Learning Deep Implicit Functions for 3D Shapes with Dynamic Code Clouds

Anonymous CVPR submission

Paper ID 1714

Abstract

Deep Implicit Function (DIF) has recently gained much popularity as an efficient 3D shape representation. To capture fine geometry details, current mainstream methods usually learn DIF using local latent codes, which discretize the space into a regular 3D grid (or octree) and store the local codes in grid points (or octree nodes). Given a query point, the local feature is computed by interpolating its neighboring local codes with their positions. However, the local codes are constrained at discrete and regular positions like grid points, which makes the code positions difficult to be optimized and limits their representation ability. To solve this problem, we propose a novel method to learn DIF for 3D shapes with Dynamic Code Cloud, named DCC-DIF. Our method explicitly associates local codes with learnable position vectors, and the position vectors are continuous and can be dynamically optimized instead of using discrete and regular grid points, which improves the representation ability. In addition, we propose a novel code position loss to optimize the positions of local codes, which heuristically guides more local codes to be distributed around complex geometric details. In contrast to previous methods, our DCC-DIF can represent 3D shapes more efficiently with a small amount of local codes, and significantly improve the shape reconstruction quality. Experiments demonstrate that our DCC-DIF can achieve the state-of-the-art performance over previous methods.

1. Introduction

Learning 3D shape representation is an important and basic prerequisite for many downstream applications in 3D computer vision. Explicit 3D representations such as meshes, voxels and point clouds have been widely used in various tasks [15, 20, 25, 26] for a long time. Recently, deep implicit function (DIF) [3, 21, 23, 32] has received much more popularity as an efficient 3D shape representation, which learns the latent code of 3D shapes by predicting a signed distance or inside/outside for each query point with a decoder. Different from explicit 3D representations, DIF can be stored compactly and learn shape priors by the network. Besides, it is simple and natural to use DIF in learning-based tasks because of its differentiable ability.

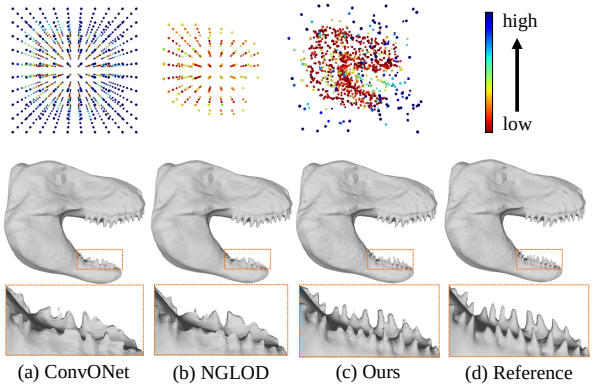


Figure 1. Illustration comparison between our method and other methods. In (a), we select the grid-based DIF (ConvONet [24]). In (b), we show the octree-based DIF (NGLOD [28]). (c) is our DCC-DIF. (d) is Reference. The first row shows the code positions of different methods, where the warmer color indicates the local codes are closer to the surface. Compared with other methods, in which code positions are discrete and regular, our code positions are continuous and more flexible. The second row shows the reconstruction results, where our method can reconstruct highly detailed geometry of complex shapes, like teeth.

Previous DIF approaches [3, 21, 23] encode the entire 3D shape into a single global latent code through the auto-encoder or auto-decoder [29] framework, which leads to information loss of local regions. As a result, those approaches can not capture local geometry details well and struggle to represent complex shapes. To address this problem, some methods [1, 14] divide the 3D space into small local volumes and assign each volume with a latent code. Then each local volume is reconstructed separately and all volumes are combined together to get the final reconstruction. Since small local volumes contain simple shapes and common patterns are shared among volumes, these methods can represent 3D shapes with high accuracy and generalize to different shapes. Similarly, some approaches [10, 11] de-

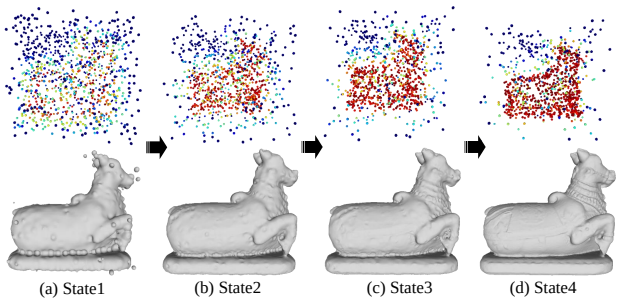


Figure 2. **Illustration of moving code positions during optimization.** Our code positions are dynamically updated during optimization, which makes 3D shape representation more efficiently. We typically show four states during optimization, where the first and second rows display code positions and reconstructions, respectively. Initial and final states are shown in (a) and (d), respectively, while (b) and (c) show two intermediate states.

compose 3D shapes into local parts, each of which is associated with a latent code for learning local details. On the other hand, more recent methods discretize the space into a regular 3D grid [4, 5, 24] (or octree) [19, 28] and store the local codes in the grid points (or octree nodes). Given a query point in 3D space, the local feature is computed by interpolating its neighboring local codes with their positional weights. Next, the local feature is fed into a decoder to predict a signed distance or inside/outside. As the resolution of grids or depth of octree increases, these methods achieve the state-of-the-art results in several shape reconstruction tasks. However, the increase of resolution or depth will result in a significant growth of the number of local codes. Moreover, the local codes in these methods are constrained at discrete and regular positions like grid points or octree nodes, which makes the code positions difficult to be optimized [4, 5, 19, 24, 28] and limits the representation ability.

To address the above-mentioned problems, we propose a novel method to learn DIF for 3D shape with Dynamic Code Cloud, named DCC-DIF. Specifically, we represent a 3D shape with a set of local latent codes, each of which is explicitly associated with a learnable position vector. Using these position vectors, the local feature for a query point is computed by interpolating local codes with their positional weights, which are computed using the distances relative to this query point. In contrast with previous local DIF methods [4, 5, 24, 28], which store the local codes in discrete and regular grids, the positions of local codes used in our method are continuous and flexible. Specially, our code positions can be dynamically optimized, where the position vectors is learnable and can be updated by back-propagation and gradient descent. Therefore, we name our method *Dynamic Code Cloud* (DCC), as shown in Fig. 1 and Fig. 2. In addition, we design a novel *Code Position* (CP) loss to optimize the positions of local codes, where more local codes are heuristically guided to distribute around complex geometric details. With the help of CP loss, our method can represent 3D shapes more efficiently with a small amount of local codes. As a result, when using the same number of local codes as previous methods, our method can achieve better results and reconstruct highly detailed geometry of 3D shapes. Our main contributions can be summarized as follows.

- We propose a novel DCC-DIF to learn deep implicit function of 3D shapes. Compared with previous methods which limit the local codes at discrete and regular grid points, the code positions in DCC-DIF are continuous and can be dynamically optimized, which improves the representation ability.
- We further propose a novel code position (CP) loss to optimize the positions of local codes, so that more local codes are distributed around complex geometric details. With the help of CP loss, our DCC-DIF can represent 3D shapes with higher quality and efficiency.
- Compared to previous methods, our method can achieve better accuracy with fewer number of local codes when reconstructing highly detailed geometry of 3D shapes. Experiments demonstrate our DCC-DIF can achieve the state-of-the-art results.

2. Related Work

The recent emerged implicit representation research has drawn a growing attention in 3D computer vision. Compared with the previous explicit representation based methods (e.g. voxel [20], mesh [15] and point cloud [25, 26]), the implicit methods can represent 3D shapes at arbitrary resolution. In this paper, we take the advantage of implicit 3D representation and focus on the task of reconstructing high-quality 3D signed distance functions (SDF). Relevant work of this area can be roughly divided into two categories, which are global DIF methods and the local DIF methods.

Global DIF methods. For the previous global DIF methods, a common practice is to take the benefit from the traditional implicit representation methods, and integrate them into the deep learning based framework. Typical method like DeepSDF [23] implicitly represents a 3D shape by its zero-level set. It optimizes a global latent code for each 3D shape, and predicts signed distances to the shape surface for sampled points using a decoder. On the other hand, OccNet [21] represents the surface of a shape by decision boundary of a deep neural network. It leverages an auto-encoder framework to predict inside/outside values for sampled points in 3D space. Following pioneers, some recent emerged methods have further improved the frontier of DIF

research. For example, Duan et al. [9] learn DIF by a curriculum strategy, and Zheng et al. [33] develop the deformation based method to predict DIF from shape templates. However, the problem is that these methods are still hard to preserve the details of local surfaces, due to the fixed dimensionality of single global code.

Local DIF methods. To overcome the limitation of global DIF methods, the local DIF methods have been developed to learn 3D shapes at more detailed geometric level. For example, LIG and DeepLS [1, 14] divide shapes/scenes into volumes, where each volume is independently reconstructed with an assigned latent code. After that, all volumes are combined together to get the final reconstruction. SIF, LDIF and PatchNets [10, 11, 31] decompose shapes into local patches and represent each patch using a latent code. More recently, IMLSNets [17] adapts implicit moving least squares surface formulation for learning based method. ConvONet [24] builds 3D grids or 2D grids on each axes plane and stores a latent code in each grid point. Then given a query point in 3D space, the positions of this point and its neighboring grid points are leveraged to interpolate the stored latent codes into a vector. IF-Nets [5] constructs hierarchical latent grids with different resolutions to capture local geometric information of different scales. Similarly, MDIF [4] also constructs hierarchical latent grids. Moreover, it sets top level latent grids to be a global latent code and connects latent codes between different levels by transposed convolutions [16] and concatenations, which enables it to do global operations like completion. However, a high grid resolution is required for latent grids based methods to achieve good results, which leads to cubic growth of number of latent codes. NGLOD [28] leverages the sparse octree instead of uniform grids to reduce the number of latent codes, and achieves state-of-the-art reconstruction accuracy. ACORN [19] also adopts the octree, and the structure of octree can be adjusted during optimization. However, this step is non-differentiable and needs to solve an integer linear program problem. These grids or octree-based methods constrain that latent codes are located at discrete and regular positions like grid points or octree nodes. And code positions are static [4,5,24,28] or non-trivial to be optimized [19]. In contrast, positions of latent codes in our method are flexible and continuous. And we dynamically optimize code positions by back-propagation and gradient descent, which is more efficient than solving an integer linear program problem.

3. Method

Our goal is to design a flexible 3D shape representation which can efficiently fit a single shape or reconstruct 3D datasets with high quality. Fig. 3 illustrates the overall architecture of our method and differences among global, local and our method. To represent a 3D shape, as shown in Fig. 3(a), global methods leverage a single global latent code, and optimize the latent code by minimizing errors between outputs and the ground truth. Local methods replace the global latent code with a set of local latent codes, as shown in Fig. 3(b). In our method, as shown in Fig. 3(c), we explicitly assign a position vector to each local code, which indicates the (x, y, z) coordinates of corresponding local code in 3D space, and a novel CP loss is further proposed to optimize position vectors. In this section, we firstly introduce background knowledge about neural signed distance functions (SDF) in Sec. 3.1. Then the design of our method is explained in detail in Sec. 3.2. Next, we present our novel CP loss in Sec. 3.3. And lastly we describe the training process in Sec. 3.4.

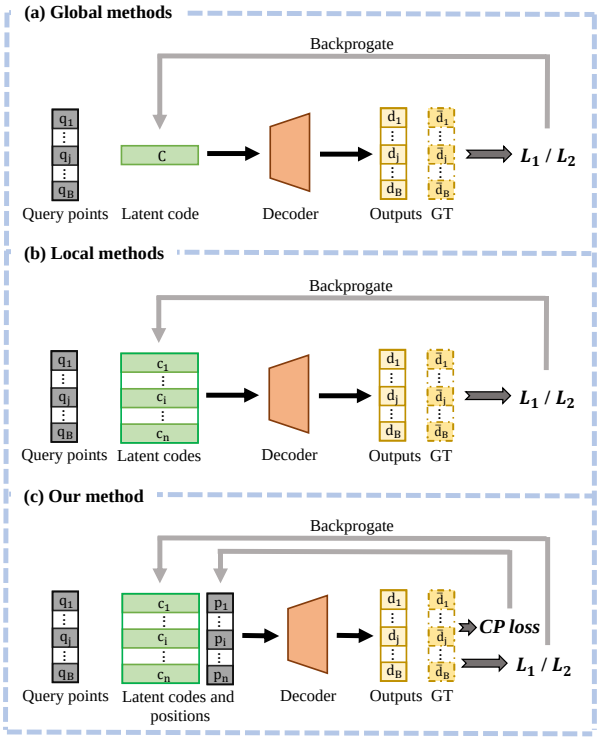


Figure 3. Illustration comparison between architecture of our method and other methods. We show the overall architecture of global DIF methods in (a), local DIF methods in (b), and our method in (c).

3.1. Deep Implicit Function

There are different approaches for deep implicit functions to represent surfaces. Mainstream approaches include occupancy functions [21] and signed distance functions (SDF) [23]. In this paper, we follow the paradigm of SDF. SDF can be formulated as $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, and $d = f(\mathbf{x})$ is the shortest signed distance from a query point $\mathbf{x}$ to the

surface of underlying 3D shape. And the sign determines whether it is inside or outside the 3D shape. Thus, the surface of a 3D shape is the zero level-set of SDF, denoted as

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\}. \tag{1}$$

Learning-based SDF usually encodes a 3D shape into a single global latant code or local latent codes. And a multi-layer perceptron is leveraged as the decoder, which takes latent codes and query points as input and predicts signed distances. Using sampled query points as training data and the corresponding ground truth signed distances as supervision, the latent codes and network parameters are optimized by minimizing the errors between predicted and ground truth signed distances. After SDF is learned, the Marching Cubes algorithm [18] is usually applied to extract an isosurface and outputs a mesh for rendering or visualization.

3.2. Dynamic Code Cloud

In our method, we leverage an auto-decoder framework [29]. And we learn DIF using a novel Dynamic Code Cloud (DCC-DIF). In Fig. 3, we show the overall architecture of our method and the differences between our method and the compared methods. We represent a 3D shape using a set of latent codes and the corresponding code positions, which are denoted by a matrix $\mathbf{C} \in \mathbb{R}^{n\times m}$ and a matrix $\mathbf{P} \in \mathbb{R}^{n\times 3}$, respectively. Here, n indicates the number of local codes that we used, and m indicates the dimension of local codes. Each row of $\mathbf{C}$ is a latent code $\mathbf{c}_i \in \mathbb{R}^m$, and each row of $\mathbf{P}$ is a position vector $\mathbf{p}_i \in \mathbb{R}^3$, where $1 \le i \le n$. Each $\mathbf{c}_i$ and $\mathbf{p}_i$ form a pair and $\mathbf{p}_i$ indicates the (x, y, z) coordinate of corresponding latent code in the 3D space.

Given a batch of query points $\mathbf{Q} \in \mathbb{R}^{B\times 3}$ in 3D space with the batch size B, in which each row is a query point $\mathbf{q}_j \in \mathbb{R}^3$ $(1 \le j \le B)$, we firstly obtain a distance matrix $\mathcal{D} \in \mathbb{R}^{B\times n}$ between query points $\mathbf{Q}$ and code positions $\mathbf{P}$, where each element of $\mathcal{D}_{ji}$ is computed as

$$\mathcal{D}_{ji} = ||\mathbf{q}_j - \mathbf{p}_i||_2. \tag{2}$$

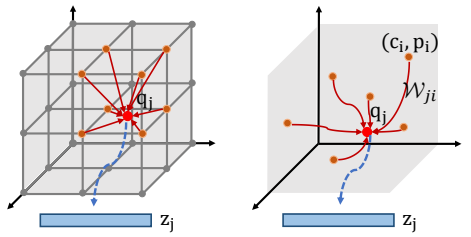
Then a weight matrix $\mathcal{W}$ is obtained based on $\mathcal{D}$, each element of which is computed as

$$\mathcal{W}_{ji} = \frac{\mathcal{W}'_{ji}}{\sum_{k=1}^{n} \mathcal{W}'_{jk}}, \tag{3}$$

where

$$\mathcal{W}'_{ji} = \frac{1}{\mathcal{D}_{ji}^3}. \tag{4}$$

As we expect that the local codes far from the query point have small weights, we take the reciprocal of the cubic distance as the weight in Eq. (4). Then we normalize the weights in Eq. (3). After that, the matrix multiplication is applied to the weight matrix $\mathcal{W}$ and latent codes $\mathbf{C}$, which produces a matrix $\mathbf{Z} \in \mathbb{R}^{B\times m}$. Intuitively, each row of $\mathbf{Z}$ is a vector $\mathbf{z}_j \in \mathbb{R}^m$ for query point $\mathbf{q}_j$, which is interpolated from latent codes $\mathbf{C}$ based on distances. Fig. 4 shows the interpolation process of our method, and the difference compared with traditional grid-based methods (e.g [24]). At last, like most methods do, we leverage a multi-layer perceptron as the decoder. We concatenate $\mathbf{Q}$ and $\mathbf{Z}$ together as the input of the decoder and get a output vector $\mathbf{d} \in \mathbb{R}^B$, in which each element d_j is a predicted signed distance for $\mathbf{q}_j$.



(a) Interpolation of latent grids (b) Interpolation of ours

Figure 4. Illustration of interpolation. (a) Given a query point $\mathbf{q}_j$, previous grid-based methods [4, 5, 24, 28] leverage the position of $\mathbf{q}_j$ and its neighboring grid points to interpolate stored latent codes into a vector $\mathbf{z}_j$, where a trilinear interpolation is usually applied in these methods. (b) In our method, each latent code $\mathbf{c}_i$ is explicitly associated with a position vector $\mathbf{p}_i$ to indicate its position in 3D space, where the positions of both $\mathbf{q}_j$ and $\mathbf{p}_i$ are used to compute the weight $\mathcal{W}_{ji}$ which is used for interpolation.

As $\mathbf{p}_i$ can be any (x, y, z) coordinates in the bounding box, our method is a more flexible representation whose code positions are continuous, while the latent codes used in other methods are usually constrained to be located at regular and discrete positions. In contrast, we can dynamically update code positions during optimization directly by back-propagation and gradient descent, as there is no essential difference between $\mathbf{p}_i$ and other trainable parameters in the network.

3.3. Code Position Loss

To fully utilize the latent codes, we further propose a novel Code Position (CP) loss. Our motivation is to guide more latent codes to be distributed near the regions with complex geometric details.

As shown in Fig. 5, we first define the prediction error of each query point $\mathbf{q}_j$ as e_j, i.e.

$$e_j = \mid d_j - \bar{d}_j \mid, \tag{5}$$

where d_j is the predicted signed distance for query point $\mathbf{q}_j$ by our method and $\bar{d}_j$ is the ground truth. Intuitively, larger e_j means it is more difficult to reconstruct the local region near the corresponding query point, which further indicates

that complex geometries details may exist on this region. As we expect the latent codes to get closer to the query points with higher prediction errors, we assume that there is a certain attraction force between query points and latent codes. Moreover, such attraction force should be directly proportional to e_j and decay with the growth of distances between latent codes and query points. As elements of the weight matrix $\mathcal{W}$ in Sec. 3.2 have the property of decreasing with increasing distances, we use this weight matrix as a decay of attraction force. Therefore, we define the *attraction matrix* $\mathcal{A} \in \mathbb{R}^{B\times n}$ between query points $\mathbf{Q}$ and latent codes $\mathbf{C}$ as

$$\mathcal{A}_{ji} = e_j * \mathcal{W}_{ji}. \tag{6}$$

Then, we apply element-wise multiplication between the attraction matrix $\mathcal{A}$ and the distance matrix $\mathcal{D}$, and take the average of all elements as the final Code Position Loss L_{CP}, denoted by

$$L_{CP} = \frac{1}{B*n}\sum_{j=1}^{B}\sum_{i=1}^{n}\mathcal{A}_{ji} * \mathcal{D}_{ji}. \tag{7}$$

Note that we cut off gradient back propagation to $\mathcal{A}$. Thus, the distances between latent codes and query points are optimized based on the attraction, leading to further update of code positions $\mathbf{P}$.

With the guidance of CP loss, more latent codes will be distributed near the regions with complex geometric details, which enables our method to capture fine local geometric details. On the other hand, since there are fewer latent codes around simple geometric regions, this allows our method to represent a 3D shape with a small amount of latent codes, compared with previous grid-based methods [4, 5, 24, 28]. Although other methods can also assign more latent codes to complex regions, such as by refining the depth of octree [19], our method is more flexible and effective since our code positions are continuous. Furthermore, we optimize the code positions directly by back-propagation and gradient descent, which is more efficient.

3.4. Training

We train our network in an auto-decoder [29] framework. To optimize latent codes and code positions, sampled query points and their corresponding ground truth signed distances should be provided as training data. For fair comparisons, we adopt different sampling schemes in different experiments to keep the same setting with the methods to be compared.

To optimize latent codes, we minimize the mean squared error (MSE) between predicted signed distances d_j and ground truth $\bar{d}_j$, denoted as

$$L_{MSE} = \frac{1}{B}\sum_{j=1}^{B}||d_j - \bar{d}_j||^2. \tag{8}$$

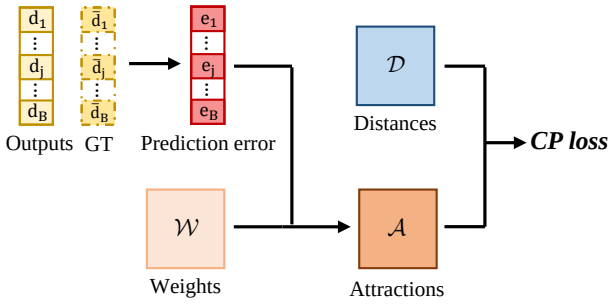


Figure 5. **Code Position Loss.** We assume query points are attractive to latent codes. And we define attractions based on prediction errors of query points. As elements of the weight matrix $\mathcal{W}$ in Sec. 3.2 have the property of decreasing with increasing distances, we use this weight matrix as a decay of attractions. Lastly, we apply the element-wise multiplication to the distance matrix and the attraction matrix, and take the average of all values as the final CP loss.

We also minimize the CP loss to optimize code positions. As a result, our final loss L is defined as

$$L = L_{MSE} + \lambda L_{CP}, \tag{9}$$

where λ is a parameter to balance L_{MSE} and L_{CP}.

4. Experiments

In this section, we conduct comprehensive experiments to evaluate the performance of the proposed DCC-DIF. Specifically, in Sec. 4.1, we demonstrate the ability of DCC-DIF for describing geometric details through the single shape fitting task. In Sec. 4.2, the ability of DCC-DIF for learning shape priors and generalizing to new objects is further evaluated, by applying DCC-DIF to reconstruct unseen shapes. In Sec. 4.3, we validate the effects of each part of DCC-DIF.

4.1. Single Shape Fitting

We apply DCC-DIF to single shape fitting task to evaluate the ability of describing geometric details. In this experiment, the latest NGLOD [28] is typically selected for our comparison, which is an octree based method and achieves state-of-the-art results in single shape fitting.

Network settings. For fair comparisons, we use the same settings with NGLOD [28]. Specifically, we set the decoder to be a multi-layer perceptron with only one hidden layer, which is 128-dimensional and leverages a ReLU [12] activation function. And we set m to 32, which is the same with NGLOD [28]. NGLOD [28] has different numbers of latent codes for different shapes after removing the empty nodes of octree that contain no surface. We simply use $n = 5600$

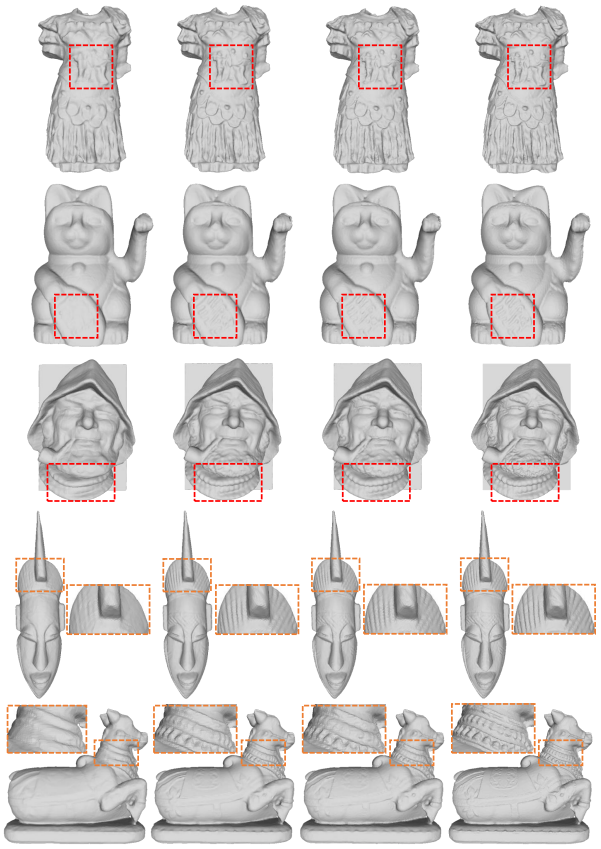
Metrics	Methods							
	DeepSDF [23]	FFN [30]	SIREN [27]	NI [8]	NGLOD3 [28]	NGLOD4 [28]	NGLOD5 [28]	Ours
IoU ↑	96.8	97.7	95.1	96.0	99.0	99.3	99.4	**99.5**
CD ↓	–	–	–	–	3.69	3.59	3.57	**3.55**
#Codes	–	–	–	–	5.7K/0.9K	41.7K/3.7K	316K/15K	5.6K
#Param.	1.8M	527K	264K	7.6K	4.7K	4.7K	4.7K	4.7K

Table 1. Results on Thingi32 [34]. We compare the reconstruction quality and efficiency between our method and others (NGLOD [28] with LODs equals to 3, 4 and 5 is denoted as NGLOD3, NGLOD4 and NGLOD5, respectively). For quality, we use the IoU and CD as metrics. For efficiency, we use *#Codes* and *#Param.* as metrics. The *#Codes* indicates the number of latent codes used in each method. And the *#Param.* means the number of network parameters used for a single distance query. In the *#Codes* row, for each LODs of NGLOD [28], we present the average number of latent codes before/after removing the empty nodes of octree. Our method simultaneously achieves the best quality and a high efficiency.

latent codes for each shape, which is approximately equal to the number of latent codes used in NGLOD [28] with 3 LODs (before removing the empty nodes of octree). As prediction errors e_j and distances $\mathcal{D}_{ji}$ tend to be small, we choose λ as a relatively large number to balance L_{MSE} and L_{CP}, where λ is typically set to 7000 in this experiment.

Data and metrics. Following NGLOD [28], we also select the same 32 shapes from Thingi10K [34] and follow the same preprocessing practice as NGLOD. Specifically, following DualSDF [13], we normalize the meshes and remove internal triangles. And we sign the distances with ray stabbing [22]. We adopt the same schemes with NGLOD [28] to obtain a point set for training. Specifically, we sample 500K points at each epoch, in which 100K points are sampled uniformly in the bounding box, 200K points from the object surface and the other points are sampled near the object surface. For metrics, we evaluate results using Chamfer Distance (CD) and Intersection over Union (IoU). Following NGLOD [28], we also pay attention to the efficiency of storage and computation. Here, the number of latent codes used in each method is denoted as *#Codes*, which roughly shows the storage cost. And we also denote the number of network parameters used for a single distance query as *#Param.*, which roughly indicates the computation cost during inference.

Results and analyses. Tab. 1 shows the result comparison of our method and other methods, including DeepSDF [23], FFN [30], SIREN [27], NI [8] and NGLOD [28]. Specially, the LODs of NGLOD are selected as 3, 4 and 5, denoted as NGLOD3, NGLOD4 and NGLOD5, respectively. As we have the exactly same experiment settings with NGLOD [28], some results in the table directly come from it. Since the CD can be influenced by some factors such as the number of surface points, we recompute the CD by ourselves. The results in Tab. 1 show that our method achieves the highest IoU and the lowest CD, which all are beyond other methods. We show the average number of latent codes used by NGLOD [28] before/after removing the empty nodes of octree. It is worth noting that, compared with NGLOD5, our method leverages a small num-



(a) NGLOD3 (b) NGLOD5 (c) Ours (d) Reference

Figure 6. Visualization results on Thingi32 [34]. We visually compare with NGLOD [28] with LODs that equals to 3 and 5, as denoted by NGLOD3 and NGLOD5, respectively. We achieve better results than NGLOD3, especially for local geometric details. Although NGLOD5 can achieve the similar results with our method, our method has a small number of latent codes than NGLOD5.

Category	Chamfer(↓)						F-Score(↑,%)					
	Occ. [21]	SIF [11]	LDIF [10]	IF. [5]	MDIF [4]	Ours	Occ. [21]	SIF [11]	LDIF [10]	IF. [5]	MDIF [4]	Ours
airplane	0.16	0.44	0.10	0.52	0.028	**0.011**	87.8	71.4	96.9	94.4	98.6	**99.7**
bench	0.24	0.82	0.17	0.31	0.052	**0.017**	87.5	58.4	94.8	92.6	96.0	**99.5**
cabinet	0.41	1.10	0.33	0.11	**0.051**	0.131	86.0	59.3	92.0	93.0	**96.6**	96.4
car	0.61	1.08	0.28	0.30	**0.088**	0.218	77.5	56.6	87.2	87.4	**93.0**	92.7
chair	0.44	1.54	0.34	0.10	**0.035**	0.037	77.2	42.4	90.9	94.5	97.6	**99.1**
display	0.34	0.97	0.28	0.07	**0.019**	0.028	82.1	56.3	94.8	96.1	98.7	**99.4**
lamp	1.67	3.42	1.80	1.17	0.795	**0.327**	62.7	35.0	84.0	89.1	93.5	**97.3**
rifle	0.19	0.42	0.09	1.07	0.057	**0.007**	86.2	70.0	97.3	93.5	96.9	**99.9**
sofa	0.30	0.80	0.35	0.13	0.037	**0.036**	85.9	55.2	92.8	92.5	98.4	**99.1**
speaker	1.01	1.99	0.68	0.14	**0.044**	0.146	74.7	47.4	84.3	90.2	**97.3**	96.1
table	0.44	1.57	0.56	0.17	0.046	**0.029**	84.9	55.7	92.4	93.4	97.6	**99.3**
telephone	0.13	0.39	0.08	0.08	**0.010**	0.027	94.8	81.8	98.1	98.8	**99.6**	99.3
watercraft	0.41	0.78	0.20	0.90	0.067	**0.042**	77.3	54.2	93.2	92.7	97.2	**98.3**
mean	0.49	1.18	0.40	0.39	0.102	**0.081**	81.9	59.0	92.2	92.9	97.0	**98.2**

Table 2. Results on ShapeNet [2]. We use the Chamfer distance (CD) and F-Score to evaluate the reconstruction results of our and compared methods. Our method achieves the lowest mean CD and the highest mean IoU, outperforming all other methods.

ber of latent codes, but still achieves slightly better results. This demonstrates that our method can represent 3D shapes more efficiently. Our method also has advantages in storage and computation efficiency. As visualized in Fig. 6, our method achieves the similar reconstruction quality compared with NGLOD5, while using a small number of latent codes. Compared with NGLOD3, our method achieves better reconstruction quality, especially for local geometric details.

4.2. Reconstructing 3D Datasets

We conduct an experiment to reconstruct 3D datasets using our method. Specifically, we optimize the latent codes, code positions and decoder parameters in the training phase. During inference, we fix decoder parameters and only optimize the latent codes and code positions on unseen shapes. This experiment shows the ability of our method to learn shape priors and generalize to new objects.

Network settings. We leverage a decoder with the same structure as IM-Net [3], which is a fully-connected network with connections between layers. We set $m = 32$ and $n = 1376$, thus we have the same number of parameters in latent codes with MDIF [4]. To balance L_{MSE} and L_{CP}, we set $\lambda = 3000$ in this experiment.

Data and metrics. We use a subset of 13 categories in ShapeNet [2] and divide the dataset with train/test splits from 3D-R^{2}N^2 [6]. And we generate watertight meshes with tools from OccNet [21]. In this experiment, we sample a point set from each shape, and use the same point set for all epochs during training. Each point set contains 200K samples in which half of the points comes from uniform sampling and the others are sampled near the object surface. We use Chamfer L2 distance and F-Score to evaluate the results, which have the identical settings with LDIF and MDIF [4, 10].

Results and analyses. The results are shown in Tab. 2. As we keep exactly the same experiment settings with MDIF [4], some results in the table directly come from it. Our method surpasses all other methods both on the mean CD and the mean IoU, which demonstrates the ability of our method to learn shape priors and generalize to new objects. Among different categories of ShapeNet [2], there are great differences in difficulty of reconstruction. Shapes in some categories tend to be complex and various, such as lamp and rifle. In contrast, shapes in some other categories are relatively simple and similar to each other, such as speaker and display. From Tab. 2, we find that our method has great advantages in reconstructing complex and various shapes. Fig. 7 visually shows the quality of our reconstruction on ShapeNet [2], compared with IF-Net [5]. Our method achieves better reconstruction results, especially in local regions with thin strips and holes. This demonstrates that our method has a ability to represent complex shapes and capture fine local geometry details.

4.3. Ablation Study

Among existing DIF methods, grid and octree based methods achieve the good performance in both single shape fitting and 3D datasets reconstruction. Compared with these methods, there are three differences in our DCC-DIF, including interpolation process, the novel position vectors and CP loss. To evaluate the effects of each difference, we conduct ablation studies on display and watercraft categories from ShapeNet [2], where display category contains relatively simple shapes and watercraft category tend to be complex. We keep other experiment settings the same as Sec. 4.2.

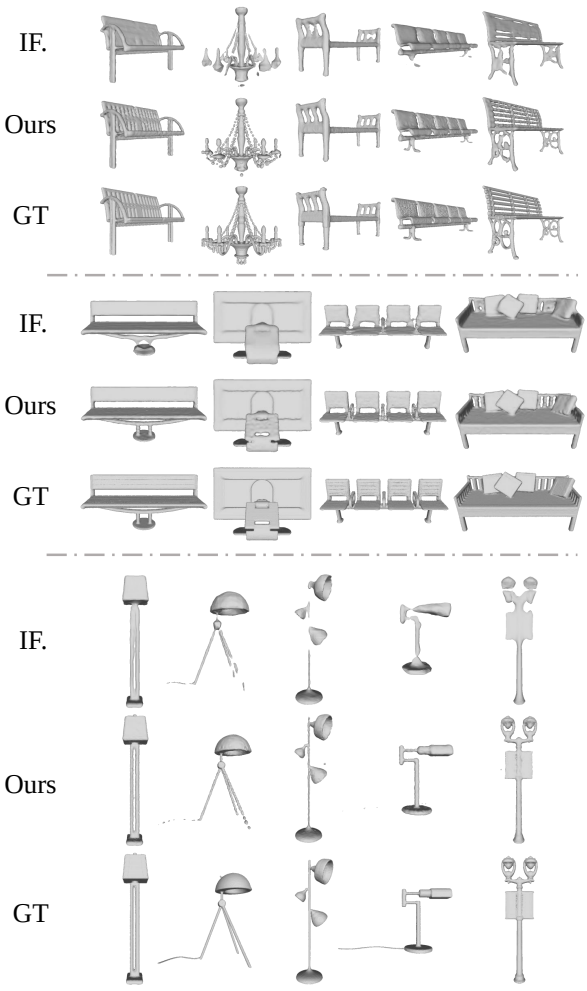


Figure 7. Visualization results on ShapeNet [2]. Compared to IF-Net [5], our method can achieve better reconstruction quality in local regions with complex geometric details, such as thin strips and holes.

To evaluate the influence of interpolation process, we design two variations of our DCC-DIF. The first variation fixes all latent codes to be located at grid points and applies trilinear interpolation. The second variation also fixes latent codes at grid points but leverages distance-based interpolation, as described in Sec. 3.2. As positions of latent codes are fixed, we remove the CP loss from both variations. As shown in Tab. 3, the variation using trilinear interpolation achieves better results. It demonstrates that the performance of our DCC-DIF does not benefit from the new interpolation algorithm, but from our proposed position vectors and CP loss.

Next, we validate the effects of our proposed position vectors and CP loss. As a baseline, we remove the CP loss from our full version pipeline and fix all latent codes at grid points. Then we evaluate the benefit of position vectors and CP loss respectively by two variations of our DCC-DIF. The first variation only adds the position vectors to the baseline and the second variation adds both position vectors and CP loss to baseline. Results are shown in Tab. 4. We can find that both position vectors and the CP loss play a positive role in 3D shape representation, which supports our proposal. Additionally, the CP loss shows more significant effects with complex shapes, which is consistent with our design to guide more latent codes to be distributed around complex geometric details.

Category	Chamfer(↓)		F-Score(↑,%)	
	Trilinear	Ours	Trilinear	Ours
display	0.038	0.045	99.2	98.8
watercraft	0.090	0.108	97.2	96.3

Table 3. Comparison of interpolation process.

Category	Chamfer(↓)			F-Score(↑,%)		
	baseline	p.v.	p.v. + CP	baseline	p.v.	p.v. + CP
display	0.045	0.033	0.028	98.8	99.3	99.4
watercraft	0.108	0.081	0.042	96.3	97.8	98.3

Table 4. Ablation of position vectors and CP loss. The 'p.v.' denotes position vectors.

5. Conclusion and Limitation

In this paper, we introduce a novel DCC-DIF to learn deep implicit functions for 3D shapes. Among existing DIF methods, the best results are achieved by grids or octree based methods. However, the latent codes in these methods are constrained to be located at discrete and regular positions, and the code positions are difficult to be optimized. In contrast, the code positions in our DCC-DIF are continuous and flexible by explicitly assigning a position vector to each latent code. We further propose a novel CP loss to optimize the positions of latent codes, so that more latent codes are distributed around complex geometric details. In experiments, our method outperforms other methods and achieves state-of-the-art results, which demonstrates its performance and efficiency. The ablation studies show effects of each part of our design, which supports our proposal.

Some limitations exist in our DCC-DIF, which are also the directions of improvement in our future work. First, the current DCC-DIF is unable to represent different levels of details like NGLOD [28]. To address this problem, we plan to design a hierarchical network, in which each level leverages a DCC-DIF and the number of latent codes growth along with the increase of level. Another limitation is that current DCC-DIF is unsuited to global operations such as completion [7]. Inspired by MDIF [4], we can further set the first level of above-mentioned hierarchical DCC-DIF to be a single global latent code and design a novel module for information exchange between global and local codes.

References

[1] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, S. Lovegrove, and Richard A. Newcombe. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. In *European Conference on Computer Vision*, 2020. 1, 3

[2] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *ArXiv*, abs/1512.03012, 2015. 7, 8

[3] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5932–5941, 2019. 1, 7

[4] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution Deep Implicit Functions for 3D Shape Representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13087–13096, October 2021. 2, 3, 4, 5, 7, 8

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6968–6979, 2020. 2, 3, 4, 5, 7, 8

[6] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *European Conference on Computer Vision*, 2016. 7

[7] Angela Dai, C. Qi, and Matthias Nießner. Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6545–6554, 2017. 8

[8] T. Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit Neural Networks as a Compact Shape Representation. *ArXiv*, abs/2009.09808, 2020. 6

[9] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J. Guibas. Curriculum DeepSDF, 2020. 3

[10] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas A. Funkhouser. Local Deep Implicit Functions for 3D Shape. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4856–4865, 2020. 1, 3, 7

[11] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas A. Funkhouser. Learning Shape Templates With Structured Implicit Functions. *IEEE/CVF International Conference on Computer Vision*, pages 7153–7163, 2019. 1, 3, 7

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2011. 5

[13] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge J. Belongie. DualSDF: Semantic Shape Manipulation Using a Two-Level Representation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7628–7638, 2020. 6

[14] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas A. Funkhouser. Local Implicit Grid Representations for 3D Scenes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6000–6009, 2020. 1, 3

[15] Hiroharu Kato, Y. Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 1, 2

[16] Sangtae Kim, Kwangjin Lee, Seungho Doo, and Byonghyo Shim. Moving Target Classification In Automotive Radar Systems Using Transposed Convolutional Networks. *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 2050–2054, 2018. 3

[17] Shilin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 3

[18] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987. 4

[19] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric Chan, Marco Monteiro, and Gordon Wetzstein. ACORN: Adaptive Coordinate Networks for Neural Scene Representation. *ACM Transactions on Graphics*, 40:58:1–58:13, 2021. 2, 3, 5

[20] Daniel Maturana and Sebastian A. Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928, 2015. 1, 2

[21] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4455–4465, 2019. 1, 2, 3, 7

[22] Fakir S. Nooruddin and Greg Turk. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9:191–205, 2003. 6

[23] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 3, 6

[24] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks. In *European Conference on Computer Vision*, 2020. 1, 2, 3, 4, 5

[25] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, 2017. 1, 2

[26] C. Qi, L. Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Conference and Workshop on Neural Information Processing Systems*, 2017. 1, 2

[27] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Conference and Workshop on Neural Information Processing Systems*, 2020. 6

[28] Towaki Takikawa, Joey Litalien, K. Yin, Karsten Kreis, Charles T. Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2, 3, 4, 5, 6, 8

[29] Shufeng Tan and Michael L. Mayrovouniotis. Reducing data dimensionality through optimizing neural network inputs. *Aiche Journal*, 41:1471–1480, 1995. 1, 4, 5

[30] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Conference and Workshop on Neural Information Processing Systems*, 2020. 6

[31] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations. In *European Conference on Computer Vision*, 2020. 3

[32] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: Deep implicit surface network for high-quality single-view 3D reconstruction. In *Conference and Workshop on Neural Information Processing Systems*, 2019. 1

[33] Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. Deep Implicit Templates for 3D Shape Representation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1439, 2021. 3

[34] Qingnan Zhou and Alec Jacobson. Thingi10K: A Dataset of 10, 000 3D-Printing Models. *ArXiv*, abs/1605.04797, 2016. 6