Neural Vector Fields: Implicit Representation by Explicit Learning

Xianghui Yang\textsuperscript{1}, Guosheng Lin\textsuperscript{2}, Zhenghao Chen\textsuperscript{1}, Luping Zhou\textsuperscript{1}
\textsuperscript{1}The University of Sydney, \textsuperscript{2}Nanyang Technological University
\{xianghui.yang, zhenghao.chen, luping.zhou\}@sydney.edu.au, gslin@ntu.edu.sg

Abstract

Deep neural networks (DNNs) are widely applied for nowadays 3D surface reconstruction tasks and such methods can be further divided into two categories, which respectively warp templates explicitly by moving vertices or represent 3D surfaces implicitly as signed or unsigned distance functions. Taking advantage of both advanced explicit learning process and powerful representation ability of implicit functions, we propose a novel 3D representation method, Neural Vector Fields (NVF). It not only adopts the explicit learning process to manipulate meshes directly, but also leverages the implicit representation of unsigned distance functions (UDFs) to break the barriers in resolution and topology. Specifically, our method first predicts the displacements from queries towards the surface and models the shapes as Vector Fields. Rather than relying on network differentiation to obtain direction fields as most existing UDF-based methods, the produced vector fields encode the distance and direction fields both and mitigate the ambiguity at “ridge” points, such that the calculation of direction fields is straightforward and differentiation-free. The differentiation-free characteristic enables us to further learn a shape codebook via Vector Quantization, which encodes the cross-object priors, accelerates the training procedure, and boosts model generalization on cross-category reconstruction. The extensive experiments on surface reconstruction benchmarks indicate that our method outperforms those state-of-the-art methods in different evaluation scenarios including watertight vs non-watertight shapes, category-specific vs category-agnostic reconstruction, category-unseen reconstruction, and cross-domain reconstruction. Our code is released at https://github.com/Wi-sc/NVF.

1. Introduction

Reconstructing continuous surfaces from unstructured, discrete and sparse point clouds is an emergent but non-trivial task in nowadays robotics, vision and graphics applications, since the point clouds are hard to be deployed into the downstream applications without recovering to high-resolution surfaces [5, 7, 38, 42].

With the tremendous success of Deep Neural Networks (DNNs), a few DNN-based surface reconstruction methods have already achieved promising reconstruction performance. These methods can be roughly divided into two categories according to whether their output representations are explicit or implicit. As shown in Fig. 1, explicit representation methods including mesh and voxel based ones denote the exact location of a surface, which learn to warp templates [3, 4, 19, 26, 29, 68] or predict voxel grids [10, 30, 59]. Explicit representations are friendly to downstream applications, but they are usually limited by resolution and topology. On the other hand, implicit representations such as

Figure 1. Common 3D representations. Explicit representations: (a) point clouds, (b) meshes, (c) voxels. Implicit representations: (c) occupancy, (d) reconstruction from the signed distance functions, and (e) reconstruction from unsigned distance functions. Our method represents continuous surfaces through (f) vector fields. (g) Vector fields can deform meshes (red) as explicit representation methods.
as Occupancy and Signed Distance Functions (SDFs) represent the surface as an isocontour of a scalar function, which receives increasing attention due to their capacity to represent surfaces with more complicated topology and at arbitrary resolution \cite{chen2022non}. However, most implicit representation methods usually require specific pre-processing to close non-watertight meshes and remove inner structures. To free from the above pre-processing requirements for implicit representation, Chibane et al. \cite{chibane2019neural} introduced Neural Unsigned Distance Fields (NDF), which employs the Unsigned Distance Functions (UDFs) for neural implicit functions (NIFs) and models continuous surfaces by predicting positive scalar between query locations and the target surface. Despite certain advantages, UDFs require a more complicated surface extraction process than other implicit representation methods (e.g., SDFs). Such a process using Ball-Pivoting Algorithm \cite{amenta1998surface} or gradient-based Marching Cube \cite{lorensen1987marching, martini2022shape} relies on model differentiation during inference (i.e., differentiation-dependent). Moreover, UDFs leave gradient ambiguities at "ridge" points, where the gradients\footnote{Learning-based methods calculate the gradients of distance fields via model differentiation. The opposite direction of gradients should point to the nearest-neighbor point on the target surface.} used for surface extraction cannot accurately point at target points as illustrated by Fig. 2a.

In this work, we propose a novel 3D representation method, Neural Vector Fields (NVF), which leverages the explicit learning process of direct manipulation on meshes and the implicit representation of UDFs to enjoy the advantages of both approaches. That is, NVF can directly manipulate meshes as those explicit representation methods as Fig. 1g, while representing the shapes with arbitrary resolution and topology as those implicit representation methods. Specifically, NVF models the 3D shapes as vector fields and computes the displacement between a point \( q \in \mathbb{R}^3 \) and its nearest-neighbor point on the surface \( \hat{q} \in \mathbb{R}^3 \) by using a learned function \( f(q) = \Delta q = \hat{q} - q : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \). Therefore, NVF could serve both as an implicit function and an explicit deformation function, since the displacement output of the function could be directly used to deform source meshes (i.e., Fig. 1g). In general, it encodes both distance and direction fields within vector fields, which can be straightforwardly obtained from the vector fields.

Different from existing UDF-based methods, our NVF representation avoids the comprehensive inference process by skipping the gradient calculation during the surface extraction procedure\footnote{Learning-based methods calculate the gradients of distance fields via model differentiation. The opposite direction of gradients should point to the nearest-neighbor point on the target surface.}, and mitigates ambiguities by directly learning displacements as illustrated by Fig. 2b. Such one-pass forward-propagation nature frees NVF from differentiation dependency, significantly reduces the inference time and memory cost, and allows our model to learn a shape codebook consisting of un-differentiable discrete shape codes in the embedded feature space. The learned shape codebook further provides cross-object priors to consequently improve the model generalization on cross-category reconstruction, and accelerates the training procedure as a regularization term during training. We use VQ as an example to demonstrate that the differentiation-free property of NVF provides more flexibility in model design in this paper.

We conduct extensive experiments on two surface reconstruction benchmark datasets: a synthetic dataset ShapeNet \cite{chang2015shapenet} and a real scanned dataset MGN \cite{liu2022mgn}. Besides category-specific reconstruction \cite{chen2022non, zhou2021fast} as demonstrated in most reconstruction methods, we also evaluate our framework by category-agnostic reconstruction, category-unseen reconstruction, and cross-domain reconstruction tasks to exploit the model generalization. Our experimental results indicate that our NVF can significantly reduce the inference time compared with other UDF-based methods as we avoid the comprehensive surface extraction step and circumvent the requirement of gradient calculation at query locations. Also, using the shape codebook, we observe a significant performance improvement and a better model generalization across categories.

Our contributions are summarized as follows.

- We propose a 3D representation NVF for better 3D field representation, which bridges the explicit learning and implicit representations, and benefits from both of their advantages. Our method can obtain the displacement of a query location in a differentiation-free way, and thus it significantly reduces the inference complexity and provides more flexibility in designing network structures which may include non-differentiable components.

- Thanks to our differentiation-free design, we further propose a learned shape codebook in the feature space, which uses VQ strategy to provide cross-object priors. In this way, each query location is encoded as a composition of discrete codes in feature space and further used to learn the NVF.

- We conduct the extensive experiments to evaluate the effectiveness of our proposed method. It consistently shows promising performance on two benchmarks.

Figure 2. Gradient ambiguities. (a) NDF \cite{chibane2019neural} cannot guarantee to pull points onto surfaces (i.e., ambiguity of gradient), while (b) our NVF address this issue by direct displacement learning.
2. Related Work

2.1. Explicit 3D Representations

Early shape representation methods are built upon voxels [17, 43, 58]. As the 3D spaces are discretized into grids, so they can be processed by adapting learning-based image processing techniques [34, 37, 53, 69, 70]. However, voxels are usually with the memory footprint scales cubically with the resolution. So recent voxel-based methods introduce adaptive discretization [10, 30, 54, 59] and voxel block hashing [57] to alleviate higher computational requirements for higher resolution.

Differently, point cloud based methods produce a compact and sparser encoding of surfaces, with lower computational cost and higher accessibility. Several learning-based point cloud processing methods [9, 49, 60, 67, 82] are proposed in recent years and achieve tremendous success on point cloud processing techniques [34, 37, 53, 69, 70]. For example, UAVS [76] adopts an intersection classification branch and CAP-UDF [83] directly optimizes models on raw point clouds. Although such UDF-based methods can intuitively improve the generalization of implicit representation, they require non-trivial surface extraction processing such as Ball-Pivoting algorithm [15] and gradient-based Marching Cubes [28, 83].

Different than aforementioned UDF-based methods, our NVF does not rely on differentiation to obtain gradients as direction fields and can simply obtain distance and direction fields by one-time forward propagation. Such effective differentiation-free strategy can significantly reduce the inference burden.

2.2. Implicit 3D Representations

To overcome the limited resolution and fixed mesh topology barriers from most explicit representation, implicit representation methods represent continuous surfaces through implicit functions. Such representations are usually implemented by multi-layer perceiving MLP, which generate binary occupancy [14, 18, 24, 44, 55, 56] and SDFs [12, 35, 47] given locations as queries. On the other hand, those methods require a heavy pre-processing stage to close the shapes artificially to obtain watertight meshes due to the characteristics of occupancy and SDFs. For one thing, it is also non-trivial to define the inside and outside of open surfaces. For another, the pre-processing results in substantive geometry information loss and lack of generalization on non-watertight meshes.

To model general non-watertight shapes, Chibane et al. [15] introduced UDFs to learn the unsigned distance. Recently, other methods improve the performance and generalization of UDFs [62, 64, 76, 81, 83]. For example, GIFS [76] adopts an intersection classification branch and CAP-UDF [83] directly optimizes models on raw point clouds. Although such UDF-based methods can intuitively improve the generalization of implicit representation, they require non-trivial surface extraction processing such as Ball-Pivoting algorithm [15] and gradient-based Marching Cubes [28, 83]. Overall, several methods employ this strategy for image generation [21, 27, 39, 51], speech synthesis [20, 73], video [23, 31, 52, 63], shape generation [13, 45, 72, 80] and compression [2, 11]. In this work, we also employ VQ and adopt a multi-head codebook in feature space to denoise, accelerate training and improve generalization on shape reconstruction.

3. Methodology

Given a 3D query location \( q \) from a query set \( Q \in \mathbb{R}^{M \times 3} \) and a sparse point cloud \( P \in \mathbb{R}^{N \times 3} \), our NVF framework predicts the displacement vector \( \Delta q \in \mathbb{R}^3 \) that moves \( q \) to the underlying surface of \( P \). This is achieved through three main modules, i.e., Feature Extraction, Multi-head Codebook, and Field Prediction, as illustrated in Fig. 3. Once the predicted vector field is obtained, we adopt Marching Cubes to generate the surface mesh. The details of each module are elaborated as follows.

3.1. Neural Vector Fields

As mentioned above, our NVF is a neural network function, which models shapes \( S \) by predicting the field of displacements \( \Delta q \) for a given query point \( q \in Q \) to its nearest
point \( \hat{q} \in S \) on the surface. It is formulated by

\[
NVF(q) = \Delta q = \min_{q \in S} \hat{q} - q. \tag{1}
\]

Please note that NVFs are directional distance fields, which encode both distance fields \( d = ||q - q||_2 \) and direction fields \( g = (\hat{q} - q) ||\hat{q} - q||_2 \) in a straightforward way. Therefore, unlike distance fields, our NVF does not require the differentiation of distance fields for direction information. This property is referred to as “differentiation-free” in this paper, which significantly reduces the inference time. On the other hand, our NVF enjoys the merits of UDFs and can represent much wider classes of surfaces and manifolds even not necessarily closed.

### 3.2. Feature Extraction

The input point cloud \( P \) is first sent to a feature encoder to extract 3D shape features for each point, which leads to the output feature maps \( F \in \mathbb{R}^{N \times C} \). Multiple point cloud networks \([9, 49, 82]\) could be used as the feature encoder, and an ablation study about the backbones is given in our experiment. For the query point \( q \), we find its \( K \) nearest points \( p_i \) \((i = 1, \ldots, k) \) in \( P \in \mathbb{R}^{N \times 3} \) based on a search using Euclidean distance, and form the feature embedding \( z_q \) of \( q \) by concatenating the signatures \( z_{qi} \) of its nearest points \( p_i \). Each signature \( z_{qi} \) is a non-linear mapping of the position features (i.e., the positions of the query point \( q \), the nearest point \( p_i \), and the relative position \( q - p_i \)) and the shape features (i.e., \( p_i \)’s corresponding feature \( f_i \) in \( F \)), which is realized by MLP. Mathematically, the query embedding \( z_q \in \mathbb{R}^D \) is obtained by

\[
z_q = z_{q1} \oplus z_{q2} \oplus \ldots z_{qi} \oplus \ldots z_{qk}, \tag{2}
\]

\[
z_{qi} = MLP(q, p_i, q_i - f_i), \quad i = 1, 2, \ldots, k.
\]

The symbol \( \oplus \) indicates the concatenation operation. As can be seen, the query embedding \( z_q \) carries the information of both the query point \( q \) and its \( K \) nearest points on \( P \).

### 3.3. Multi-head Codebook

After obtaining the embedding \( z_q \) whose dimension is \( D \), we learn a multi-head codebook \( C \in \mathbb{R}^{H \times R \times \frac{D}{H}} \) to produce cross-object priors. Our codebook contains \( H \) sub-codebooks (i.e., heads). Each sub-codebook \( C^h \) \((h = 1, \ldots, H) \) contains \( R \) discrete codes \( c^h_r \) \((r = 1, \ldots, R) \) and each code has a dimension of \( \frac{D}{H} \). The codebook is randomly initialized.

Accordingly, we split the continuous embedding \( z_q \) into \( H \) segments along the channel dimension. For each embedding segment \( z^h_q \), we search its closest code from the corresponding sub-codebook \( C^h \) according to their Euclidean distance. In this way, we could discretize the continuous embedding \( z_q \) to \( \hat{z}_q \) that is composed of discrete segment codes. This process can be formulated as follows,

\[
z_q = z^1_q \oplus z^2_q \oplus \ldots \oplus z^h_q \ldots \oplus z^H_q,
\]

\[
\hat{z}_q = c^1_{r^1} \oplus c^2_{r^2} \oplus \ldots \oplus c^h_{r^h} \ldots \oplus c^H_{r^H},
\]

\[
r^h_r = \arg \min_{r \in \{1, \ldots, R\}} \| z^h_q - c^h_r \|_2. \tag{3}
\]

Compared with using only one codebook \( C \in \mathbb{R}^{(H \times R) \times D} \), the multi-head codebook enables \( R^H \) permutations, which extends the feature space from \( \mathbb{R}^{H \times R} \) to \( \mathbb{R}^{R^H} \), and enhances the codebook representation capacity. Note that the nearest search stops the gradient here.

The multi-head codebook can be jointly learned during training by minimizing the distance between the continuous embedding \( z \) and its discretization \( \hat{z}_q \) as follows,

\[
L_{code} = \sum_{h \in \{1, \ldots, H\}} (\| sg(c^h_{r^h}) - z^h_q \|^2_2 + \beta \| sg(z^h_q) - c^h_{r^h} \|^2_2), \tag{4}
\]

where \( sg \) stands for stop gradient operation and \( \beta \) is the weight to the commitment loss in the second term. Alternatively, we can update the discretized embedding \( \hat{z}_q \) and therefore its corresponding codes \( (z^h_q \equiv c^h_{r^h}) \) in the codebook via Exponential Moving Average \([61]\),

\[
c^h_{r^h} := \gamma c^h_{r^h} + (1 - \gamma) z^h_q, \tag{5}
\]

with \( \gamma \) is a value between 0 and 1.

### 3.4. Field Prediction

Last, we take the continuous embedding \( z_q \) and its discretization \( \hat{z}_q \) to predict a vector as the displacement for
the query point \( q \) to move to the ground truth surface \( S \). By combining the embeddings \( z_q \) and \( \hat{z}_q \), the final vector field is predicted using MLPs, i.e., \( NVF(q) \approx \Delta q = MLP(z_q, \hat{z}_q) \).

### 3.5. Optimization

We optimize NVF by minimizing the difference between the predicted displacement \( \Delta q \) and the ground-truth displacement \( \hat{q} - q \), as well as the error of discretizing the continuous embedding \( z_q \) into \( \hat{z}_q \). We use stop-gradient operation \((i.e., sg)\) to cut the gradient after vector quantization. In sum, our overall objective function is,

\[
L = \sum_{q \in Q} ||\hat{q} - q - \Delta q||_1 + \lambda \sum ||z_q - sg(\hat{z}_q)||_2^2. \tag{6}
\]

The hyperparameter \( \lambda \) balances the two loss terms. The commitment loss term \( ||sg(z) - z'||_2^2 \) used in Eq. 4 is replaced by the Exponential Moving Average algorithm to update corresponding codes in the multi-head codebook.

### 3.6. Surface Extraction

Both signed and unsigned distance functions define the surfaces as a 0-level set. SDFs extract surfaces through Marching Cubes [42] faithfully due to the intersection being easily captured via sign flipping. Recently, MeshUDF [28] and CAP-UDF [83] propose to detect the opposite gradient directions to replace the sign in SDFs and successfully apply the Marching Cubes [42] on UDFs. Note that these two methods learn distance fields and they need to differentiate the distance field to obtain the gradient direction. In contrast, our NVF can similarly extract surfaces using Marching Cubes while avoiding the differentiation of the distance field. This is because our NVF directly encodes both distance and direction fields. The surface extraction algorithm divides the space into grids and decides whether the adjacent corners locate on the same grid or two sides. NVF predicts the vectors \( \Delta q \) of all lattices and normalizes them into normal vectors \( g_i = \Delta q_i / ||\Delta q_i||_2 \). Given a lattice gradient \( g_i \) and its adjacent lattices’ gradients \( g_j \), the algorithm checks if their gradients have opposite orientations, and assigns the pseudo sign \( s_i \) to the lattice. For more details, please refer to MeshUDF [28].

## 4. Experiments

### 4.1. Experimental Protocol

**Tasks.** We evaluate the effectiveness of our framework on four tasks: 1) category-specific, 2) category-agnostic, 3) category-unseen and 4) cross-domain reconstruction. We first demonstrate the ability of our NVF to reconstruct non-watertight meshes by category-specific reconstruction in Sec. 4.2. Next, to evaluate the generalization ability, we compare our NVF with existing methods on category-agnostic and category-unseen reconstruction in Sec. 4.3. We also test cross-domain reconstruction by reconstructing real scanned data without training or fine-tuning in Sec. 4.4.

**Implementations.** We employ PointTransformer [82] as our feature encoder and set \( k = 16 \) for the nearest points. The multi-head codebook consists of 4 sub-codebooks, each containing 128 codes and 64 dimension channels. We set the grid resolution for surface extraction as 256. To train our network, the learning rate is set as \( l_r = 0.001 \) and decays with the factor 0.3 for the subsequent 30, 70, and 120 steps. The loss weight \( \lambda \) is set to be 0.001.

**Datasets.** Most experiments are conducted on a large-scale synthetic dataset ShapeNet [8] and a real scanned dataset MGN [6]. We conduct our category-specific, category-agnostic, and category-unseen reconstruction experiments on ShapeNet [8] while conducting cross-domain reconstruction experiments on MGN [6] to demonstrate our model generalization in the wild. All meshes are normalized to a unit cube with a range \([-0.5, 0.5]\).

ShapeNet is a synthetic dataset with 55 object categories. We choose cars for category-specific reconstruction following UDF literature, and cars, chairs, planes, and tables for category-agnostic reconstruction. For category-unseen reconstruction, we take speakers, bench, lamps, and watercraft as the testing set. The above 4 categories for category-agnostic reconstruction are referred to as base classes and the ones for category-unseen reconstruction are referred to as novel classes in Tab. 2 and Tab. 3.
through NDF, GIFS, and our NVF are able to recover inner structures, our NVF could yield smoother surfaces and fewer holes in the finer parts (e.g., the back of driving seats).

### 4.3. Category-agnostic and Category-unseen Cases

#### Base vs Novel categories on watertight shapes

We conduct category-agnostic and category-unseen reconstruction experiments to explore the generalization ability of our NVF. For a fair comparison with the previous methods (e.g., OccNet [44] and IF-Net [14]) which cannot handle non-watertight shapes as the input, we report the quantitative experimental results on watertight shapes pre-processed by DISN [71] in Tab. 2 and give qualitative visualization in Fig. 5. We use the same training/testing split as in IF-Net [14]. The input of all methods is 3k points in this experiment.

Table 2 indicates that the surface reconstruction results produced by UDF-based methods are much better than occupancy representation, especially for novel classes. Among them, our NVF outperforms others by a large margin, no matter in base classes or novel classes. The qualitative comparison in Fig. 5 shows that OccNet and IF-Net fail to construct a lot of details. NDF and GIFS are better than them, but they still fail to capture some details, e.g., undercarriages of planes, or little bumps from watercraft. In contrast, our method can reconstruct flat and faithful surfaces, and thereby achieves a better visual effect.

#### Base vs Novel categories on non-watertight shapes

We also test on non-watertight shape reconstruction with the same setting above and the results are in Tab. 3 and Fig. 6. For the training/testing split, we randomly sample 3000 shapes from each base class as the training set and 200 shapes from each base and novel class as the validation and testing set. The input is 10k points sampled from the raw shapes.

### 4.2. Category-specific Cases

We first report the results of category-specific reconstruction on ShapeNet cars to demonstrate the representation ability to one category of our NVF. We perform the same training/testing split and sample 10k points as the input as in NDF [15] and GIFS [76]. The quantitative comparison in Tab. 1 shows that our method achieves better results with previous state-of-the-art methods including NDF and GIFS. Specifically, on CD and EMD, our NVF outperforms the second-best GIFS; and on F-scores, ours outperforms the second-best NDF. Moreover, we provide the qualitative comparison in Fig. 4. The visualization indicates that although NDF, GIFS, and our NVF are able to recover inner structures, our NVF could yield smoother surfaces and fewer holes in the finer parts (e.g., the back of driving seats).

### 4.1. Category-agnostic and Category-unseen Cases

#### Base vs Novel categories on non-watertight shapes

We first report the results of category-agnostic and category-unseen reconstructions on non-watertight shapes of ShapeNet. We train all models on the base classes and evaluate them on the base and the novel classes, respectively.

**Table 2. Quantitative results of category-agnostic and category-unseen reconstructions on watertight shapes of ShapeNet.** We train all models on the base classes and evaluate them on the base and the novel classes, respectively.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{2.5 \times 10^{-5}}↑</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>F1_{2 \times 10^{-5}}↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>0.840</td>
<td>1.045</td>
<td>14.148</td>
<td>25.111</td>
<td>0.800</td>
<td>1.024</td>
<td>17.576</td>
<td>29.815</td>
</tr>
<tr>
<td>OccNet [44]</td>
<td>2.766</td>
<td>1.694</td>
<td>30.877</td>
<td>46.644</td>
<td>44.762</td>
<td>4.013</td>
<td>15.943</td>
<td>24.433</td>
</tr>
<tr>
<td>IF-Net [14]</td>
<td>0.190</td>
<td>1.120</td>
<td>65.975</td>
<td>85.421</td>
<td>0.596</td>
<td>1.608</td>
<td>61.670</td>
<td>81.106</td>
</tr>
<tr>
<td>NDF [15]</td>
<td>0.169</td>
<td>1.538</td>
<td>66.802</td>
<td>84.809</td>
<td>0.169</td>
<td>1.741</td>
<td>65.622</td>
<td>84.069</td>
</tr>
<tr>
<td>GIFS [76]</td>
<td>0.179</td>
<td>1.280</td>
<td>56.188</td>
<td>78.458</td>
<td>0.194</td>
<td>1.534</td>
<td>56.644</td>
<td>78.016</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.091</strong></td>
<td><strong>1.079</strong></td>
<td><strong>78.503</strong></td>
<td><strong>91.408</strong></td>
<td><strong>0.144</strong></td>
<td><strong>1.145</strong></td>
<td><strong>80.883</strong></td>
<td><strong>91.836</strong></td>
</tr>
</tbody>
</table>

**Table 3. Quantitative results of category-agnostic and category-unseen reconstructions on non-watertight shapes of ShapeNet.** We train all models on the base classes, and evaluate them on the base and the novel classes, respectively.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{2.5 \times 10^{-5}}↑</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>F1_{2 \times 10^{-5}}↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>0.317</td>
<td>0.867</td>
<td>32.875</td>
<td>51.105</td>
<td>0.289</td>
<td>0.843</td>
<td>39.902</td>
<td>58.092</td>
</tr>
<tr>
<td>NDF [15]</td>
<td>0.099</td>
<td>1.372</td>
<td>72.425</td>
<td>88.754</td>
<td>0.093</td>
<td>1.532</td>
<td>76.162</td>
<td>89.977</td>
</tr>
<tr>
<td>GIFS [76]</td>
<td>0.118</td>
<td>1.260</td>
<td>64.915</td>
<td>85.115</td>
<td>0.296</td>
<td>1.499</td>
<td>69.252</td>
<td>86.518</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.085</strong></td>
<td><strong>1.197</strong></td>
<td><strong>75.372</strong></td>
<td><strong>90.266</strong></td>
<td><strong>0.078</strong></td>
<td><strong>1.340</strong></td>
<td><strong>79.723</strong></td>
<td><strong>91.576</strong></td>
</tr>
</tbody>
</table>

**Table 4. Quantitative results of cross-domain reconstruction on MGN [6].** We train our models based on ShapeNet with the base classes and evaluate them on the raw data from MGN.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{2.5 \times 10^{-5}}↑</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>CD↓</th>
<th>EMD↓</th>
<th>F1_{1 \times 10^{-5}}↑</th>
<th>F1_{2 \times 10^{-5}}↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>0.124</td>
<td>0.157</td>
<td>52.189</td>
<td>72.969</td>
<td>0.289</td>
<td>0.843</td>
<td>39.902</td>
<td>58.092</td>
</tr>
<tr>
<td>NDF [15]</td>
<td>0.025</td>
<td>0.216</td>
<td>96.338</td>
<td>98.678</td>
<td>0.093</td>
<td>1.532</td>
<td>76.162</td>
<td>89.977</td>
</tr>
<tr>
<td>GIFS [76]</td>
<td>0.039</td>
<td>0.192</td>
<td>91.330</td>
<td>97.295</td>
<td>0.296</td>
<td>1.499</td>
<td>69.252</td>
<td>86.518</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.014</strong></td>
<td><strong>0.184</strong></td>
<td><strong>98.499</strong></td>
<td><strong>99.498</strong></td>
<td><strong>0.078</strong></td>
<td><strong>1.340</strong></td>
<td><strong>79.723</strong></td>
<td><strong>91.576</strong></td>
</tr>
</tbody>
</table>

MGN is a real scanned dataset containing 5 garment categories, (i.e., long coat, pants, shirt, short pants, and T-shirt), which are all open surfaces. We evaluate our model trained by ShapeNet on MGN to demonstrate the reconstruction ability in the wild. The results are in Sec. 4.4.

**Metrics.** We use Chamfer Distance (CD), Earth Mover Distance (EMD) and F-score as our evaluation metrics. Specifically, we sample 100k points from the reconstructed surfaces for the Chamfer Distance (CD $\times 10^{-4}$), 2048 points for the Earth Mover Distance (EMD $\times 10^{-2}$), and 100k points for F-score ($\times 10^{-7}$) with thresholds $1 \times 10^{-5}$ and $2 \times 10^{-5}$.

### 4.2. Category-specific Cases

We first report the results of category-specific reconstruction on ShapeNet cars to demonstrate the representation ability to one category of our NVF. We perform the same training/testing split and sample 10k points as the input as in NDF [15] and GIFS [76]. The quantitative comparison in Tab. 1 shows that our method achieves better results with previous state-of-the-art methods including NDF and GIFS. Specifically, on CD and EMD, our NVF outperforms the second-best GIFS; and on F-scores, ours outperforms the second-best NDF. Moreover, we provide the qualitative comparison in Fig. 4. The visualization indicates that...
Figure 5. Visualization of category-agnostic and category-unseen reconstructions on watertight shapes from the ShapeNet dataset. The 1st row, planes, is from the base classes. The 2nd row, watercraft, is from the novel classes.

Figure 6. Visualization of category-agnostic and category-unseen reconstructions on non-watertight shapes from the ShapeNet dataset. The 1st row, tables, is from the base classes. The 2nd and 3rd rows, benches and watercraft, are from the novel classes.

Figure 7. Visualization of cross-domain reconstruction on the MGN dataset. All models are trained based on ShapeNet with the base classes and evaluated directly on MGN.

data of these shapes.

The quantitative comparison in Tab. 3 shows that our NVF outperforms the NDF and GIFS on either the category-agnostic or the category-unseen reconstruction. For example, we decrease ~10% CD on base classes and ~15% CD on novel classes from NDF. The qualitative visualization in Fig. 6 further supports the results, in which NDF and GIFS miss some important details (e.g., seams on benches) and leave a number of holes (e.g., legs of tables, sides of benches), and their surfaces are not as smooth as ours (e.g., the board of watercraft).

<table>
<thead>
<tr>
<th>K</th>
<th>Codebook</th>
<th>CD</th>
<th>EMD</th>
<th>F11×10^{-5}</th>
<th>F12×10^{-5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=8</td>
<td>X</td>
<td>0.089</td>
<td>1.195</td>
<td>74.139</td>
<td>89.392</td>
</tr>
<tr>
<td>k=8</td>
<td>✓</td>
<td>0.087</td>
<td>1.172</td>
<td>75.104</td>
<td>90.057</td>
</tr>
<tr>
<td>k=16</td>
<td>X</td>
<td>0.121</td>
<td>1.212</td>
<td>73.642</td>
<td>88.962</td>
</tr>
<tr>
<td>k=16</td>
<td>✓</td>
<td>0.085</td>
<td>1.197</td>
<td>75.372</td>
<td>90.266</td>
</tr>
<tr>
<td>k=8</td>
<td>X</td>
<td>0.080</td>
<td>1.334</td>
<td>78.701</td>
<td>90.972</td>
</tr>
<tr>
<td>k=8</td>
<td>✓</td>
<td>0.083</td>
<td>1.354</td>
<td>79.522</td>
<td>91.434</td>
</tr>
<tr>
<td>k=16</td>
<td>X</td>
<td>0.081</td>
<td>1.329</td>
<td>78.800</td>
<td>91.084</td>
</tr>
<tr>
<td>k=16</td>
<td>✓</td>
<td>0.078</td>
<td>1.340</td>
<td>79.723</td>
<td>91.576</td>
</tr>
</tbody>
</table>

Table 5. Effect of feature number \(K\) and multi-head codebook. The multi-head codebook improves the performance and achieves the best for \(K = 16\).

4.4. Cross-domain Cases

Rather than only concentrating on intra-domain performance as most previous reconstruction methods, we also explore the cross-domain ability of our method, in which we design a cross-domain reconstruction experiment and demonstrate the quantitative results and qualitative visualization respectively in Tab. 4 and Fig. 7. Specifically, we directly deploy our category-agnostic models trained based on ShapeNet base classes to MGN [6] to evaluate its performance. We randomly sample 20 shapes from the 5 categories in MGN and 3,000 points from the raw data as the input.

Tab. 4 indicates that our NVF outperforms other methods by a large margin under all metrics. For example, our method reduces almost 40% CD and 60% CD respectively from NDF and GIFS. The qualitative visualization in Fig. 7 also supports this observation, where only our NVF can reconstruct the crinkles of the shirts and pants faithfully.

5. Ablation Study

We conduct comprehensive ablation studies on non-watertight category-agnostic and category-unseen reconstruction (same settings as Sec. 4.3.) to evaluate the effectiveness of the proposed modules including multi-head codebook, and the influence of hyper-parameters like the feature number \(K\) (i.e., the number of the nearest points in Sec. 3.2) on the results. Complexity analysis is also provided to demonstrate the efficiency of our model.

Feature number. The number of features for query points
Table 6. Comparison of different backbones on non-watertight ShapeNet. The point cloud based backbones outperform the 3D convolution backbone. PointTransformer performs the best.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>Codebook</th>
<th>Runtime</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDF [15]</td>
<td>3D Conv</td>
<td>X</td>
<td>0.75s</td>
<td>13.44G</td>
</tr>
<tr>
<td></td>
<td>Pointnet++</td>
<td>✓</td>
<td>0.55s</td>
<td>9.21G</td>
</tr>
<tr>
<td></td>
<td>PointTransformer</td>
<td>✓</td>
<td>0.34s</td>
<td>9.28G</td>
</tr>
</tbody>
</table>

Table 7. Inference analysis. The runtime and memory are time cost and peak memory during the inference of 200k queries. NVF is more efficient on inference runtime and memory.

(determined by the number of nearest points on the point cloud $P$) is an important factor of the model. We report the performance of our network using $k = 8, 16$ in Tab. 5, and observe that a larger feature number $k$ yields better results but at the cost of memory.

**Codebook.** We also demonstrate the effectiveness of the introduced codebook mechanism in Tab. 5. The results indicate that using codebook can generally improve the overall performance regarding all metrics.

**Training Effectiveness.** We would like to highlight that the introduced multi-head codebook could also work as regularization to reduce the training time. As shown in Fig. 8, the loss curves of the models with (w/) codebooks (i.e., solid lines) generally converge faster than their corresponding models without (w/o) codebooks (i.e., dashed lines).

**3D Feature Extraction.** We also report our methods with different 3D feature extraction backbones (i.e., 3D convolution [14, 15, 76], Pointnet++ [49], and PointTransformer [82]) in Tab. 6. The results show that our NVF can already achieve comparable (if not better) results with NDF using the same extraction backbone (e.g., 3D convolution), while ours is more efficient on runtime and memory cost. Using PointTransformer can consistently improve the overall performance.

**Complexity.** We report the inference time and peak GPU memory on the machine with one RTX 3090Ti in Tab. 7 for obtaining the distance and direction of 200k query points. The results show that our method only requires 50% inference time and 70% memory compared with NDF when using the same feature extraction backbone. This is due to the cost from the differentiation of distance field in NDF, while our method avoids this issue and introducing multi-head codebook only brings negligible overhead of memory and time as shown in the Tab. 7.

**Limitations.** Our method can handle watertight or non-watertight shapes with arbitrary resolution and topology. However, the reconstruction for very thin or complex structures is still far from perfect as visualized in Fig. 9.

6. Conclusion

In this work, we introduce a novel surface reconstruction representation NVF, which leverages the advantages of both the implicit representations and the explicit learning process. It allows the reconstruction of high-quality general object shapes including watertight, non-watertight, and multi-layer shapes. Thanks to our differentiation-free design, we also introduce vector quantization and build a multi-head codebook to improve the generalization of cross-category reconstruction. Experiments demonstrate that our method not only achieves state-of-the-art reconstruction performance for varied topology, but also improves the training and inference efficiency.

7. Grant Acknowledgement

This research is partly supported by Australian Research Council (ARC) DP200103223 and the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20220-0007) and Tier 1 (RG1422).
References


[23] Songwei Ge, Thomas Hayes, Harry Yang, Xiaoyue Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vQGAN and time-sensitive transformer. In ECCV, 2022. 3


