HYVE: Hybrid Vertex Encoder for Neural Distance Fields

Stefan R. Jeske*, Jonathan Klein†, Dominik Michels† and Jan Bender*
*Visual Computing Institute, RWTH Aachen University Aachen, Germany
{jeske,bender}@cs.rwth-aachen.de
†Computational Sciences Group, KAUST Thuwal, Saudi Arabia
{jonathan.klein,dominik.michels}@kaust.edu.sa

hyve.physics-simulation.org

***Abstract*—Neural shape representation generally refers to representing 3D geometry using neural networks, e.g., computing a signed distance or occupancy value at a specific spatial position. In this paper we present a neural-network architecture suitable for accurate encoding of 3D shapes in a single forward pass. Our architecture is based on a multi-scale hybrid system incorporating graph-based and voxel-based components, as well as a continuously differentiable decoder. The hybrid system includes a novel way of voxelizing point-based features in neural networks by projecting the point "feature-field" onto a grid. This projection is insensitive to local point density, and we show that it can be used to obtain smoother and more detailed reconstructions, in particular when combined with oriented point clouds as input. Our architecture also requires only a single forward pass, instead of the latent-code optimization used in auto-decoder methods. Furthermore, our network is trained to solve the well-established eikonal equation and only requires knowledge of the zero-level set for training and inference. We additionally propose a modification to the aforementioned loss function for the case that surface normals are not well defined, e.g., in the context of non-watertight surfaces and non-manifold geometry. Overall, our method consistently outperforms other baselines on the surface reconstruction task across a wide variety of datasets, while being more computationally efficient and requiring fewer parameters.**

***Index Terms*—neural shape representation, neural distance fields, eikonal equation, surface point-cloud, encoder-decoder**

I. Introduction

Algorithms processing 3D geometric data have become omnipresent and an integral part of many systems. These include, for example, systems evaluating LiDAR sensor data, game engines, 3D asset visualization, and physical simulation used in engineering prototypes. In recent years, deep learning methods have been increasingly investigated to assist in solving problems pertaining to 3D geometry.

In particular, neural shape representation is the task of using neural networks to predict shape occupancies or surface distances at arbitrary spatial coordinates. Recent works have shown the ability to capture intricate details of 3D geometry with ever-increasing fidelity [2], [3]. However, a significant number of such works employ an auto-decoder-based architecture, which requires solving an optimization problem when representing new geometry. Additionally, the auto-decoder still has to be evaluated for all query points individually, which can become very costly when evaluating these systems for high-resolution reconstruction [4]. Finally, many of these methods also require annotated and densely sampled ground truth data. Meta-learning approaches, e.g., by Sitzmann et al. [5] and Ousafi et al. [6], mitigate this problem, but also have to run several iterations of gradient descent to specialize the network for each new model before inference. Encoder-decoders can instead encode the shape in a single forward pass [7]–[9], and typically employ computationally cheaper decoder networks for evaluating query points. Nevertheless, these approaches also often require elaborate data preprocessing pipelines, and rely on labeled training data.

One of our core motivations for this work was to develop a neural distance field based method that could be used for signed distance computation of both rigid and deforming shapes. Both are represented as points and the latter can change often, requiring frequent recomputation of the distance field. The deformations and dynamics of these shapes are not explicitly learned by the network but instead implicitly captured through geometric changes in the input shape. As a result, we wanted to keep the decoder network computationally efficient and, instead of latent optimization, we wanted to develop an encoder capable of generating fast and accurate latent codes for this decoder network.

Therefore, we propose an end-to-end learnable encoder-decoder system that is not bound by previous data preprocessing constraints and can be trained using only the zero-level set, i.e., surface samples as labeled data, which are readily available with a minimal amount of preprocessing. This kind of training was previously introduced for auto-decoders [10], and is enabled by using the eikonal equation as a training target. Similar approaches have been attempted for encoders by Atzmon et al. [11], [12], using global encoders augmented by latent-optimization at test-time. Yet to the best of our knowledge, this kind of training is not well-established using encoders without latent-optimization.

We summarize our contributions as follows:

- We derive an encoder architecture for representing 3D

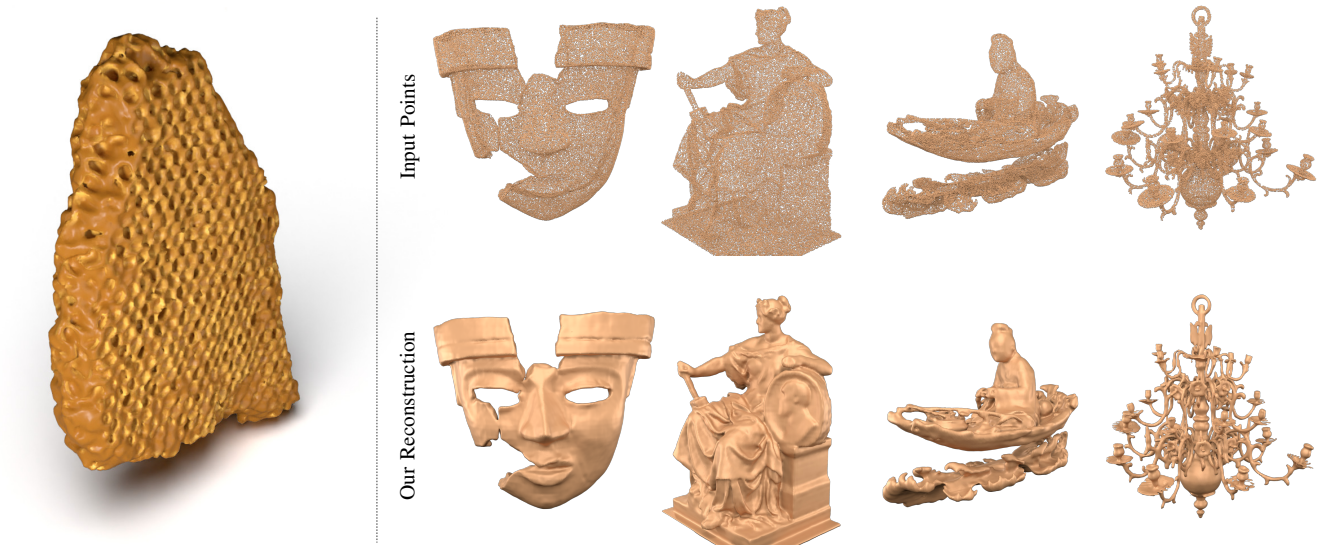

Fig. 1. The reconstruction of a 3D scan of a beehive (left) and comparisons of input points and our respective reconstructions on the Objaverse dataset (right) [1]. These examples show the capability of our neural distance field encoder to capture a large amount of detail in a single forward pass, using only oriented point clouds as input.

shapes. The key component of this encoder is the hybrid and interleaved execution of graph-level convolutions and 3D grid convolutions, as well as back-and-forth feature projections. Our motivation for this hybrid approach is the ability to accurately *extract* information from surface-points and to efficiently *process* information in grid (Euclidean) space.

- We introduce a novel way of voxelizing point-features within network architectures, in which the pooling operator is replaced by projection of the entire "feature-field" onto the grid. We show that this results in less-noisy and more detailed surfaces, particularly when point normals are used as additional input.
- We show that the accuracy of our architecture is intuitively controllable. Using a model with as little as 38K parameters (including the decoder) can already achieve excellent visual quality while being very fast to evaluate. This makes it potentially useful for practical applications within resource constrained computing platforms.
- We show the feasibility of training encoder-decoder networks on the eikonal equation for high-fidelity 3D shape encoding. We also propose a simple yet effective modification to the loss function that can gracefully handle poorly oriented surface normals in the training data, e.g., caused by non-manifold or non-watertight geometry.

In the evaluation, we show that we are able to reconstruct better quality surfaces than other state-of-the-art methods. Sample reconstructions of our method are shown in Figure 1.

II. Related Work

Neural fields have become an integral part of research in geometric deep learning, with hundreds of papers published in recent years. A comprehensive overview is given by Xie et al. [4]. One of the seminal works on deep learning for unstructured data was the introduction of PointNet [13]. From today's perspective, one of the major limitations of this work is the difficulty of learning high-frequency functions from low-dimensional data [14], [15]. The solution to the problem is addressed by more recent approaches such as NeRFs [16] and Fourier Feature Networks [17]. In essence, the idea is to use positional embeddings, inspired by the embeddings proposed by Vaswani et al. [18] for transformer networks. These embeddings compute a mapping from low dimensional positional information (typically 2D or 3D) into higher dimensional spaces using a specific number of Fourier basis functions [17]. A concurrent work shows that using periodic activation functions inside an MLP also significantly improves reconstruction quality and surface detail [10], the single layer of which can again be seen as a kind of positional encoding [4]. Subsequent works improve the usage of positional encodings, e.g., by controlling the frequency through a feedback loop [19] or modulating the periodic activations using a separate ReLU-activated MLP [20]. Other benefits of using periodic activations are the ability to better learn high-frequency mappings and the continuous differentiability of these activations which is useful for evaluating network derivatives as training targets [2], [21], [22].

There are many approaches for representing 3D shapes using neural networks. For clarity of exposition we will classify them into global-prior and local-prior based methods. In addition, we will discuss overfitting-based methods, general unsupervised shape encoding and neural scene representation. Finally, we will differentiate our work from previous works.

a) Global methods: These do not make use of geometric structures in the network itself, and can generally be used irrespective of the discretized representation of the geometry. Typically auto-decoder methods, in which the latent representation is optimized during training and testing, are in this category [2], [10]–[12], [20], [23]–[25]. The network can then be queried using both the latent feature and a 3D coordinate

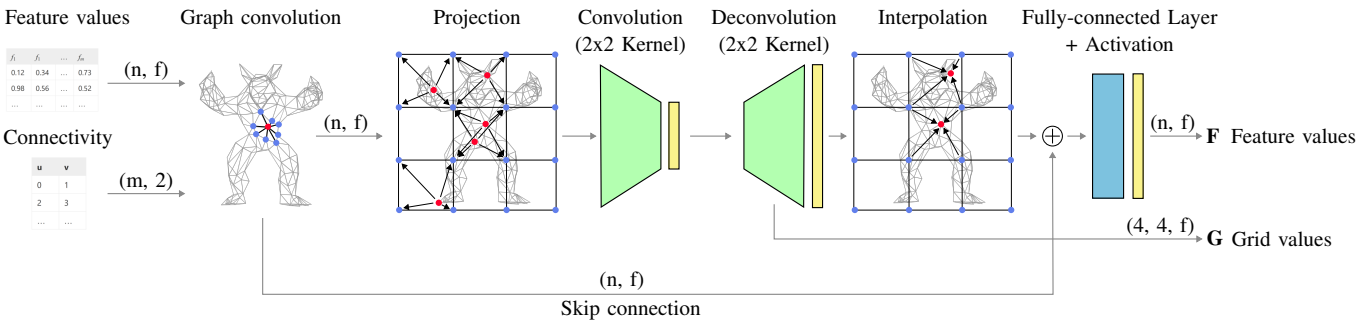

Fig. 2. Convolution block that extracts features for a specific grid resolution. For clarity of illustration, a 2D rather than 3D grid is shown here. The input is a set of vertices (with position / feature data) and edges (encoded as vertex indices). The + denotes element-wise vector addition. The block has two outputs, feature values on the vertices and grid values for each grid cell. For all resolutions, a 2×2 convolution kernel is used. n: number of vertices. m: number of edges. f: number of features (on the first level, the features are the spatial coordinate of each vertex).

to evaluate either a distance metric or an occupancy value. Meta learning approaches also fall into this category. A few iterations of gradient descent are used to specialize the weights of a generalized network to a new shape [5], [6]. An approach that has both discretization-dependent and independent components was presented by Chen et al. [22], where the discretization-dependent encoder is typically discarded during inference. The amount of encoded details by these methods is naturally bounded by the number of network weights. It has also been shown that using pooling-based set encoders for global conditioning frequently underfits the data [26].

b) Local methods: This group of methods typically relies on using spatial structures, namely spatial grids or point clouds, within the network itself for the extraction of meaningful information [7]–[9], [27]–[37]. This has proven to be a valuable approach since it is quite difficult for neural networks to encode the high-frequency functions needed to represent detailed fields in 3D. Previous works make efficient use of discretized structures, e.g., point clouds, meshes or voxel-grids as either inputs or outputs [13], [19], [26], [38]. There are also exceptions using other discretized structures. One such work uses triplanes instead of volumetric grids for latent diffusion 3D generation tasks [39], while another makes use of grid "patches" representing local geometric features that are merged through a global octree structure [40]. A similar approach is used by Yariv et al. [41], which uses patches of small volumetric grids covering the object surface. For encoder-type methods, extracting local features has been shown to generally improve network performance over global ones.

More recently, kernel-based methods have evolved, combining local features in spatial structures with an inference-time sparse linear solve [34]–[36]. This allows local methods to more strictly adhere to the input point cloud and scale to larger inputs, at the cost of the linear solve in addition to inferencing a "convolutional-backbone" network. Finally, Müller et al. [42] propose an efficient way to use sparsity through multiresolution hashing and demonstrate its potential applicability for many of the local methods discussed here.

c) Unsupervised shape encoding: A number of recent works have investigated the ability to train networks for shape representation without direct supervision [3], [10]–[12], [25], [43]–[47]. Most of these works focus on auto-decoders and generally use optimization of latent codes during inference, while others "overfit" small networks to specific shapes [43]. Some also use unsigned distance fields, which enables training on non-manifold geometry but increases reconstruction difficulty. While Tang et al. [31] also use sign-agnostic optimization of occupancy fields, they still require ground-truth occupancy values for pre-training. Notably, Gropp et al. [25] introduced the formulation of the unsupervised eikonal loss, which was further refined in the work of Sitzmann et al. [10]. In our work we extend this loss to improve training on data with inconsistent normal orientations.

d) Neural scene representations and diffusion-based 3D generation: Apart from our focus on shape reconstruction, recent neural scene representations reconstruct scenes from posed multi-view images. Prominent approaches include neural implicit fields, often neural radiance fields (NeRFs) [16], [48]–[50] and explicit primitive-based methods such as 3D Gaussian splatting [51]–[53]. The dominant objective is photorealistic novel-view synthesis and interactive editing. While geometric reconstruction is possible (e.g., implicit-surface methods like Neuralangelo), most pipelines prioritize rendering quality and speed over watertight mesh extraction.

For generating 3D shapes or assets from text or images, diffusion models have become the most common paradigm [54]–[56]. Nevertheless, a recent autoregressive model has shown competitive performance to the prevalent diffusion models for shape generation [57].

e) Differentiation from Related Works: Many previous works focus on either purely point-based or grid-based approaches with prior voxelization of input points. Early works by Liu et al. [58], [59] explore the use of hybrid point-voxel architectures for object segmentation and classification and show promising results in terms of accuracy and efficiency. In the latter work, the authors particularly focus on the efficiency aspect, presenting a method to find an optimal neural architecture to fit into a specific computational budget, again with application to classification and segmentation of 3D scenes. In contrast to previous work, we expand the capabilities of hybrid approaches to the unsupervised 3D reconstruction task and explore important design decisions. This includes details on network construction and the often-overlooked point-to-

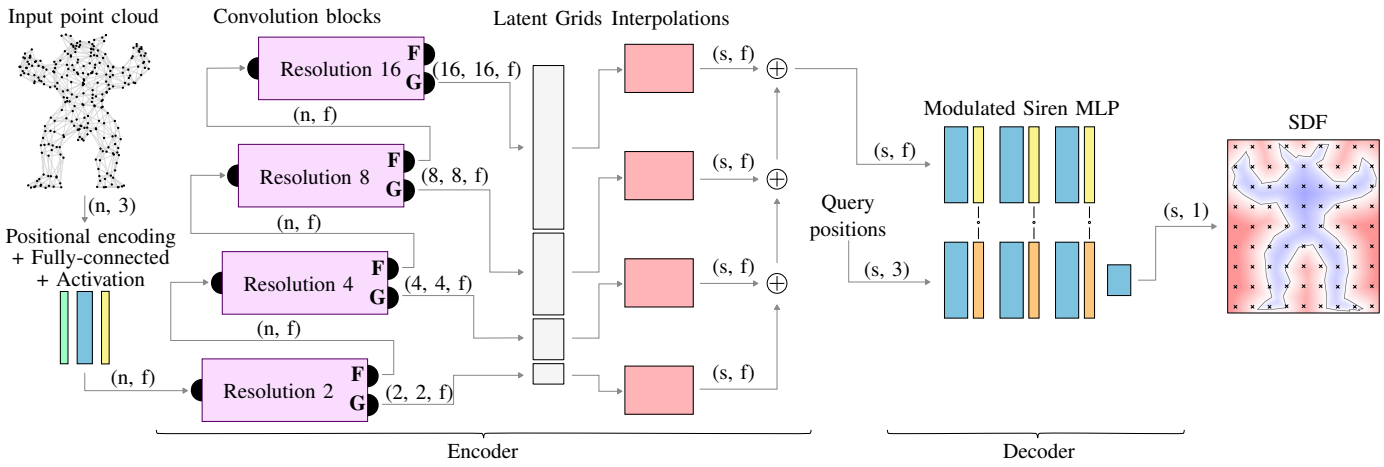

Fig. 3. The encoder-decoder architecture of our network. The encoder computes vertex and volumetric features at multiple resolutions. By passing the feature vector through the convolution blocks, neighbor information is collected. The implementation of the convolution blocks is shown in Figure 2. After the last block, the vertex feature vector is discarded. The + denotes element-wise vector addition. n: number of vertices. f: number of features. s: number of SDF sample points.

grid feature transfer methods to maintain consistency between grid and points.

As opposed to other works operating purely on point-based data [8], [9], we have found the usage of a grid to be beneficial both in terms of efficiency and accuracy. We also use graph-convolutions at the point-level, extracting information at the smallest possible scale, while Wang et al. [33] use graph-convolutions at the voxel-level, as a surrogate for sparse grid-convolutions. Our formulation also retains information about the local positioning of the points in the grid cells by *not* voxelizing the input as is common with purely grid-based approaches [7], [33], [37]. Using these components we propose and ablate an architecture that outperforms previous baselines on a number of popular datasets and behaves predictably when changing internal resolutions and latent sizes. We further propose to approximate a continuous feature field that adheres to the input points by coupling point-centered and grid-centered convolutions through feature projections. In contrast to recent works [23], [34], [36], we maintain this "continuity" without any additional optimization or linear solves at inference-time.

III. Zero-Level-Set Encoder

In the following we will present our encoder-decoder architecture, including our *convolution block* for the encoder, the decoder structure, the loss function along with our modification to support non-manifold geometries, and the overall training process. We will also propose a new way to transfer information from points to voxels within neural networks, which is inspired by hybrid Eulerian-Lagrangian physical simulation methods.

Our inputs are surface point-clouds of 3D objects, given by a set of vertices $\mathcal{V} = \left\{V \in \mathbb{R}^3\right\}$. In order to use graph convolutions, we create edges $\mathcal{E} = \left\{E \in (\mathbb{N} \times \mathbb{N})\right\}$ between vertices, using e.g. k-nearest-neighbor (k-NN) or radius graph connectivity. Within the network, the surface points store abstract f-dimensional feature vectors ($V^f \in \mathbb{R}^f$), rather than 3D coordinates. This input representation also allows for utilization of known point connectivity, e.g., for partially meshed point cloud inputs or triangle soups.

A. Encoder-Decoder Architecture

a) Convolution block: We introduce our hybrid point-grid convolution block in Figure 2 as the main building block for our encoder. This is in contrast to many previous encoder approaches that use either only point data [8], [9], [13], [24] or transfer point data to voxel grids for further processing [7], [28]. To get the best of both, we instead interleave these approaches. The intuition behind this idea is that the points represent exactly the shape that should be encoded. In order to reason about the surrounding space of the object, the features are projected onto a grid and propagated by using shallow convolutional layers. Then, to extract as much information as possible about the shape, features are interpolated from the grid to the input points and propagated between the nodes. The usage of shallow 3D convolutions makes the network faster to evaluate and use less intermediate storage.

First, a graph convolution operator (e.g., EdgeConv by Wang et al. [60]) transforms each vertex feature V^f using the edge connectivity information. Next, we project the feature vectors onto a grid. We do this using our projection method described in Section III-B. A $2 \times 2 \times 2$ convolution and subsequent deconvolution with activation (where the feature count is doubled for the latent space to retain information) is then used to exchange information between neighboring cells. We map the features back onto the vertices through tri-linear interpolation using the 8 closest cell centers. Here, they are combined with the original output of the graph convolution before finally being processed through a single per-vertex fully-connected layer. This output serves as the input of the next convolution block, while the deconvolved grid values are cached for later use.

We reason that the grid is suitable for distances in Euclidean space, because the regular grid structure implicitly assumes regular distances between cells and across convolution weights. This information can be used implicitly to more

TABLE I
METADATA ABOUT EACH OF OUR DATASETS. $\mathcal{O}$ DENOTES A ROUGHLY EVEN SPLIT OF MANIFOLD AND NON-MANIFOLD INSTANCES.

	Trained Shapes	Tested Shapes	Trained Vertices	Tested Vertices	Manifold	Noisy	Sparse
Dragon	2,400	300	2,210	2,210	✓	✗	✗
Armadillo	2,400	300	25,441	25,441	✓	✗	✗
DFAUST	6,258	2,038	30,000	100,000	✗	✓	✗
ScanNet	1,513	100	30,000	100,000	✗	✓	✓
Thingi10k	2,000	200	4 - 4,995	282 - 4,890	✓	✗	✓
ShapeNet v2 planes	1,632	421	434 - 14,879	457 - 13,888	✗	✗	✗
Objaverse	2,000	200	50,000	100,000	$\mathcal{O}$	✗	✗

accurately locate patches of points, projected to individual grid cells, within the global structure. In turn, the graph convolution can maximize information extraction from the input points at a smaller scale. Connecting the points based on geometric proximity and using the relative distance in graph convolutions makes the information extraction more "fine-grained" within local patches of points. Combining both multiple times throughout the network makes the feature extraction time and memory efficient.

b) Encoder: The overall encoder-decoder architecture is shown in Figure 3. At the beginning of the encoder, the input vertex positions $\mathcal{V}$ are transformed to per-vertex features V^f through positional encoding [17] and a single linear layer with ReLU activations. We feed the encoded positions through a fixed number (typically 4-5) of our hybrid convolution blocks (as described above) with increasing resolution (contrary to the usual decreasing order of convolutional neural networks). The concatenated outputs of the grids form the grid vector (the vertex-level feature output of the last convolutional block can be discarded). Values from each level of the grid vector are extracted for each of the s SDF sample positions by tri-linear interpolation. The results are summed element-wise to form the final latent vector. We note that when latent vectors for additional sample positions are computed, there is no need to recompute the grid vector — it can be cached for future use.

c) Decoder: The decoder receives the sample features and sample positions as input, see Section III-D, and processes them individually to compute a single signed distance value for each sample. As architecture we make use of the proposed decoder of Mehta et al. [20], which is an augmentation of the SIREN layer proposed by Sitzmann et al. [10]. The feature vector is passed through a ReLU-activated MLP and the sample position through a sine-activated MLP. The activations of the ReLU MLP are then used to modulate the activations of the sine MLP whose output is the final SDF value. In our experiments we have found sine-based decoders to consistently outperform ReLU-based decoders. In our testing, regular SIREN layers often had instabilities, which was remedied by using the modulation proposed by Mehta et al. [20].

The decoder is specifically chosen to be a smaller network, since per our motivation, it should be efficient to evaluate for a very large number of query points.

B. Point-to-Grid Transfer

Instead of the usual voxelization schemes applied in neural networks, typically occupancy indicators, max pooling or average pooling, we propose to use the particle-in-cell interpolation approach [61], [62]. This approach is commonly used in hybrid physical simulation methods in order to transfer momentum from particles to an enclosing grid. The concept of this transfer is sketched in Figure 2. As opposed to commonly used pooling operations, which only transfer features to the closest grid node, our projection operation splits up the feature vector across the eight closest grid nodes (for linear basis functions), shown by the black arrows. This intuitively allows for the reconstruction of a more accurate and consistent "feature-field" by using a more meaningful geometric operation. The feature vector $\boldsymbol{f}_i$ at each grid point i can be computed using

$$\hat{\boldsymbol{f}}_i = \sum_p w_{ip} m_p \boldsymbol{f}_p, \quad w_i = \sum_p w_{ip}, \quad \boldsymbol{f}_i = \frac{\hat{\boldsymbol{f}}_i}{w_i},$$

where $\boldsymbol{f}_p \in \mathbb{R}^f$ denotes the feature vector at vertex p. We further assume unit masses $m_p = 1$, meaning that each vertex carries the same amount of "weight" as all others. This quantity could theoretically be used to relatively weight contributions according to local point density or variability, or even to learn via, e.g., an attention-based mechanism. Lastly,

$$w_{ip} = N\left(\frac{x_p - x_i}{h}\right) N\left(\frac{y_p - y_i}{h}\right) N\left(\frac{z_p - z_i}{h}\right)$$

denotes the interpolation weights, h the grid spacing and N the linear interpolation function given by

$$N(x) = \begin{cases} 1 - |x|, & 0 \leq |x| < 1 \\ 0, & 1 \leq |x|. \end{cases}$$

Using this interpolation scheme instead of a pooling approach allows for a smoother transition between graph-convolutions and grid-convolutions, since both of them now approximate the same continuous "feature-field". We show in Section IV-C that this improves reconstruction quality and reduces noise in reconstructed surfaces when compared to max pooling, especially when used in conjunction with input normals.

This projection scheme can also be interpreted as a kind of "cross-attention" layer, that computes grid features using a weighted sum of point features. Namely, the dot-product between key and query is replaced by the geometric weights w_{ip} computed from the interpolation function N. In contrast to regular scaled dot product attention, one can directly obtain grid features from point features, without "initializing" the grid features in any way.

C. Loss Function

An SDF can be defined as the unique solution Φ of the following eikonal equation:

$$\begin{aligned} \|\nabla\Phi(\boldsymbol{x})\| &= 1 \text{ for } \boldsymbol{x} \in \Omega \setminus \Omega_S \subset \mathbb{R}^3 \\ \Phi(\boldsymbol{x}) &= 0 \text{ for } \boldsymbol{x} \in \Omega_S, \end{aligned} \tag{1}$$

where Ω is the SDF domain and Ω_S is the surface of the object. Related to the work of Smith et al. [63], Sitzmann et al. [10] proposed the following loss function as a measure of how well the eikonal equation is satisfied

$$\begin{aligned} \mathcal{L}_{\text{eikonal}} = &\int_{\Omega} \big|\|\nabla\Phi(\boldsymbol{x})\| - 1\big| \, \mathrm{d}\boldsymbol{x} + \int_{\Omega_S} |\Phi(\boldsymbol{x})| \, \mathrm{d}\boldsymbol{x} \\ &+ \int_{\Omega_S} (1 - \langle \nabla\Phi(\boldsymbol{x}), \boldsymbol{n}_x \rangle) \, \mathrm{d}\boldsymbol{x} \\ &+ \int_{\Omega \setminus \Omega_S} \exp(-\alpha |\Phi(\boldsymbol{x})|) \, \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{2}$$

Here $\Phi(\boldsymbol{x})$ denotes the predicted signed distance value of our neural network at position $\boldsymbol{x} \in \Omega$, and $\boldsymbol{n}_x$ denotes the target surface normal. The exponential in the last term is a weighting function which "pushes" signed distance values away from zero for points not on the surface. The original paper notes that the value of the constant α should be chosen as $\alpha \gg 1$, however we have found that this is sometimes detrimental to the desired effect, while choosing a lower value of $\alpha \approx 10$ yielded the best results.

If the input surface is non-manifold or self-intersecting, the inside and outside of the object's volume are not well defined and normal vectors do not necessarily point in the right direction. For these cases we introduce a simple modification that ignores the sign of the normal:

$$\mathcal{L}_{\text{surface normal}} = \int_{\Omega_S} (1 - |\langle \nabla\Phi(\boldsymbol{x}), \boldsymbol{n}_x \rangle|) \, \mathrm{d}\boldsymbol{x}. \tag{3}$$

This change alone still allows for the result to be a valid *signed* distance field. As we discuss in Section IV-B, this greatly improves performance on meshes with poor quality. An example of multiple reconstructed SDF contours from a model trained using all of these losses on the ShapeNet v2 planes dataset is shown in Figure A-2 of the supplemental document.

D. Training

A major advantage of our proposed architecture is that ground truth SDF values for the input point cloud *never* have to be computed. However, it does need sample points to train the network. While these could theoretically be generated on the fly, we sample a number of evaluation points in advance to reduce the training overhead. In order to evaluate the eikonal loss, only the sample positions and classification into (1) surface and (2) non-surface samples are required. After normalizing the input point clouds to the unit cube, the training samples can be trivially generated.

We sample them in the following ways: (1) Surface points. If the source is a triangle mesh rather than a point-cloud, its surface is randomly sampled to create additional on-surface samples. If it is a point-cloud, we either take all available points or subsample to a reasonable number (typically 100k-200k). (2a) Random points in the surrounding volume are sampled as off-surface samples (no check is performed if these points actually lie on the surface per chance since the surface has zero measure as a volume). (2b) If surface normal information is available, additional close-to-surface samples are generated from random points on the surface by displacing them along the surface normal. These additional samples are not strictly needed but aid the training process, as the SDF is most detailed close to the surface.

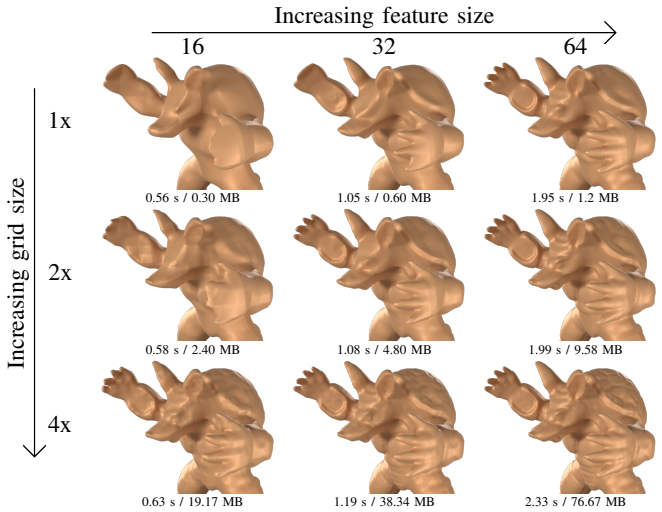

Fig. 4. From top to bottom, we increase the size of all latent grids, while from left to right the size of the latent feature is increased. Below each figure the inference time for 17M points (256^3 regular grid) and the storage requirements for the latent grid is shown. The number of trainable network parameters for feature sizes 16, 32, and 64 are 38K, 151K, and 604K, respectively, regardless of the specific grid sizes.

We train our encoder-decoder architecture end to end by inputting batches of surface point-clouds and sample points and computing output signed distance values. The set of input points is always disjoint from the set of sample points. Automatic differentiation is used to provide spatial derivatives for the SDF gradients in $\mathcal{L}_{\text{eikonal}}$.

IV. Results

The evaluation of our method is split into three parts: First, we compare different model sizes (i.e., number of learned parameters and grid resolutions) with respect to the amount of detail they can reconstruct. Second, we compare our results to a number of recent encoder architectures to highlight the advantages of our approach. Finally, we showcase the improvement of using our proposed point-to-grid feature projection over pooling-based voxelization. Details on general model ablation can be found in Section A of the supplemental document.

a) Datasets: To cover a variety of relevant scenarios, we use seven different datasets, which are summarized in Table I. The first two datasets consist of synthetic deformations of two different simulated solid objects, a low-resolution dragon and a high-resolution armadillo, which were computed using the IPC simulation framework [65]. We additionally use the raw point scan data of the dynamic FAUST (DFAUST) dataset

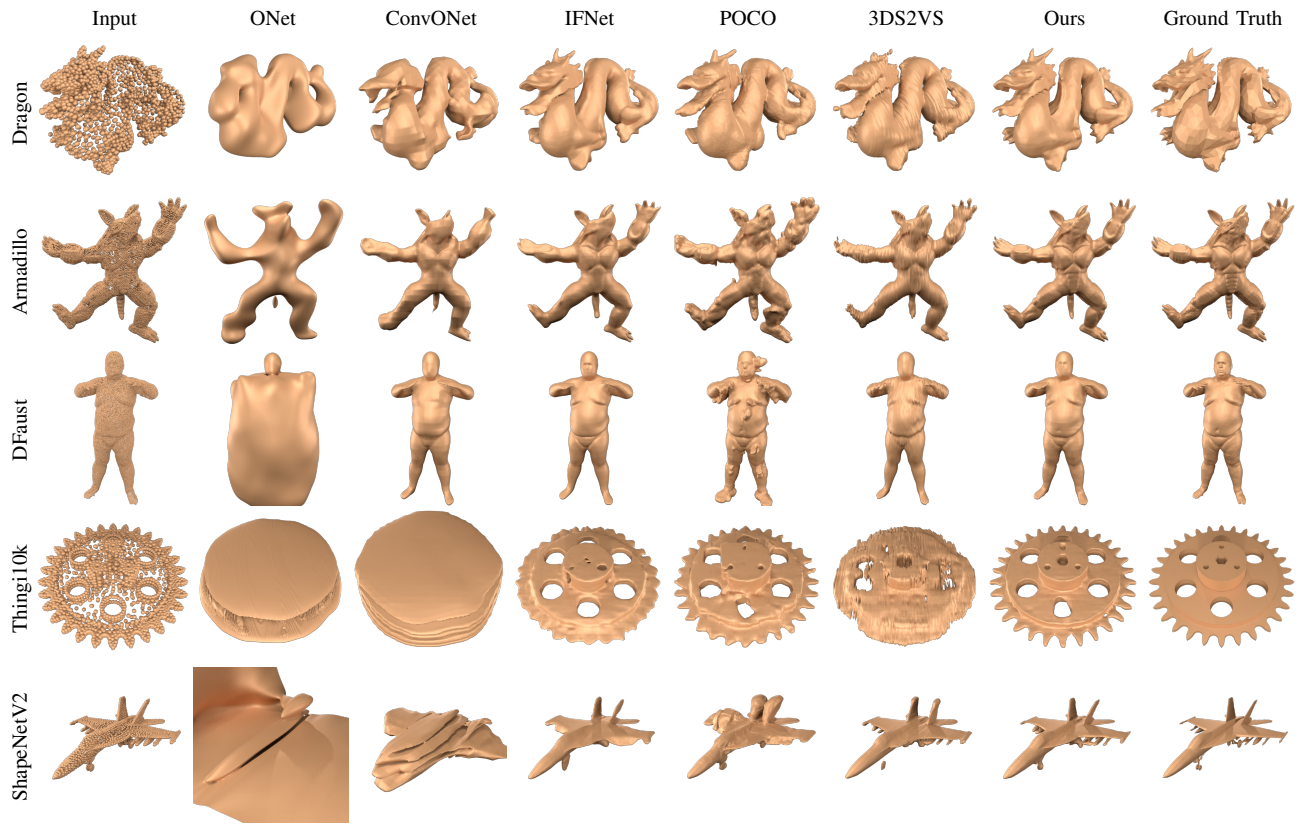

Fig. 5. Comparing our method to related baselines. All methods use a variable number of input vertices for the Thingi10k and ShapeNet v2 datasets, while POCO and 3DS2VS use a fixed number of input vertices. Please refer to the accompanying video for a more immersive comparison.

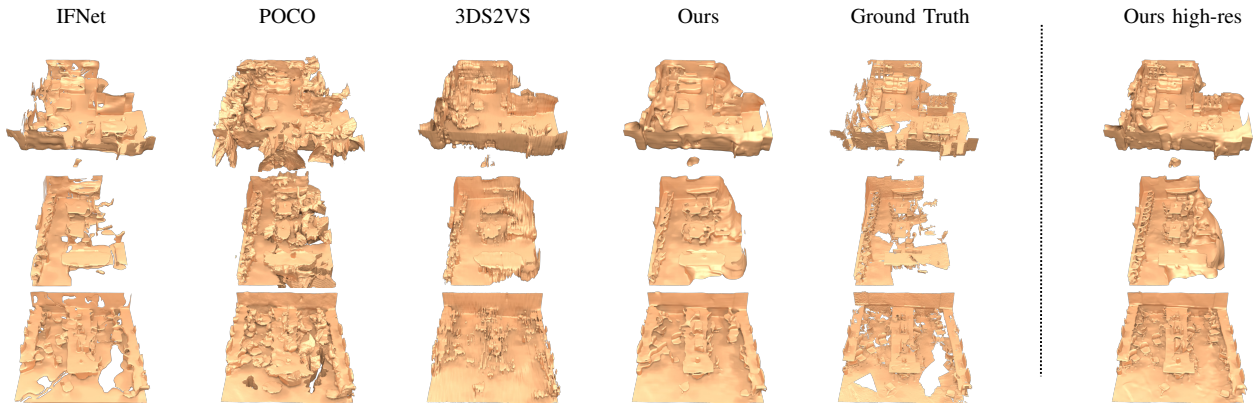

Fig. 6. Comparing our method to related baselines on the indoor scene scan dataset ScanNet [64]. The results of ONet and ConvONet are omitted due to poor quality. In the rightmost column, we additionally show the results obtained with our method utilizing double resolution grid sizes and oriented point clouds. Best viewed under magnification in the digital document. Please refer to the accompanying video for a more immersive comparison.

[66], which consists of temporally deforming 3D shapes and the ScanNet dataset [64] which contains a large variety of scanned indoor scenes. Due to the origin of these datasets, they contain varying amounts of holes and noise. To analyze performance on variable vertex count as well as single-class and multi-class encoding, we employ the Thingi10k dataset as provided by Hu et al. [67] as well as the "planes" category of the ShapeNet v2 dataset [68]. Finally, we use the public-domain models of the Objaverse dataset [1] to test the ability of our method to fit more complex shapes.

b) Baselines: We focus our comparison on other encoder-based 3D shape representation methods, as they are most similar to our work. To that end we implemented the encoder components of Occupancy Networks (ONet) [24], Convolutional Occupancy Networks (ConvONet) [28], Implicit Feature Networks (IFNet) [7], Point Convolution for Surface Reconstruction (POCO) [8] and 3DShape2VecSet (3DS2VS) [9]. To ensure a fair comparison, we use the same training and testing data for all models as well as the same modulated-SIREN decoder as discussed in Section III. In addition, we use the same eikonal loss for all baselines,

TABLE II
A SUMMARY OF THE COMPARISONS BETWEEN OUR ENCODER METHOD, ONET [24], CONVONET [28], IFNET [7], POCO [8] AND 3DSHAPE2VECSET [9]. THE TABLE SHOWS THE L1 CHAMFER DISTANCE (CD-L1), L2 CHAMFER DISTANCE (CD-L2), NORMAL CONSISTENCY (NC), INTERSECTION-OVER-UNION (IOU) AND F-SCORE (F). THE ARROW INDICATES WHETHER LOWER OR HIGHER VALUES ARE BETTER. THE BEST SCORE IS MARKED USING A BOLD FONT. ALL VALUES ARE GIVEN SCALED TO $\cdot 10^{-2}$. FOOTNOTES FOR SPECIFIC VALUES: [1] EVALUATED ON A SUBSET OF 200 RANDOM SHAPES DUE TO COMPUTATIONAL CONSTRAINTS. [2] BETTER METRIC BUT LESS DESIRABLE SOLUTION. [3] TRAINED AND EVALUATED USING A FIXED NUMBER OF INPUT POINTS INSTEAD OF A VARIABLE NUMBER.

Metric	Method	Dragon	Armadillo	DFAUST	ScanNet	Thingi10k	ShapeNet v2 w/ Eq. (3)	ShapeNet v2 w/o Eq. (3)
CD_{L1} ↓ mean	ONet	5.581	4.451	16.50	15.64	12.69	34.89	10.02
	ConvONet	3.960	2.377	1.627	10.69	10.62	3.734	4.424
	IFNet	2.681	1.259	1.015	1.559 [2]	3.888	2.180	**2.724**
	POCO	2.981	2.594	3.480 [1]	29.82	3.235 [3]	3.843 [3]	4.382 [3]
	3DS2VS	2.823	1.291	1.181	7.722	6.196 [3]	2.362 [3]	6.104 [3]
	Ours	**2.406**	**0.934**	**0.662**	**4.721**	**3.222**	**1.888**	3.547
CD_{L2} ↓ mean	ONet	0.281	0.322	3.667	3.000	2.298	19.76	1.161
	ConvONet	0.143	0.096	0.070	1.421	1.407	0.164	0.235
	IFNet	0.056	0.048	0.101	0.022 [2]	0.135	0.035	**0.055**
	POCO	0.067	0.067	0.217 [1]	14.53	0.120 [3]	0.264 [3]	0.243 [3]
	3DS2VS	0.062	0.012	0.049	1.520	0.596 [3]	0.050 [3]	0.600 [3]
	Ours	**0.049**	**0.005**	**0.004**	**0.679**	**0.112**	**0.027**	0.092
NC ↑ mean	ONet	82.20	83.35	82.70	64.51	62.16	59.61	58.11
	ConvONet	88.89	90.80	94.50	63.91	67.11	75.43	72.31
	IFNet	95.63	95.87	96.52	83.36 [2]	92.27	82.34	**81.61**
	POCO	92.39	89.06	84.93 [1]	65.13	93.57 [3]	78.01 [3]	73.66 [3]
	3DS2VS	92.64	94.12	95.33	74.22	79.74 [3]	83.35 [3]	70.11 [3]
	Ours	**96.33**	**98.22**	**97.41**	**82.27**	**94.26**	**86.75**	78.67
$IoU_{0.01}$ ↑ mean	ONet	32.41	32.60	20.90	9.428	11.65	17.70	11.97
	ConvONet	51.12	53.72	56.78	8.687	14.51	36.68	29.93
	IFNet	74.20	76.84	76.25	52.26	50.71	55.10	**39.69**
	POCO	63.80	37.25	39.52 [1]	31.83	78.42 [3]	38.76 [3]	30.57 [3]
	3DS2VS	66.79	71.32	72.96	35.49	38.66 [3]	52.54 [3]	23.98 [3]
	Ours	**87.73**	**89.76**	**85.82**	**52.29**	**80.02**	**66.96**	27.30
$F_{0.01}$ ↑ mean	ONet	48.04	49.56	23.44	16.15	16.59	18.29	18.85
	ConvONet	68.44	74.68	82.04	13.54	21.30	56.87	45.84
	IFNet	94.29	96.86	96.50	73.10 [2]	68.32	79.17	**58.06**
	POCO	86.53	56.65	68.94 [1]	20.15	91.73 [3]	61.94 [3]	43.52 [3]
	3DS2VS	87.75	93.12	95.31	52.33	54.14 [3]	78.80 [3]	40.32 [3]
	Ours	**99.47**	**99.82**	**97.89**	**71.01**	**93.86**	**93.23**	40.44

TABLE III
NUMBER OF PARAMETERS AND INFERENCE TIME OF THE DIFFERENT ENCODER BASELINES COMPARED IN TABLE II.

	ONet	ConvONet	IFNet	POCO	3DS2VS	Ours
Params	626K	1.1M	1.9M	12.8M	106M	747K
Time	2.45 s	0.7 s	15.3 s	120 s - 330 s	12.3 s - 20.1 s	2.34 s

which trains all models to compute signed distance fields. We should note that while we use 3DS2VS for reconstruction, it is more strongly targeted towards latent compression and generation. Furthermore, the implementations of POCO and 3DS2VS were unfortunately not trivially adjustable to account for a variable number of input points during training, which is why they are evaluated using a fixed number of vertices for our versions of the Thingi10K and ShapeNet v2 dataset. Specifically, we sample the mean number of points for each geometry across the respective datasets to train these models. For all comparisons with the other methods, our model utilizes five layers of latent grids with resolutions [4,8,16,32,64], a latent size of 64, $k = 8$ for kNN connectivity and EdgeConv [60] as point convolution operator. Please refer to Section A in the supplemental document for details on model component ablation.

c) Metrics: We use four different metrics in our evaluation: The widely used Chamfer distance (CD) in L1 and L2 norm, normal consistency (NC), intersection-over-union (IoU) and F-score. The Chamfer distance compares distances between two point clouds A and B by finding the distance to the closest point in B for each point in A and taking the average of the L1 and L2 norms respectively. While the values typically correlate strongly, the L2 Chamfer distance penalizes outliers more strongly than the L1 Chamfer distance and is therefore also reported for completeness. The normal consistency instead compares orientations of normals by also finding the closest point in B for each point in A, but instead

computing the cosine similarity of the normal vectors and taking the average. The same is done for the reverse direction and the results are summed together in the case of the Chamfer distance, and averaged in the case of the normal consistency.

The IoU and F-score are computed using a "thick-shell" surface representation of prediction and ground truth. For the IoU, a voxel grid of 256^3 is sampled and cells marked as 1 that are within a distance of 0.01 for both the prediction and ground truth. The IoU is then the number of cells that are 1 in both grids divided by the number of cells that are 1 in either grid.

The F-score is computed in a similar manner, but uses sampled points instead of a regular grid. In specific, we sample 100k points on the surface of the predicted and ground-truth mesh respectively. Recall is computed as the number of ground-truth points that are within 0.01 distance to the predicted surface, while precision is the reverse, i.e., the number of predicted points within 0.01 to the ground-truth surface. The F-score is finally computed as usual.

For all tests we use 100-2000 individual reconstructions and report mean values for all metrics.

d) Hardware and Framework: We train all models on RTX 4000 Ada GPUs with 20GB of memory (they also fit on an RTX 2080 Ti which increases the training times reported below by $\approx 30\%$). The training time for 400 epochs of our model using a batch size of 16 on a single GPU is between 1-2 days. Most other models trained in similar time, apart from IFNet, POCO, and 3DS2VS which required between 3-16 days on a single GPU. We implemented all of the models in this paper in the PyTorch framework [69]. In addition, we make use of PyTorch-Geometric for its 3D graph and set learning capabilities [70]. All models were trained using the Adam optimizer using a learning rate of 5e-4 and a batch size of 16. We have also implemented the tri-linear interpolation and point-to-grid projection using custom CUDA kernels. For the former, this includes the double backward pass, which is not supported by the native PyTorch interpolation function, but necessary because we are using network gradients in our training target. We limit the impact of the additional GPU memory and compute-load of this double-backward pass, by only differentiating the "small" decoder network twice, instead of the whole encoder-decoder structure.

A. Model Size

The number of features per vertex and the resolutions of the grids are tunable parameters of our architecture. Increasing them enables the network to capture more surface details, at the cost of increased storage requirements or computation time. The results are shown in Figure 4. We find that the network behaves very intuitively and that results degrade gracefully when the model size is reduced. Most notably, the models in the first column with a latent size of 16 all contain only 38K network parameters in total, of which the decoder contains just 1.4K. High-frequency features vanish first which makes small model sizes particularly appealing for applications such as approximate collision tests, e.g., in physical simulation, or deployment in resource constrained environments, e.g., edge devices.

B. Comparison to Related Work

We now discuss comparisons to the other baseline shape encoder methods on six datasets. They are shown for all methods and datasets in the accompanying video, as well as in Figure 5, in Figure 6, in Table II, and in Table III. We can summarize that our method outperforms the other baselines with respect to reconstructed surface detail in terms of all metrics on nearly all datasets. There is however one exception, where IFNet seemingly performs better on ScanNet. On closer inspection, this is due to IFNet finding a less-desirable minimum to the eikonal loss, by "wrapping" thin sheets around the sparsely distributed geometry. While this results in smaller loss values, this is clearly a less desirable and less useful result. The visual comparison in Figure 6 and the supplementary video confirm this observation.

A potential risk of using k-nearest neighbor connectivity in our method is for fine detail to become "connected". However, out of all the baselines, only our method was able to reconstruct the individual teeth of the cog wheel in the Thingi10k dataset. In addition, some methods also struggle with the hand of the of armadillo and the claws of the dragon, while our method is able to reproduce them in better detail. We attribute this to the effectiveness of our proposed architecture and point-to-grid feature projection at extracting point-level information and detail. This assumption is supported by the results in Figure 7.

From Table III we can see, that our method also is able to reconstruct a dense grid of approx. 17M points in only 2.35 s, while IFNet takes more than 6x longer. Only ConvONet is faster on the reconstruction task, however at significantly lower quality. We can conclude that our approach offers the highest quality per parameter.

Our model consistently exhibits very low L2 Chamfer distance on all datasets. This indicates that there are fewer outliers, which means that the performance of our network is more predictable for a given level of accuracy. This is underlined by the comparison in Figure 4, which shows the effect of changing latent size and grid resolutions. Very small networks with our architecture can already have remarkable representational abilities.

Finally, we introduced a simple modification to the loss function (Equation 3), to be able to compute distance fields for surfaces with inconsistent surface normals, which we tested on the ShapeNet v2 dataset. Results without this modification are shown in the grey column in Table II. We observed an improvement on all metrics for five of the six tested methods, with the exception of ONet [24], which partially performed better using the original loss. Although it should be noted that the model did not seem to be able to deal well with the changing number of vertices in the dataset, which is why we interpret this result as an outlier.

C. Point-to-Grid Feature Projection

As described in Section III-B, we propose a physically-inspired method to project particle features to the convolutional grid. We find, that the feature projection generally makes reconstructions smoother and details sharper, as can

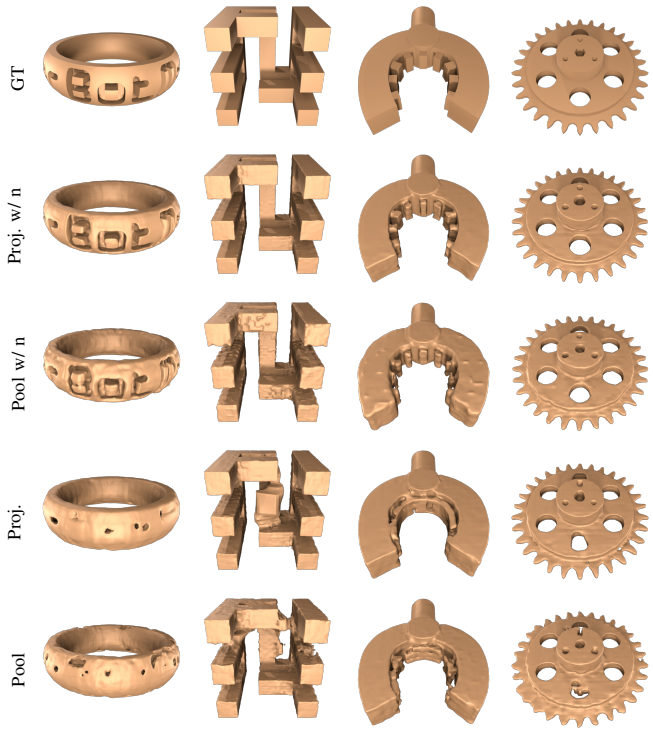

Fig. 7. Visual comparison of our architecture using pooling-based and projection-based point-to-grid transfers. Row two and three from the top use point normals as extra input.

be seen in Figure 7. We further compare our method using projection with the well-established max pooling operator in the top two rows for each metric in Table A-V of the supplemental document. Simply exchanging the pooling operator for our projection already results in improved metrics for most datasets. This exchange comes at identical network complexity and memory requirements and nearly identical training and inference speed.

In the case that the input point clouds contain information about orientation, i.e., point normals, these can be used as additional inputs to our network. To do so, they are concatenated to the surface-point positions (cf. Figure 3). Results using normals are shown in the bottom two rows for each metric in Table A-V for the pooling-based and projection-based variants of our network, respectively. Again, projection measurably improves performance, outperforming almost all other variants of our model across all datasets. The reconstructed surfaces in Figure 7 again show improved surface smoothness and detail sharpness. We also apply projection-based pooling using input normals to the ScanNet dataset, using twice the resolution of the baseline model, and show the results in the right column of Figure 6. It is again possible to see improved surface detail and smoothness. The reader is encouraged to view examples from the other datasets in the second half of the supplementary video.

Since projection maps a single point to eight grid cells in contrast to the one to one mapping of pooling, we have at times found it useful to apply dropout within the projection operator. In practice we have found a larger dropout factor to be necessary, when the shapes within a dataset are more diverse. For the Objaverse dataset, we in fact only keep one of the eight points during training, whereas for the dragon and armadillo datasets we keep all eight.

V. Conclusion and Future Work

We have shown that our hybrid approach of interleaving graph and grid convolutions, including point-to-grid projection and feature interpolation, is capable of outperforming other state-of-the-art encoders on the surface reconstruction task from point cloud inputs. We are able to do this using only surface information and additional unlabeled samples in the surrounding volume, which prevents us from having to solve for a ground truth signed distance field in advance. We have also introduced a physically-inspired projection operator for particle-to-grid feature transfers in neural networks and shown its effectiveness. Using this projection enables the reconstruction of both more detailed and less noisy surfaces, by improving "feature-field" consistency between point-based and grid-based representations.

We believe that this work could be foundational for further research regarding efficient models for 3D shape encoding. For instance, exploring the sparsity of our latent grids could enable the usage of ever higher latent resolutions, resulting in accurate encoding of almost arbitrarily complex shapes. Combined with small latent sizes, this could result in a more scalable architecture, where the accuracy can be determined by available compute capabilities. Another interesting direction of further study is the use of the latent grid as a basis for shape generation. Multiple latent representations could also be blended, or operators could be learned on the latent codes to achieve specific effects. Finally, in the spirit of using traditional geometric techniques within neural architectures, it could be explored whether projecting surface features to outer grid cells using methods such as fast marching or Laplacian smoothing could further improve predictions at distant points.

Please refer to the project website hyve.physics-simulation.org for implementation details.

Acknowledgments

This work has been partially supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) project number 310833819 and the individual baseline funding of the Computational Sciences Group within the KAUST Visual Computing Center. "Honeycomb" (shown in Fig. 1) (https://skfb.ly/onPMp) by RISD Nature Lab is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

References

[1] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A Universe of Annotated 3D Objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[2] Y. Wang, L. Rahmann, and O. Sorkine-Hornung, "Geometry-consistent neural shape representation with implicit displacement fields," in *The Tenth International Conference on Learning Representations*, 2022.

[3] X. Long, C. Lin, L. Liu, Y. Liu, P. Wang, C. Theobalt, T. Komura, and W. Wang, "NeuralUDF: Learning Unsigned Distance Fields for Multi-View Reconstruction of Surfaces With Arbitrary Topologies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[4] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural Fields in Visual Computing and Beyond," *Computer Graphics Forum*, 2022.

[5] V. Sitzmann, E. Chan, R. Tucker, N. Snavely, and G. Wetzstein, "MetaSDF: Meta-Learning Signed Distance Functions," in *Advances in Neural Information Processing Systems*, 2020.

[6] A. Ouasfi and A. Boukhayma, "Few 'Zero Level Set'-Shot Learning of Shape Signed Distance Functions in Feature Space," in *Computer Vision – ECCV 2022*, 2022.

[7] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[8] A. Boulch and R. Marlet, "POCO: Point Convolution for Surface Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[9] B. Zhang, J. Tang, M. Nießner, and P. Wonka, "3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models," *ACM Transactions on Graphics*, Jul. 2023.

[10] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," in *Advances in Neural Information Processing Systems*, 2020.

[11] M. Atzmon and Y. Lipman, "SAL: Sign Agnostic Learning of Shapes From Raw Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[12] ——, "SALD: Sign Agnostic Learning with Derivatives," in *9th International Conference on Learning Representations, ICLR 2021*, Oct. 2020.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, "Training Behavior of Deep Neural Network in Frequency Domain," in *Neural Information Processing*, 2019.

[15] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the Spectral Bias of Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, May 2019.

[16] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Computer Vision – ECCV 2020*, 2020.

[17] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[19] A. Hertz, O. Perel, R. Giryes, O. Sorkine-Hornung, and D. Cohen-Or, "SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization," in *Advances in Neural Information Processing Systems*, 2021.

[20] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker, "Modulated Periodic Activations for Generalizable Local Functional Representations," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Apr. 2021.

[21] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, Feb. 2019.

[22] P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. A. Pershing, H. T. Maia, M. M. Chiaramonte, K. T. Carlberg, and E. Grinspun, "CROM: Continuous reduced-order modeling of PDEs using implicit neural representations," in *The Eleventh International Conference on Learning Representations*, 2023.

[23] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[24] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[25] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *Proceedings of the 37th International Conference on Machine Learning*, Nov. 2020.

[26] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Liò, "Graph Neural Networks with Adaptive Readouts," *Advances in Neural Information Processing Systems*, Dec. 2022.

[27] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *ACM Transactions on Graphics*, Jul. 2019.

[28] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional Occupancy Networks," in *Computer Vision – ECCV 2020*, 2020.

[29] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction," in *Computer Vision – ECCV 2020*, 2020.

[30] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local Implicit Grid Representations for 3D Scenes," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[31] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang, "SA-ConvONet: Sign-Agnostic Optimization of Convolutional Occupancy Networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[32] B. Zhang, M. Niessner, and P. Wonka, "3DILG: Irregular Latent Grids for 3D Generative Modeling," *Advances in Neural Information Processing Systems*, Dec. 2022.

[33] P.-S. Wang, Y. Liu, and X. Tong, "Dual octree graph networks for learning adaptive volumetric shape representations," *ACM Trans. Graph.*, Jul. 2022.

[34] F. Williams, Z. Gojcic, S. Khamis, D. Zorin, J. Bruna, S. Fidler, and O. Litany, "Neural Fields As Learnable Kernels for 3D Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[35] J. Huang, H.-X. Chen, and S.-M. Hu, "A Neural Galerkin Solver for Accurate Surface Reconstruction," *ACM Trans. Graph.*, Nov. 2022.

[36] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural Kernel Surface Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[37] X. Ren, J. Huang, X. Zeng, K. Museth, S. Fidler, and F. Williams, "XCube: Large-Scale 3D Generative Modeling using Sparse Voxel Hierarchies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[38] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] A. Gupta, W. Xiong, Y. Nie, I. Jones, and B. Oğuz, "3DGen: Triplane Latent Diffusion for Textured Mesh Generation," Mar. 2023.

[40] G. Lin, L. Yang, C. Zhang, H. Pan, Y. Ping, G. Wei, T. Komura, J. Keyser, and W. Wang, "Patch-Grid: An Efficient and Feature-Preserving Neural Implicit Surface Representation," Aug. 2023.

[41] L. Yariv, O. Puny, O. Gafni, and Y. Lipman, "Mosaic-SDF for 3D Generative Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[42] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, Jul. 2022.

[43] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-Pull: Learning Signed Distance Function from Point clouds by Learning to Pull Space onto Surface," in *Proceedings of the 38th International Conference on Machine Learning*, Jul. 2021.

[44] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape As Points: A Differentiable Poisson Solver," in *Advances in Neural Information Processing Systems*, 2021.

[45] J. Chibane, M. A. mir, and G. Pons-Moll, "Neural Unsigned Distance Fields for Implicit Function Learning," in *Advances in Neural Information Processing Systems*, 2020.

[46] C. Chen, Y.-S. Liu, and Z. Han, "NeuralTPS: Learning Signed Distance Functions Without Priors From Single Sparse Point Clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2025.

[47] S. Li, G. Gao, Y. Liu, M. Gu, and Y.-S. Liu, "Implicit Filtering for Learning Neural Signed Distance Functions from 3D Point Clouds," in *Computer Vision – ECCV 2024*, 2025.

[48] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[49] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-Fidelity Neural Surface Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
[50] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-Planes: Explicit Radiance Fields in Space, Time, and Appearance," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
[51] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, Jul. 2023.
[52] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4D Gaussian Splatting for Real-Time Dynamic Scene Rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.
[53] Y. Jiang, C. Yu, T. Xie, X. Li, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang, and C. Jiang, "VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality," in *ACM SIGGRAPH 2024 Conference Papers*, Jul. 2024.
[54] Z. Wang, C. Lu, Y. Wang, F. Bao, C. LI, H. Su, and J. Zhu, "ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation," in *Advances in Neural Information Processing Systems*, 2023.
[55] Y.-C. Cheng, H.-Y. Lee, S. Tuyakov, A. Schwing, and L. Gui, "SDFusion: Multimodal 3D shape completion, reconstruction, and generation," in *CVPR*, 2023.
[56] Z. Zhao *et al.*, "Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation," Feb. 2025.
[57] S.-T. Wei, R.-H. Wang, C.-Z. Zhou, B. Chen, and P.-S. Wang, "OctGPT: Octree-based Multiscale Autoregressive Models for 3D Shape Generation," Apr. 2025.
[58] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-Voxel CNN for Efficient 3D Deep Learning," in *Advances in Neural Information Processing Systems*, 2019.
[59] Z. Liu, H. Tang, S. Zhao, K. Shao, and S. Han, "PVNAS: 3D Neural Architecture Search With Point-Voxel Convolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov. 2022.
[60] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics*, Oct. 2019.
[61] F. Harlow, "The particle-in-cell method for numerical solution of problems in fluid dynamics," Tech. Rep., Mar. 1962.
[62] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Transactions on Graphics*, Jul. 2015.
[63] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, "EikoNet: Solving the Eikonal Equation With Deep Neural Networks," *IEEE Transactions on Geoscience and Remote Sensing*, Dec. 2021.
[64] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[65] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics," *ACM Transactions on Graphics*, Aug. 2020.
[66] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering Human Bodies in Motion," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.
[67] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo, "Tetrahedral meshing in the wild," *ACM Transactions on Graphics*, Jul. 2018.
[68] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep., 2015.
[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019.
[70] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Stefan R. Jeske received his BSc and MSc degrees in Computational Engineering Science from RWTH Aachen University. He is pursuing his PhD degree in Computer Science under the supervision of Prof. Jan Bender. His research interest lies in physical simulation and deep-learning on particle-based representations.

Jonathan Klein is a Research Scientist in the Computational Sciences Group at KAUST, working with Prof. Dominik L. Michels. His research is focused on computational simulation, machine learning, procedural modeling & synthetic data generation, smart agriculture, and computational photography. He completed his Ph.D. on Computational Non-Line-of-Sight Imaging at the University of Bonn, working with Prof. Matthias B. Hullin, and his MSc and BSc degrees in Computer Science at the University of Siegen, working with Prof. Andreas Kolb.

Dominik L. Michels received the BSc degree in computer science and physics from the University of Bonn, the MSc degree in computer science, and the PhD degree from the Faculty of Mathematics and Natural Sciences. He is an associate professor of computer science and applied mathematics with KAUST, and the principal investigator of the KAUST Computational Sciences Group. Together with his team he develops principled computational methods for the accurate and efficient simulation of natural phenomena solving practically relevant problems in Scientific and Visual Computing. Previously, he joined Stanford University heading the High Fidelity Algorithmics Group within the Max Planck Center for Visual Computing and Communication. Prior to this, he did postdoctoral studies in Computing and Mathematical Sciences with the California Institute of Technology.

Jan Bender is professor of computer science and leader of the Computer Animation Group at RWTH Aachen University. He received his diploma, PhD and habilitation in computer science from the University of Karlsruhe. His research interests include the physically-based simulation of rigid body systems, deformable solids, and fluids, collision handling, cutting, fracturing, and real-time simulation methods.

TABLE A-I
THE RESULTS OF COMPARING DIFFERENT DESIGN CHOICES WITHIN THE STRUCTURE OF OUR NETWORK (SEE FIGURE 2). THE TABLE SHOWS THE L1 CHAMFER DISTANCE, NORMAL CONSISTENCY, INTERSECTION-OVER-UNION (IOU) AND F-SCORE. GNN REFERS TO GRAPH-CONVOLUTIONS AND CNN REFERS TO GRID-CONVOLUTIONS. THE CELL COLOR CODING IS DERIVED FROM THE MEAN VALUE, DIFFERENT METRICS USE DIFFERENT COLOR MAPS. THE ARROW INDICATES WHETHER LOWER OR HIGHER VALUES ARE BETTER. A DARKER COLOR CORRESPONDS TO A BETTER VALUE.

	Interpolation on		Interpolation off	
GNN	CNN on	CNN off	CNN on	CNN off
on	2.607 / 0.253	4.637 / 1.936	2.771 / 0.355	4.819 / 2.026
off	3.141 / 0.736	12.40 / 1.270	3.176 / 0.774	12.67 / 1.388

(a) Chamfer distance L1 ↓ mean / std ($\cdot 10^{-2}$)

	Interpolation on		Interpolation off	
GNN	CNN on	CNN off	CNN on	CNN off
on	95.53 / 1.480	85.75 / 7.193	94.88 / 2.055	85.54 / 8.190
off	92.50 / 4.472	85.85 / 5.997	92.30 / 4.683	85.81 / 5.969

(b) Normal Consistency ↑ mean / std ($\cdot 10^{-2}$)

	Interpolation on		Interpolation off	
GNN	CNN on	CNN off	CNN on	CNN off
on	76.69 / 8.015	34.21 / 15.19	70.35 / 12.30	38.89 / 19.61
off	61.29 / 18.40	27.74 / 10.30	59.42 / 17.65	28.92 / 11.07

(c) $\text{IoU}_{0.01}$ ↑ mean / std ($\cdot 10^{-2}$)

	Interpolation on		Interpolation off	
GNN	CNN on	CNN off	CNN on	CNN off
on	95.87 / 4.836	51.74 / 21.45	91.48 / 9.472	57.37 / 25.49
off	81.76 / 18.26	35.51 / 12.04	80.59 / 18.49	36.00 / 12.20

(d) $\text{F}_{0.01}$ ↑ mean / std ($\cdot 10^{-2}$)

TABLE A-II
ABLATION STUDY OF THE INFLUENCE OF THE GRID ORDER, GRAPH CONVOLUTION AND THE NUMBER OF NEAREST NEIGHBORS ON THE PERFORMANCE OF OUR METHOD. CD-L1 DENOTES THE L1 CHAMFER DISTANCE, NC THE NORMAL CONSISTENCY, IOU THE INTERSECTION-OVER-UNION AND F THE F-SCORE. THE BEST VALUES ARE HIGHLIGHTED IN BOLD. ALL VALUES DENOTE THE MEAN.

GNN	Res.	k-NN	CD-L1 ↓	NC ↑	$\text{IoU}_{0.01}$ ↑	$\text{F}_{0.01}$ ↑
PointConv	Decr.	None	1.045	96.603	82.651	99.165
EdgeConv	Decr.	2	0.994	97.423	85.719	99.613
EdgeConv	Incr.	2	0.953	97.765	88.754	99.684
EdgeConv	Incr.	4	0.947	97.807	89.197	99.724
EdgeConv	Incr.	8	**0.942**	**97.938**	**89.542**	**99.771**

TABLE A-III
COMPARISON OF THE L1 CHAMFER DISTANCE, NORMAL CONSISTENCY, INTERSECTION-OVER-UNION (IOU) AND F-SCORE (F) ACROSS DIFFERENT NOISY PERTURBATIONS IN RANDOM DIRECTIONS OF THE INPUT POINTS. ROWS INDICATE THE NOISE LEVEL USED DURING TRAINING AND COLUMNS DURING TESTING. THE BEST VALUE FOR EACH METRIC IS MARKED IN BOLD FONT.

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	3.349	**3.348**	3.357	3.397	4.198
5e-4	3.358	3.358	3.361	3.403	4.262
5e-3	3.381	3.379	3.397	3.448	4.268
1e-2	3.403	3.404	3.415	3.464	4.345

(a) Chamfer Distance L1 ↓ mean / std ($\cdot 10^{-2}$)

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	**93.200**	93.189	93.063	92.702	81.880
5e-4	93.114	93.097	93.016	92.633	81.777
5e-3	93.001	93.017	92.918	92.616	82.344
1e-2	93.130	93.117	93.037	92.809	83.813

(b) Normal Consistency ↑ mean / std ($\cdot 10^{-2}$)

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	**77.000**	76.950	75.272	71.615	36.144
5e-4	76.598	76.603	75.115	71.503	35.673
5e-3	76.356	76.375	75.261	72.393	36.860
1e-2	73.923	73.934	73.509	71.860	37.767

(c) $\text{IoU}_{0.01}$ ↑ mean / std ($\cdot 10^{-2}$)

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	91.725	**91.743**	91.480	90.537	48.793
5e-4	91.614	91.606	91.425	90.443	47.880
5e-3	91.459	91.469	91.244	90.532	49.494
1e-2	91.254	91.252	91.110	90.451	50.569

(d) $\text{F}_{0.01}$ ↑ mean / std ($\cdot 10^{-2}$)

APPENDIX

This section provides information on the various ablation experiments that we conducted in the context of our work.

A. Design Ablation

To highlight the impact of different design choices within our network architecture, we first conduct a number of ablation studies. The first covers three decisions within our convolution block in Figure 2: 1) using the nearest neighbor instead of linear interpolation to map values from latent grid back to the input points, 2) enabling/ disabling the graph convolutions (GNN), and 3) enabling/ disabling the grid convolutions (CNN).

The results for different combinations are reported in Table A-I. The most impactful component is clearly using the grid convolutions, which also contains the majority of trainable weights. The next important decision is enabling the graph convolution. Furthermore, interpolation seems to have a greater impact on network performance when the graph convolution is also enabled. This is because without the graph convolution, the interpolated features are immediately projected back to the grid, eliminating most implicit information gain. For maximal performance, all options should enabled.

Next, we investigate the impact of varying the number of

TABLE A-IV
COMPARISON OF THE L1 CHAMFER DISTANCE, NORMAL CONSISTENCY, INTERSECTION-OVER-UNION (IOU) AND F-SCORE (F) ACROSS DIFFERENT NOISY PERTURBATIONS IN THE DIRECTION OF VERTEX NORMALS OF THE INPUT POINTS. ROWS INDICATE THE NOISE LEVEL USED DURING TRAINING AND COLUMNS DURING TESTING. THE BEST VALUE FOR EACH METRIC IS MARKED IN BOLD FONT.

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	3.381	3.381	3.429	3.534	4.684
5e-4	3.370	3.371	3.410	3.517	4.642
5e-3	**3.365**	**3.365**	3.415	3.531	4.667
1e-2	3.417	3.417	3.452	3.567	4.731

(a) Chamfer Distance L1 ↓ mean / std $(\cdot 10^{-2})$

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	93.070	93.059	92.770	91.850	77.978
5e-4	93.035	93.033	92.768	91.804	77.722
5e-3	**93.254**	93.244	93.058	92.503	78.971
1e-2	93.200	93.213	93.165	92.915	80.882

(b) Normal Consistency ↑ mean / std $(\cdot 10^{-2})$

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	74.459	74.396	71.118	63.869	19.128
5e-4	**75.429**	75.417	71.846	64.361	19.056
5e-3	75.375	75.372	73.537	68.036	19.557
1e-2	72.430	72.527	72.392	70.457	20.812

(c) $\text{IoU}_{0.01}$ ↑ mean / std $(\cdot 10^{-2})$

Train \Test	none	5e-4	5e-3	1e-2	5e-2
none	91.198	91.196	90.369	86.334	25.295
5e-4	91.508	91.496	90.706	86.788	25.213
5e-3	**91.518**	91.512	90.996	88.918	25.550
1e-2	91.138	91.144	90.944	89.934	26.985

(d) $\text{F}_{0.01}$ ↑ mean / std $(\cdot 10^{-2})$

nearest neighbors (k) for connecting the input point cloud, using point convolutions [13] instead of edge convolutions [60], and finally reversing the resolution of latent grids. We show the results in Table A-II. In the depicted parameter sequence, the grid order has the single largest impact. This suggests that coarse features are extracted earlier and fine features later in the network. Using an increasing number of k consistently improves both CD-L1 and NC. For all experiments we use the settings corresponding to the best result.

In addition to this, we investigate the impact of increasing the number of nearest neighbors (k) in combination with changing the point-to-grid interpolation scheme. The results of this ablation are shown in Figure A-1. It becomes clear that increasing the number of nearest neighbors (k) has a positive impact on all metrics regardless of the transfer method used. However, we also observe that our proposed projection method appears to scale better with increasing k, and it outperforms pooling-based point-to-grid transfer for all values of $k > 2$. For all of our experiments we otherwise use $k = 8$, as increasing k further appears to result in diminishing returns.

B. Noise Ablation

We investigate our architecture's sensitivity to noise by training and testing using different noise levels. We add noise by perturbing input vertices by a random amount drawn from $\mathcal{U}(-\sigma, \sigma)$ in random direction (see Table A-III) or in normal direction (see Table A-IV). Here, σ is the noise level shown in the column labels and row labels of the respective tables. Since the coordinates have been normalized to the unit box, the maximum evaluated noise of 5e-2 corresponds to a shift of $1/40$ of the bounding box edge length. Our model remains stable during training up to noise levels of 1e-2 both in normal or random directions and even benefits from data augmentation using noise in normal direction.

C. Point-to-Grid Feature Projection

In addition to the visual comparison in Figure 7, Table A-V shows the metrics comparing pooling- and projection-based feature transfers across all datasets.

D. Contours of Volumetric Signed Distance Field

Despite mostly showing only the zero level-set of the computed signed distance field, out network also produces a reasonable volumetric signed distance field. We show this briefly in Figure A-2, where contours are generated at multiple iso-values for the signed distance field.

E. Non-manifold Handling

Non-manifold meshes, mostly in the form of non-watertightness or self-intersections appear in a number of different datasets (Table I). We show in Figure A-3 using an example from the Objaverse dataset, that our model is able to deal with these self-intersections fairly well.

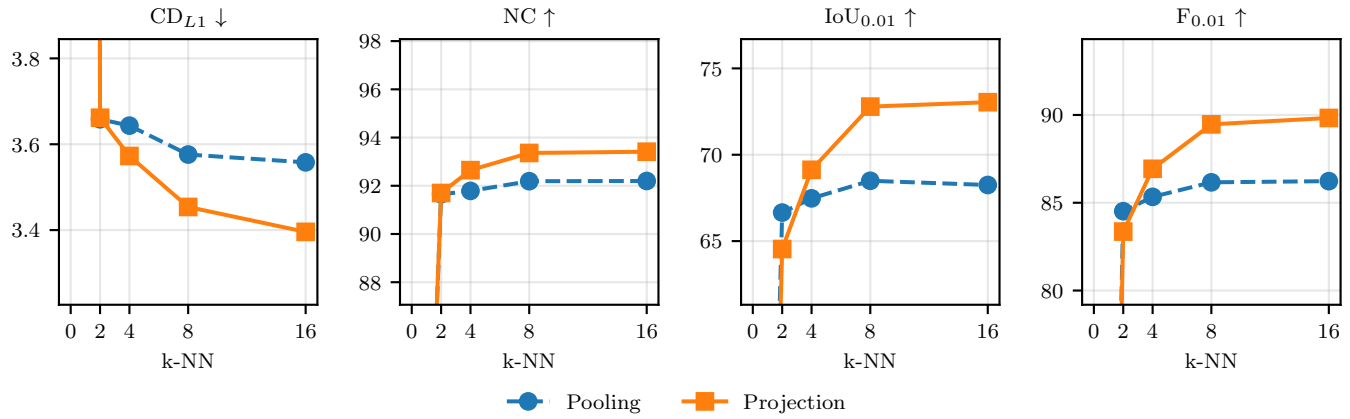

Fig. A-1. Ablation study comparing pooling and our projection operator when using different numbers of nearest neighbors (k) to connect input points for the graph convolution. Increasing k consistently results in improved metrics, where our proposed projection operator benefits more and consistently outperforms pooling for k> 2. We compare using the L1 Chamfer distance, normal consistency, intersection-over-union (IoU) and F-score.

TABLE A-V
COMPARING OUR METHOD WITH POOLING-BASED AND PROJECTION-BASED POINT-TO-GRID TRANSFERS. OPTIONALLY, INPUT VERTEX NORMALS ARE ALSO USED. WE EVALUATE THE L1 CHAMFER DISTANCE (CD-L1), NORMAL CONSISTENCY (NC), INTERSECTION-OVER-UNION (IOU) AND F-SCORE (F). BOLD FONT MARKS THE BEST VALUE IN EACH METRIC AND DATASET. ALL VALUES ARE GIVEN SCALED TO $\cdot 10^{-2}$.

Metric	Method	Dragon	Armadillo	DFAUST	ScanNet	Thingi10k	ShapeNet v2	Objaverse
CD_{L1} ↓ mean	Pool w/o n	2.455	0.956	0.627	4.339	3.270	2.011	1.222
	Proj. w/o n	**2.406**	0.934	0.662	4.721	3.222	**1.888**	1.363
	Pool w/ n	2.486	0.934	0.700	**4.156**	3.446		**1.118**
	Proj. w/ n	2.427	**0.906**	**0.626**	4.269	**3.075**		1.604
NC ↑ mean	Pool w/o n	96.25	97.86	97.65	82.98	93.65	86.11	92.95
	Proj. w/o n	96.33	98.22	97.41	82.27	94.26	**86.75**	92.00
	Pool w/ n	96.37	98.27	97.61	85.15	95.53		94.87
	Proj. w/ n	**96.53**	**98.47**	**97.83**	**85.33**	**95.89**		**95.25**
$IoU_{0.01}$ ↑ mean	Pool w/o n	84.84	88.63	88.40	54.88	78.85	61.24	65.75
	Proj. w/o n	**87.73**	89.76	85.82	52.29	80.02	**66.96**	61.49
	Pool w/ n	80.15	89.01	84.76	**57.33**	61.74		66.79
	Proj. w/ n	87.30	**92.00**	**88.76**	56.50	**87.58**		**86.53**
$F_{0.01}$ ↑ mean	Pool w/o n	98.97	99.82	**98.17**	73.68	92.45	88.53	89.35
	Proj. w/o n	99.47	99.82	97.89	71.01	93.86	**93.23**	85.28
	Pool w/ n	98.69	99.87	97.53	**75.84**	81.09		81.02
	Proj. w/ n	**99.71**	**99.89**	97.88	74.72	**97.47**		**93.45**

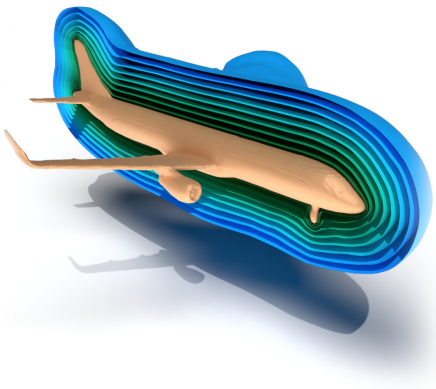

Fig. A-2. Visualization of various level-sets using our method on an instance of the planes category from the ShapeNet v2 dataset [68].

Fig. A-3. Our model is able to resolve a large number of self-intersections in the Objaverse dataset [1] without special treatment, as long as surface normals have consistent orientation. Left shows our reconstruction without self-intersections, while the right shows the ground-truth shape containing self-intersections. The transparent renders show a closeup of the marked red region to make self-intersections visible.