-Score
FoldingNet	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	2.13	2.29	1.02	1.85	2.06	4.5	2.86	1.33	0.89	1.32	1.94	1.96	4.08	2.66	0.133
TopNet	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	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
Snowflake	0.98	1.12	0.54	0.98	1.02	1.93	1.08	0.57	0.48	0.61	0.7	1.06	1.96	1.24	0.398
PoinTr	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
AdaPoinTr	**0.62**	**0.69**	**0.33**	0.81	**0.63**	1.33	0.68	0.38	**0.33**	**0.34**	**0.49**	**0.69**	**1.24**	**0.81**	0.503
ProxyFormer	0.70	0.83	0.34	**0.78**	0.69	-	-	-	-	-	0.49	0.75	1.55	0.93	0.483
SeedFormer	0.72	0.81	0.4	0.89	0.71	-	-	-	-	-	0.5	0.77	1.49	0.92	0.472
Ours	0.66	0.79	0.41	0.84	0.69	1.49	0.73	0.42	**0.33**	0.49	**0.49**	0.75	1.46	0.90	0.432

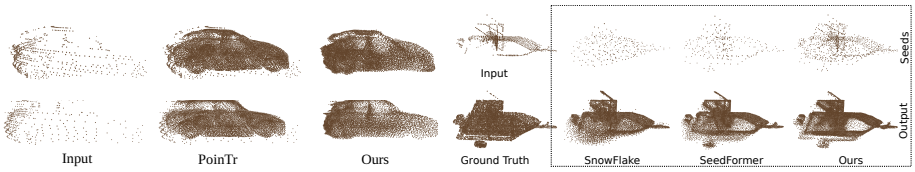
even competitive to recent SOTA algorithms like SeedFormer [44] and AdaPoinTr [35]. Then, Figure 4 includes several visual examples of the completion results. From the figure, it becomes evident that our proposed algorithm can correctly generate shape details for the missing parts. In contrast, other algorithms, such as SnowFlake and SeedFormer, may generate ambiguous shapes, often accompanied by a considerable number of outliers. For example, in the second row of Figure 4, our method can reconstruct the missing chair legs with better shapes, while for comparison, SnowFlake generates only ambiguous shapes for the legs and SeedFormer tends to generate many noise points.

Then, to showcase the generalization capability of the proposed method, we perform additional experiments on the ShapeNet-55/34 dataset. In Tables 3 and 2, we present the performance of EINet on the ShapeNet-55/34 dataset. Impressively, our method achieves an average CD-L2 score of 0.90×10^{-3} on the ShapeNet-55 dataset, demonstrating a remarkable advancement over the performance of previous counterparts. Furthermore, even when dealing with the most challenging ShapeNet-34 seen and ShapeNet-21 unseen dataset, our method shows quite a robust performance. For example, we achieve an average CD-L2 score of 0.79×10^{-3} on seen data and an average CD-L2 score of 1.26×10^{-3} on unseen data.

Finally, we examine the robustness of the proposed EINet on the real-world KITTI dataset. As the KITTI dataset contains only real-collected Lidar point clouds, we do not have ground-truth point clouds for training. Instead, we use a pre-trained model on the PCN car dataset and directly test it on the KITTI dataset. Table 4 shows the quantitative completion results, and Figure 5a shows some visual examples. We see that our method achieves an average Fidelity score of 1.48 and an average MMD of 0.512, showing robust performance comparable to previous foundation models like PCN and PoinTr. Besides, from the visual examples, we see that the previous set-to-set translation method PoinTr [36] tends to generate outlier points. This is because PoinTr directly merges input with output, and the input partial point cloud does not match the generated point cloud, resulting in discontinuities. In contrast, our method can generate cleaner point clouds and reconstruct better structure of the cars.

Table 4: Quantitative results on the KITTI dataset.

Method	PCN	GR-Net	PoinTr	AdaPoinTr	ProxyFormer	SeedFormer	Ours
FD ($\times 10^{-3}$)	2.235	0.816	0.00	0.237	0.00	0.151	1.48
MMD ($\times 10^{-3}$)	1.366	0.568	0.526	0.392	0.508	0.516	0.512



(a) Visual comparisons of generated point clouds on the KITTI dataset. Our results contain fewer outliers than PoinTr's [36].

(b) The visualization of generated seed point clouds from different methods. Please zoom in for details.

Fig. 5: The visual comparisons on the KITTI dataset and the quality of generated seed point clouds on the PCN dataset.

4.4 Ablation Study

Since we introduce a new paradigm for the point cloud completion task, to obtain a better understanding of the proposed method, we perform several ablation studies on the model complexity and the effectiveness of each component on the PCN dataset.

Quality of Seeds: Since the extrapolation module is one of the key designs of the proposed EINet, we then visualize the generated seed points to understand the effectiveness of the extrapolation module. Besides, we also present the corresponding dense outcomes, in order to elucidate the quality of the upsampling stage. Figure 5b illustrates the seed point clouds and the dense points generated by different methods on one sample in the PCN dataset. In contrast to the SnowFlake [30], wherein seed points encompass only ambiguous shapes, our generated seed points are imbued with intricate shape details. While SeedFormer [44] can produce commendable seeds, its final completion result exhibits many noise points. This underscores the superiority of the proposed Interpolation module, which excels in generating a complete point cloud with clear borders and well-defined shapes.

Model Complexity: One major concern is the model complexity of the proposed method. Thus, we evaluate its number of trainable parameters (Params) and the real inference speed (pc/s, point clouds per second) with a batch size of 1 compared to previous methods. This experiment is conducted on an RTX A4000 GPU and Table 5a shows the result. Our model achieves a testing speed of 21 pc/s and its size is quite small with only 2.80 M trainable parameters, which is almost 10 times smaller than PointTr with 28.9 M parameters.

Architecture Design: Furthermore, we investigate the importance of the different components designed in this paper. To begin with, we ablate the feature propagation module by switching it with the three nearest-neighbor based feature propagation layer [20, 44]. Then, we remove the Extrapolation-Based

Table 5: Ablation study on the model complexity and network design.

Methods	Params	Speed	CD_{PCN}
PCN [37]	5.04	241	9.64
SnowFlake [30]	19.30	59	7.21
PoinTr [36]	28.9	48	7.26
SeedFormer [44]	3.24	18	6.74
Ours	2.80	21	6.63

(a) Model complexity of different methods. We report the number of parameters (M) and the testing speed (pc/s).

Module	Combination			
Feature Propagation		✓	✓	✓
Extrapolation Module	✓		✓	✓
Mapping Constraints	✓	✓		✓
CD-L1 ($\times 10^{-3}$)	6.87	7.02	6.69	6.63

(b) Effectiveness of different components.

seed generation head and substitute it with the SnowFlake's seed generation head [30]. Finally, we study the effect of the proposed Mapping Constraints. As depicted in Table 5b, the optimal performance is attained when all the designed modules are applied. Especially, we observe a notable performance drop when the Feature Propagation module is removed, where the CD-L1 increases from 6.63×10^{-3} to 6.87×10^{-3}. Furthermore, without the Extrapolation module, we observe that the performance further drops to 7.02×10^{-3}, which proves the effectiveness of the proposed extrapolation strategy. Finally, the proposed Mapping Constraints also play an important role, with which the performance improves from 6.69×10^{-3} to 6.63×10^{-3}.

5 Conclusion and Disscussion

In this paper, we contribute a new algorithm for the point cloud completion task. Especially we introduced a novel paradigm that reframes the shape completion task as an extrapolation problem and the point cloud upsampling as an interpolation problem. Based on this paradigm, we designed the EINet, which integrates newly proposed Extrapolation and Interpolation modules to generate the missing shape features and upsample the point cloud in the feature space. The experiment results on multiple datasets prove the effectiveness of the proposed method.

Limitations and Future Work: Point cloud completion is an important research problem and can benefit many downstream applications. A limitation of current research is its focus on objects without background scenes, which makes it challenging to apply to real point cloud data collected in large environments, such as the KITTI dataset. Another concern is the assumption that the input point cloud is clean and contains little noise. In reality, noise can significantly affect the overall shape quality, especially when the input is sparse, resulting in degraded performance. Therefore, an interesting direction for future research is to design a pipeline for completing point clouds of large, real scenes with various objects and backgrounds. Additionally, another promising direction is to increase the robustness of point cloud completion algorithms.

References

1. Cai, P., Scott, D., Li, X., Wang, S.: Orthogonal dictionary guided shape completion network for point cloud. In: AAAI (2024) 2
2. Cai, P., Wu, Z., Wu, X., Wang, S.: Parametric surface constrained upsampler network for point cloud. In: AAAI (2023) 4
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository (2015) 9
4. Chen, J., Liu, Y., Liang, Y., Long, D., He, X., Li, R.: Sd-net: Spatially-disentangled point cloud completion network. In: ACM MM (2023) 4, 11
5. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: CVPR (2017) 2, 4
6. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) **32**(11), 1231–1237 (2013) 3, 9
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) 2, 4
8. Hong, S., Yavartanoo, M., Neshatavar, R., Lee, K.M.: Acl-spc: Adaptive closed-loop system for self-supervised point cloud completion. In: CVPR (2023) 2
9. Hu, W., Fu, Z., Guo, Z.: Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting. IEEE Transactions on Image Processing **28**(8) (2019) 3
10. Kim, V.G., Li, W., Mitra, N.J., Chaudhuri, S., DiVerdi, S., Funkhouser, T.: Learning part-based templates from large collections of 3d shapes. ACM TOG **32**(4) (2013) 3
11. Li, R., Li, X., Heng, P.A., Fu, C.W.: Pointaugment: An auto-augmentation framework for point cloud classification. In: CVPR (2020) 1
12. Li, S., Gao, P., Tan, X., Wei, M.: Proxyformer: Proxy alignment assisted point cloud completion with missing part sensitive transformer. In: CVPR (2023) 2, 4, 10, 11
13. Li, Y., Dai, A., Guibas, L., Nießner, M.: Database-assisted object retrieval for real-time 3d reconstruction. In: Computer Graphics Forum. vol. 34, pp. 435–446 (2015) 3
14. Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M.A., Cao, D., Li, J.: Deep learning for lidar point clouds in autonomous driving: A review. IEEE Transactions on Neural Networks and Learning Systems **32**(8), 3412–3432 (2021) 1
15. Lin, H.W., Tai, C.L., Wang, G.J.: A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. Computer-Aided Design **36**(1) (2004) 1
16. Mitra, N.J., Pauly, M., Wand, M., Ceylan, D.: Symmetry in 3d geometry: Extraction and applications. Computer Graphics Forum **32**(6), 1–23 (2013) 3
17. Nan, L., Xie, K., Sharf, A.: A search-classify approach for cluttered indoor scene understanding. ACM TOG **31**(6) (2012) 3
18. Pauly, M., Mitra, N.J., Wallner, J., Pottmann, H., Guibas, L.J.: Discovering structural regularity in 3d geometry. In: ACM SIGGRAPH. ACM (2008) 3
19. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017) 4
20. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017) 1, 4, 5, 6, 13

21. Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M.: Towards 3d point cloud based object maps for household environments. Robotics and Autonomous Systems **56**(11), 927–941 (2008) 1
22. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: CVPR (2020) 1
23. Sung, M., Kim, V.G., Angst, R., Guibas, L.: Data-driven structural priors for shape completion. ACM TOG **34**(6) (2015) 3
24. Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I., Savarese, S.: Topnet: Structural point cloud decoder. In: CVPR (2019) 2, 10, 11
25. Wang, Y., Tan, D.J., Navab, N., Tombari, F.: Learning local displacements for point cloud completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2022) 2
26. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM TOG (2019) 1
27. Wen, X., Li, T., Han, Z., Liu, Y.S.: Point cloud completion by skip-attention network with hierarchical folding. In: CVPR (2020) 4
28. Wen, X., Xiang, P., Han, Z., Cao, Y.P., Wan, P., Zheng, W., Liu, Y.S.: Pmp-net: Point cloud completion by learning multi-step point moving paths. In: CVPR (2021) 10
29. Wenxiao Zhang, Qingan Yan, C.X.: Detail preserved point cloud completion via separated feature aggregation. In: ECCV (2020) 10
30. Xiang, P., Wen, X., Liu, Y.S., Cao, Y.P., Wan, P., Zheng, W., Han, Z.: SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer. In: ICCV (2021) 2, 4, 5, 7, 10, 11, 13, 14
31. Xie, H., Yao, H., Zhou, S., Mao, J., Zhang, S., Sun, W.: Grnet: Gridding residual network for dense point cloud completion. In: ECCV (2020) 2, 4, 10, 11
32. Yan, X., Yan, H., Wang, J., Du, H., Wu, Z., Xie, D., Pu, S., Lu, L.: Fbnet: Feedback network for point cloud completion. In: ECCV (2022) 4, 10
33. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: CVPR (2018) 10, 11
34. Yavartanoo, M., Hung, S.H., Neshatavar, R., Zhang, Y., Lee, K.M.: Polynet: Polynomial neural network for 3d shape recognition with polyshape representation. In: 2021 International Conference on 3D Vision (3DV) (2021) 1
35. Yu, X., Rao, Y., Wang, Z., Lu, J., Zhou, J.: Adapointr: Diverse point cloud completion with adaptive geometry-aware transformers. IEEE Transactions on Pattern Analysis **45**(12), 14114–14130 (2023) 2, 4, 10, 11, 12
36. Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., Zhou, J.: Pointr: Diverse point cloud completion with geometry-aware transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 12498–12507 (2021) 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14
37. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: International Conference on 3D Vision (3DV) (2018) 2, 3, 4, 9, 10, 11, 14
38. Zeng, Y., Hu, Y., Liu, S., Ye, J., Han, Y., Li, X., Sun, N.: RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. IEEE Robotics and Automation Letters **3**(4), 3434–3440 (2018) 1
39. Zhang, C., Wu, Z., Wu, X., Zhao, Z., Wang, S.: Few-shot 3d point cloud semantic segmentation via stratified class-specific attention based transformer network. In: AAAI (2023) 1

40. Zhang, W., Dong, Z., Liu, J., Yan, Q., Xiao, C., et al.: Point cloud completion via skeleton-detail transformer. IEEE Transactions on Visualization and Computer Graphics (2022) 2
41. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: CVPR (2021) 4, 5, 8
42. Zhao, W., Gao, S., Lin, H.: A robust hole-filling algorithm for triangular mesh. In: IEEE International Conference on Computer-Aided Design and Computer Graphics (2007) 3
43. Zhao, Z., Wu, Z., Wu, X., Zhang, C., Wang, S.: Crossmodal few-shot 3d point cloud semantic segmentation. In: ACM MM (2022) 1
44. Zhou, H., Cao, Y., Chu, W., Zhu, J., Lu, T., Tai, Y., Wang, C.: Seedformer: Patch seeds based point cloud completion with upsample transformer. In: ECCV (2022) 2, 4, 5, 6, 10, 11, 12, 13, 14
45. Zhou, Y., Tuzel, O.: VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In: CVPR (2018) 1, 2, 4