# **Adaptive Local Basis Functions for Shape Completion**

Hui Ying
State Key Lab of CAD&CG, Zhejiang
University
Hangzhou, China
huiying@zju.edu.cn

Tianjia Shao\*
State Key Lab of CAD&CG, Zhejiang
University
Hangzhou, China
tianjiashao@gmail.com

He Wang University of Leeds Leeds, UK H.E.Wang@leeds.ac.uk

Yin Yang University of Utah Salt Lake City, USA yin.yang@utah.edu

Kun Zhou
State Key Lab of CAD&CG, Zhejiang
University
Hangzhou, China
kunzhou@acm.org

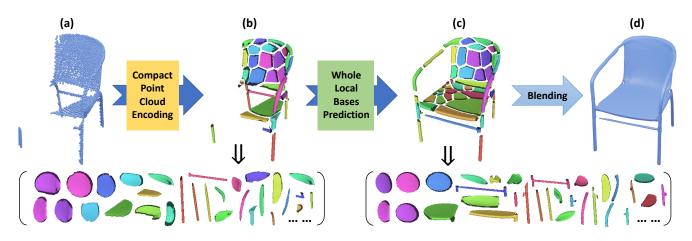


Figure 1: Given an input partial point cloud (a), we first encode the visible observation as a set of local basis functions (32 in this example) (b), and then use them to predict the local basis functions for the overall shape (64 in this example) (c), which are blended to form the final shape (d). We visualize the implicit surfaces inferred from the local basis functions in the bottom of (b) and (c), while (b) and (c) show the positioned local basis functions without the overlapped regions.

#### **ABSTRACT**

In this paper, we focus on the task of 3D shape completion from partial point clouds using deep implicit functions. Existing methods seek to use voxelized basis functions or the ones from a certain family of functions (e.g., Gaussians), which leads to high computational costs or limited shape expressivity. On the contrary, our method employs adaptive local basis functions, which are learned end-to-end and not restricted in certain forms. Based on those basis functions, a local-to-local shape completion framework is presented. Our algorithm learns sparse parameterization with a small number of basis functions while preserving local geometric details during completion. Quantitative and qualitative experiments demonstrate that our method outperforms the state-of-the-art methods in shape completion, detail preservation, generalization to unseen geometries, and computational cost. Code and data are at https://github.com/yinghdb/Adaptive-Local-Basis-Functions.

#### **KEYWORDS**

shape completion, deep implicit functions, adaptive local basis functions  $% \left( 1\right) =\left( 1\right) \left( 1\right)$ 

### 1 INTRODUCTION

3D Shape completion from partially scanned point clouds has been widely studied due to its importance to various applications such as automatic driving, augmented reality, and robotics. Naturally, one needs to rely on certain schemes to represent the 3D shapes we want to complete such as point clouds [Liu et al. 2020; Mazur and Lempitsky 2021; Wang et al. 2022; Xiang et al. 2021; Xie et al. 2020; Yuan et al. 2018], deformable meshes [Litany et al. 2018; Rock et al. 2015], and voxels [Choy et al. 2016; Dai et al. 2017b; Han et al. 2017; Häne et al. 2017]. On the downside, those classic representations also exhibit several intrinsic limitations. For instance, a point cloud often needs extra post-processing; the deformable template mesh may not fit the topology of target object; while processing voxel-based shapes is much more expensive. Deep implicit functions or DIFs have recently attracted more attention, which have been proven to be highly effective for the completion of 3D

<sup>\*</sup>Corresponding author.

objects [Genova et al. 2020, 2019; Mescheder et al. 2019; Park et al. 2019].

Traditionally, an implicit function is regarded as a weighted combination of multiple basis functions [Turk and O'Brien 2002; Walder et al. 2006]. Those basis functions and the associated weights can be computed with respect to an individual geometry. In the context of deep learning, a DIF encodes an input observation using a latent vector  $\mathbf{z}$  and adopts a network-based embodiment to estimate the function value  $f(\mathbf{x}, \mathbf{z})$  for a given 3D query location  $\mathbf{x}$ . Most existing DIF methods follow this modality but use different choices of implicit functions (i.e., either as a global basis function or multiple local basis functions) and weighting mechanisms, which collectively determine the expressivity of the DIF.

Early DIF methods [Genova et al. 2019; Mescheder et al. 2019; Park et al. 2019] estimate a signed distance or an occupancy function utilizing a single latent code, with a global basis function. This representation is later proven to be limited in describing complex shapes [Chibane et al. 2020; Genova et al. 2020]. Therefore, researchers switched to localized basis functions for shape completion by dividing the whole shape into multiple regions and regionconditioned latent codes. One line of research is to discretize the space into a regular voxel grid and embed the local latent codes in the voxels [Chen et al. 2021; Chibane et al. 2020]. While being able to achieve the state-of-the-art results in shape completion, the required grid resolution leads to a significant growth of computational cost. Alternatively, adaptive parameterization is sought for more compact representations (LDIF) [Genova et al. 2020], which learns to decompose a shape into a collection of overlapping regions represented by 3D Gaussian basis functions. In each region a latent code is assigned to learn a residual coefficient function for the Gaussian basis function to produce details. Despite the impressive results in shape completion, as the final shape is based on a mixture of refined Gaussians, it inherently has a limited capacity to capture full details, and therefore can still miss geometric details in difficult cases (see Fig. 3 for example).

In this paper, we argue that the specific form of basis functions should be learnable without being restricted to a certain family of functions. Such basis functions can potentially bring multiple benefits. Since the basis functions are learnable and local, they are more likely to capture local fine details due to the data-driven nature. Because the center and the shape of basis functions are learnable, fewer basis functions can achieve equal or better representation for the same geometry compared to analytical functions. For this reason, we aim to learn arbitrarily shaped basis functions so that we can complete 3D shapes with more local details and lower computational costs.

Learning basis functions for shape completion however is a challenging task. It is known that global DIFs can still miss local details [Chibane et al. 2020; Genova et al. 2020], and we would like to keep our learned functions local. Enforcing such *locality* is non-trivial during the training process. In addition, unlike learning DIFs from full observations [Chabra et al. 2020; Jiang et al. 2020; Li et al. 2022; Yao et al. 2021], we only have partial observations in shape completion tasks, which impose further difficulties. To this end, our method leverages a progressive, observed-to-unobserved process. We first encode the visible shape as a sequence of local basis functions and then use them to predict the basis functions in

the missing region in a sequence-to-sequence manner. This strategy tends to preserve the fine details for the visible area while learning the correlations between the visible and missing parts.

We address the locality problem by learning the function domain of each basis. Following the intuition that points located near a basis should be more likely to be inside its domain, we adopt the Radial Basis Function (RBF) kernels with learnable parameters to parameterize the domain. In our implementation, the RBF-based domains and DIF-based basis functions are learned jointly in an endto-end manner for a compact shape representation and preserved local details. Based on such shape formulations, we build the shape completion pipeline in two main steps. As shown in Fig. 1, the first step is to map the partial input points into a collection of local bases which encode the visible shape compactly with details. In the second step, we predict the local bases for the missing areas and refine the local bases of the visible areas by adopting Transformer encoders [Vaswani et al. 2017]. The self-attention mechanism of Transformers mimics the pairwise interaction between local bases, thus enabling the accurate sequence-to-sequence translation among visible local bases and missing local bases.

To summarize, our main contributions include the following aspects. First, we propose DIF-based local basis functions for effective and efficient shape representation, which can capture fine details with a small number of local basis functions. Second, we introduce a local-to-local shape completion pipeline, which is both efficient and geometry-rich. Experiments demonstrate that our method outperforms previous state-of-the-art methods by a large margin.

# 2 RELATED WORK

# 2.1 Deep Implicit Shape Representation

A large amount of learning-based methods has achieved promising results using implicit shape representation. With the strength of deep learning, neural networks serve as a powerful tool to fit various implicit functions, such as signed/unsigned distance fields [Park et al. 2019; Venkatesh et al. 2020], occupancy indicator functions [Chen and Zhang 2019; Mescheder et al. 2019; Peng et al. 2020], deformation functions [Deng et al. 2021; Hui et al. 2022; Paschalidou et al. 2021] or other specifically defined implicit functions [Aumentado-Armstrong et al. 2022; Chen et al. 2022a; Morreale et al. 2021].

Pioneering works such as OccNet [Mescheder et al. 2019], IM-Net [Chen and Zhang 2019], and DeepSDF [Park et al. 2019] show that many simple shapes can be represented by a latent code and the corresponding deep implicit function. However, such deep representations often fail to capture local geometries for more complex shapes. Recent works overcome the problem by focusing on the localized basis functions. Some methods divide the 3D space into voxel grids and assign each voxel with a latent code [Chabra et al. 2020; Jiang et al. 2020], while some store the latent codes in the grid points [Chen et al. 2021; Chibane et al. 2020; Peng et al. 2020] (or octree [Takikawa et al. 2021]) and interpolates them for query points within the voxel. Then the local basis functions are learned separately for each voxel and all local bases are combined for the final shape reconstruction. While state-of-the-art results can be achieved, increased resolution yields a significant growth in the number of codes, resulting in high computational costs.

In addition to the grid-based DIFs, some methods [Chen et al. 2022b; Genova et al. 2020, 2019; Hertz et al. 2022; Li et al. 2022; Tretschk et al. 2020; Xiao et al. 2022; Yao et al. 2021; Zhang et al. 2022] seek to formulate the local bases with irregular positions. SIF [Genova et al. 2019] decomposes a shape into a collection of overlapping regions represented by 3D Gaussian basis functions, and LDIF [Genova et al. 2020] learns adaptive weights with DIFs for further refinement. LGCL [Yao et al. 2021] samples a set of key points and divide the 3D space into local regions based on Euclidean distance, within which it learns a DIF for local shape representation. Some other methods [Chen et al. 2022b; Li et al. 2022] store the latent codes in multiple irregularly distributed key points, and the new code is interpolated from these codes. The advantage of these methods is that less computation is used to represent a complex shape. In our method, we follow this strategy but propose a novel formulation such that the DIF-based local bases are learned along with the combining weights so as to capture more details.

# 2.2 Shape Completion

Recently, neural networks have been used to predict the whole shape from partial input with the help of data priors. The shape completion methods can be classified according to the output representations, such as voxels, meshes, point clouds, and deep implicit functions. Voxel-based methods [Choy et al. 2016; Dai et al. 2017b; Han et al. 2017; Häne et al. 2017; Sun et al. 2022] can directly generate output data thanks to 3D convolution networks, but the memory and time costs are too high when dealing with high-resolution shape grids. And mesh-based methods [Litany et al. 2018; Rock et al. 2015] are hard to handle shapes with arbitrary topology. Therefore, a mass of methods [Liu et al. 2020; Mazur and Lempitsky 2021; Wang et al. 2022; Xiang et al. 2021; Xie et al. 2020; Yuan et al. 2018] focus on performing shape completion in point clouds which do not have those problems. But usually what we want are the mesh outputs rather than the point clouds. Other popular methods recently for shape completion is using DIF [Genova et al. 2020, 2019; Mescheder et al. 2019; Park et al. 2019]. For preserving details and using the convenient 3D convolution, most methods [Chen et al. 2021; Chibane et al. 2020; Mittal et al. 2022; Yan et al. 2022; Zheng et al. 2022] employ grid-based features to process data and express the implicit function of the output shape. ShapeFormer [Yan et al. 2022], AutoSDF [Mittal et al. 2022] and SDF-StyleGAN [Zheng et al. 2022] propose to model the shape completing as a generative task which aims to generate a series of voxelized latent codes for representing the complete shape. However, due to the large amount of latent codes and the use of 3D convolution, these methods suffer the problem of high computational cost. SIF [Genova et al. 2019] and LDIF [Genova et al. 2020] perform 2D convolutions on the input partial depth map(s) to extract the features which encode the whole shape. But their Gaussian-based local bases limit their expression capacity of arbitrary shapes. Without the above issues, our shape completion method utilizes DIF-based local bases with arbitrary shapes to preserve better details and avoid the use of 3D convolution to consume lower computation.

#### 2.3 Transformers

Transformer [Vaswani et al. 2017] is a powerful framework for sequence-to-sequence translation tasks, which has been proved useful in natural language processing [Devlin et al. 2018; Radford et al. 2019] and image processing [Carion et al. 2020; Dosovitskiy et al. 2020; Parmar et al. 2018]. Most recently, a number of methods [Mittal et al. 2022; Yan et al. 2022; Yu et al. 2021] model the shape completion as a sequence-to-sequence task by taking advantage of Transformers. PoinTr [Yu et al. 2021], as a point cloud completing method, uses the Transformer encoder-decoder architecture to predict point proxies for missing parts. ShapeFormer [Yan et al. 2022] and AutoSDF [Mittal et al. 2022], as implicit-function based methods, use the Transformer-based autoregressive model to predict the complete shape conditioned on the partial inputs. In our method, we utilize the Transformer encoder to model the dependencies among the visible and missing parts and predict the local bases for complete shape representation.

#### 3 IMPLICIT FIELD FORMULATION

A surface can be described as an SDF and represented implicitly as  $\{\mathbf{x}|f_{\phi}(\mathbf{x},\mathbf{z})=0\}$ , where  $f_{\phi}(\mathbf{x},\mathbf{z})$  can be implemented by a neural network with learnable parameters  $\phi$ . Unlike previous methods which assume a global latent code  $\mathbf{z}$  [Park et al. 2019], we represent the SDF as a weighted sum of multiple local basis functions. Each DIF  $f_{\phi}(\mathbf{x}-\mu_i,\mathbf{z}_i)$ , or  $f_i(\mathbf{x})$  for simplicity, is defined with a center position  $\mu_i$ , and a latent code  $\mathbf{z}_i$  is used for expressing the local SDF. For a given query point  $\mathbf{x} \in \mathbb{R}^3$ , its final signed distance is decided by a linear combination of N DIF-based basis functions with weights  $\alpha_i$ ,

$$sdf(\mathbf{x}) = \sum_{i \in [N]} \alpha_i f_{\phi}(\mathbf{x} - \boldsymbol{\mu}_i, \mathbf{z}_i),$$

$$\alpha_i = \frac{g(\mathbf{x} - \boldsymbol{\mu}_i, \mathbf{A}_i)}{\sum_{j \in [N]} g(\mathbf{x} - \boldsymbol{\mu}_j, \mathbf{A}_j)},$$

$$g(\mathbf{x} - \boldsymbol{\mu}_i, \mathbf{A}_i) = \exp(-||\mathbf{A}_i(\mathbf{x}_i - \boldsymbol{\mu}_i)||_2^2),$$
(1)

where  $g(\mathbf{x} - \boldsymbol{\mu}_i, \mathbf{A}_i)$ , or  $g_i(\mathbf{x})$  for simplicity, is an RBF function with learnable parameter  $\mathbf{A}_i$ .  $\mathbf{A}_i$  is a linear transform matrix, which is constructed by the product of a scaling matrix  $\mathbf{S}_i$  and a 3D rotation matrix  $\mathbf{R}_i$ . Practically,  $\mathbf{S}_i$  is mapped from a 3-dimensional vector, and  $\mathbf{R}_i$  is mapped from a 6-dimensional vector as in [Zhou et al. 2019], in which these vectors are predicted from networks directly. Eq. 1 naturally encourages sparsity through  $\alpha_i$ , which increases exponentially when  $\mathbf{x}$  is close to  $\boldsymbol{\mu}_i$ , but quickly becomes damped when  $\mathbf{x}$  moves away from  $\boldsymbol{\mu}_i$ .

For complex shapes, one may still need to use many basis functions to capture local shape variations, and  $\alpha_i$  alone is insufficient to guarantee the sparsity. To this end, we require each  $f_\phi$  only parameterizes a local neighborhood around it. As a result, only a small number of  $f_\phi$ s contribute the actual value of  $sdf(\mathbf{x})$  for a given  $\mathbf{x}$ . In our experiments, we found that only two nearest  $f_\phi$ s to  $\mathbf{x}$  will give reasonably good results. Let p and q be the indices of two largest RBFs  $g_i(\mathbf{x})$ , and  $sdf(\mathbf{x})$  becomes the linear combining the local bases with these two indices (i.e., replacing [N] with  $\{p,q\}$  in Eq. 1).

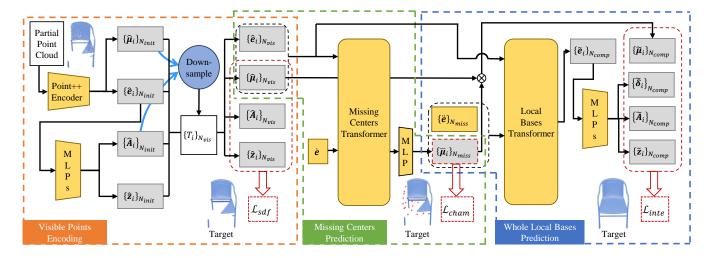


Figure 2: The overall network architecture of shape completion is divided into three parts. 'Visible Points Encoding' takes a partial point cloud as input and encodes it into  $N_{vis}$  local bases for visible regions, which is supervised by  $\mathcal{L}_{sdf}$ . 'Missing Centers Prediction' predicts  $N_{miss}$  centers of local bases for invisible regions based on the encoding and centers of visible local bases, which is supervised by  $\mathcal{L}_{cham}$ . With the above predictions, 'Whole Local Bases Prediction' predicts the final  $N_{comp}$  local bases for the complete shape, which is supervised by  $\mathcal{L}_{inte}$ . Within the figure, yellow blocks stand for learnable networks or parameters,  $\dot{e}$  and  $\ddot{e}$  and  $\ddot{e}$   $N_{miss}$  are query embeddings, and  $N_{comp}$  means a concatenate operation. Both Missing Centers Transformer and Local Bases Transformer use the Transformer encoder architecture with multiple self-attention layers. Note that  $N_{comp} = N_{vis} + N_{miss}$ .

#### 4 COMPLETION PIPELINE

As shown in Fig. 2, we first encode the input partial points into a series of local bases as the shape representation for the visible area (see Sec. 4.1), and then utilize the power of Transformers [Vaswani et al. 2017] to generate the whole local bases (see Sec. 4.2), which can further be optimized for better results (see Sec. 4.3).

# 4.1 Compact Point Cloud Encoding

As shown in the orange dashed box in Fig 2, the PointNet++ encoder [Qi et al. 2017] serves to downsample and encode input points into  $N_{init}$  center points with coordinates  $\hat{\mu}_i$  and embeddings  $\hat{\mathbf{e}}_i$ . Then Multi-Layer Perceptrons (MLPs) are used to decode the embeddings into latent codes  $\hat{\mathbf{z}}_i$  and domain parameters  $\hat{\mathbf{A}}_i$  which together with the centers  $\hat{\boldsymbol{\mu}}_i$  form the local bases for the shape representation.

In the PointNet++ encoder, key points are sampled uniformly from the input points. Such sampling is unnecessary. Ideally, regions with complex geometry should be densely sampled while regions with simple geometry should be sparsely sampled, as the domain of the local basis will be smaller for complex geometry and larger for simple geometry. Therefore, we propose an adaptive downsampling strategy based on the predicted domains of local bases after the uniform sampling in PointNet++ encoder. The detailed downsampling algorithm is shown in Alg. 1.  $g_i(\mu_j)$  can be regarded approximately as the the probability that the center of j-th local basis is inside the domain of the i-th local basis. So the higher s(j) implies the higher probability that the domain of j-th local basis can be covered by the other local bases, so we eliminate the key point with the highest s(j) sequentially. Note that after eliminating one key point, all the other s(j) needs to be updated.

### ALGORITHM 1: Domain-based Downsampling

```
Let S = \{s(j) = \sum_{i \in [N_{init}], i \neq j} g_i(\mu_j)\}_{j \in [N_{init}]};

Let T = [N_{init}] be the reserved indices;

for iter = 1 to N_{init} - N_{vis} do

Find the max s(k) in S;

Eliminate k from T, and s(k) from S;

Update s(j) = s(j) - g_k(\mu_j) for s(j) \in T;

end
```

In order to learn the compact encoding for the input partial point cloud, we perform end-to-end training for partial shape representation with the following loss:

$$\mathcal{L}_{sdf} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}, \mathbf{y} \in \mathcal{X}, \mathcal{Y}} \alpha_p |f_p(\mathbf{x}) - \mathbf{y}| + \alpha_q |f_q(\mathbf{x}) - \mathbf{y}|,$$
(2)

where X and Y stand for the set of query points and target signed distances, and  $\{p,q\}$  are as described in Sec. 3. For partial point cloud encoding, the query points X for training are sampled near the input points. With the loss function, we want each local basis function  $f_i$  to accurately learn the local SDF function. And at the same time, the learning of  $\alpha_i$  and  $f_i$  is adaptive: if  $|f_i(\mathbf{x}) - \mathbf{y}|$  is smaller, the corresponding  $\alpha_i$  will be learned to be larger to reduce  $\mathcal{L}_{sdf}$  since  $\alpha_p + \alpha_q = 1$ . This means that if a position can be more accurately described by a local basis function, it will be more likely in the local domain of the basis.

#### 4.2 Whole Local Bases Prediction

In Sec. 4.1, we get  $N_{vis}$  centers  $\hat{\mu}_i$  and embeddings  $\hat{\mathbf{e}}_i$  which can be decoded into local bases for visible shape representation. Taking these results as input, we aim to predict more centers and embeddings for complete shape representation in two steps as shown in the green and blue dashed boxes in Fig. 2.

Missing Centers Prediction. First, the  $N_{vis}$  pairs of embeddings  $\hat{\mathbf{e}}_i$  and centers  $\hat{\boldsymbol{\mu}}_i$ , as well as one learnable query embedding  $\hat{\mathbf{e}}$ , are input into Missing Centers Transformer, and the output is then fed to the MLP to get  $N_{miss}$  center points coordinates  $\bar{\boldsymbol{\mu}}_i$  for the missing area. We supervise the prediction of missing centers with the following chamfer distance loss:

$$\mathcal{L}_{cham} = \frac{1}{|\mathcal{M}|} \sum_{\mu \in \mathcal{M}} \min_{\nu \in \mathcal{N}} ||\mu - \nu||_2 + \frac{1}{|\mathcal{N}|} \sum_{\nu \in \mathcal{N}} \min_{\mu \in \mathcal{M}} ||\nu - \mu||_2,$$
(3)

where  $\mathcal{M}$  is the set of  $N_{miss}$  predicted missing center point coordinates, and  $\mathcal{N}$  is the set of target missing center point coordinates. The target centers are fetched by using Furthest Point Sampling (FPS) in the surface points of the missing region.

Whole Local Bases Prediction. Local Bases Transformer takes as input  $N_{vis}$  pairs of embeddings  $\hat{\mathbf{e}}_i$  and centers  $\hat{\boldsymbol{\mu}}_i$  for visible region, as well as  $N_{miss}$  pairs of query embeddings  $\ddot{\mathbf{e}}$  and centers  $\tilde{\boldsymbol{\mu}}_i$  for missing region, to predict the final  $N_{comp}$  embeddings  $\tilde{\mathbf{e}}_i$ . Then the output embeddings are sent to MLPs to get latent codes  $\tilde{\boldsymbol{z}}_i$  and domain parameters  $\tilde{\mathbf{A}}_i$  for complete shape representation. Note that the  $N_{miss}$  query embeddings  $\ddot{\mathbf{e}}$  share the same parameter. In addition, center offsets  $\tilde{\boldsymbol{\delta}}_i$  are predicted and added to the centers  $\tilde{\boldsymbol{\mu}}_i$  for the reason of providing optimal positions for local basis functions.

To learn the whole local bases, we use four loss functions to supervise the parameters of whole local bases, which are  $\mathcal{L}^{dom}_{sdf}$ ,  $\mathcal{L}^{euc}_{sdf}$ ,  $\mathcal{L}^{euc}_{sdf}$ ,  $\mathcal{L}^{euc}_{sdf}$ , and  $\mathcal{L}_{reg}$ .  $\mathcal{L}^{dom}_{sdf}$  has the same form and function as Eq. 2. However, only using  $\mathcal{L}^{dom}_{sdf}$  will cause problems in learning missing local basis functions. As with prior input embeddings, visible local bases are learned more quickly than the missing ones so that their domains soon expand to a wide range that even covers the missing coordinates, making the weights  $\alpha_i$  close to 0 for missing local basis functions. We address the problem by adding the loss function  $\mathcal{L}^{euc}_{sdf}$ , which simply discards the use of weights. The loss function  $\mathcal{L}^{euc}_{smooth}$  is used to make the transition between two adjacent local bases smooth, and  $\mathcal{L}_{reg}$  is used for keeping the coordinate offsets  $\tilde{\delta}_i$  small. The final loss functions are shown below,

$$\mathcal{L}_{inte} = \mathcal{L}_{sdf}^{dom} + \mathcal{L}_{sdf}^{euc} + \lambda_1 \mathcal{L}_{smooth} + \lambda_2 \mathcal{L}_{reg}, \tag{4}$$

$$\mathcal{L}_{smooth} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}, \mathbf{y} \in \mathcal{X}, \mathcal{Y}} |f_{p}(\mathbf{x}) - f_{q}(\mathbf{x})|, \tag{5}$$

$$\mathcal{L}_{sdf}^{euc} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}, \mathbf{y} \in \mathcal{X}, \mathcal{Y}} |f_k(\mathbf{x}) - \mathbf{y}|, \ \mathcal{L}_{reg} = \frac{1}{N_{comp}} \sum_{i \in [N_{comp}]} |\delta_i|,$$
(6)

where  $\lambda_1$  is 0.5, and  $\lambda_2$  is 0.01 in the first epoch of training and 0.0 in the other epochs. k is defined as the index of the  $\mu_i$  which is closest to the query point  $\mathbf{x}$  in Euclidean distance. X are the mixture of uniformly sampled points and the points sampled near the complete surface as in DeepSDF [Park et al. 2019].

# 4.3 Post Optimization

From Sec. 4.2, we get a collection of local bases defined by latent codes  $\mathbf{z}_i$ , domain parameters  $\mathbf{A}_i$  and coordinates  $\boldsymbol{\mu}_i$  to represent a complete shape. The represented shapes already have good quality (see Tab. 1 and Fig. 3), but they can be further optimized. In short, we optimize  $\mathbf{z}_i$  and  $\boldsymbol{\mu}_i$  to minimize the following loss, where  $\lambda_1, \lambda_2, \lambda_3, \lambda_4 = 1.0, 10.0, 10.0, 0.1$ .

$$\mathcal{L}_{opt} = \lambda_1 \mathcal{L}_{face} + \lambda_2 \mathcal{L}_{pos} + \lambda_3 \mathcal{L}_{adj} + \lambda_4 \mathcal{L}_{stable}, \tag{7}$$

 $\mathcal{L}_{face}$  is used to guarantee the predicted signed-distance values of input points  $\mathcal{X}_{in}$  are close to 0.  $\mathcal{L}_{pos}$  is used to ensure the signed distance value of sampled points, whose signed distance value is confirmed as positive, is larger than 0.  $\mathcal{L}_{adj}$  is used to make the signed distance value change smoothly between two adjacent local bases, and  $\mathcal{L}_{stable}$  is used to keep the optimized parameters close to the original ones. More details are in the supplementary material.

### 5 EXPERIMENT

We execute a series of experiments to evaluate our method for 3D completion. By default, we use  $N_{init} = 128$ ,  $N_{vis} = 64$  and  $N_{miss} = 32$ , and test with unoptimized results.

Dataset. The experiments are run on the ShapeNet dataset [Chang et al. 2015]. We preprocess the shapes to make them water-tight following the instructions from Occupancy Networks [Mescheder et al. 2019]. We first generate the  $224 \times 224$  depth scans of 16 random views around the objects. The input point clouds are fetched by reprojecting the 2D pixels and sampling within the 3D points using FPS. We set the number of input points M = 2048 in all our experiments. Except in the ablation study, we use the official training splits of 8 classes as [Yuan et al. 2018] for training, and for testing, we perform two kinds of experiments. One experiment is for the trained 8 classes where we use 3000 partial inputs randomly sampled from the testing splits of the 8 classes, and the other is for the unknown classes where we use 3000 partial inputs from 5 unseen classes. As for the ablation study, we use the chair class for training and testing. More details are in the supplementary material.

*Metrics*. For the evaluation we use the metrics of Intersection-over-Union (IoU) [Mescheder et al. 2019], Chamfer L2 Distance (CD) [Mescheder et al. 2019] and F-score%1 (F1) [Tatarchenko et al. 2019]. If there is no notation, we sampled 100k points on each mesh surface for the computation of CD and F1.

Baselines. We compare our method with the state-of-the-art implicit-function based 3D completion methods: IF-Net [Chibane et al. 2020], decoder-only DeepSDF [Park et al. 2019], LDIF [Genova et al. 2020], and ShapeFormer [Yan et al. 2022]. For a fair comparison, we set the number of shape elements in LDIF to be 96 which is the same as ours. For the training of ShapeFormer, in which the latent resolution is set to 16<sup>3</sup>, the scaling augmentation is used, or it will fall into overfitting quickly. We also compare our method with

Table 1: Quantitative comparison on 8 trained classes. 'Ours (opt.)' means the the optimized results of our method. 'PoinTr-NDC' and 'SnowflakeNet-NDC' convert the point cloud results of 'PoinTr' and 'SnowflakeNet' to meshes using Neural Dual Contouring (NDC) [Chen et al. 2022c]. As the reconstructed meshes are not watertight, their IoU scores are not listed.

Metric	Method	airplane	cabinet	car	chair	lamp	sofa	table	vessel	mean
	DeepSDF	0.174	0.490	0.611	0.376	0.286	0.437	0.287	0.356	0.377
	IF-Net	0.811	0.656	0.845	0.765	0.676	0.825	0.646	0.799	0.753
IoU ↑	LDIF	0.637	0.657	0.785	0.630	0.454	0.760	0.534	0.639	0.637
100	ShapeFormer	0.662	0.586	0.802	0.629	0.526	0.747	0.552	0.671	0.647
	Ours	0.853	0.778	0.867	0.830	0.763	0.872	0.766	0.851	0.822
	Ours (opt.)	0.843	0.778	0.863	0.822	0.756	0.868	0.759	0.845	0.817
	DeepSDF	9.55	2.96	2.31	3.81	8.76	6.51	7.55	8.58	6.25
	IF-Net	0.142	1.12	1.08	0.523	0.687	0.338	1.44	0.645	0.747
	LDIF	0.385	1.10	0.365	0.840	2.00	0.437	1.36	0.661	0.893
CD↓	ShapeFormer	0.219	2.03	0.423	0.945	0.899	0.625	1.42	0.320	0.860
$(\times 10^{-3})$	PoinTr-NDC	0.122	0.540	0.205	0.286	0.315	0.252	0.615	0.146	0.310
	SnowflakeNet-NDC	0.128	0.536	0.202	0.300	0.400	0.251	0.580	0.173	0.321
	Ours	0.0834	0.772	0.299	0.298	0.323	0.250	0.825	0.118	0.371
	Ours (opt.)	0.0791	0.771	0.312	0.296	0.333	0.256	0.782	0.116	0.368
	DeepSDF	0.238	0.375	0.402	0.346	0.262	0.326	0.320	0.308	0.322
	IF-Net	0.889	0.647	0.773	0.792	0.780	0.793	0.741	0.830	0.781
	LDIF	0.691	0.537	0.651	0.579	0.484	0.637	0.606	0.572	0.595
F1 ↑	ShapeFormer	0.764	0.552	0.678	0.653	0.640	0.665	0.661	0.664	0.659
F1	PoinTr-NDC	0.826	0.642	0.737	0.728	0.699	0.767	0.725	0.788	0.739
	SnowflakeNet-NDC	0.817	0.657	0.732	0.728	0.692	0.750	0.737	0.779	0.737
	Ours	0.920	0.749	0.795	0.858	0.848	0.848	0.841	0.883	0.843
	Ours (opt.)	0.918	0.778	0.789	0.855	0.845	0.847	0.844	0.883	0.841

Table 2: Quantitative comparison on unseen classes.

Method	IoU↑	CD↓ (×10 <sup>-3</sup> )	F1↑
DeepSDF	0.335	6.74	0.311
IF-Net	0.674	0.837	0.721
LDIF	0.458	1.15	0.436
ShapeFormer	0.572	1.02	0.607
PoinTr-NDC	-	0.511	0.656
SnowflakeNet-NDC	-	0.648	0.652
Ours	0.771	0.611	0.812
Ours(opt.)	0.770	0.596	0.818

PoinTr [Yu et al. 2021] and SnowflakeNet [Xiang et al. 2021] which are 3D point cloud completion methods that also use Transformer architectures. All these baseline methods use the same training data and testing data as ours. But the input formats are a little different for some methods. The input of LDIF is a scanned depth map which is the source of our partial point cloud, and the input of IF-Net is a  $128 \times 128 \times 128$  discrete occupancy grid from the input point cloud.

#### 5.1 Results on Trained Classes

As shown in Tab. 1, our method achieves much better scores than other methods. The IoU and F1 scores of our method outperform the second place by 6.2% and 6.9%. We can find that DeepSDF is very difficult to restore the shapes because of the limitation of a single

latent code. IF-Net, LDIF, and ShapeFormer achieve better scores but are still not comparable with ours. PoinTr-NDC and SnowflakeNet-NDC obtain slightly better scores in CD than ours as the target of PoinTr and SnowflakeNet is to minimize the chamfer distance, but they have lower scores in F1. An interesting phenomenon is that after optimization, our method achieves better scores in CD but worse scores in IoU and F1. It illustrates that the initial shapes directly predicted from networks are accurate enough.

We also show the qualitative comparison in Fig. 3. From the results, we can find that DeepSDF fails to repair shapes when facing complex shapes. IF-Net preserves fine details for the visible part because the input points are transferred into a high-resolution voxel grid which saves the detail information in an explicit way, but it cannot avoid the generation of noises for invisible parts. LDIF, and ShapeFormer fail to recover fine details. We can see that details such as small holes and thin curves are missing in their results. It is mainly because of the low capacity of their shape basis (i.e., Gaussians and vector quantized DIF) for capturing details. Obviously, our method owns a better capacity in preserving details for shape completion. For example, in Row 1 and Row 3, our method preserves the correct holes on the chair back and the very thin line correctly. Row 2 shows the power of our method in handling complex surfaces, and it also shows the benefit of post optimization, which is flattening the uneven surface.

Table 3: Comparison on computational complexity. Floating Point Operations (FLOPs) are counted according to a single forward of each method with one partial input and one query point.

Methods	DeepSDF	IF-Net	LDIF	ShapeFormer	Ours
Latent Resolution FLOPs	1	$128^{3}$	96	16 <sup>3</sup>	96
FLOPs	1.84 M	20.6 B	7.07 B	112.5 B	1.65 B

Table 4: Ablation study on the number of local bases for encoding.

Local Basis Number	128	96	64	32
$ \begin{array}{c} \operatorname{IoU} \uparrow \\ \operatorname{CD} (\times 10^{-3}) \downarrow \\ \operatorname{F1} \uparrow \end{array} $	0.836	0.828	0.818	0.793
CD (×10 <sup>-3</sup> ) ↓	0.303	0.330	0.358	0.369
F1 ↑	0.866	0.861	0.852	0.829

#### 5.2 Results on Unseen Classes

The results on unseen classes can represent the generalization ability of the shape completing methods on other shapes. From Tab. 2, we can find that our method shows outstanding generalization ability in completing the shapes of unseen classes. It is mainly because of our adaptive local bases and the local-to-local translation mechanism which enables predicting missing parts locally. As local parts across different classes may share a similar shape distribution (e.g. the legs of chairs and tables), our method can share the learned local-to-local translation priors from the trained classes with the unseen classes to improve the generalization ability. Again, PoinTr-NDC obtains a slightly better score in CD than ours as PoinTr aims to minimize the chamfer distance. We provide more qualitative comparisons in the supplementary materials.

### 5.3 Comparison with PatchRD

PatchRD [Sun et al. 2022] is a voxel-based completion method. While it uses 3D convolutional encoding, its output is based on retrieval rather than direct convolutional decoding. We follow PatchRD to crop out small regions to generate input data, and perform qualitative comparison with it. The results are shown in Fig. 5. Our method can produce more accurate and smoother results than PatchRD.

# 5.4 Different Levels of Completeness

In order to validate the robustness of our method, we conduct experiments on input points with different levels of completeness, as shown in Fig. 6. Our method is robust to different levels of completeness and can preserve fine details in general. But we still observe floating crossbars near the chair legs (Column 1&5 in Row 3). It may be caused by the ambiguity of crossbar existence in chairs. One interesting future work is how to avoid such ambiguities.

# 5.5 Results on Real Scans

We have investigated the completion ability of our method on real-world scans. Fig. 4 shows the results of different objects from

Table 5: Ablation study on smooth loss and domain-based downsampling.

Smooth Loss	Domain-based Downsampling	IoU↑	CD↓ (×10 <sup>-3</sup> )	F1↑
×	<b>✓</b>	0.797	0.409	0.832
~	×	0.808	0.394	0.841
<b>✓</b>	<b>✓</b>	0.808	0.364	0.844

ScanNet [Dai et al. 2017a] and scanned by ourselves using Kinect v2. Our method achieves high-quality completion results on real scan data.

# 5.6 Comparison on Computational Cost

Tab. 3 gives the comparison of computational cost among DIF-based baseline methods and ours. Although DeepSDF achieves the fewest latent codes and FLOPs, it fails in completing complex shapes as illustrated above. Among the other methods, it can be easily found that our method has a much lower computational cost, because we use a compact shape representation which requires a small number of latent codes and avoids the use of 2D and 3D convolutions which consume a lot of computations in LDIF, IF-Net, and ShapeFormer.

#### 5.7 Ablation Studies

Weight Strategies. To prove that our learnable weights  $\alpha_i$  can improve the learning of DIF-based basis functions and therefore help with preserving details, we compare the completion results with different weight strategies. The comparison results can be found in Fig. 7 (right). The learnable strategy is what we used, while the soft strategy means using the same formulation as Eq. 1 but the weights  $\alpha_i$  are unlearnable which means the domain parameters  $A_i$  and the offsets  $\delta_i$  are fixed (we set  $\mathbf{R}_i$  to be an identity matrix,  $\sigma_i = [500, 500, 500]$  and  $\delta_i = 0$ .). The hard strategy is more direct that  $\alpha_i$  is 1 if the center  $\mu_i$  is closest to the query point  $\mathbf{x}$  or 0 otherwise. Obviously, the learnable strategy can help provide more details.

Local Basis Number. In order to verify that our local bases are compact, we do an ablation study using different numbers of local bases during partial point cloud encoding. Specifically, we fix the initial local basis number ( $N_{init}$ ) to 128 but vary the downsampling number ( $N_{vis}$ ) for partial points encoding and shape completion. From Tab. 4, we can find that the scores keep close with different local basis numbers. When the local basis number decreases from 128 to 32, the scores only drop a little. That means our local bases can be learned adaptively to faithfully represent the shapes.

*Nearest Basis Number.* In our framework, the signed distance of a query point is determined by two nearest local basis functions. We have also tried to use three nearest bases, and the IoU score decreases by 0.2% and F1 score remains the same. Therefore, two nearest local bases are sufficient to express the signed distance.

Loss Functions. We conduct ablation studies on different loss functions. As shown in Row 1 and 3 in Tab. 5, by using the smooth loss, the IoU score increases by 1.1% and F1 score increases by

1.2%. As shown in Fig 7 (left), without  $\mathcal{L}_{sdf}^{euc}$ , the prediction of missing parts degrades. Removing  $\mathcal{L}_{reg}$  makes the training unable to converge.

Domain-based Downsampling. We validate the domain-based downsampling by replacing it with uniform downsampling. As shown in Row 2 and 3 in Tab. 5, domain-based downsampling obtains the best scores

### 6 CONCLUSION AND LIMITATION

We have proposed a new shape completion method based on implicit function with adaptive local basis functions. These local basis functions provide an effective and efficient compact representation for complex shapes, preserving geometric details while reducing computational costs. One limitation of our method is that the whole local bases prediction does not guarantee to recover the target topology of the shape, shown in (Fig. 1 (d)) where the cross-bar is not tightly connected with one leg of the chair. This happens when the connectivity of the local bases in the missing region is different from the ground-truth. In the future, we will look into augmenting our method with a high-level graph-based representation that focuses on the global topology of the shape.

#### **ACKNOWLEDGMENTS**

The authors would like to thank reviewers for their insightful comments. This work was supported by NSF China (62227806), the XPLORER PRIZE, and the 100 Talents Program of Zhejiang University.

#### **REFERENCES**

- Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan D Jepson. 2022. Representing 3D Shapes with Probabilistic Directed Distance Fields. In CVPR. 19343–19354.
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In ECCV. Springer, 213–229.
- Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Love-grove, and Richard Newcombe. 2020. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In ECCV. Springer, 608–625.
- Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015).
- Chao Chen, Yu-Shen Liu, and Zhizhong Han. 2022b. Latent partition implicit with surface codes for 3D representation. arXiv preprint arXiv:2207.08631 (2022).
- Weikai Chen, Cheng Lin, Weiyang Li, and Bo Yang. 2022a. 3PSDF: Three-Pole Signed Distance Function for Learning Surfaces with Arbitrary Topologies. In CVPR. 18522– 18531.
- Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022c. Neural dual contouring. ACM Transactions on Graphics (TOG) 41, 4 (2022), 1–13.
- Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In CVPR. 5939–5948.
- Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. 2021. Multiresolution deep implicit functions for 3d shape representation. In ICCV. 13087–13096
- Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. 2020. Implicit functions in feature space for 3d shape reconstruction and completion. In CVPR. 6970–6981.
- Christopher B Choy, Danfei Xu, Jun Young Gwak, Kevin Chen, and Silvio Savarese. 2016. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In ECCV. Springer, 628–644.
- Angel Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017a. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In Proc. Computer Vision and Pattern Recognition (CVPR), IEEE.
- Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. 2017b. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In CVPR. 5868–5877.

- Yu Deng, Jiaolong Yang, and Xin Tong. 2021. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In CVPR. 10286–10296.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020).
- Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. 2020. Local deep implicit functions for 3d shape. In CVPR. 4857–4866.
- Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019. Learning shape templates with structured implicit functions. In ICCV. 7154–7164.
- Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. 2017. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In ICCV. 85–93.
- Christian Häne, Shubham Tulsiani, and Jitendra Malik. 2017. Hierarchical surface prediction for 3d object reconstruction. In 3DV. IEEE, 412–420.
- Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. arXiv preprint arXiv:2201.13168 (2022).
- Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. 2022. Neural Template: Topology-Aware Reconstruction and Disentangled Generation of 3D Meshes. In CVPR. 18572– 18582.
- Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. 2020. Local implicit grid representations for 3d scenes. In CVPR. 6001–6010.
- Tianyang Li, Xin Wen, Yu-Shen Liu, Hua Su, and Zhizhong Han. 2022. Learning deep implicit functions for 3D shapes with dynamic code clouds. In CVPR. 12840–12850.
- Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. 2018. Deformable shape completion with graph convolutional autoencoders. In CVPR. 1886–1895.
- Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. 2020. Morphing and sampling network for dense point cloud completion. In AAAI, Vol. 34. 11596–11603.
- Kirill Mazur and Victor Lempitsky. 2021. Cloud transformers: A universal approach to point cloud processing tasks. In ICCV. 10715–10724.
- Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In CVPR. 4460–4470.
- Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. 2022. Autosdf: Shape priors for 3d completion, reconstruction and generation. In CVPR. 306–315.
- Luca Morreale, Noam Aigerman, Vladimir G Kim, and Niloy J Mitra. 2021. Neural surface maps. In CVPR. 4639–4648.
- Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In CVPR. 165–174.
- Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *ICML*. PMLR, 4055–4064.
- Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. 2021. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In CVPR. 3204–3215.
- Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In ECCV. Springer, 523–540.
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS 30 (2017).
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog 1, 8 (2019), 9.
- Jason Rock, Tanmay Gupta, Justin Thorsen, Jun Young Gwak, Daeyun Shin, and Derek Hoiem. 2015. Completing 3d object shape from one depth image. In CVPR. 2484– 2403
- Bo Sun, Vladimir G Kim, Noam Aigerman, Qixing Huang, and Siddhartha Chaudhuri. 2022. PatchRD: Detail-Preserving Shape Completion by Learning Patch Retrieval and Deformation. arXiv preprint arXiv:2207.11790 (2022).
- Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In CVPR. 11358–11367
- Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. 2019. What do single-view 3d reconstruction networks learn? In CVPR. 3405–3414.
- Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. 2020. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In ECCV. Springer, 293–309.
- Greg Turk and James F O'Brien. 2002. Modelling with implicit surfaces that interpolate. TOG 21, 4 (2002), 855–873.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. NIPS

- 30 (2017).
- Rahul Venkatesh, Sarthak Sharma, Aurobrata Ghosh, Laszlo Jeni, and Maneesh Singh. 2020. Dude: Deep unsigned distance embeddings for hi-fidelity representation of complex 3d surfaces. arXiv preprint arXiv:2011.02570 (2020).
- Christian Walder, Bernhard Schölkopf, and Olivier Chapelle. 2006. Implicit surface modelling with a globally regularised basis of compact support. In CGF, Vol. 25. Amsterdam: North Holland, 1982-, 635–644.
- Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. 2022. Learning Local Displacements for Point Cloud Completion. In CVPR. 1568–1577.
- Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. 2021. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *ICCV*. 5499–5509.
- Yuting Xiao, Jiale Xu, and Shenghua Gao. 2022. TaylorImNet for Fast 3D Shape Reconstruction Based on Implicit Surface Function. arXiv preprint arXiv:2201.06845 (2022).
- Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. 2020. Grnet: Gridding residual network for dense point cloud completion. In ECCV. Springer, 365–381.
- Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2022. Shapeformer: Transformer-based shape completion via sparse representation. In *CVPR*. 6239–6249.
- Shun Yao, Fei Yang, Yongmei Cheng, and Mikhail G Mozerov. 2021. 3d shapes local geometry codes learning with sdf. In ICCV. 2110–2117.
- Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. 2021. Pointr: Diverse point cloud completion with geometry-aware transformers. In ICCV. 12498– 12507
- Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. 2018. Pcn: Point completion network. In *3DV*. IEEE, 728–737.
- Biao Zhang, Matthias Nießner, and Peter Wonka. 2022. 3DILG: Irregular Latent Grids for 3D Generative Modeling. arXiv preprint arXiv:2205.13914 (2022).
- X Zheng, Yang Liu, P Wang, and Xin Tong. 2022. SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation. In CGF, Vol. 41. Wiley Online Library, 52–63.
- Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In CVPR. 5745–5753.

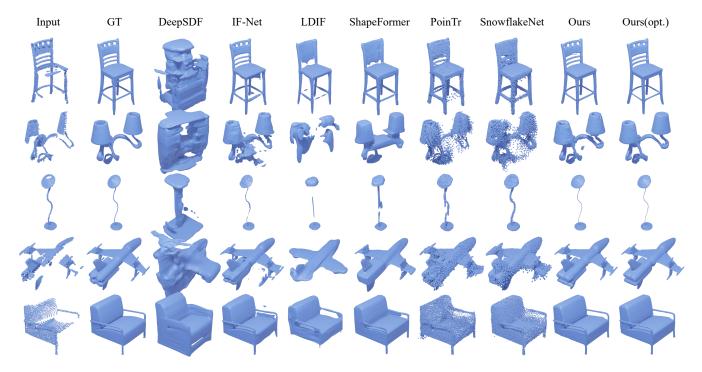


Figure 3: Qualitative comparison with other methods. Ours (opt.)' means the optimized results of our method.

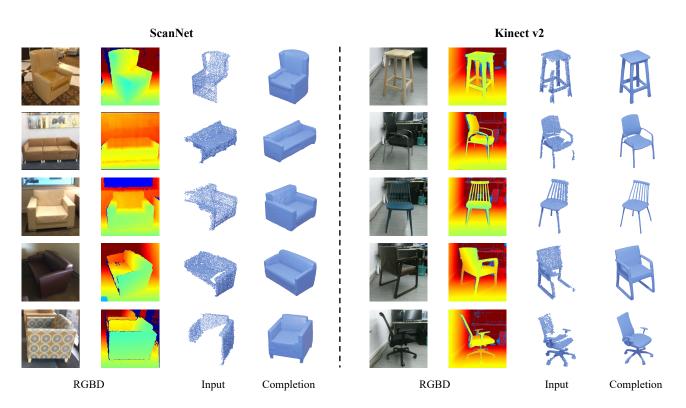


Figure 4: Shape completion results on real-world scans.



Figure 5: Qualitative comparison with PatchRD. For each pair, the left is the input and the right is the output.



Figure 6: Qualitative results on point clouds with different levels of completeness. For each pair, the left is the input and the right is the output.



Figure 7: Ablation study on  $\mathcal{L}_{sdf}^{euc}$  (left) and different weight strategies (right).

#### A IMPLEMENTATION DETAILS

### A.1 Networks Details

In Tab. 6, we show the detailed network architecture of the Point-Net++ encoder [Qi et al. 2017] and head networks. For the Point-Net++ encoder, the sampling layer performs farthest point sampling (FPS) to the input points to choose a subset of input points, and the grouping layer uses *K* nearest neighbors (KNN) to find a fixed number of neighboring points. The hierarchical sampling and grouping operations in the PointNet++ encoder guarantee that the output key point embeddings focus on the local region features.

Given N input embeddings with feature size 256, four head networks are used to decode them into the required parameters for local bases, which are latent codes  $\mathbf{z}_i$ , domain parameters  $\mathbf{A}_i$ , and offsets  $\boldsymbol{\delta}_i$ . Note that the domain parameter  $\mathbf{A}_i$  consists of sigma  $\boldsymbol{\sigma}_i$  and rotation  $\mathbf{R}_i$ , so there are two head networks to predict these two parameters separately. Also, the head networks used in the partial point cloud encoding and the whole local bases prediction share the same architecture but have different weights. In addition to these four head networks, the coordinate head network takes as input one 256-dimension embedding and predicts  $N_{miss} \times 3$  dimension features that can be reshaped as  $N_{miss}$  coordinates for the missing region.

For the implicit decoder, we follow the same architecture as DeepSDF [Park et al. 2019]. Both Missing Centers Transformer and Local Bases Transformer consist of several self-attention blocks as the encoder in [Vaswani et al. 2017]. We use 3 and 6 blocks for Missing Centers Transformer and Local Bases Transformer, respectively. All of these blocks have 8 heads self-attention, and the embedding dimension is 256. All input coordinates  $\mu_i$  are mapped into 256-dimension positional embeddings by 2 linear layers.

# A.2 Training Details

Directly training the whole model makes it hard to converge, so we adopt a two-phase training. The first phase is to train the networks and parameters used in partial point cloud encoding and missing centers prediction simultaneously. The used loss function is  $\mathcal{L}_{sdf}$  +  $\lambda \mathcal{L}_{cham}$  with  $\lambda = 0.01$ . It is worth mentioning that during training, the embeddings  $\mathbf{e}_i$  and coordinates  $\boldsymbol{\mu}_i$  after downsampling will be inputted into Missing Centers Transformer in forward propagation, but their gradients will not be returned from the Transformer in backward propagation in order to avoid the influence of missing centers prediction on partial point cloud encoding. After the first phase of training, those trained networks and parameters are fixed, and the next phase is to train the query embedding **\vec{e}** and Local Bases Transformer, as well as the head networks and implicit decoder used in whole local bases prediction, with the loss function  $\mathcal{L}_{inte}$ . Before the start of the second phase training, the weight of the untrained implicit decoder and head networks can be initialized with the ones trained in the first phase who share the same architectures.

In the both training phases, we utilize Adam optimizer and multistep learning rate scheduler to train the networks and parameters with the batch size 16 and the initial learning rate 0.0005. In the main experiments, during each phase we perform training for 20 epochs and reduce the learning rate by 50% in the epochs of {10, 15, 18}. While in the ablation study, during each phase we perform training for 30 epochs and reduce the learning rate by 50% in the epochs

of {15, 22, 27}. More implementation details can be found in the supplementary materials.

In the experiments, the trained classes for ShapeNet [Chang et al. 2015] include airplane, cabinet, car, chair, lamp, sofa, table, and vessel, while the unseen classes include telephone, loudspeaker, display, bench, and rifle.

# A.3 Post Optimization

During inference, after predicting the parameters of local bases from networks, we can further optimize  $\mathbf{z}_i$  and  $\boldsymbol{\mu}_i$  to get a more accurate and smoother shape by minimizing the following loss. We utilize Adam optimizer to perform optimization for 1000 iterations with a learning rate of 0.001.

$$\mathcal{L}_{opt} = \lambda_1 \mathcal{L}_{face} + \lambda_2 \mathcal{L}_{pos} + \lambda_3 \mathcal{L}_{adj} + \lambda_4 \mathcal{L}_{stable}. \tag{8}$$

The function of  $\mathcal{L}_{face}$  shown below is to guarantee the predicted signed-distance values of input points  $\mathcal{X}_{in}$  are close to 0. We use a margin  $\epsilon$  to relax the loss that only the items with  $|sdf(\mathbf{x})|$  larger than  $\epsilon$  will be punished.

$$\mathcal{L}_{face} = \frac{1}{|X_{in}|} \sum_{\mathbf{x} \in X_{in}} \min(|sdf(\mathbf{x})| - \epsilon, 0.0)^2.$$
 (9)

The function of  $\mathcal{L}_{pos}$  shown below is to ensure the signed distance value of positive points  $\mathcal{X}_{pos}$  is larger than 0. Similar to Eq. 9, a margin  $\epsilon$  is also used to relax the loss. The positive points  $\mathcal{X}_{pos}$  are the points that are sampled around the key point coordinates and whose signed distance value is confirmed as positive. For example, if the sampled points are located between the camera and the observed depth map or outside silhouette of the depth map, they will be regarded as positive.

$$\mathcal{L}_{pos} = \frac{1}{|\mathcal{X}_{pos}|} \sum_{\mathbf{x} \in \mathcal{X}_{pos}} min(-sdf(\mathbf{x}) - \epsilon, 0.0)^2.$$
 (10)

The function of  $\mathcal{L}_{adj}$  shown below is to make the signed distance value change smoothly between two adjacent local bases. In the function,  $\mathcal{X}_{samp}$  are the points sampled around the key point coordinates. p and q are the same as the ones introduced in Sec. 3 of the main paper.  $\omega_1(\mathbf{x})$  and  $\omega_2(\mathbf{x})$  are the weight factors, that  $\omega_1(\mathbf{x})$  gets larger when  $\mathbf{x}$  is close to the surface and  $\omega_2(\mathbf{x})$  gets larger when  $g_p(\mathbf{x})$  and  $g_q(\mathbf{x})$  are close. In practice, we set  $a_1 = 10000$  and  $a_2 = 1000$ .

$$\mathcal{L}_{adj} = \frac{1}{|\mathcal{X}_{samp}|} \sum_{\mathbf{x} \in \mathcal{X}_{samp}} \omega_1(\mathbf{x}) \omega_2(\mathbf{x}) (f_{\phi,p}(\mathbf{x}) - f_{\phi,q}(\mathbf{x}))^2,$$
where  $\omega_1(\mathbf{x}) = \exp(-a_1 \min(f_{\phi,p}(\mathbf{x}), f_{\phi,q}(\mathbf{x}))^2)$ 

$$\omega_2(\mathbf{x}) = \exp(-a_2(g_p(\mathbf{x}) - g_q(\mathbf{x}))^2).$$
(11)

The last loss  $\mathcal{L}_{stable}$  shown below is to keep the optimized parameters close to the original latent codes  $\hat{\mu}_i$  and coordinates  $\hat{\mathbf{z}}_i$ .

$$\mathcal{L}_{stable} = \frac{1}{N_{comp}} \sum_{i \in [N_{comp}]} ||\mu_i - \hat{\mu}_i||^2 + ||\mathbf{z}_i - \hat{\mathbf{z}}_i||^2.$$
(12)

Table 6: The detailed architecture information of our method. N is the number of local bases. For linear layer (LinearLayer and LinearLayer\*), 'i' and 'o' stand for input channel size and output channel size. For convolutional layer (ConvLayer), the 'k', 's', and 'p' stand for kernel size, stride, and padding. For grouping layer (Grouping), 'n' stands for the number of points in the neighborhood of centroid points, and the input consists of two point sets, which are the point set with features before sampling (left) and the coordinates of a set of centroids (right). Also 'LinearLayer' denotes a single linear layer, while 'LinearLayer\*' denotes the composition of linear layer + layer normalization + leaky ReLU. 'ConvLayer' denotes the composition of convolutional layer + group normalization + leaky ReLU.

Layer Name	Notes	Input Size	Output Size
PointNet++ Encoder			
LinearLayer	i3o16	$2048 \times 3$	$2048 \times 16$
Grouping	n16	$2048 \times (3 + 16)/2048 \times 3$	$2048 \times 16 \times 19$
ConvLayer	k1s0p0	$2048 \times 16 \times 19$	$2048 \times 16 \times 32$
ConvLayer	k1s0p0	$2048 \times 16 \times 32$	$2048 \times 16 \times 64$
ConvLayer	k1s0p0	$2048 \times 16 \times 64$	$2048 \times 16 \times 128$
MaxPooling	-	$2048 \times 16 \times 128$	$2048 \times 128$
Sampling		$2048 \times 3$	$512 \times 3$
Grouping	n16	$2048 \times (3 + 128)/512 \times 3$	$512 \times 16 \times 131$
ConvLayer	k1s0p0	$512 \times 16 \times 131$	$512 \times 16 \times 128$
ConvLayer	k1s0p0	$512 \times 16 \times 128$	$512 \times 16 \times 128$
ConvLayer	k1s0p0	$512 \times 16 \times 128$	$512 \times 16 \times 128$
MaxPooling	•	$512 \times 16 \times 128$	$512 \times 128$
Sampling		$512 \times 3$	$128 \times 3$
Grouping	n16	$512 \times (3 + 128)/128 \times 3$	$128 \times 16 \times 131$
ConvLayer	k1s0p0	$128 \times 16 \times 131$	$128 \times 16 \times 128$
ConvLayer	k1s0p0	$128 \times 16 \times 128$	$128 \times 16 \times 128$
ConvLayer	k1s0p0	$128 \times 16 \times 128$	$128 \times 16 \times 128$
MaxPooling	1	$128 \times 16 \times 128$	$128 \times 128$
LinearLayer	i128o256	$2048 \times 256$	$128 \times 256$
Latent Code Head			
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer	i256o256	$N \times 256$	$N \times 256$
Scaling Head			
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer	i256o3	$N \times 256$	$N \times 3$
Rotation Head			
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer	i256o6	$N \times 256$	$N \times 6$
Offset Head			
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer*	i256o256	$N \times 256$	$N \times 256$
LinearLayer	i256o3	$N \times 256$	$N \times 3$
Coordinates Head			
LinearLayer*	i256o512	256	512
LinearLayer*	i256o512	512	512
LinearLayer	i512o96	512	$N_{miss} \times 3$

Table 7: Comparison with PoinTr and SnowflakeNet.

	Trained		Unseen	
	$CD \downarrow (\times 10^{-3})$	F1 ↑	$CD \downarrow (\times 10^{-3})$	F1 ↑
PoinTr	0.279	0.585	0.405	0.558
SnowflakeNet	0.252	0.628	0.374	0.633
Ours	0.447	0.637	0.678	0.640

Table 8: Shape reconstruction results. 'Lat. Res.' means latent resolution, and 'ShapeF.' means ShapeFormer.

Method	Lat. Res.	IoU↑	$CD \downarrow (\times 10^{-4})$	F1↑
DeepSDF	1 1	0.560	18.5	0.425
IFNet	128 <sup>3</sup>	0.915	0.312	0.985
LDIF	32	0.769	3.25	0.834
ShapeF.	16 <sup>3</sup>	0.814	0.641	0.884
ShapeF.	$32^{3}$	0.844	0.671	0.928
3DILG	128	0.864	0.782	0.920
3DILG	256	0.895	0.485	0.956
3DILG	512	0.920	0.374	0.978
Ours	32	0.892	0.804	0.936
Ours	64	0.928	0.437	0.968
Ours	128	0.948	0.348	0.981
Ours	256	0.957	0.274	0.987

#### **B** EXPERIMENTS

# B.1 Comparison with PoinTr and SnowflakeNet

We also compare our method directly with the point-based completion method PoinTr and SnowflakeNet [Xiang et al. 2021] which also adopts a Transformer architecture. As the number of output points is fixed to be 8192 in PoinTr and SnowflakeNet, we sample the same number of points on the meshes of our method for computing CD and F1. The results are listed in Tab. 7. We can find that the PoinTr and SnowflakeNet have better scores in CD, and our method achieves better scores in F1. That means the whole point cloud generated by PoinTr and SnowflakeNet obtain lower average errors in L2 distance, but there are numbers of outliers that reduce the F1 score. Our methods can produce more stable results with fewer outliers. It can also be proven in the visualization results shown in the main paper. Notably, for both methods, there are small gaps between the scores on trained classes and unseen classes. It further proves that the generalization ability of shape completion can be improved by building local-to-local translation modules among local shape representations.

# **B.2** Shape Reconstruction on Complete Input Points

To show the efficiency of our shape representation method, we perform shape reconstruction using complete point clouds and compared the results with other DIF-based methods. For this task, we only use the 'Visible Points Encoding' module and use the loss

function  $\mathcal{L}_{sdf} + \lambda \mathcal{L}_{smooth}$  in which  $\lambda = 0.5$ . Here, our post optimization is not used. All the experiments are run on the chair class of ShapeNet dataset [Chang et al. 2015], and the results are shown in Tab. 8. The methods for comparison include DeepSDF [Park et al. 2019], IF-Net [Chibane et al. 2020], LDIF [Genova et al. 2020], ShapeFormer [Yan et al. 2022], and 3DILG [Zhang et al. 2022], which also adopt deep implicit functions and auto-encoder architectures. Our method achieves the best scores when using 256 local bases. Notably, our method can achieve similar scores with much fewer bases. For example, our method with 128 local bases can achieve similar scores as IFNet with 128 $^3$  resolution, and our method with 64 local bases can achieve similar scores as 3DILG with 512 latent features. Even with 32 local bases, our method outperforms LDIF and ShapeFormer in most metrics.

# **B.3** More Qualitative Results

We show more qualitative comparisons on the trained and unseen classes in Fig. 8 and Fig. 9. We can find that our completion results clearly outperform the ones from other methods. DeepSDF [Park et al. 2019] fails to generate reasonable shapes when meeting complex cases. LDIF [Genova et al. 2020] and ShapeFormer [Yan et al. 2022] are difficult to preserve details. Also, ShapeFormer may generate shapes that do not match the input points. PoinTr [Yu et al. 2021] may generate outlier points. IF-Net [Chibane et al. 2020] can preserve fine details for visible parts but is difficult to predict smooth results for invisible parts. In contrast, our method can predict reasonable shapes for invisible parts while preserving fine details. Fig. 9 demonstrates the strong generalization ability of our method.

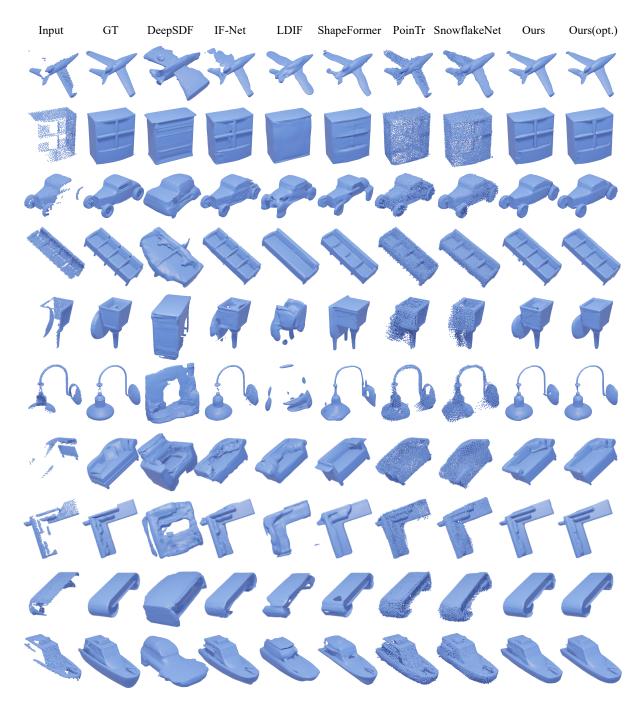


Figure 8: More qualitative comparison with other methods on trained classes. Ours (opt.)' means the optimized results of our method.

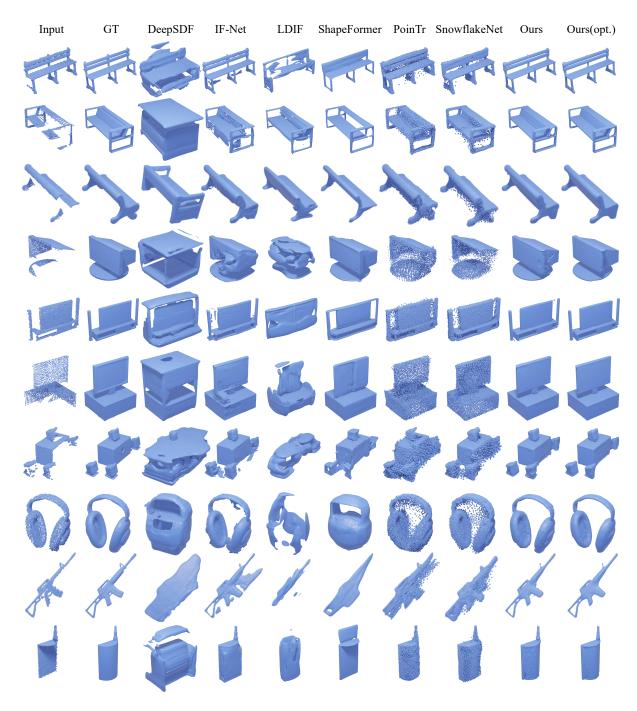


Figure 9: More qualitative comparison with other methods on unseen classes. Ours (opt.)' means the optimized results of our method.