3D Equivariant Graph Implicit Functions

Yunlu Chen[1], Basura Fernando[2], Hakan Bilen[3],
Matthias Nießner[4], and Efstratios Gavves[1]

[1] University of Amsterdam
[2] CFAR, IHPC, A*STAR
[3] University of Edinburgh
[4] Technical University of Munich

ychen9201@gmail.com, fernandopbc@ihpc.a-star.edu.sg,
hbilen@ed.ac.uk, niessner@tum.de, e.gavves@uva.nl

Abstract. In recent years, neural implicit representations have made remarkable progress in modeling of 3D shapes with arbitrary topology. In this work, we address two key limitations of such representations, in failing to capture local 3D geometric fine details, and to learn from and generalize to shapes with unseen 3D transformations. To this end, we introduce a novel family of graph implicit functions with equivariant layers that facilitates modeling fine local details and guaranteed robustness to various groups of geometric transformations, through local k-NN graph embeddings with sparse point set observations at multiple resolutions. Our method improves over the existing rotation-equivariant implicit function from 0.69 to 0.89 (IoU) on the ShapeNet reconstruction task. We also show that our equivariant implicit function can be extended to other types of similarity transformations and generalizes to unseen translations and scaling.

Keywords: Implicit neural representations; equivariance; graph neural networks; 3D reconstruction; transformation.

1 Introduction

Neural implicit representations are effective at encoding 3D shapes of arbitrary topology [32,30,10]. Their key idea is to represent a shape by a given latent code in the learned manifold and for each point in space, the neural implicit function checks whether a given coordinate location is occupied within the shape or not. In contrast to traditional discrete 3D representations such as triangle meshes or point clouds, this new paradigm of implicit neural representations has gained significant popularity due to the advantages such as being continuous, grid-free, and the ability to handle various topologies.

Despite their success, latent-code-conditioned implicit representations have two key limitations. First, the latent code of the shape captures coarse high-level shape details (i.e., the global structure) without any explicit local spatial information, hence it is not possible to learn correlations between the latent code and local 3D structural details of the shape. As a result, the surface reconstruction from latent-code-conditioned implicit functions tends to be over-smoothed and they are not good at capturing local

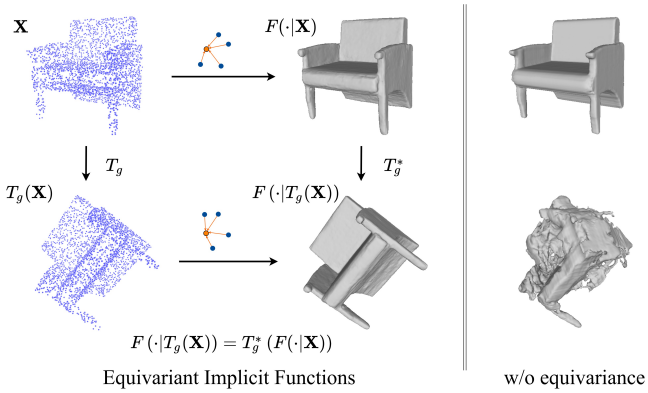

Fig. 1: **Our equivariant graph implicit function** infers the implicit field $F(\cdot|\mathbf{X})$ for a 3D shape, given a sparse point cloud observation $\mathbf{X}$. When a transformation T_g (rotation, translation, or/and scaling) is applied to the observation $\mathbf{X}$, the resulting implicit field $F(\cdot|T_g(\mathbf{X}))$ is *guaranteed* to be the same as applying a corresponding transformation T_g^* to the inferred implicit field from the untransformed input (*middle*). The property of equivariance enables generalization to unseen transformations, under which existing models such as ConvONet [34] often struggle (*right*).

surface detail [34,11,25,21,18]. Second, implicit representations are sensitive to various geometric transformations, in particular to rotations [16]. The performance of the implicit representations heavily relies on the assumption that shape instances in the same category are required to be in the same canonical orientation such that shape structures of planes and edges are in line with the coordinate axes. While data augmentation loosely addresses this second issue to some degree, a principled approach is to enable the representations to be inherently aware of common geometric operations such as rotations, translations, and scaling, which are found commonly in real-world 3D objects.

To address the first challenge of modeling local spatial information, recent methods [34,11] first discretize 3D space into local 2D or 3D grids and then store implicit codes locally in the respective grid cells. However, these methods are still sensitive to transformations as the grid structure is constructed in line with the chosen coordinate axes. This results in deteriorated performance under transformations as shown in Fig. 1. In addition, the grid discretization often has to trade fine details of the shape, hence the quality of shape reconstruction, for better computational efficiency through a low resolution grid. Deng *et al.* [17] propose VN-ONet to tackle the second challenge of pose-sensitivity with a novel vector neuron formulation, which enables the network architecture to have a rotation equivariant representation. Nevertheless, similar to implicit representations, the VN-ONet encodes each shape with a global representation and hence fails to capture local details. As grid discretization is not robust to transformations, this solution is not compatible with the grid-based local implicit methods. Thus, integrating VN-ONet with grid-approach is not a feasible solution.

In this work, our goal is to simultaneously address both challenges of encoding local details in latent representations and dealing with the sensitivity to geometric transfor-

mations such as rotations, translations, and scaling. To this end, we propose a novel equivariant graph-based local implicit function that unifies both of these properties. In particular, we use graph convolutions to capture local 3D information in a non-Euclidean manner, with a multi-scale sampling design in the architecture to aggregate global and local context at different sampling levels. We further integrate equivariant layers to facilitate generalization to unseen geometric transformations. Unlike the grid-based methods [34,11] that requires discretization of 3D space into local grids, our graph-based implicit function uses point features from the input point cloud observation directly without interpolation from grid features. Our graph mechanism allows the model to attend detailed information from fine areas of the shape surface points, while the regularly-spanned grid frame may place computations to less important areas. In addition, our graph structure is not biased towards the canonical axis directions of the given Cartesian coordinate frame, hence less sensitive than the grid-local representations [34,11]. Therefore, our graph representation is maximally capable of realizing an equivariant architecture for 3D shape representation. In summary, our contributions are as follows:

- We propose a novel graph-based implicit representation network that enables effective encoding of local 3D information in a multi-scale sampling architecture, and thus modeling of high-fidelity local 3D geometric detail. Our model features a non-Euclidean graph representation that naturally adapts with geometric transformations.
- We incorporate equivariant graph layers in order to facilitate inherent robustness against geometric transformations. Together with the graph embedding, our equivariant implicit model significantly improves the reconstruction quality from the existing rotation equivariant implicit method [17].
- We extend our implicit method to achieve a stronger equivariant model that handles more types of similarity transformations simultaneously with guaranteed perfect generalization, including rotation, translation and scaling.

2 Related Work

Implicit 3D representations. Neural implicits have been shown to be highly effective for encoding continuous 3D signals of varying topology [32,30,10]. Its variants have been used in order to reconstruct shapes from a single image [38,52,53], or use weaker supervision for raw point clouds [1,2,3] and 2D views [29,31,26].

Local latent implicit embeddings. ConvONet [34] and IF-Net [11] concurrently propose to learn multi-scale local grid features with convolution layers to improve upon global latent implicit representations. Other variants of grid methods [25,5] takes no global cues, hence restricted by requiring additional priors during inference such as normals [25] or the partial implicit field [5]. The paradigm is extended to adaptive grids or octrees [44,45,47]. Inspired by non-grid local embeddings in point cloud networks [36,48,28,20], we aim for non-grid local implicit methods, which are less explored. The existing non-grid local implicits [22,18] are limited without multi-scale hierarchical designs, and thus restricted to single objects. The pose-sensitivity problem is not addressed in all these local methods. In contrast, we use a hierarchical graph embedding that effectively encodes multi-scale context, as the first local implicit method to address the pose-sensitivity problem.

Pose-sensitivity in implicit functions. As generalization becomes a concern for 3D vision [42,4,9,8], Davies *et al.* [16] first point out that the implicit 3D representations are biased towards canonical orientations. Deng *et al.* [17] introduce a rotation equivariant implicit network VN-ONet that generalises to random unseen rotations, but yet with the restrictions from the global latent. Concurrent to our work, [41,7,54] extend equivariance of implicit reconstruction to SE(3) group for reconstruction and robotic tasks, while our method further handles scaling transformation, and recovers significantly better local details with the graph local embedding design. As grid embeddings are sensitive to rotation, seeking a compatible local latent embedding is a non-trivial problem.

Rotation equivariance with 3D vector features. Equivariance has drawn attention in deep learning models with inductive priors of physical symmetries, e.g., the success of ConvNets are attributed to translation equivariance. Advanced techniques are developed for equivariance to rotation [14,51,46], scale [43,56] and permutation [55,35]. Recently, a new paradigm for rotation equivariance uses 3D vectors as neural features [17,40] with improved effectiveness and efficiency upon methods based on spherical harmonics [51,46,50,49,19]. Shen *et al.* [40] first introduced pure quaternion features that are equivalent to 3D vectors. Deng *et al.* [17] proposed a similar design with improved nonlinear layers. Satorras *et al.* [39] proposed to aggregate vector inputs in graph message passing, but without vector nonlinearities involved. Leveraging on existing work [40,17,39], we introduce hybrid vector and scalar neural features for better performance and efficiency. We also adapt the paradigm for scale equivariance, for the first time in literature.

3 A Definition of Equivariance for Implicit Representations

While equivariance to common geometric transformations is widely studied for explicit representations of 2D and 3D data [13,51,46], the property for implicit representations that encode signals in a function space is more challenging since continuous queries are involved, yet an important problem for 3D reconstruction. We discuss standard 3D implicit functions and then define equivariance for the representation.

3D implicit representations. We build our model on neural 3D occupancy field functions [30], widely used as a shared implicit representation for a collection of 3D shapes. Given an observation $\mathbf{X} \in \mathcal{X}$ of a 3D shape, the conditional implicit representation of the shape, $F(\cdot|\mathbf{X}) : \mathbb{R}^3 \rightarrow [0, 1]$, is a 3D scalar field that maps the 3D Euclidean domain to occupancy probabilities, indicating whether there is a surface point at the coordinate. In this work, we consider $\mathbf{X}$ as a sparse 3D point cloud, such that the information from the observation is indifferent under an arbitrary global transformation, as required for equivariance. For each 3D query coordinate $\vec{p} \in \mathbb{R}^3$, the conditioned implicit representation is in the form of

$$F(\vec{p}|\mathbf{X}) = \Psi(\vec{p}, \vec{z}) = \Psi(\vec{p}, \Phi(\mathbf{X})), \tag{1}$$

where $\Phi(\mathbf{X}) = \vec{z} \in \mathbb{R}^{C_{\vec{z}}}$ is the latent code in the form of a $C_{\vec{z}}$-dimensional vector from the observation $\mathbf{X}$, and Φ is the *latent feature extractor* that encodes the observation data $\mathbf{X}$. Ψ is the implicit decoder implemented using a multi-layered perceptron with

ReLU activations. Occupancy probabilities are obtained by the final sigmoid activation function of the implicit decoder. Following [30,34], the model is trained with binary cross-entropy loss supervised by ground truth occupancy. The underlying shape surface is the 2-manifold $\{\vec{p}|F(\vec{p}|\mathbf{X}) = \tau\}$, where $\tau \in (0, 1)$ is the surface decision boundary.

Preliminary of equivariance. Consider a set of transformations $T_g : \mathcal{X} \to \mathcal{X}$ on a vector space $\mathcal{X}$ for $g \in G$, where G is an abstract group. Formally, $T_g = T(g)$ where T is a representation of group G, such that $\forall g, g' \in G, T(gg') = T(g)T(g')$. In the case that G is the 3D rotation group SO(3), T_g instantiates a 3D rotation matrix for a rotation denoted by g. We say a function $\Xi : \mathcal{X} \to \mathcal{Y}$ is equivariant with regard to group G if there exists $T_g^* : \mathcal{Y} \to \mathcal{Y}$ such that for all $g \in G$: $T_g^* \circ \Xi = \Xi \circ T_g$. We refer to [13,46] for more detailed background theory. Next, we define equivariance for implicit representations as follows:

Definition 1 (Equivariant 3D implicit functions). *Given a group G and the 3D transformations T_g with $g \in G$, the conditioned implicit function $F(\cdot|\mathbf{X})$ is equivariant with regard to G, if*

$$F\big(\cdot\,|T_g(\mathbf{X})\big) = T_g^*\big(F(\cdot|\mathbf{X})\big), \quad \textit{for all } g \in G, \mathbf{X} \in \mathcal{X}. \tag{2}$$

where the transformation T_g^* applied on the implicit function associated to T_g is applying the inverse coordinate transform on query coordinates $T_g^*(F(\cdot|\mathbf{X})) \equiv F(T_g^{-1}(\cdot)|\mathbf{X})$.

Remark 1 *Eq. (2) can be reformulated more intuitively as:*

$$F(\cdot|\mathbf{X}) = F\big(T_g(\cdot)|T_g(\mathbf{X})\big), \quad \textit{for all } g \in G, \mathbf{X} \in \mathcal{X}. \tag{3}$$

Eq. (3) indicates that the equivariance is satisfied if for any observation $\mathbf{X}$ and query $\vec{p}$, the implicit output of $F(\vec{p}|\mathbf{X})$ is *locally* invariant to any T_g applied jointly to $\mathbf{X}$ and $\vec{p}$ in the implicit model.

4 Transformation-robust Graph Local Implicit Representations

Our goal is to design an equivariant implicit function model using local feature embeddings to capture fine details of the 3D geometry. However, existing grid-based local implicit functions are sensitive to geometric transformations such as rotations, thus not suitable for equivariant implicit representations. To address this limitation, we propose a graph-based local embedding which is robust to geometric transformations.

Background: grid local implicit representations. To overcome the limitation of a global latent feature, recent methods, such as ConvONet [34] and IF-Net [11], propose to learn spatially-varying latent features $\mathbf{z_p} = \Phi_{\text{grid}}(\mathbf{p}; \mathbf{X})$. The main idea is to partition the 3D space into a grid and compute latent codes locally. Specifically, these methods formulate the local latent implicit function on the grid as $F_{\text{grid}}(\mathbf{p}|\mathbf{X}) = \Psi(\mathbf{p}, \mathbf{z_p}) = \Psi(\mathbf{p}, \Phi_{\text{grid}}(\mathbf{p}; \mathbf{X}))$, where the *grid local latent extractor* Φ_{grid} is further decomposed as $\Phi_{\text{grid}}(\mathbf{p}; \mathbf{X}) = \psi_{\text{grid}}\left(\mathbf{p}, \phi_{\text{grid}}(\mathbf{X})\right)$. The function ϕ_{grid} is the *grid feature encoder* that learns to generate a 2D or 3D grid-based feature tensor $\mathbf{M}$ from the entirety of point observations $\mathbf{X}$. For each grid location, point features are aggregated for all $\mathbf{x}_i \in \mathbf{X}$ in

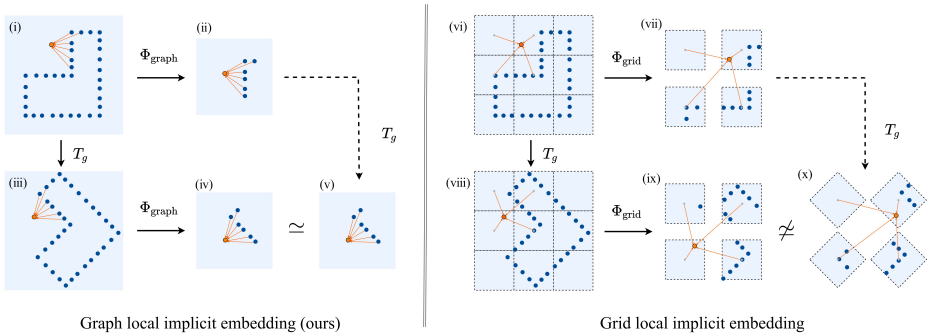

Fig. 2: **Graph (*left*) vs. grid (*right*) local implicit feature embeddings under rotation.** Φ_{graph} extracts k-NN graphs and applies graph convolutions; Φ_{grid} partitions points into regular grids and applies regular convolutions. *Left*: given a point cloud observation $\mathbf{X}$ (*navy*) (i), our method aggregates the local latent feature at any query coordinate $\mathbf{p}$ (*orange*) from a local k-NN graph connecting its neighbours in $\mathbf{X}$ (ii). Moreover, when a transformation T_g, *e.g.* rotation, is applied to the shape, (iv) the constructed local graph is in the same structure as (v) applying T_g to the graph from untransformed data. *Right*: in contrast, (vi) visualizes discretized grid features. (vii) The off-the-grid query location $\mathbf{p}$ interpolates the neighboring on-grid features. However, (viii) with T_g applied to the raw observation, often (ix) the sub-grid point patterns for the local features are different from (x) applying T_g to the untransformed local grids. This makes the grid local implicit models sensitive to transformations such as rotation.

the corresponding bin. Convolutional layers are applied to the grid-based $\mathbf{M}$ to capture multi-scale information and maintain translation equivariance. Given the local 3D feature tensor $\mathbf{M}$, the *local latent aggregator* ψ_{grid} computes local latent feature $\mathbf{z}_\mathbf{p}$ on any off-the-grid query coordinate $\mathbf{p}$ using a simple trilinear interpolation. We refer to [25,5] for other variants of grid-based representations using purely local information without global cues, while restricted by requiring additional priors during inference such as the normals. Overall, grid partitioning is not robust to general transformations such as rotations, especially when the sub-grid structure is considered for the resolution-free implicit reconstruction. Fig. 2 (*right*) shows an illustration of this limitation, which we will address in our method.

4.1 Graph-structured local implicit feature embeddings

We propose to use graphs as a non-regular representation, such that our local latent feature function is robust to these transformations and free from feature grid resolutions. The graph-local implicit function extends the standard form of Eq. (1) to

$$F_{\text{graph}}(\mathbf{p}|\mathbf{X}) = \Psi(\mathbf{p}, \mathbf{z}_\mathbf{p}) = \Psi(\mathbf{p}, \Phi_{\text{graph}}(\mathbf{p}; \mathbf{X})) = \Psi(\mathbf{p}, \psi\left(\mathbf{p}, \phi(\mathbf{X})\right)). \tag{4}$$

Φ_{graph} is a deep network that extracts *local* latent features $\mathbf{z}_\mathbf{p}$ on the graph, composed of two sub-networks ϕ and ψ. The *point feature encoder* ϕ maps the point set $\mathbf{X} =$

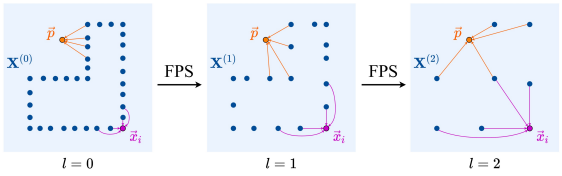

Fig. 3: **Multi-scale design,** with enlarged receptive fields of local k-NN graphs for the graph point encoder ϕ (*orange*) and the graph local latent aggregator ψ (*violet*).

$\{\mathbf{x}_i\}$ to the associated features $\{\mathbf{h}_i\}$, and is invariant to the sampled query location $\mathbf{p}$. The *graph local latent feature aggregator* ψ propagates the input point feature to the query coordinate $\mathbf{p}$. Unlike grid-based methods, we directly aggregate local information from the point cloud feature $\{(\mathbf{x}_i, \mathbf{h}_i)\}$ without an intermediate grid feature tensor. In particular, we construct a local k-nearest neighbor (k-NN) graph $(\mathcal{V}, \mathcal{E})$ for every query point $\mathbf{p}$, where the vertices $\mathcal{V} = \mathbf{X} \cup \{\mathbf{p}\}$ include the point set elements and the query coordinate. The edges $\mathcal{E} = \{(\mathbf{p}, \mathbf{x}_i)\}$ are between the query point $\mathbf{p}$ and its k-NN points from the observation point set $\mathbf{x}_{i'} \in \mathcal{N}_k(\mathbf{p}, \mathbf{X})$, with $\mathcal{N}_k(\mathbf{p}, \mathbf{X})$ denoting the set of the k-NN points of $\mathbf{p}$ from $\mathbf{X}$. Last, with graph convolutions, we aggregate into the local feature vector $\mathbf{z}_\mathbf{p}$ the point features of the neighbors of $\mathbf{p}$. We adopt a simple spatial graph convolution design in the style of Message Passing Neural Network (MPNN) [23], which is widely used for 3D shape analysis [48,24]. For each neighboring point $\mathbf{x}_{i'}$ from the query $\mathbf{p}$, messages are passed through a function η as a shared two-layer ReLU-MLP, where the inputs are the point features $\mathbf{h}_{i'}$ and the query coordinate $\mathbf{p}$ as node feature as well as the displacement vector $\mathbf{x}_{i'}$ - $\mathbf{p}$ as the edge feature, followed by a permutation-invariant aggregation AGGRE over all neighboring nodes, e.g., max- or mean-pooling:

$$\mathbf{z}_\mathbf{p} = \underset{i'}{\text{AGGRE}} \ \eta(\mathbf{p}, \mathbf{h}_{i'}, \mathbf{x}_{i'} - \mathbf{p}). \tag{5}$$

For each neighboring point as a graph node, the edge function η take as inputs, the query coordinate $\mathbf{p}$, the node point feature $\mathbf{h}_{i'}$, and the relative position $\mathbf{x}_{i'} - \mathbf{p}$.

As all the graph connections are relative between vertices, the local latent feature aggregation is robust to transformations like rotations, as illustrated in Fig. 2 (*left*).

4.2 Learning multi-scale local graph latent features

To capture the context of the 3D geometry at multiple scales, both ConvONet [34] and IF-Net [11] rely on a convolutional U-Net [37], with progressively downsampled and then upsampled feature grid resolutions to share neighboring information at different scales. Our graph model enables learning at multiple scales by farthest point sampling (FPS). That is, we downsample the point set $\mathbf{X}$ to $\mathbf{X}^{(l)}$ at sampling levels $l = 1, \ldots, L$ with a progressively smaller cardinality $|\mathbf{X}^{(l)}| < |\mathbf{X}^{(l-1)}|$, where $\mathbf{X}^{(0)} = \mathbf{X}$ is the original set.

Moreover, we use a graph encoder for the point encoder ϕ, instead of PointNet [35]. This way, without involving regular grid convolutions, we can still model local features and facilitate a translation-equivariant encoder, which is beneficial in many scenarios,

especially for learning scene-level implicit surfaces [34]. Next, we sketch the multi-scale graph point encoder and latent feature aggregator, with Fig. 3 as a conceptual illustration.

The *graph point encoder* ϕ learns point features $\{\mathbf{h}_i^{(l)}\}$ for the corresponding points $x_i \in \mathbf{X}^{(l)}$ at each sampling level $l = 0 \ldots, L$. The encoder starts from the initial sampling level $l = 0$ where the input features are the raw coordinates. At each sampling level, a graph convolution is applied to each point to aggregate message from its local k-nearest neighbor point features, followed by an FPS operation to the downsampled level $l + 1$. The graph convolution is similar to that in Eq. (5), with the point features from both sides of the edge and the relative position as inputs. The graph convolutions and FPS downsampling are applied until the coarsest sampling level $l = L$. Then the point features are sequentially upsampled back from $l = L$ to $l = L - 1$, until $l = 0$. At each sampling level l, the upsampling layer is simply one linear layer followed by ReLU activation. For each point, the input of the upsampling layer is the nearest point feature from the last sampling level $l + 1$, and the skip-connected feature of the same point at the same sampling level from the downsampling stage. Thus far, we obtain multi-scale point features $\{(\mathbf{x}_i, \mathbf{h}_i^{(l)})\}$ for $\mathbf{x}_i \in \mathbf{X}^{(l)}$ at sampling levels $l = 0, \ldots, L$, as the output from the graph point encoder ϕ.

For the *graph latent feature aggregator* ψ, at each query coordinate $\mathbf{p}$, we use graph convolutions to aggregate the k-neighboring features $\{(\mathbf{x}_i, \mathbf{h}_i^{(l)})\}$ at different sampling levels l, as described in Sec 4.1. The aggregated features from all sampling levels l are concatenated to yield the local latent vector $\mathbf{z}_\mathbf{p}$ as output. The detailed formulations of ϕ and ψ are provided in the supplementary material.

5 Equivariant Graph Implicit Functions

The local graph structure of the proposed implicit function, with $\mathbf{X} \cup \{\mathbf{p}\}$ as the set of vertices, is in line with the requirement of equivariance in Sec. 3 and Eq. (3). As a result, the local graph implicit embedding can be used for an equivariant model to achieve theoretically guaranteed generalization to unseen transformations. To do this, we further require all the graph layers in the latent extractor Φ_{graph} to be equivariant in order to obtain equivariant local latent feature. We present the equivariant layers for 3D rotation group $G = SO(3)$, a difficult case for implicit functions [16].

To build the equivariant model from equivariant layers, we additionally remove the query coordinate input $\mathbf{p}$ to ensure the implicit decoder Ψ spatially invariant in Eq. (4), as the local spatial information is already included in the latent $\mathbf{z}_\mathbf{p}$. See Appendix A.1 for details. We also extend the method to other similarity transformations, such as translation and scaling. See Appendix A.2.

5.1 Hybrid feature equivariant layers

Our equivariant layers for the graph convolution operations is inspired by recent methods [17,40] that lift from a regular neuron feature $h \in \mathbb{R}$ to a vector $\mathbf{v} \in \mathbb{R}^3$ to encode rotation, and the list of 3D vector features $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{C_\mathbf{v}}]^\top \in \mathbb{R}^{C_\mathbf{v} \times 3}$ that substitutes the regular features $\mathbf{h} \in \mathbb{R}^{C_h}$. However, using only vector features in the

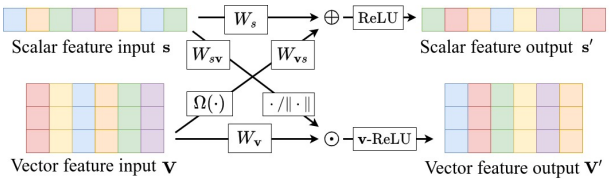

Fig. 4: **Hybrid feature equivariant layers.** Visualization of how vector and scalar features share information in linear and nonlinear layers. Vector features go through an invariant function Ω that is added to the scalar part. Scalar features are transformed with normalizing $\cdot/\|\cdot\|$ to scale the vector feature channels.

network is non-optimal for both effectiveness and efficiency, with highly regularized linear layers and computation-demanding nonlinearity projections.

To this end, we extend the method and propose hybrid features $\{\mathbf{s}, \mathbf{V}\}$ to replace the regular feature $\mathbf{h}$, where vector features $\mathbf{V}$ encode rotation equivariance, and scalar features $\mathbf{s} \in \mathbb{R}^{C_s}$ are rotation-invariant. In practice, hybrid features show improved performance and computation efficiency by transferring some learning responsibility to the scalar features through more powerful and efficient standard neural layers.

Linear layers. For input hybrid hidden feature $\{\mathbf{s}, \mathbf{V}\}$ with $\mathbf{s} \in \mathbb{R}^{C_s}$, and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{C_\mathbf{v}}]^\top \in \mathbb{R}^{C_\mathbf{v} \times 3}$, we define a set of weight matrices, $\mathbf{W}_s \in \mathbb{R}^{C'_s \times C_s}$, $\mathbf{W}_\mathbf{v} \in \mathbb{R}^{C'_\mathbf{v} \times C_\mathbf{v}}$, $\mathbf{W}_{s\mathbf{v}} \in \mathbb{R}^{C'_\mathbf{s} \times C_s}$, and $\mathbf{W}_{\mathbf{v}s} \in \mathbb{R}^{C'_s \times C_\mathbf{v}}$ for the linear transformation, with information shared between scalar and vector features in the inputs. The resulting output features $\{\mathbf{s}', \mathbf{V}'\}$ become:

$$\mathbf{s}' = \mathbf{W}_s \mathbf{s} + \mathbf{W}_{\mathbf{v}s}\, \Omega(\mathbf{V}) \tag{6}$$

$$\mathbf{V}' = \mathbf{W}_\mathbf{v} \mathbf{V} \odot (\mathbf{W}_{s\mathbf{v}}\, \mathbf{s} \,/\, \|\mathbf{W}_{s\mathbf{v}}\, \mathbf{s}\|) \tag{7}$$

where $\odot$ is channel-wise multiplication between $\mathbf{W}_\mathbf{v}\mathbf{V} \in \mathbb{R}^{C'_\mathbf{s} \times 3}$ and $\mathbf{W}_{s\mathbf{v}}\mathbf{h} \in \mathbb{R}^{C'_\mathbf{v} \times 1}$, and the normalized transformed scalar feature $\mathbf{W}_{s\mathbf{v}}\, \mathbf{s} \,/\, \|\mathbf{W}_{s\mathbf{v}}\, \mathbf{s}\|$ learns to scale the output vectors in each channel; $\Omega(\cdot)$ is the invariance function that maps the rotation equivariant vector feature $\mathbf{V} \in \mathbb{R}^{C_\mathbf{v} \times 3}$ to the rotation-invariant scalar feature $\Omega(\mathbf{V}) \in \mathbb{R}^{C_\mathbf{v}}$, to be added on the output scalar feature. The design of the invariance function $\Omega(\cdot)$ is introduced later in Eq. (9).

Nonlinearities. Nonlinearities apply separately to scalar and vector features. For scalar features, it is simply a ReLU$(\cdot)$. For vector features, the nonlinearity $\mathbf{v}$-ReLU$(\cdot)$ adopts the design from Vector Neurons [17]: the vector feature $\mathbf{v}_c$ at each channel c takes an inner-product with a learnt direction $\mathbf{q} = [\mathbf{W}_\mathbf{q}\mathbf{V}]^\top \in \mathbb{R}^3$ from a linear layer $\mathbf{W}_\mathbf{q} \in \mathbb{R}^{1 \times C_\mathbf{v}}$. If the inner-product is negative, $\mathbf{v}_c$ is projected to the plane perpendicular to $\mathbf{q}$.

$$[\mathbf{v}\text{-ReLU}(\mathbf{V})]_c = \begin{cases} \mathbf{v}_c & \text{if } \left\langle \mathbf{v}_c, \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\rangle \geq 0, \\ \mathbf{v}_c - \left\langle \mathbf{v}_c, \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\rangle \frac{\mathbf{q}}{\|\mathbf{q}\|}. & \text{otherwise.} \end{cases} \tag{8}$$

Invariance layer. The invariance function $\Omega(\cdot)$ maps the rotation equivariant vector feature $\mathbf{V} \in \mathbb{R}^{C_\mathbf{v} \times 3}$ to a rotation-invariant scalar feature $\Omega(\mathbf{V}) \in \mathbb{R}^{C_\mathbf{v}}$ with the same

Table 1: **ShapeNet implicit reconstruction from sparse noisy point clouds.** The grid resolutions are marked for ConvONet and IF-Net.

	SO(3) equiv.	Mean IoU ↑	Chamfer-ℓ_1 ↓	Normal consist. ↑
ONet	×	0.736	0.098	0.878
ConvONet-2D (3×64^2)	×	0.884	0.044	0.938
ConvONet-3D (32^3)	×	0.870	0.048	0.937
IF-Net (128^3)	×	0.887	0.042	0.941
GraphONet (ours)	×	**0.904**	**0.038**	**0.946**
VN-ONet	✓	0.694	0.125	0.866
E-GraphONet-SO(3) (ours)	✓	0.890	0.041	0.936

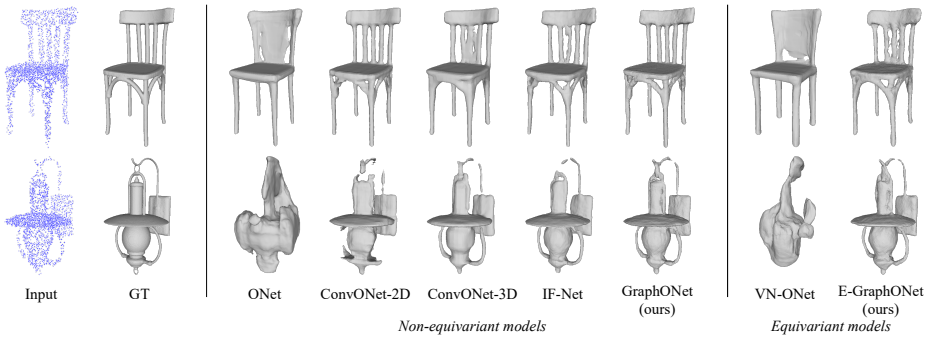

Fig. 5: **ShapeNet object reconstruction in canonical space.** GraphONet is among the best performing non-equivariant implicit representation methods; E-GraphONet significantly improves on the existing equivariant method VN-ONet [17].

channel dimension $C_{\mathbf{v}}$. At each layer $c = 1, \ldots, C_{\mathbf{v}}$, $[\Omega(\mathbf{V})]_c \in \mathbb{R}$ takes the inner product of $\mathbf{v}_c$ with the channel-averaged direction:

$$[\Omega(\mathbf{V})]_c = \left\langle \mathbf{v}_c, \frac{\overline{\mathbf{v}}}{\|\overline{\mathbf{v}}\|} \right\rangle, \tag{9}$$

where $\overline{\mathbf{v}} = \frac{1}{C_{\mathbf{v}}} \sum_{c'=1}^{C_{\mathbf{v}}} \mathbf{v}_{c'}$ is the channel-averaged vector feature. One can verify that applying any rotation on the vector feature does not change the inner product. Our Ω adopts from [17] with modification. Ours is parameter-free using averaged direction, while [17] learns this vector.

The invariance function is applied in each linear layer in Eq. (7) to share the information between vector and scalar parts of the feature. In addition, at the end of equivariant graph feature network Φ_{graph}, $\Omega(\mathbf{V})$ is concatenated with the scalar feature as the final invariant local latent feature $\mathbf{z}_{\mathbf{p}} = \Omega(\mathbf{V}) \| \mathbf{s}$, as to be locally invariant with transformed $\mathbf{p}$ and $\mathbf{X}$ in line with Eq. (3).

6 Experiments

We experiment on implicit surface reconstruction from sparse and noisy point observations. In addition, we evaluate the implicit reconstruction performance under random

Table 2: **Implicit surface reconstruction with random rotation, IoU↑.** The *left* table evaluates non-equivariant methods, and the *right* one compares the best performing model with equivariant methods. *I* denotes canonical pose and *SO(3)* random rotation. *I / SO(3)* denotes training with canonical pose and test with random rotation, and so on. *: models not re-trained with augmentation due to guaranteed equivariance.

training / test	*I / I*	*I / SO*(3)	*SO*(3) / *SO*(3)
ONet	0.742	0.271 [-0.471]	0.592 [-0.150]
ConvONet-2D	0.884	0.568 [-0.316]	0.791 [-0.093]
ConvONet-3D	0.870	0.761 [-0.109]	0.838 [-0.032]
GraphONet (ours)	**0.904**	**0.846 [-0.058]**	**0.887 [-0.017]**

training / test	equiv.	*I / I*	*I / SO*(3)	*SO*(3) / *SO*(3)
GraphONet (ours)	×	**0.904**	0.846 [-0.058]	0.887 [-0.017]
VN-ONet	SO(3)	0.694	0.694 [-0.000]	0.694* [-0.000]
E-GraphONet (ours)	SO(3)	0.890	**0.890 [-0.000]**	**0.890*** **[-0.000]**

transformations of rotation, translation and scaling. Our method is referred to as Graph Occupancy networks, or GraphONet, while E-GraphONet is the equivariance model with 3 variants: SO(3), SE(3) and similarity transformations (Sim.).

Implementation details. We implement our method using PyTorch [33]. The number of neighbours in k-NN graph is set as 20. For the multi-scale graph implicit encoder, we take $L = 2$, *i.e.*, the point set is downsampled twice with farthest point sampling (FPS) to 20% and 5% of the original cardinality respectively. The permutation invariant function AGGRE adopts mean-pooling for vector features and max-pooling for scalar features. We use an Adam optimizer [27] with $\gamma = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Main experiments are conducted on the ShapeNet [6] dataset with human designed objects, where the train/val/test splits follow prior work [12,34] with 13 categories. Following [34], we sample 3000 surface points per shape and apply Gaussian noise with 0.005 standard deviation. More details are provided in the Appendix. Code is available at https://github.com/yunlu-chen/equivariant-graph-implicit.

6.1 Canonical-posed object reconstruction

We experiment on ShapeNet object reconstruction following the setups in [34]. For quantitative evaluation in Table 1, we evaluate the IoU, Chamfer-ℓ_1 distance and normal consistency, following [30,34]. The qualitative results are shown in Fig. 5. Our GraphONet outperforms the state-of-the-arts methods ConvONet [34] and IF-Net [11], as our graph-based method aggregates local feature free of spatial grid resolution and captures better local details. IF-Net is better than ConvONet but at large memory cost per batch with 128^3 resolution grid feature. For rotation equivariant models, our E-GraphONet significantly outperforms VN-ONet [17], benefiting from the graph local features.

6.2 Evaluation under geometric transformations

Rotation. First, we investigate how implicit models perform under random rotations, which is challenging for neural implicits [16]. In Table 2 left, GraphONet shows the smallest performance drop among all non-equivariant methods under rotations, either with (*SO(3) / SO(3)*) or without augmentation (*I / SO(3)*) during training, since the graph structure is more robust to rotations. ConvONet [34]-2D is more sensitive than the 3D

Table 3: **Implicit surface reconstruction performance under various types of seen/unseen transformations, mIoU↑.**

Transformation(s)	equiv.	-	*translation*		*scale*		*rot. & transl.*		*rot. & scale*		*transl. & scale*		*all*	
Training augmentation		-	×	✓	×	✓	×	✓	×	✓	×	✓	×	✓
ONet	×	0.738	0.221	0.716	0.423	0.685	0.154	0.585	0.235	0.591	0.202	0.713	0.121	0.573
ConvONet-2D (3×128^2)	×‡	0.882	0.791	0.878	0.812	0.850	0.532	0.771	0.542	0.789	0.723	0.838	0.481	0.728
ConvONet-3D (64^3)	×‡	0.861	0.849	0.856	0.797	0.837	0.759	0.836	0.742	0.832	0.771	0.835	0.721	0.803
GraphONet (ours)	×‡	**0.901**	**0.884**	**0.898**	0.857	**0.893**	0.837	0.881	0.798	0.880	0.852	**0.888**	0.798	0.874
VN-ONet	SO(3)	0.682	0.354	0.667	0.516	0.662	0.357	0.658	0.511	0.666	0.360	0.638	0.309	0.615
E-GraphONet (ours)	SO(3)‡	0.887	0.823	0.876	0.824	0.880	0.825	0.877	0.825	**0.882**	0.726	0.872	0.729	0.870
E-GraphONet (ours)	SE(3)	0.884	**0.884**	0.884*	0.840	0.880	**0.884**	**0.884***	0.838	0.880	0.841	0.878	0.840	0.878
E-GraphONet (ours)	Sim.†	0.882	0.882	0.882*	**0.882**	0.882*	0.882	0.882*	**0.882**	**0.882***	**0.882**	0.882*	**0.882**	**0.882***

† Similarity transformation group. ‡ The (graph) convolution subnetwork is translation equivariant. * with no augmentation due to guaranteed equivariance.

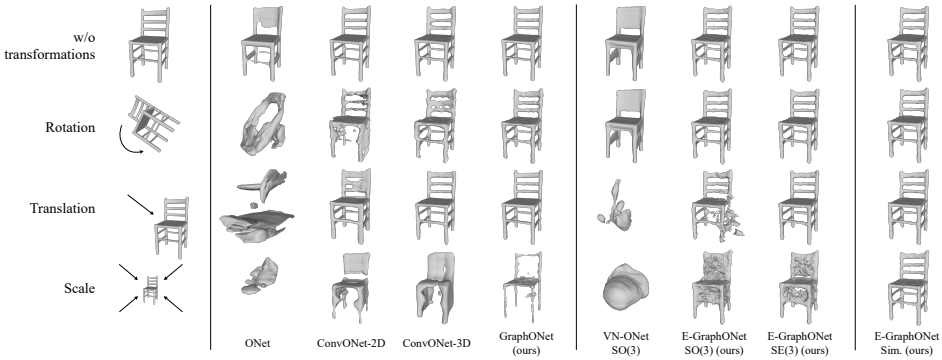

Fig. 6: **Implicit surface reconstruction under *unseen* transforms.** Visualizing back-transformed shapes. Shapes are scaled by a factor of 0.25, and translation is from the unit cube center to the corner. E-GraphONet-Sim is robust to all similarity transformations. See the appendix video for reconstructions under different poses and scales.

version, as 3D rotation would lead to highly distinct 2D projections. In Table 2 right, E-GraphONet, equipped with equivariant layers, achieves better performance under random rotations, even when the non-equivariant methods are trained with augmentation. It also outperforms the previous equivariant method VN-ONet [17] by a large margin.

Scale, translation and combinations. We evaluate how implicit methods perform under various similarity transformations besides SO(3) rotation, including scale, translation and combinations. We apply random scales and rotations in a bounded unit cube $[-0.5, 0.5]^3$, as assumed by grid methods, and set the canonical scale to be half of the cube. ConvONet resolutions are doubled to keep the effective resolution. Random scaling and translation are added under the constraint of the unit bound, with the minimum scaling factor of 0.2.

From the results in Table 3, GraphONet is more robust to transformations than other non-equivariant models. For equivariant models, VN-ONet and the SO(3) E-GraphONet models perform poorly on other types of transformations, as they are optimized towards the rotation around origin only. Similarly, the SE(3) E-GraphONet does not generalize to scaling. Our model with full equivariance performs well on all similarity transformations

Table 4: **Parameter- and data-efficiency.** Evaluated on the full test set with 5 runs of 130 randomly sampled training examples.

	#param.	IoU↑
ConvONet-2D	2.0×10^6	0.727 ±0.009
ConvONet-3D	1.1×10^6	0.722 ±0.008
GraphONet (ours)	1.9×10^5	0.867 ±0.004
E-GraphONet (ours)	$\mathbf{6.5 \times 10^4}$	**0.873** ±0.002

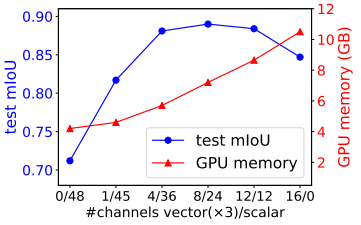

Fig. 7: **Ablation on hybrid feature channels.** Mixed vectors and scalars are more effective and efficient than pure vectors.

with numerically the same performance. Fig 6 shows qualitative examples, where our E-GraphONet-Sim handles all types of unseen similarity transformations.

6.3 Analysis

We show some ablation experiments while more results are provided in the Appendix.

Learning from very few training examples. We show that our graph method is both parameter- and data-efficient, and the transform-robust modeling inherently benefits generalization. As reported in Table 4, we use less than 10% of the parameters of ConvONet as the graph conv kernel shares parameters for all directions. We evaluate the test set performance when training on only 130 examples - 10 per class - instead of the full training set size of 30661. While ConvONets fail to achieve good performance, GraphONets does not drop by far from the many-shot results in Table 1. The E-GraphONet demonstrates even better performance, with more parameter-sharing from the equivariance modeling, indicating better power of generalization.

Ablation on vector and scalar feature channels. We validate our design of hybrid features by experimenting different ratio of vector and scalar channels. We constrain in total 48 effective channels, with one vector channel counted as three scalars. In Fig. 7, using both vectors and scalars with a close-to-equal ratio of effective channels obtains higher performance with less memory cost than using pure vectors, i.e., in the Vector Neurons [17]. This indicates the expressive power of the scalar neuron functions. We provide additional results for non-graph-based equivariant models in the Appendix.

6.4 Scene-level reconstructions

In addition to the ability of handling object shape modeling under transformations, our graph implicit functions also scale to scene-level reconstruction. We experiment on two datasets: (i) *Synthetic Rooms* [34], a dataset provided by [34], with rooms constructed with walls, floors, and ShapeNet objects from five classes: chair, sofa, lamp, cabinet and table.

Table 5: **Indoor scene reconstructions.**

Dataset	**Synthetic room**			**ScanNet**
	IoU↑	Chamfer↓	Normal↑	Chamfer↓
ONet	0.514	0.135	0.856	0.546
ConvONet-2D (3×128^2)	0.802	0.038	0.934	0.162
ConvONet-3D (64^3)	0.847	0.035	0.943	0.067
GraphONet (ours)	**0.883**	**0.032**	**0.944**	**0.061**
E-GraphONet (ours)	0.851	0.035	0.934	0.069

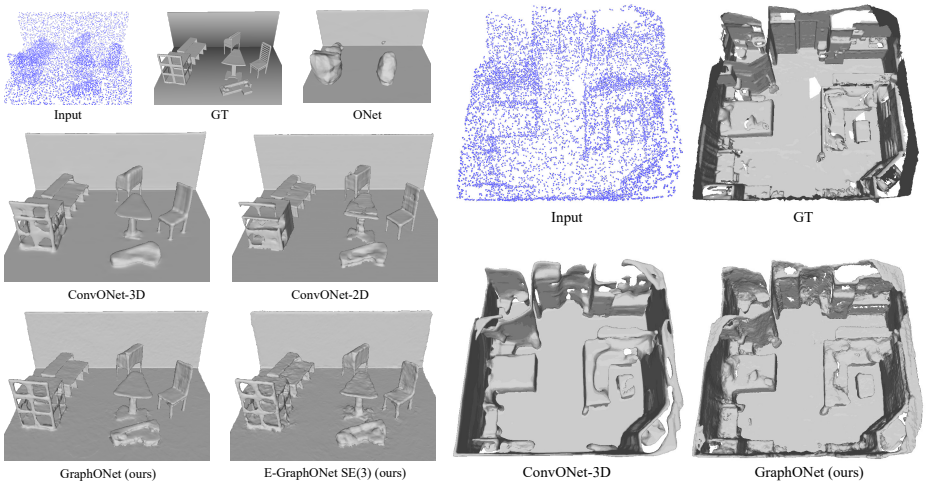

Fig. 8: **Indoor scene reconstructions on Synthetic rooms (*left*) and ScanNet (*right*).** Our GraphONet produces shaper edges and better finer details.

(ii) *ScanNet* [15], a dataset of RGB-D scans of real-world rooms for testing synthetic-to-real transfer performance.

We train and evaluate our model on Synthetic room dataset [34] using 10,000 sampled points as input. The quantitative results are shown in Table 5 and qualitative results in Fig. 8 (left). Our GraphONet performs better than ConvONets [34] at recovering detailed structures. We evaluate the SE(3) variant of our equivariance model, and it performs generally well, but less smooth at flat regions. In addition, we evaluate the model transfer ability of our method on ScanNet [15] with the model trained on synthetic data, for which we report the Chamfer measure in Table 5. Our GraphONet ourperforms other methods. Fig. 8 (right) shows a qualitative example of our reconstructions.

7 Conclusion

In this paper, we introduce graph implicit functions, which learn local latent features from k-NN graph on sparse point set observations, enabling reconstruction of 3D shapes and scenes in fine detail. By nature of graphs and in contrast to regular grid representations, the proposed graph representations are robust to geometric transformations. What is more, we extend the proposed graph implicit functions with hybrid feature equivariant layers, thus guarantee theoretical equivariance under various similarity transformations, including rotations, translations and scales, and obtain models that generalize to arbitrary and unseen transformations.

Acknowledgement This research was supported in part by SAVI/MediFor project, ERC Starting Grant Scan2CAD (804724), EPSRC programme grant Visual AI EP/T028572/1, and National Research Foundation Singapore and DSO National Laboratories under its AI Singapore Programme (AISG Award No: AISG2-RP-2020-016). We thank Angela Dai for the video voice over.

References

1. Atzmon, M., Haim, N., Yariv, L., Israelov, O., Maron, H., Lipman, Y.: Controlling neural level sets. arXiv preprint arXiv:1905.11911 (2019) 3
2. Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2565–2574 (2020) 3
3. Atzmon, M., Lipman, Y.: Sal++: Sign agnostic learning with derivatives. arXiv preprint arXiv:2006.05400 (2020) 3
4. Bautista, M.A., Talbott, W., Zhai, S., Srivastava, N., Susskind, J.M.: On the generalization of learning-based 3d reconstruction. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2180–2189 (2021) 4
5. Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.: Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In: European Conference on Computer Vision. pp. 608–625. Springer (2020) 3, 6
6. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 11
7. Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., Daniilidis, K.: Se (3)-equivariant attention networks for shape reconstruction in function space. arXiv preprint arXiv:2204.02394 (2022) 4
8. Chen, Y., Fernando, B., Bilen, H., Mensink, T., Gavves, E.: Neural feature matching in implicit 3d representations. In: International Conference on Machine Learning. pp. 1582–1593. PMLR (2021) 4
9. Chen, Y., Hu, V.T., Gavves, E., Mensink, T., Mettes, P., Yang, P., Snoek, C.G.: Pointmixup: Augmentation for point clouds. arXiv preprint arXiv:2008.06374 (2020) 4
10. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019) 1, 3
11. Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3d shape reconstruction and completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6970–6981 (2020) 2, 3, 5, 7, 11, 26
12. Choy, C.B., Gwak, J., Savarese, S., Chandraker, M.: Universal correspondence network. arXiv preprint arXiv:1606.03558 (2016) 11
13. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: International conference on machine learning. pp. 2990–2999. PMLR (2016) 4, 5
14. Cohen, T.S., Welling, M.: Steerable cnns. arXiv preprint arXiv:1612.08498 (2016) 4
15. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017) 14
16. Davies, T., Nowrouzezahrai, D., Jacobson, A.: On the effectiveness of weight-encoded neural implicit 3d shapes. arXiv preprint arXiv:2009.09808 (2020) 2, 4, 8, 11
17. Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., Guibas, L.: Vector neurons: A general framework for so (3)-equivariant networks. arXiv preprint arXiv:2104.12229 (2021) 2, 3, 4, 8, 9, 10, 11, 12, 13, 18, 22, 23
18. Erler, P., Guerrero, P., Ohrhallinger, S., Mitra, N.J., Wimmer, M.: Points2surf learning implicit surfaces from point clouds. In: European Conference on Computer Vision. pp. 108–124. Springer (2020) 2, 3
19. Fuchs, F., Worrall, D., Fischer, V., Welling, M.: Se (3)-transformers: 3d roto-translation equivariant attention networks. Advances in Neural Information Processing Systems **33** (2020) 4

20. Fujiwara, K., Hashimoto, T.: Neural implicit embedding for point cloud analysis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11734–11743 (2020) 3
21. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Deep structured implicit functions. arXiv preprint arXiv:1912.06126 (2019) 2
22. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Local deep implicit functions for 3d shape. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4857–4866 (2020) 3
23. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International conference on machine learning. pp. 1263–1272. PMLR (2017) 7
24. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019) 7
25. Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3d scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 2, 3, 6
26. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1251–1261 (2020) 3
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 11, 22
28. Li, J., Chen, B.M., Lee, G.H.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9397–9406 (2018) 3
29. Liu, S., Saito, S., Chen, W., Li, H.: Learning to infer implicit surfaces without 3d supervision. In: Advances in Neural Information Processing Systems. pp. 8295–8306 (2019) 3
30. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019) 1, 3, 4, 5, 11, 22
31. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3504–3515 (2020) 3
32. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019) 1, 3
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019) 11, 22
34. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 523–540. Springer (2020) 2, 3, 5, 7, 8, 11, 13, 14, 22, 23, 24, 26
35. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017) 4, 7
36. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017) 3
37. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015) 7

38. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2304–2314 (2019) 3
39. Satorras, V.G., Hoogeboom, E., Welling, M.: E(n) equivariant graph neural networks. arXiv preprint arXiv:2102.09844 (2021) 4
40. Shen, W., Zhang, B., Huang, S., Wei, Z., Zhang, Q.: 3d-rotation-equivariant quaternion neural networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16. pp. 531–547. Springer (2020) 4, 8, 18
41. Simeonov, A., Du, Y., Tagliasacchi, A., Tenenbaum, J.B., Rodriguez, A., Agrawal, P., Sitzmann, V.: Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 6394–6400. IEEE (2022) 4
42. Sitzmann, V., Chan, E., Tucker, R., Snavely, N., Wetzstein, G.: Metasdf: Meta-learning signed distance functions. Advances in Neural Information Processing Systems **33**, 10136–10147 (2020) 4
43. Sosnovik, I., Szmaja, M., Smeulders, A.: Scale-equivariant steerable networks. In: International Conference on Learning Representations (2019) 4
44. Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11358–11367 (2021) 3
45. Tang, J.H., Chen, W., Wang, B., Liu, S., Yang, B., Gao, L., et al.: Octfield: Hierarchical implicit functions for 3d modeling. Advances in Neural Information Processing Systems **34**, 12648–12660 (2021) 3
46. Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., Riley, P.: Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arXiv preprint arXiv:1802.08219 (2018) 4, 5, 18
47. Wang, P.S., Liu, Y., Tong, X.: Dual octree graph networks for learning adaptive volumetric shape representations. arXiv preprint arXiv:2205.02825 (2022) 3
48. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019) 3, 7
49. Weiler, M., Geiger, M., Welling, M., Boomsma, W., Cohen, T.: 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In: NeurIPS (2018) 4
50. Weiler, M., Hamprecht, F.A., Storath, M.: Learning steerable filters for rotation equivariant cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 849–858 (2018) 4
51. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic networks: Deep translation and rotation equivariance. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5028–5037 (2017) 4
52. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In: Advances in Neural Information Processing Systems. pp. 492–502 (2019) 3
53. Xu, Y., Fan, T., Yuan, Y., Singh, G.: Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry. In: European Conference on Computer Vision. pp. 248–263. Springer (2020) 3
54. Yuan, Y., Nüchter, A.: An algorithm for the se (3)-transformation on neural implicit maps for remapping functions. IEEE Robotics and Automation Letters (2022) 4
55. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep sets. arXiv preprint arXiv:1703.06114 (2017) 4
56. Zhu, W., Qiu, Q., Calderbank, R., Sapiro, G., Cheng, X.: Scale-equivariant neural networks with decomposed convolutional filters. arXiv preprint arXiv:1909.11193 (2019) 4

A Equivariance: Discussions and Proofs

A.1 From layer equivariance to model equivariance

We review the definition of the equivariance of the implicit function model in Section 3. In Eq. (3), we show that equivariance is satisfied if the implicit function is locally invariant to any T_g applied jointly to the observation $\mathbf{X}$ and the query $\mathbf{p}$, for any $\mathbf{X}$ and $\mathbf{p}$. Invariance is a special case of equivariance, where the transformation T_g^* on the output domain is the identity function, i.e., the output is invariant regardless of the input transformation T_g.

Here we clarify the use of layer equivariance in creating the equivariant implicit function model. As in Eq. (4), the graph implicit function model F_{graph} is composed of the graph latent feature extractor Φ_{graph} and the implicit decoder Ψ, where Φ_{graph} can be further decomposed to the point encoder ϕ and the graph local latent aggregator ψ. We integrate the equivariant layers in ϕ and ψ that process the input point set $\mathbf{X}$ and the queries $\mathbf{p}$. As shown by the literature, the composition of two equivariant functions is also an equivariant function [46]. Thus, the graph local feature extractor Φ_{graph} that stacks sequential equivariant graph layers in ϕ and ψ is equivariant. Note that all the operations other than the graph convolutions involved in ϕ and ψ, such as the k-NN graph extration and the farthest-point-sampling for the multi-scale feature, are equivariant to the similarity transformations as well. At the end of all the equivariant graph layers in ϕ and ψ, we apply the invariance function Ω to obtain the locally invariant latent feature. The implicit decoder Ψ, which processes the invariant local latent feature and predicts the occupancy probability, is simply a standard ReLU-MLP as the non-equivariant implicit models. We do not include the query coordinate input $\mathbf{p}$ in the implicit decoder Ψ in order to satisfy translation equivariance, while the local latent feature already contains the position information. Since the local latent feature is locally invariant, the output is locally invariant as well, which satisfies Eq. (3). Thus far, we have shown how the equivariant graph layers help to build the equivariant implicit function model.

A.2 Extension to translation and scale equivariance

While existing vector-based equivariance methods in 3D vision [17,40] apply only to the $\mathrm{SO}(3)$ group[5], we extend our method to be equivariant to the similarity transformation group that further includes translation and scale transformations as subgroups.

Translation. The local graph structure is robust to rotation by design. The method can further achieve numerically guaranteed translation equivariance simply by removing the absolute coordinates input $\mathbf{p}$ from the graph layers in Eq. (5), keeping only the relative positions as the spatial cue.

Scale. As vectors hold scale information from their norms, we extend the method for scale equivariance by modifying to normalize the invariance function $\Omega(\mathbf{V}) \leftarrow \Omega(\mathbf{V})/\|\Omega(\mathbf{V})\|$ based on Eq. (9). Likewise, in each layer the scalar features are scale invariant and the vectors are scale equivariant.

[5] More generally, it is the $\mathrm{O}(3)$ group. Reflection is handled as well.

A.3 Proof of translation equivariance

We discuss the translation equivariance property in separate from other transformation groups. Because the translation equivariance property in our method is from the use of the local graph structure, while the equivariant properties of other similarity transformations, including rotations, reflections and scaling, rely on the hybrid features, especially the vector part. Each of the graph layers are locally invariant to the continuous translation group.

For any input points and queries, we discard the absolute global coordinate inputs, and manipulate on the relative positions within a local graph structure in all graph layers in ϕ and ψ. We consider an edge connects an input point $\mathbf{x}_{i'}$ and a query $\mathbf{p}$ in an equivariant graph layer in the graph latent aggregator ψ, then the input vector feature is $\mathbf{x}_{i'} - \mathbf{p}$. if a translation vector $\mathbf{t}$ is applied on all the points, then the input feature becomes

$$(\mathbf{x}_{i'} - \mathbf{t}) - (\mathbf{p} - \mathbf{t}) = \mathbf{x}_{i'} - \mathbf{p}. \tag{10}$$

Thus, the input is invariant to the translation $\mathbf{t}$, so is the output of the graph layer. And similarly for the graph layers in the point encoder ϕ. Therefore, the whole graph function is translation-invariant for local predictions from any 3D point inputs, which means that our graph implicit function, local to each of the query locations, is translation equivariant.

A.4 Proof of rotation, scaling and reflection equivariance

Next, we show the equivariance properties on other transformations including rotations, reflections and scaling. We assume that the scalar feature $\mathbf{h}$ is invariant to these transformations, while the vector feature $\mathbf{V}$ is equivariant, before the invariance function Ω is applied. Formally, we consider an arbitrary orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{3\times3}$ encoding a 3D rotation with a possible reflection, and an arbitrary positive scalar $m \in \mathbb{R}^+$ for the scale transformation applied on the features. The scalar and vector features $\mathbf{h}$ and $\mathbf{V}$ are then transformed into $s\mathbf{V}\mathbf{Q}^\top$ and $\mathbf{h}$ respectively. Note here the orthogonal matrix applying to a stack of vector features $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{C_\mathbf{v}}]^\top \in \mathbb{R}^{C_\mathbf{v}\times3}$ returns $(\mathbf{Q}\mathbf{V}^\top)^\top = \mathbf{V}\mathbf{Q}^\top$. We study how the output of each layer changes with the change of the inputs.

Invariance layer We first show that the invariance function works for any 3×3 orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{3\times3}$ encoding 3D rotation and reflection and any random scaling factor $s \in \mathbb{R}$, such that $\Omega(s\mathbf{V}\mathbf{Q}^\top) = \Omega(\mathbf{V})$:

$$\begin{aligned}
\Omega(m\mathbf{V}\mathbf{Q}^\top) &= \frac{\left\langle m\mathbf{V}\mathbf{Q}^\top, \frac{m\mathbf{Q}\bar{\mathbf{v}}}{\|m\mathbf{Q}\bar{\mathbf{v}}\|}\right\rangle}{\left\|\left\langle s\mathbf{V}\mathbf{Q}^\top, \frac{m\mathbf{Q}\bar{\mathbf{v}}}{\|m\mathbf{Q}\bar{\mathbf{v}}\|}\right\rangle\right\|} = \frac{m\left\langle \mathbf{V}\mathbf{Q}^\top, \frac{\mathbf{Q}\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle}{m\left\|\left\langle \mathbf{V}\mathbf{Q}^\top, \frac{\mathbf{Q}\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle\right\|} \\
&= \frac{\left\langle \mathbf{Q}^{-1}\mathbf{Q}\mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle}{\left\|\left\langle \mathbf{Q}^{-1}\mathbf{Q}\mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle\right\|} = \frac{\left\langle \mathbf{I}\mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle}{\left\|\left\langle \mathbf{I}\mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle\right\|} \\
&= \frac{\left\langle \mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle}{\left\|\left\langle \mathbf{V}, \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}\right\rangle\right\|} \\
&= \Omega(\mathbf{V}),
\end{aligned} \tag{11}$$

where in the second row of Eq. (11), the orthogonal matrices $\mathbf{Q}$ are cancelled out in the inner product. Here we adopt a slightly abused notation to have the inner product between a stack of vector features $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{C_\mathbf{v}}]^\top$ and the average vector $\overline{\mathbf{v}}$, such that $\langle \mathbf{V}, \overline{\mathbf{v}} \rangle = [\langle \mathbf{v}_1, \overline{\mathbf{v}} \rangle, \langle \mathbf{v}_2, \overline{\mathbf{v}} \rangle, \ldots, \langle \mathbf{v}_{C_\mathbf{v}}, \overline{\mathbf{v}} \rangle]^\top$.

Linear layer Next, we show that our hybrid feature linear layer is equivariant to the rotation, reflection, and scaling transformations, encoded by arbitrary $\mathbf{Q}$ and m. Without these transformations, we consider $\{\mathbf{s}', \mathbf{V}'\}$ as the outputs for $\{\mathbf{s}, \mathbf{V}\}$ from the layer denoted by f, i.e., $\{\mathbf{s}', \mathbf{V}'\} = f(\{\mathbf{s}, \mathbf{V}\})$; while under the transformations encoded by $\mathbf{Q}$ and m, we denote the outputs as $\{\mathbf{s}'', \mathbf{V}''\} = f\left(\{\mathbf{s}, m\mathbf{V}\mathbf{Q}^\top\}\right)$. For the equivariance of f, we need to show that:

$$\mathbf{s}'' = \mathbf{s}', \text{ and } \mathbf{V}'' = m\mathbf{V}'\mathbf{Q}^\top, \tag{12}$$

in which we assume that the vector feature $\mathbf{V}$ is equivariant for rotations, reflections and scaling, while the scalar feature $\mathbf{h}$ is invariant to these transformations.

For the scalar feature output in Eq. (6), one can simply verify the invariance

$$\begin{aligned} \mathbf{s}'' &= \mathbf{W}_s \mathbf{s} + \mathbf{W}_{\mathbf{v}s}\, \Omega(m\mathbf{V}\mathbf{Q}^\top) \\ &= \mathbf{W}_s \mathbf{s} + \mathbf{W}_{\mathbf{v}s}\, \Omega(\mathbf{V}) \\ &= \mathbf{s}'. \end{aligned} \tag{13}$$

Here the invariance function returns

$$\Omega(m\mathbf{V}\mathbf{Q}^\top) = \Omega(\mathbf{V}), \tag{14}$$

as shown in Eq. (11). For the vector feature output in Eq. (6),

$$\begin{aligned} \mathbf{V}'' &= \mathbf{W}_\mathbf{v}(m\mathbf{V}\mathbf{Q}^\top) \odot (\mathbf{W}_{s\mathbf{v}}\, \mathbf{s} \,/\, \|\mathbf{W}_{s\mathbf{v}}\, \mathbf{s}\|) \\ &= m\left(\mathbf{W}_\mathbf{v}\mathbf{V} \odot (\mathbf{W}_{s\mathbf{v}}\, \mathbf{s} \,/\, \|\mathbf{W}_{s\mathbf{v}}\, \mathbf{s}\|)\right)\mathbf{Q}^\top \\ &= m\mathbf{V}'\mathbf{Q}^\top \end{aligned} \tag{15}$$

Thus far we have shown the equivariance of the linear layers.

Non-linearity For the non-linearity, the scalar features take simple ReLU activation, hence the invariance is easily ensured as no equivariant vector feature is involved for the output scalar feature.

For the vector non-linearity $\mathbf{v}$-ReLU in Eq. (8), we first reason that the transformations $\mathbf{Q}$ and m does not influence whether the vector feature $\mathbf{v}_c$ at each channel c falls in the positive or the negative part of the piecewise non-linearity. With the untransformed feature $\mathbf{v}_c$, the positive case is judged by $\left\langle \mathbf{v}_c, \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\rangle \geq 0$; while with the transformed feature vector $m\mathbf{Q}\mathbf{v}_c$, the learned direction vector $\mathbf{q}$ is transformed accordingly into $m\mathbf{Q}\mathbf{q}$. Then, the condition of the positive case becomes $\left\langle m\mathbf{Q}\mathbf{v}_c, \frac{m\mathbf{Q}\mathbf{q}}{\|m\mathbf{Q}\mathbf{q}\|} \right\rangle \geq 0$, which is equivalent to the condition without the transformations, as the orthogonal matrices $\mathbf{Q}$ are cancelled out and the positive scaling factor s does not change the sign. The same for the negative case.

Next, we show the equivariance in both positive and negative cases of the non-linearity in Eq. (8). When transformations $\mathbf{Q}$ and m are applied, in the positive case,

$$\begin{aligned}[\mathbf{v}\text{-ReLU}(m\mathbf{V}\mathbf{Q}^\top)]_c &= m\mathbf{Q}\mathbf{v}_c \\ &= m\mathbf{Q}[\mathbf{v}\text{-ReLU}(\mathbf{V})]_c;\end{aligned} \tag{16}$$

while in the negative case,

$$\begin{aligned}[\mathbf{v}\text{-ReLU}(m\mathbf{V}\mathbf{Q}^\top)]_c &= m\mathbf{Q}\mathbf{v}_c - \left\langle m\mathbf{Q}\mathbf{v}_c, \frac{m\mathbf{Q}\mathbf{q}}{\|m\mathbf{Q}\mathbf{q}\|} \right\rangle \frac{m\mathbf{Q}\mathbf{q}}{\|m\mathbf{Q}\mathbf{q}\|} \\ &= m\mathbf{Q}\left(\mathbf{v}_c - \left\langle \mathbf{v}_c, \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\rangle \frac{\mathbf{q}}{\|\mathbf{q}\|}\right) \\ &= m\mathbf{Q}[\mathbf{v}\text{-ReLU}(\mathbf{V})]_c\end{aligned} \tag{17}$$

Thus,

$$\mathbf{v}\text{-ReLU}(m\mathbf{V}\mathbf{Q}^\top) = m\left(\mathbf{v}\text{-ReLU}(\mathbf{V})\right)\mathbf{Q}^\top, \quad \text{or} \tag{18}$$

$$\mathbf{V}'' = m\mathbf{V}'\mathbf{Q}^\top \tag{19}$$

has been proven in both the positive and the negative cases of the vector ReLU function, indicating the equivariance of the non-linear layer with regard to rotation, reflection and scaling.

B Detailed formulations of graph fuctions in multiple scales

We provide the detailed formulation of the layers in the multi-scale graph point encoder ϕ and the multi-scale graph latent feature decoder ψ in Sec 4.2. Here we show the formulations with only the scalar features for the non-equivariant graph implicit model. All these formulations can be easily adapted to the equivariant model by replacing the hidden features $\mathbf{h}$ to the hybrid features of both scalars and vectors $\{\mathbf{s}, \mathbf{V}\}$.

Graph point encoder. The point encoder ϕ is composed of graph convolution layers in the downsampling stage, starting from $l = 0$ to $l = L$, followed by the upsampling layers from $l = L - 1$ back to $l = 0$.

In each graph convolution layer with the sampled point set $\mathbf{X}^{(l)}$ by farthest-point-sampling in the downsampling stage, we obtain the hidden feature $\mathbf{h}_i^{(l)}$ for any sampled point at this level $\mathbf{x}_i \in \mathbf{X}^{(l)}$. To do this, we use graph convolution to aggregate information from the k-nearest neighbor points of $\mathbf{x}_i$, denoted as $\mathbf{x}_{i'}$. The information from the neighboring points to be aggregated is the hidden feature from the previous layer $\mathbf{h}_{i'}^{(l-1)}$. Within the local k-NN graph structure, messages are passed through $\eta_\downarrow^{(l)}$, hence concatenating the inputs for a shared two-layer ReLU-MLP, and a permutation-invariant aggregation function AGGRE; e.g., max- or mean-pooling operator:

$$\mathbf{h}_i^{(l)} = \underset{i'}{\text{AGGRE}}\ \eta_\downarrow^{(l)}(\mathbf{h}_i^{(l-1)}, \mathbf{h}_{i'}^{(l-1)}, \mathbf{x}_{i'} - \mathbf{x}_i). \tag{20}$$

Note that there is an exception in Eq. 20 with $l = 0$, where the input features are the raw coordinates.

In each upsampling layer, the point feature $\mathbf{h}_i^{(l)}$ for any $\mathbf{x}_i \in \mathbf{X}^{(l)}$ at a finer sampling level l takes information from $\mathbf{h}_{i'}^{(l+1)}$, the hidden feature associated to $\mathbf{x}_i$'s 1-nearest neighbor point $\mathbf{x}_{i'}$ from the sampled point set $\mathbf{X}^{(l+1)}$ from the previous sampling level. In addition, we skip-connect $\mathbf{h}_i^{(l)}$, the feature at the same level from the downsampling stage, which is akin to the U-Net structure in grid-based methods:

$$\mathbf{h}_i^{(l)} \leftarrow \eta_{\uparrow}^{(l)}(\mathbf{h}_{i'}^{(l+1)}, \mathbf{h}_i^{(l)}), \tag{21}$$

where $\eta_{\uparrow}^{(l)}$ is a linear layer with a ReLU activation for the concatenation of inputs.

Graph local latent feature aggregator. Given the query coordinate $\mathbf{p}$, we use graph convolutions to aggregate the k-neighboring features $\{(\mathbf{x}_{i'}, \mathbf{h}_{i'}^{(l)})\}$ at different sampling levels l. The aggregated features $\mathbf{z}_{\mathbf{p}}^{(l)}$ from all sampling levels l are concatenated to yield the local latent vector $\mathbf{z}_{\mathbf{p}}$ as output:

$$\mathbf{z}_{\mathbf{p}} = \mathop{\Big\|}_{l=0}^{L} \mathbf{z}_{\mathbf{p}}^{(l)}, \quad \text{where} \tag{22}$$

$$\mathbf{z}_{\mathbf{p}}^{(l)} = \underset{i'}{\text{AGGRE}}\, \eta^{(l)}(\mathbf{p}, \mathbf{h}_{i'}^{(l)}, \mathbf{x}_{i'}^{(l)} - \mathbf{p}), \tag{23}$$

Likewise, $\eta^{(l)}$ is a two-layer ReLU-MLP for the concatenated inputs, and $\|$ denotes concatenation over sampling levels.

C More Implementation Details

We use PyTorch [33] to implement our method and run experiments on a single NVIDIA GeForce GTX 1080 Ti GPU. We train the network using the Adam optimizer [27] with the initial learning rate is set as 10^{-3} for fast convergence for 200K iterations, followed by a finetuning of 100K iterations with the learning rate 10^{-4}. Other hyperparameters and initializations follow the default setups in PyTorch. For the reconstruction of ShapeNet objects, we follow [30,34] to sample 3000 points from the mesh and apply the Gaussian noise with standard deviation 0.005. For scene-level indoor room reconstruction, the number of input points is 10000, as in [34]. We reduce the Gaussian noise level to 0.001 standard deviation, in order to match the change of the scale to have comparable level-of-detail information as the object dataset.

The number of neighbors in k-NN graphs is set as $k = 20$ for all the graph convolution layers. For the multi-scale graph structure, the point set is downsampled twice with farthest point sampling (FPS) to 20% and 5% of the original cardinality respectively. The permutation invariant function AGGRE is a mean-pooling aggregation for vector features and a max-pooling for scalar features. Empirically, we find that using vector max-pooling function as in [17] generates artifacts in the qualitative results, so we simply take the average of the vector features. For the non-equivariant GraphONet, the number of output feature channels is set as 64 for all the layers in the graph latent feature extractor function Φ. For the equivariance model E-GraphONets with hybrid features, the number of output channels for the vector features is 8, and 32 for scalar features. For the input geometric features, the 3D coordinates or the relative position are

Table 6: **Ablation on vector vs. scalar channels.** We evaluate on ShapeNet surface reconstruction with non-graph structured equivariant implicit models, and ModelNet40 point cloud classification with equivariant point cloud networks, for which we follow the VN-PointNet and the VN-ONet architectures in [17] and use hybrid layers instead of pure vector neuron layers.

Ratio of vector channels	0%	12.5%	25%	50%	75%	87.5%	100%
ShapeNet implicit surface reconstruction (mIoU)	0.408	0.630	0.707	**0.719**	**0.719**	0.704	0.694
ModelNet40 point cloud classification (mAcc)	0.808	0.830	0.852	**0.856**	0.855	0.852	0.847

Table 7: **Graph functions with and without multi-scale graph neighbor samplings.** The multi-scale structure improves more on scene reconstruction performance.

Model	GraphONet		E-GraphONet	
Dataset	ShapeNet	SynRoom	ShapeNet	SynRoom
Multi-scale sampling	**0.904**	**0.883**	0.890	0.848
Single-scale sampling	0.897 [-0.007]	0.859 [-0.024]	0.884 [-0.006]	0.814 [-0.034]

considered as 3 channels for the GraphONet, or 1 vector channel and 0 scalar channel for the equivariant layer. For both equivariant and non-equivariant models, the implicit decoder F is the same as that in the ConvONet [34], which is a light-weight ReLU-MLP architechture with skip-connections.

D Additional Experiments and Results

D.1 Vector vs. scalar channels in hybrid feature equivariant layers.

We extend the ablation experiments on hybrid feature channels in Fig. 7 of the main paper. Here we show that our hybrid feature paradigm benefits different architectures and tasks. For implicit surface reconstruction, we evaluate the equivariant implicit model without a graph embedding. We follow the VN-ONet architecture and the implementation details from [17], and use hybrid layers instead of pure vector neuron layers. In addition, we evaluate point cloud classification on the ModelNet40 dataset. Similarly, the architecture and the experimental setups follow VN-PointNet from [17], and we replace a portion of vector channels with scalars in each layer. We evaluate the performance with different ratios of vector channels, where one vector channel is equivalent to three scalar channels.

In Table 6, we report the performance with different ratio of vector channels, where one vector channel is considered equivalent to three scalar channels. In both cases, our method with hybrid features achieves higher accuracy than pure vector features (100%) as in [17]. The conclusion is consistent with the ablation experiments in Fig. 7 of the main paper, and our hybrid feature paradigm is advantageous in general cases.

Table 8: **Implicit functions with different point encoders.** Graph models with graph point encoders replaced by PointNets would lead to more performance drop on scenes than on objects, because PointNet models no locality or translation equivariance which are more crucial for scenes; ConvONets with different point encoders show similar performance, because in such methods, feature embedding relies more on the grid encoder than the point encoder.

Model	GraphONet		E-GraphONet		ConvONet-2D		ConvONet-3D	
Dataset	ShapeNet	SynRoom	ShapeNet	SynRoom	ShapeNet	SynRoom	ShapeNet	SynRoom
Graph encoder	**0.904**	**0.883**	0.890	0.848	0.881 [-0.003]	0.803 [+0.001]	0.872 [+0.002]	0.853 [+0.006]
PointNet encoder	0.887 [-0.017]	0.826 [-0.057]	0.879 [-0.011]	0.797 [-0.051]	0.884	0.802	0.870	0.847

Table 9: **Graph implicit functions with different numbers of neighboring points k and layers L.** Reporting IoU ↑ on the ShapeNet dataset.

Evaluating IoU↑	$k = 6$	$k = 12$	$k = 20$	$k = 32$	$k = 64$
GraphONet / E-GraphONet	0.860 / 0.813	0.885 / 0.867	**0.904** / **0.890**	0.902 / **0.890**	0.891 / 0.872
Evaluating IoU↑	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
GraphONet / E-GraphONet	0.824 / 0.659	**0.904** / **0.890**	0.900 / 0.887	0.890 / 0.874	0.884 / 0.865

D.2 Ablation on the architecture.

We ablate the implementation choices of our models. First, we explore how our models perform without the multi-scale sampling design on the ShapeNet object reconstruction and the Synthetic Room (SynRoom) scene reconstruction tasks. In Table 7, we show that the scene reconstruction performance drops more without the multi-scale architecture, while the difference in object reconstruction performance is subtle. We argue that scene reconstruction is a more complex task, so the multi-scale design plays a more important role to aggregate global and local context in different scales.

Then, we show the effect of using different point cloud encoders in our graph models and the baseline methods ConvONets [34]. The results are in Table 8. For the graph models, the scene-level reconstruction performance drops much more when the graph encoder is replaced by a PointNet encoder. The results indicate that both the locality modelling and the awareness of the translation equivariance from the graph encoder are more crucial for scene-level reconstruction as a more comlex task. However, in ConvONets, using the graph point encoder instead of the PointNet encoder does not lead to a significantly improved performance. Unlike our graph methods, ConvONets learn the latent feature with an intermediate grid feature tensor. So feature embedding in ConvONet relies more on the regular convolution layers applied on the grid feature, while the point encoder plays a less important role.

Next, we show how the model performs under different numbers of neighboring points k and layers L. In Table 9 we report the mean IoU ↑ on the ShapeNet dataset. From the results, using too smaller k and L could suffer from underfitting, while using too large values increases computation cost and may cause over-smoothed graph features.

D.3 Learning curve with limited training data.

We explore how the implicit model performs with very few training data of 130 examples, and provide the validation loss curve, as illustrated in Fig. 9. This result is a supplement to the test performance in Table 4 of the main paper. Both ConvONet-2D and ConvONet-3D suffer from overfitting in the very early stage of training, prior to 500 training steps. By contrast, our graph methods are able to learn properly from very few training data. The equivariant model is with better validation loss and more stable learning curve, which indicates that the equivariance property works as a regularization that controls the model complexity.

D.4 More qualitative and quantitative results.

We evaluate ShapeNet reconstruction performance by each object category. The results are shown in Table 10, where our graph model shows better performance with most of the object categories.

In Fig. 10, we show some additional ShapeNet reconstruction examples under transformations. Our final equivariance model guarantees equivariance to all kinds of similarity transformations.

Though not the main focus of this paper, our method scale to scene-level reconstructions, and we show some more room reconstruction examples in Fig. 11. Our GraphONet models better details. The equivariant model E-GraphONet achieves relatively good performance but generates more noisy artifacts, especially on the synthetic-to-real evaluation on the ScanNet dataset with corrupt areas in the point cloud scans. We argue that the restricted representation power of the equivariant layers limits the model to learn denoising and completion alongside reconstruction while generalize to more complex corrupted scenes. See the discussion on the limitation in the main paper.

E Limitations and Future Work

A limitation that comes with equivariance is that, by constraining the model complexity to conform to equivariant designs, expressive power may be affected as well. As a consequence, accuracy drops in stylized settings and datasets. In particular, we observe that the shapes generated from equivariance models are usually less smooth than non-equivariant methods, especially for the more complex scenes. See Fig. 8 in the main paper and Fig. 11 in the Appendix. We argue that the restricted power of equivariance models limits the ability to identify the denoised geometry from the noisy point observations, while at the same time, the equivariant model are designed to avoid leveraging the prior of the flat straight and planar structures aligned with the Cartesian coordinate axes. To this end, relevant future directions include exploring more powerful equivariant models, or incorporating filtering techniques for implicit fields.

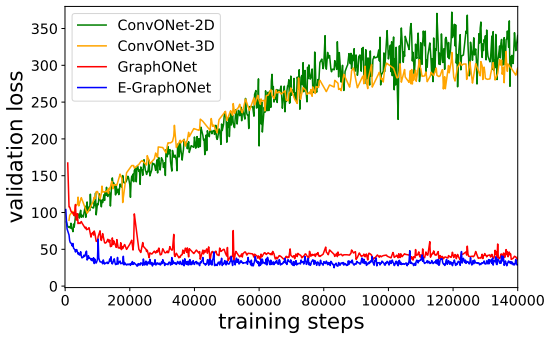

Fig. 9: **Learning curve with 130 training examples.** We show the validation loss curves. Graph methods can learn properly from very few training data, while ConvONets suffer from overfitting.

Table 10: **Category-specific ShapeNet reconstruction performance.** We evaluate our methods and compare with the baseline implicit representation models.

	ConvONet-2D [34]			ConvONet-3D [34]			IF-Net [11]			GraphONet			E-GraphONet-SO(3)		
	IoU	Chamfer	Normal	IoU	Chamfer	Normal	IoU	Chamfer	Normal	IoU	Chamfer	Normal	IoU	Chamfer	Normal
airplane	0.849	0.034	0.931	0.849	0.033	0.932	0.862	0.031	0.936	**0.881**	**0.027**	**0.941**	0.867	0.028	0.930
bench	0.830	0.035	0.921	0.791	0.041	0.911	0.815	0.037	0.915	**0.836**	**0.034**	**0.924**	0.807	0.037	0.906
cabinet	0.940	0.046	0.956	0.923	0.054	0.953	0.936	0.048	0.956	**0.943**	**0.047**	**0.958**	0.927	0.050	0.944
car	0.886	0.075	0.893	0.877	0.080	0.891	0.890	0.072	0.894	**0.897**	**0.068**	**0.895**	0.890	0.072	0.886
chair	0.871	0.046	0.943	0.853	0.049	0.942	0.878	0.043	0.946	**0.895**	**0.039**	**0.951**	0.879	0.043	0.939
display	0.927	0.036	0.968	0.904	0.042	0.965	0.923	0.036	0.968	**0.936**	**0.034**	**0.972**	0.922	0.036	0.963
lamp	0.785	0.059	0.900	0.792	0.066	0.910	0.820	0.047	0.916	0.847	0.042	**0.922**	**0.848**	**0.040**	0.915
loudspeaker	0.918	0.064	0.939	0.914	0.065	0.942	0.928	0.056	0.945	**0.938**	**0.053**	**0.946**	0.936	0.055	0.941
rifle	0.846	0.028	0.929	0.826	0.031	0.924	0.842	0.028	0.928	**0.877**	**0.022**	**0.943**	0.868	0.023	0.933
sofa	0.936	0.042	0.958	0.923	0.046	0.956	0.938	0.040	0.959	**0.946**	**0.037**	**0.963**	0.931	0.041	0.951
table	0.888	0.038	0.959	0.860	0.043	0.956	0.880	0.038	0.959	**0.896**	**0.036**	**0.963**	0.869	0.040	0.950
telephone	**0.955**	0.027	**0.983**	0.942	0.030	0.981	0.949	0.027	**0.983**	0.954	**0.026**	**0.983**	0.946	0.027	0.979
vessel	0.865	0.043	0.919	0.860	0.045	0.919	0.876	0.040	0.923	**0.901**	**0.033**	**0.934**	0.892	0.035	0.924
mean	0.884	0.044	0.938	0.870	0.048	0.937	0.887	0.042	0.941	**0.904**	**0.038**	**0.946**	0.890	0.041	0.936

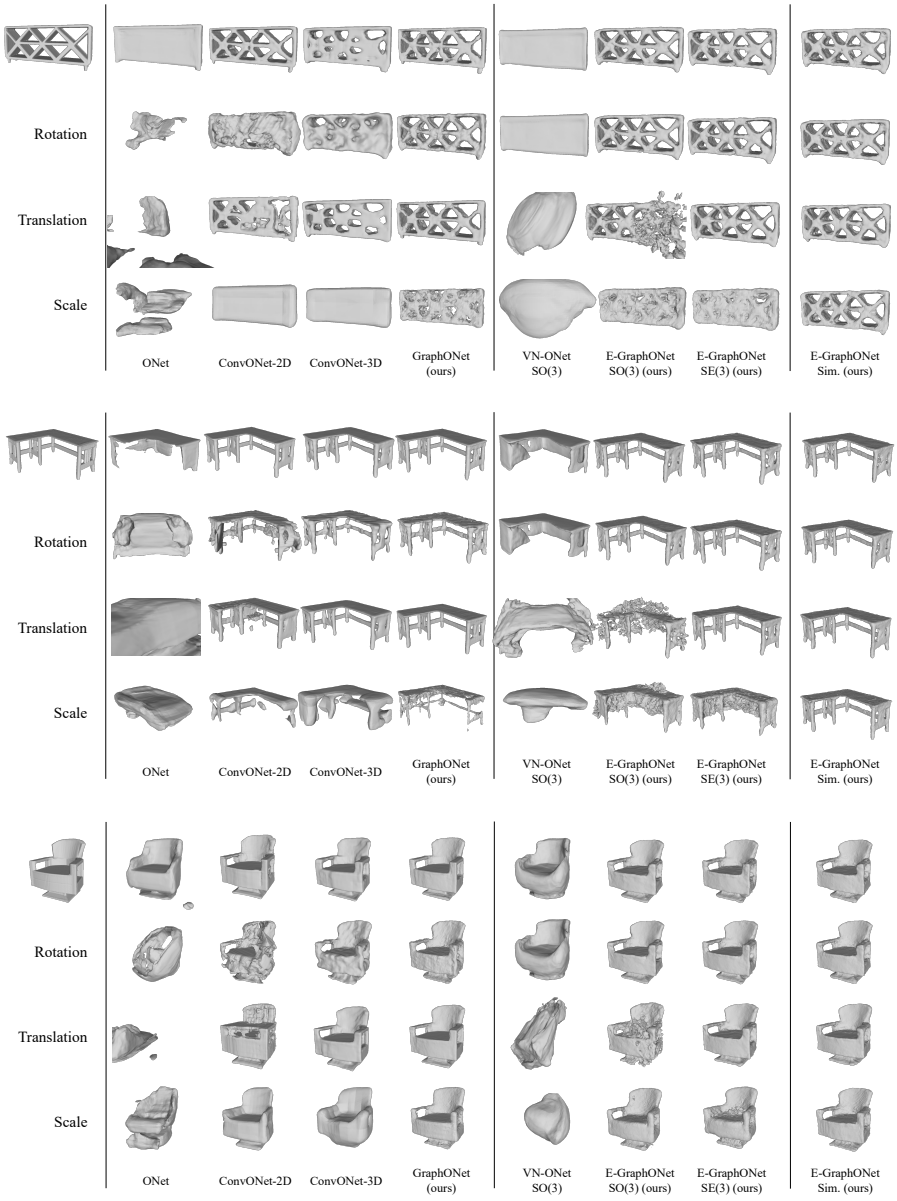

Fig. 10: **More examples on ShapeNet object reconstruction under unseen transformations.** With transformations we show the back-transformed shapes.

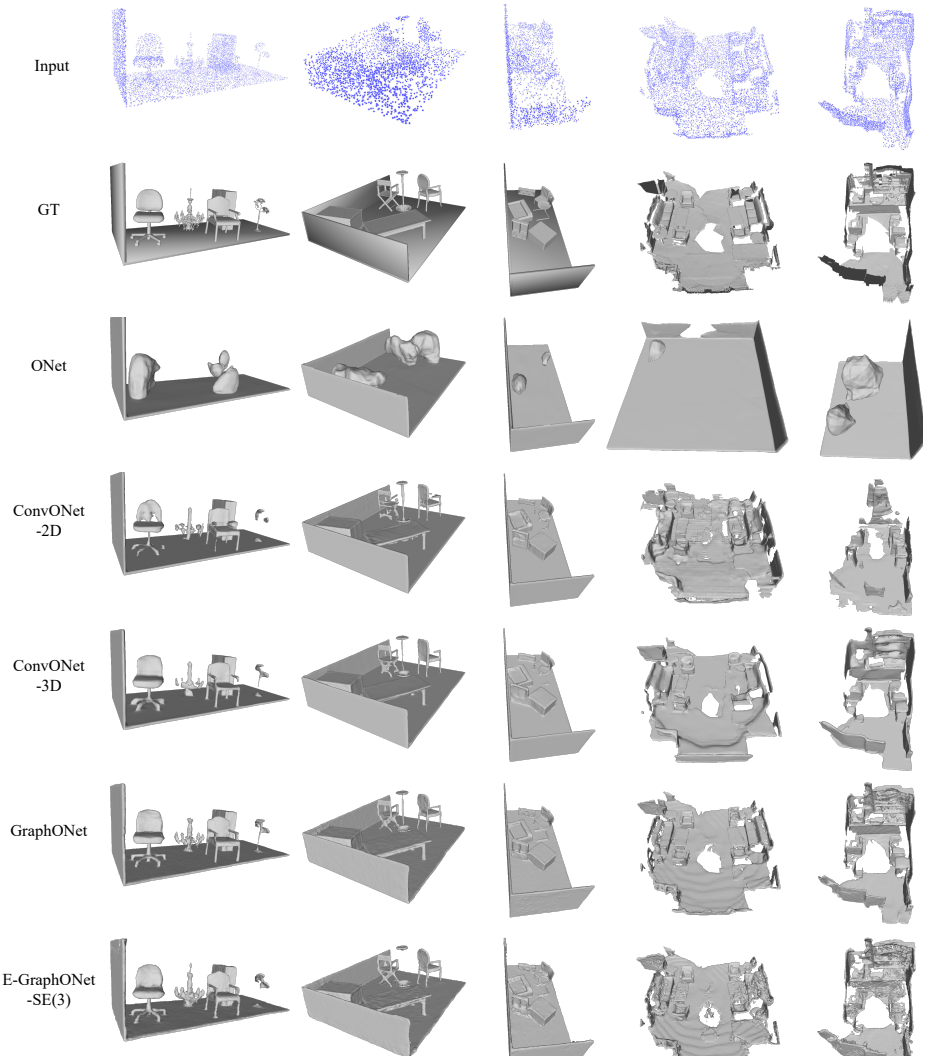

Fig. 11: **More scene reconstruction examples.** The left three columns are from the Synthetic Room dataset. The right two columns are from ScanNet.