PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations

Edgar Tretschk[1] Ayush Tewari[1] Vladislav Golyanik[1]
Michael Zollhöfer[2] Carsten Stoll[2] Christian Theobalt[1]

[1] Max Planck Institute for Informatics, Saarland Informatics Campus
[2] Facebook Reality Labs

Abstract. Implicit surface representations, such as signed-distance functions, combined with deep learning have led to impressive models which can represent detailed shapes of objects with arbitrary topology. Since a continuous function is learned, the reconstructions can also be extracted at any arbitrary resolution. However, large datasets such as ShapeNet are required to train such models.

In this paper, we present a new mid-level patch-based surface representation. At the level of patches, objects across different categories share similarities, which leads to more generalizable models. We then introduce a novel method to learn this patch-based representation in a canonical space, such that it is as object-agnostic as possible. We show that our representation trained on one category of objects from ShapeNet can also well represent detailed shapes from any other category. In addition, it can be trained using much fewer shapes, compared to existing approaches. We show several applications of our new representation, including shape interpolation and partial point cloud completion. Due to explicit control over positions, orientations and scales of patches, our representation is also more controllable compared to object-level representations, which enables us to deform encoded shapes non-rigidly.

Keywords: implicit functions, patch-based surface representation, intra-object class generalizability

1 Introduction

Several 3D shape representations exist in the computer vision and computer graphics communities, such as point clouds, meshes, voxel grids and implicit functions. Learning-based approaches have mostly focused on voxel grids due to their regular structure, suited for convolutions. However, voxel grids [5] come with large memory costs, limiting the output resolution of such methods. Point cloud based approaches have also been explored [23]. While most approaches assume a fixed number of points, recent methods also allow for variable resolution outputs [28, 17]. Point clouds only offer a sparse representation of the surface. Meshes with fixed topology are commonly used in constrained settings with known object categories [32]. However, they are not suitable for representing objects with varying topology. Very recently, implicit function-based representations were introduced [21, 17, 4]. DeepSDF [21] learns a network which represents the continuous signed distance functions for a class of objects. The surface is represented as the 0-isosurface. Similar approaches [17, 4] use occupancy networks, where only the occupancy values are learned (similar to voxel grid-based approaches), but in a continuous representation. Implicit functions allow for representing (closed) shapes of arbitrary topology. The reconstructed surface can be extracted at any resolution, since a continuous function is learned.

All existing implicit function-based methods rely on large datasets of 3D shapes for training. Our goal is to build a generalizable surface representation which can be trained with much fewer shapes, and can also generalize to different object categories. Instead of learning an object-level representation, our *PatchNet* learns a mid-level representation of surfaces, at the level of patches. At the level of patches, objects across different categories share similarities. We learn these patches in a canonical space to further abstract from object-specific details. Patch extrinsics (position, scale and orientation of a patch) allow each patch to be translated, rotated and scaled. Multiple patches can be combined in order to represent the full surface of an object. We show that our patches can be learned using very few shapes, and can generalize across different object categories, see Fig. 1. Our representation also allows to build object-level models, *ObjectNets*, which is useful for applications which require an object-level prior.

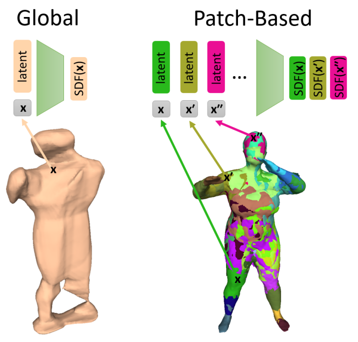

Fig. 1. In contrast to a global approach, our patch-based method generalizes to human shapes after being trained on rigid ShapeNet objects.

We demonstrate several applications of our trained models, including partial point cloud completion from depth maps, shape interpolation, and a generative model for objects. While implicit function-based approaches can reconstruct high-quality and detailed shapes, they lack controllability. We show that our patch-based implicit representation *natively* allows for controllability due to the explicit control over patch extrinsics. By user-guided rigging of the patches to the surface, we allow for articulated deformation of humans without re-encoding the deformed shapes. In addition to the generalization and editing capabilities, our representation includes all advantages of implicit surface modeling. Our patches can represent shapes of any arbitrary topology, and our reconstructions can be extracted at any arbitrary resolution using *Marching Cubes* [16]. Similar to DeepSDF [21], our network uses an auto-decoder architecture, combining classical optimization with learning, resulting in high-quality geometry.

2 Related Work

Our patch-based representation relates to many existing data structures and approaches in classical and learning-based visual computing. In the following, we focus on the most relevant existing representations, methods and applications.

Global Data Structures. There are multiple widely-used data structures for geometric deep learning such as voxel grids [5], point clouds [23], meshes [32] and implicit functions [21]. To alleviate the memory limitations and speed-up training, improved versions of voxel grids with hierarchical space partitioning [24] and tri-linear interpolation [27] were recently proposed. A mesh is an explicit discrete surface representation which can be useful in monocular rigid 3D reconstruction [32, 13]. Combined with patched-based policies, this representation can suffer from stitching artefacts [12]. All these data structures enable a limited level of detail given a constant memory size. In contrast, other representations such as sign distance functions (SDF) [6] represent surfaces implicitly as the zero-crossing of a volumetric level set function.

Recently, neural counterparts of implicit representations and approaches operating on them were proposed in the literature [21, 17, 4, 18]. Similarly to SDFs, these methods extract surfaces as zero level sets or decision boundaries, while differing in the type of the learned function. Thus, DeepSDF is a learnable variant of SDFs [21], whereas Mescheder *et al.* [17] train a spatial classifier (indicator function) for regions inside and outside of the scene. In theory, both methods allow for surface extraction at unlimited resolution. Neural implicit functions have already demonstrated their effectiveness and robustness in many follow-up works and applications such as single-view 3D reconstruction [25, 15] as well as static [28] and dynamic [19] object representation. While SAL [1] perform shape completion from noisy full raw scans, one of our applications is shape completion from partial data with local refinement. Unlike the aforementioned global approaches, PatchNets generalize much better, for example to new categories.

Patch-Based Representations. Ohtake *et al.* [20] use a combination of implicit functions for versatile shape representation and editing. Several neural techniques use mixtures of geometric primitives as well [31, 11, 7, 9, 33]. The latter have been shown as helpful abstractions in such tasks as shape segmentation, interpolation, classification and recognition, as well as 3D reconstruction. Tulsiani *et al.* [31] learn to assemble shapes of various categories from explicit 3D geometric primitives (*e.g.,* cubes and cuboids). Their method discovers a consistent structure and allows to establish semantic correspondences between the samples. Genova *et al.* [11] further develop the idea and learn a general template from data which is composed of implicit functions with local support. Due to the function choice, *i.e.,* scaled axis-aligned anisotropic 3D Gaussians, shapes with sharp edges and thin structures are challenging for their method. In *CVXNets* [7], solid objects are assembled in a piecewise manner from convex elements. This results in a differentiable form which is directly usable in physics and graphics engines. Deprelle *et al.* [9] decompose shapes into learnable combinations of deformable elementary 3D structures. VoronoiNet [33] is a deep generative network which operates on a differentiable version of Voronoi diagrams. The concurrent NASA method [8] focuses on articulated deformations, which is one of our applications. In contrast to other patch-based approaches, our learned patches are not limited to hand-crafted priors but instead are more flexible and expressive.

3 Proposed Approach

We represent the surface of any object as a combination of several surface patches. The patches form a mid-level representation, where each patch represents the surface within a specified radius from its center. This representation is generalizable across object categories, as most objects share similar geometry at the patch level. In the following, we explain how the patches are represented using artificial neural networks, the losses required to train such networks, as well as the algorithm to combine multiple patches for smooth surface reconstruction.

3.1 Implicit Patch Representation

We represent a full object i as a collection of $N_P = 30$ patches. A patch p represents a surface within a sphere of radius $r_{i,p} \in \mathbb{R}$, centered at $\mathbf{c}_{i,p} \in \mathbb{R}^3$. Each patch can be oriented by a rotation about a canonical frame, parametrized by Euler angles $\phi_{i,p} \in \mathbb{R}^3$. Let $\mathbf{e}_{i,p} = (r_{i,p}, \mathbf{c}_{i,p}, \phi_{i,p}) \in \mathbb{R}^7$ denote all extrinsic patch parameters. Representing the patch surface in a canonical frame of reference lets us normalize the query 3D point, leading to more object-agnostic and generalizable patches.

The patch surface is represented as an implicit signed-distance function (SDF), which maps 3D points to their signed distance from the closest surface. This offers several advantages, as these functions are a continuous representation of the surface, unlike point clouds or meshes. In addition, the surface can be extracted at any resolution without large memory requirement, unlike for voxel grids. In contrast to prior work [33, 11], which uses simple patch primitives, we parametrize the patch surface as a neural network (PatchNet). Our network architecture is based on the auto-decoder of DeepSDF [21]. The input to the network is a *patch latent code* $\mathbf{z} \in \mathbb{R}^{N_z}$ of length $N_z = 128$, which describes the patch surface, and a 3D query point $\mathbf{x} \in \mathbb{R}^3$. The output is the scalar SDF value of the surface at $\mathbf{x}$. Similar to DeepSDF, we use eight weight-normalized [26] fully-connected layers with 128 output dimensions and ReLU activations, and we also concatenate $\mathbf{z}$ and $\mathbf{x}$ to the input of the fifth layer. The last fully-connected layer outputs a single scalar to which we apply *tanh* to obtain the SDF value.

3.2 Preliminaries

Preprocessing: Given a watertight mesh, we preprocess it to obtain SDF values for 3D point samples. First, we center each mesh and fit it tightly into the unit sphere. We then sample points, mostly close to the surface, and compute their truncated signed distance to the object surface, with truncation at 0.1. For more details on the sampling strategy, please refer to [21].

Auto-Decoding: Unlike the usual setting, we do not use an encoder that regresses patch latent codes and extrinsics. Instead, we follow DeepSDF [21] and auto-decode shapes: we treat the patch latent codes and extrinsics of each object as free variables to be optimized for during training. I.e., instead of back-propagating into an encoder, we employ the gradients to learn these parameters directly during training.

Initialization: Since we perform auto-decoding, we treat the patch latent codes and extrinsics as free variables, similar to classical optimization. Therefore, we can directly initialize them. All patch latent codes are initially set to zero, and the patch positions are initialized by greedy farthest point sampling of point samples of the object surface. We set each patch radius to the minimum such that each surface point sample is covered by its closest patch. The patch orientation aligns the z-axis of the patch coordinate system with the surface normal.

3.3 Loss Functions

We train PatchNet by auto-decoding N full objects. The patch latent codes of an object i are $\mathbf{z}_i = [\mathbf{z}_{i,0}, \mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,N_P-1}]$, with each patch latent code of length N_z. Patch extrinsics are represented as $\mathbf{e}_i = [\mathbf{e}_{i,0}, \mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,N_P-1}]$. Let θ denote the trainable weights of PatchNet. We employ the following loss function:

$$\mathcal{L}(\mathbf{z}_i, \mathbf{e}_i, \theta) = \mathcal{L}_{\text{recon}}(\mathbf{z}_i, \mathbf{e}_i, \theta) + \mathcal{L}_{\text{ext}}(\mathbf{e}_i) + \mathcal{L}_{\text{reg}}(\mathbf{z}_i) \ . \tag{1}$$

Here, $\mathcal{L}_{recon}$ is the surface reconstruction loss, $\mathcal{L}_{ext}$ is the extrinsic loss guiding the extrinsics for each patch, and $\mathcal{L}_{reg}$ is a regularizer on the patch latent codes.

Reconstruction Loss: The reconstruction loss minimizes the SDF values between the predictions and the ground truth for each patch:

$$\mathcal{L}_{\text{recon}}(\mathbf{z}_i, \mathbf{e}_i, \theta) = \frac{1}{N_P} \sum_{p=0}^{N_P-1} \frac{1}{|S(\mathbf{e}_{i,p})|} \sum_{\mathbf{x} \in S(\mathbf{e}_{i,p})} \left\| f(\mathbf{x}, \mathbf{z}_{i,p}, \theta) - s(\mathbf{x}) \right\|_1, \tag{2}$$

where $f(\cdot)$ and $s(\mathbf{x})$ denote a forward pass of the network and the ground truth truncated SDF values at point $\mathbf{x}$, respectively; $S(\mathbf{e}_{i,p})$ is the set of all (normalized) point samples that lie within the bounds of patch p with extrinsics $\mathbf{e}_{i,p}$.

Extrinsic Loss: The composite extrinsic loss ensures all patches contribute to the surface and are placed such that the surfaces are learned in a canonical space:

$$\mathcal{L}_{\text{ext}}(\mathbf{e}_i) = \mathcal{L}_{\text{sur}}(\mathbf{e}_i) + \mathcal{L}_{\text{cov}}(\mathbf{e}_i) + \mathcal{L}_{\text{rot}}(\mathbf{e}_i) + \mathcal{L}_{\text{scl}}(\mathbf{e}_i) + \mathcal{L}_{\text{var}}(\mathbf{e}_i) \ . \tag{3}$$

$\mathcal{L}_{\text{sur}}$ ensures that every patch stays close to the surface:

$$\mathcal{L}_{\text{sur}}(\mathbf{e}_i) = \omega_{\text{sur}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} \max(\min_{\mathbf{x} \in \mathbf{O}_i} \left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2^2, t) \ . \tag{4}$$

Here, $\mathbf{O}_i$ is the set of surface points of object i. We use this term only when the distance between a patch and the surface is greater than a threshold $t = 0.06$.

A symmetric coverage loss $\mathcal{L}_{\text{cov}}$ encourages each point on the surface to be covered by a at least one patch:

$$\mathcal{L}_{\text{cov}}(\mathbf{e}_i) = \omega_{\text{cov}} \cdot \frac{1}{|\mathbf{U}_i|} \sum_{\mathbf{x} \in \mathbf{U}_i} \frac{w_{i,p,\mathbf{x}}}{\sum_p w_{i,p,\mathbf{x}}} (\left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2 - r_{i,p}) \ , \tag{5}$$

where $\mathbf{U}_i \subseteq \mathbf{O}_i$ are all surface points that are not covered by any patch, *i.e.*, outside the bounds of all patches. $w_{i,p,\mathbf{x}}$ weighs the patches based on their distance from $\mathbf{x}$, with $w_{i,p,\mathbf{x}} = \exp\left(-0.5 \cdot ((\left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2 - r_{i,p})/\sigma)^2\right)$ where $\sigma = 0.05$.

We also introduce a loss to align the patches with the surface normals. This encourages the patch surface to be learned in a canonical frame of reference:

$$\mathcal{L}_{\text{rot}}(\mathbf{e}_i) = \omega_{\text{rot}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} (1 - \langle \phi_{i,p} \cdot [0,0,1]^T, \mathbf{n}_{i,p} \rangle)^2 \ . \tag{6}$$

Here, $\mathbf{n}_{i,p}$ is the surface normal at the point $\mathbf{o}_{i,p}$ closest to the patch center, *i.e.,* $\mathbf{o}_{i,p} = \operatorname*{argmin}_{\mathbf{x} \in \mathbf{O}_i} \left\| \mathbf{x} - \mathbf{c}_{i,p} \right\|_2$.

Finally, we introduce two losses for the extent of the patches. The first loss encourages the patches to be reasonably small. This prevents significant overlap between different patches:

$$\mathcal{L}_{\text{scl}}(\mathbf{e}_i) = \omega_{\text{scl}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} r_{i,p}^2 \ . \tag{7}$$

The second loss encourages all patches to be of similar sizes. This prevents the surface to be reconstructed only using very few large patches:

$$\mathcal{L}_{\text{var}}(\mathbf{e}_i) = \omega_{\text{var}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} (r_{i,p} - m_i)^2 \ , \tag{8}$$

where m_i is the mean patch radius of object i.

Regularizer: Similar to DeepSDF, we add an ℓ_2-regularizer on the latent codes assuming a Gaussian prior distribution:

$$\mathcal{L}_{\text{reg}}(\mathbf{z}_i) = \omega_{\text{reg}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} \left\| \mathbf{z}_{i,p} \right\|_2^2 \ . \tag{9}$$

Optimization: At training time, we optimize the following problem:

$$\underset{\theta,\{\mathbf{z}_i\}_i,\{\mathbf{e}_i\}_i}{\operatorname{argmin}} \sum_{i=0}^{N-1} \mathcal{L}(\mathbf{z}_i, \mathbf{e}_i, \theta) \ . \tag{10}$$

At test time, we can reconstruct any surface using our learned patch-based representation. Using the same initialization of extrinsics and patch latent codes, and given point samples with their SDF values, we optimize for the patch latent codes and the patches extrinsics with fixed network weights.

3.4 Blended Surface Reconstruction

For a smooth surface reconstruction of object i, *e.g.* for Marching Cubes, we blend between different patches in the overlapping regions to obtain the blended SDF prediction $g_i(\mathbf{x})$. Specifically, $g_i(\mathbf{x})$ is computed as a weighted linear combination of the SDF values $f(\mathbf{x}, \mathbf{z}_{i,p}, \theta)$ of the overlapping patches:

$$g_i(\mathbf{x}) = \sum_{p \in P_{i,\mathbf{x}}} \frac{w_{i,p,\mathbf{x}}}{\sum_{p \in P_{i,\mathbf{x}}} w_{i,p,\mathbf{x}}} f(\mathbf{x}, \mathbf{z}_{i,p}, \theta), \tag{11}$$

with $P_{i,\mathbf{x}}$ denoting the patches which overlap at point $\mathbf{x}$. For empty $P_{i,\mathbf{x}}$, we set $g_i(\mathbf{x}) = 1$. The blending weights are defined as:

$$w_{i,p,\mathbf{x}} = \exp\left(-\frac{1}{2}\left(\frac{\left\|\mathbf{c}_{i,p} - \mathbf{x}\right\|_2}{\sigma}\right)^2\right) - \exp\left(-\frac{1}{2}\left(\frac{r_{i,p}}{\sigma}\right)^2\right), \tag{12}$$

with $\sigma = r_{i,p}/3$. The offset ensures that the weight is zero at the patch boundary.

4 Experiments

In the following, we show the effectiveness of our patch-based representation on several different problems. For an ablation study of the loss functions, please refer to the supplemental.

4.1 Settings

Datasets We employ *ShapeNet* [3] for most experiments. We perform preprocessing with the code of Stutz *et al.* [29], similar to [17, 10], to make the meshes watertight and normalize them within a unit cube. For training and test splits, we follow Choy *et al.* [5]. The results in Tables 1 and 2 use the full test set. Other results refer to a reduced test set, where we randomly pick 50 objects from each of the 13 categories. In the supplemental, we show that our results on the reduced test set are representative of the full test set. In addition, we use Dynamic FAUST [2] for testing. We subsample the test set from DEMEA [30] by concatenating all test sequences and taking every 20th mesh. We generate 200k SDF point samples per shape during preprocessing.

Metrics We use three error metrics. For Intersection-over-Union (IoU), higher is better. For Chamfer distance (Chamfer), lower is better. For F-score, higher is better. The supplementary material contains further details on these metrics.

Training Details We train our networks using *PyTorch* [22]. The number of epochs is 1000, the learning rate for the network is initially $5 \cdot 10^{-4}$, and for the patch latent codes and extrinsics 10^{-3}. We half both learning rates every 200 epochs. For optimization, we use Adam [14] and a batch size of 64. For each object in the batch, we randomly sample 3k SDF point samples. The weights for the losses are: $\omega_{\text{scl}} = 0.01$, $\omega_{\text{var}} = 0.01$, $\omega_{\text{sur}} = 5$, $\omega_{\text{rot}} = 1$, $\omega_{\text{sur}} = 200$. We linearly increase ω_{reg} from 0 to 10^{-4} for 400 epochs and then keep it constant.

Baseline We design a "global-patch" baseline similar to DeepSDF, which only uses a single patch without extrinsics. The patch latent size is 4050, matching ours. The learning rate scheme is the same as for our method.

4.2 Surface Reconstruction

We first consider surface reconstruction.

Results We train our approach on a subset of the training data, where we randomly pick 100 shapes from each category. In addition to comparing with our baseline, we compare with DeepSDF [21] as setup in their paper. Both DeepSDF and our baseline use the subset. Qualitative results are shown in Fig. 2 and 3.

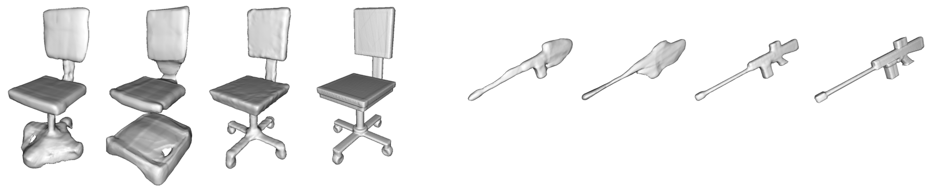

Fig. 2. Surface Reconstruction. From left to right: DeepSDF, baseline, ours, groundtruth.

Table 1 shows the quantitative results for surface reconstruction. We significantly outperform DeepSDF and our baseline almost everywhere, demonstrating the higher-quality afforded by our patch-based representation.

We also compare with several state-of-the-art approaches on implicit surface reconstruction, OccupancyNetworks [17], Structured Implicit Functions [11] and Deep Structured Implicit Functions [10][1]. While they are trained on the full

[1] DSIF is also known as *Local Deep Implicit Functions for 3D Shape.*

Table 1. Surface Reconstruction. We significantly outperform DeepSDF [21] and our baseline on all categories of ShapeNet almost everywhere.

	IoU			Chamfer			F-score		
Category	DeepSDF	Baseline	Ours	DeepSDF	Baseline	Ours	DeepSDF	Baseline	Ours
airplane	84.9	65.3	**91.1**	0.012	0.077	**0.004**	83.0	72.9	**97.8**
bench	78.3	68.0	**85.4**	0.021	0.065	**0.006**	91.2	80.6	**95.7**
cabinet	92.2	88.8	**92.9**	**0.033**	0.055	0.110	**91.6**	86.4	91.2
car	87.9	83.6	**91.7**	**0.049**	0.070	**0.049**	82.2	74.5	**87.7**
chair	81.8	72.9	**90.0**	0.042	0.110	**0.018**	86.6	75.5	**94.3**
display	91.6	86.5	**95.2**	**0.030**	0.061	0.039	93.7	87.0	**97.0**
lamp	74.9	63.0	**89.6**	0.566	0.438	**0.055**	82.5	69.4	**94.9**
rifle	79.0	68.5	**93.3**	0.013	0.039	**0.002**	90.9	82.3	**99.3**
sofa	92.5	85.4	**95.0**	0.054	0.226	**0.014**	92.1	84.2	**95.3**
speaker	91.9	86.7	**92.7**	**0.050**	0.094	0.243	87.6	79.4	**88.5**
table	84.2	71.9	**89.4**	0.074	0.156	**0.018**	91.1	79.2	**95.0**
telephone	96.2	95.0	**98.1**	0.008	0.016	**0.003**	97.7	96.2	**99.4**
watercraft	85.2	79.1	**93.2**	0.026	0.041	**0.009**	87.8	80.2	**96.4**
mean	77.4	76.5	**92.1**	0.075	0.111	**0.044**	89.9	80.6	**94.8**

ShapeNet shapes, we train our model only on a small subset. Even in this disadvantageous and challenging setting, we outperform these approaches on most categories, see Table 2. Note that we compute the metrics consistently with Genova *et al.* [10] and thus can directly compare to numbers reported in their paper.

Table 2. Surface Reconstruction. We outperform OccupancyNetworks (OccNet) [17], Structured Implicit Functions (SIF) [11], and Deep Structured Implicit Functions (DSIF) [10] almost everywhere.

	IoU				Chamfer				F-score			
Category	OccNet	SIF	DSIF	Ours	OccNet	SIF	DSIF	Ours	OccNet	SIF	DSIF	Ours
airplane	77.0	66.2	**91.2**	91.1	0.016	0.044	0.010	**0.004**	87.8	71.4	96.9	**97.8**
bench	71.3	53.3	**85.6**	85.4	0.024	0.082	0.017	**0.006**	87.5	58.4	94.8	**95.7**
cabinet	86.2	78.3	**93.2**	92.9	0.041	0.110	**0.033**	0.110	86.0	59.3	**92.0**	91.2
car	83.9	77.2	90.2	**91.7**	0.061	0.108	**0.028**	0.049	77.5	56.6	87.2	**87.7**
chair	73.9	57.2	87.5	**90.0**	0.044	0.154	0.034	**0.018**	77.2	42.4	90.9	**94.3**
display	81.8	69.3	94.2	**95.2**	0.034	0.097	**0.028**	0.039	82.1	56.3	94.8	**97.0**
lamp	56.5	41.7	77.9	**89.6**	0.167	0.342	0.180	**0.055**	62.7	35.0	83.5	**94.9**
rifle	69.5	60.4	89.9	**93.3**	0.019	0.042	0.009	**0.002**	86.2	70.0	97.3	**99.3**
sofa	87.2	76.0	94.1	**95.0**	0.030	0.080	0.035	**0.014**	85.9	55.2	92.8	**95.3**
speaker	82.4	74.2	90.3	**92.7**	0.101	0.199	**0.068**	0.243	74.7	47.4	84.3	**88.5**
table	75.6	57.2	88.2	**89.4**	0.044	0.157	0.056	**0.018**	84.9	55.7	92.4	**95.0**
telephone	90.9	83.1	97.6	**98.1**	0.013	0.039	0.008	**0.003**	94.8	81.8	98.1	**99.4**
watercraft	74.7	64.3	90.1	**93.2**	0.041	0.078	0.020	**0.009**	77.3	54.2	93.2	**96.4**
mean	77.8	66.0	90.0	**92.1**	0.049	0.118	**0.040**	0.044	81.9	59.0	92.2	**94.8**

Generalization Our patch-based representation is more generalizable compared to existing representations. To demonstrate this, we design several experiments with different training data. We modify the learning rate schemes to equalize the number of network weight updates. For each experiment, we compare our method with the baseline approaches described above. We use a reduced ShapeNet test set, which consists of 50 shapes from each category. Fig. 3 shows qualitative results and comparisons. We also show cross-dataset generalization by evaluating on 647 meshes from the Dynamic FAUST [2] test set. In the first experiment, we train the network on shapes from the *Cabinet* category and try to reconstruct shapes from every other category. We significantly outperform the baselines almost everywhere, see Table 3. The improvement is even more noticeable for cross dataset generalization with around 70% improvement in the F-score compared to our global-patch baseline.

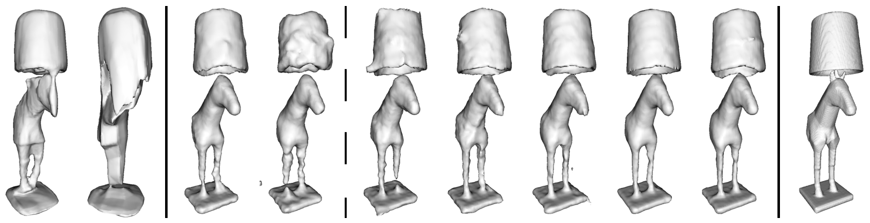

Fig. 3. Generalization. From left to right: DeepSDF, baseline, ours on one category, ours on one shape, ours on 1 shape per category, ours on 3 per category, ours on 10 per category, ours on 30 per category, ours on 100 per category, and groundtruth.

In the second experiment, we evaluate the amount of training data required to train our network. We train both our network as well as the baselines on 30, 10, 3 and 1 shapes per-category of ShapeNet. In addition, we also include an experiment training the networks on a single randomly picked shape from ShapeNet. Fig. 4 shows the errors for ShapeNet (mean across categories) and Dynamic FAUST. The performance of our approach degrades only slightly with a decreasing number of training shapes. However, the baseline approach of DeepSDF degrades much more severely. This is even more evident for cross dataset generalization on Dynamic FAUST, where the baseline cannot perform well even with a larger number of training shapes, while we perform similarly across datasets.

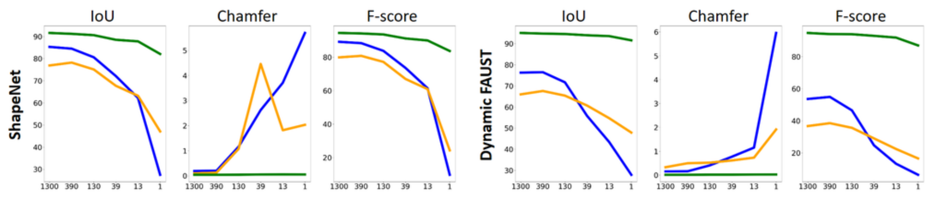

Fig. 4. Generalization. We train our PatchNet (green), the global-patch baseline (orange), and DeepSDF (blue) on different numbers of shapes (x-axis). Results on different metrics on our reduced test sets are shown on the y-axis. For IoU and F-score, higher is better. For Chamfer distance, lower is better.

Table 3. Generalization. Networks trained on the *Cabinet* category, but evaluated on every category of ShapeNet, as well as on Dynamic FAUST. We significantly outperform the baseline (BL) and DeepSDF (DSDF) almost everywhere.

	IoU			Chamfer			F-score		
Category	BL	DSDF	Ours	BL	DSDF	Ours	BL	DSDF	Ours
airplane	33.5	56.9	**88.2**	0.668	0.583	**0.005**	33.5	61.7	**96.3**
bench	49.1	58.8	**80.4**	0.169	0.093	**0.006**	63.6	76.3	**93.3**
cabinet	86.0	91.1	**91.4**	0.045	**0.025**	0.121	86.4	**92.6**	91.7
car	78.4	83.7	**92.0**	0.101	0.074	**0.050**	62.7	73.9	**87.2**
chair	50.7	61.8	**86.9**	0.473	0.287	**0.012**	49.1	65.2	**92.5**
display	83.2	87.6	**94.4**	0.111	0.065	**0.052**	83.9	89.6	**96.9**
lamp	49.7	59.3	**86.6**	0.689	2.645	**0.082**	50.4	64.5	**93.4**
rifle	56.4	56.1	**91.8**	0.114	2.669	**0.002**	71.0	54.7	**99.1**
sofa	81.1	87.3	**94.8**	0.245	0.193	**0.010**	74.2	84.6	**95.2**
speaker	83.2	88.3	**90.5**	0.163	**0.080**	0.232	71.8	80.1	**84.9**
table	55.0	73.6	**88.4**	0.469	0.222	**0.020**	61.8	82.8	**95.0**
telephone	90.4	94.7	**97.3**	0.051	0.015	**0.004**	90.8	96.1	**99.2**
watercraft	66.5	73.5	**91.8**	0.115	0.157	**0.006**	63.0	74.2	**96.2**
mean	66.4	74.8	**90.3**	0.263	0.547	**0.046**	66.3	76.6	**93.9**
DFAUST	57.8	71.2	**94.4**	0.751	0.389	**0.012**	25.0	45.4	**94.0**

Table 4. Ablative Analysis. We evaluate the performance using different numbers of patches, as well as using variable sizes of the patch latent code/hidden dimensions, and the training data. The training time is measured on an Nvidia V100 GPU.

	IoU	Chamfer	F-score	Time
$N_P = 3$	73.8	0.15	72.9	1h
$N_P = 10$	85.2	0.049	88.0	1.5h
size 32	82.8	0.066	84.7	1.5h
size 512	95.3	0.048	97.2	8h
full dataset	92.2	0.050	94.8	156h
ours	91.6	0.045	94.5	2h

Ablation Experiments We perform several ablative analysis experiments to evaluate our approach. We first evaluate the number of patches required to reconstruct surfaces. Table 4 reports these numbers on the reduced test set. The patch networks here are trained on the reduced training set, consisting of 100 shapes per ShapeNet category. As expected, the performance becomes better with a larger number of patches, since this would lead to smaller patches which can capture more details and generalize better. We also evaluate the impact of different sizes of the latent codes and hidden dimensions used for the patch network. Larger latent codes and hidden dimensions lead to higher quality results. Similarly, training on the full training dataset, consisting of $33k$ shapes leads to higher quality. However, all design choices with better performance come at the cost of longer training times, see Table 4.

4.3 Object-Level Priors

We also experiment with *category-specific* object priors. We add ObjectNet (four FC layers with hidden dimension 1024 and ReLU activations) in front of PatchNet and our baselines. From object latent codes of size 256, ObjectNet regresses patch latent codes and extrinsics as an intermediate representation usable with PatchNet. ObjectNet effectively increases the network capacity of our baselines.

Training We initialize all object latents with zeros and the weights of ObjectNet's last layer with very small numbers. We initialize the bias of ObjectNet's last layer with zeros for patch latent codes and with the extrinsics of an arbitrary object from the category as computed by our initialization in Sec. 3.2. We pretrain PatchNet on ShapeNet. For our method, the PatchNet is kept fixed from this point on. As training set, we use the full training split of the ShapeNet

category for which we train. We remove $\mathcal{L}_{\text{rot}}$ completely as it significantly lowers quality. The $L2$ regularization is only applied to the object latent codes. We set $\omega_{\text{var}} = 5$. ObjectNet is trained in three phases, each lasting 1000 epochs. We use the same initial learning rates as when training PatchNet, except in the last phase, where we reduce them by a factor of 5. The batch size is 128.
Phase I: We pretrain ObjectNet to ensure good patch extrinsics. For this, we use the extrinsic loss, $\mathcal{L}_{\text{ext}}$ in Eq. 3, and the regularizer. We set $\omega_{\text{scl}} = 2$.
Phase II: Next, we learn to regress patch latent codes. First, we add a layer that multiplies the regressed scales by 1.3. We then store these extrinsics. Afterwards, we train using $\mathcal{L}_{\text{recon}}$ and two $L2$ losses that keep the regressed position and scale close to the stored extrinsics, with respective weights $1, 3$, and 30.
Phase III: The complete loss $\mathcal{L}$ in Eq. 1, with $\omega_{\text{scl}} = 0.02$, yields final refinements.

Coarse Correspondences Fig. 5 shows that the learned patch distribution is consistent across objects, establishing coarse correspondences between objects.

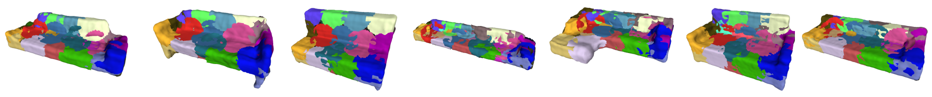

Fig. 5. Coarse Correspondences. Note the consistent coloring of the patches.

Interpolation Due to the implicitly learned coarse correspondences, we can encode test objects into object latent codes and then linearly interpolate between them. Fig. 6 shows that interpolation of the latent codes leads to a smooth morph between the decoded shapes in 3D space.

Generative Model We can explore the learned object latent space further by turning ObjectNet into a generative model. Since auto-decoding does not yield an encoder that inputs a known distribution, we have to estimate the unknown input distribution. Therefore, we fit a multivariate Gaussian to the object latent codes obtained at training time. We can then sample new object latent codes from the fitted Gaussian and use them to generate new objects, see Fig. 6.

Partial Point Cloud Completion Given a partial point cloud, we can optimize for the object latent code which best explains the visible region. ObjectNet acts as a prior which completes the missing parts of the shape. For our method, we pretrained our PatchNet on a different object category and keep it fixed, and then train ObjectNet on the target category, which makes this task more challenging for us. We choose the versions of our baselines where the eight final layers are pretrained on all categories and finetuned on the target shape category. We

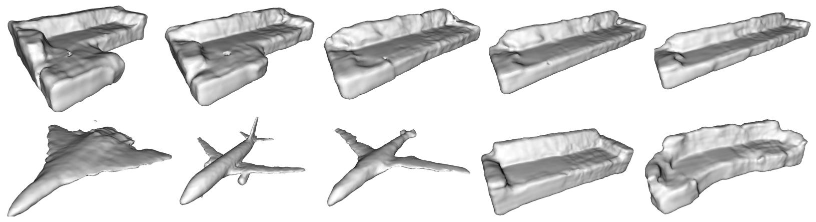
Fig. 6. Interpolation (top). The left and right end points are encoded test objects. Generative Models (bottom). We sample object latents from ObjectNet's fitted prior.

evaluated several other settings, with this one being the most competitive. See the supplemental for more on surface reconstruction with object-level priors.

Optimization: We initialize with the average of the object latent codes obtained at training time. We optimize for 600 iterations, starting with a learning rate of 0.01 and halving it every 200 iterations. Since our method regresses the patch latent codes and extrinsics as an intermediate step, we can further refine the result by treating this intermediate patch-level representation as free variables. Specifically, we refine the patch latent code for the last 100 iterations with a learning rate of 0.001, while keeping the extrinsics fixed. This allows to integrate details not captured by the object-level prior. Fig. 7 demonstrates this effect. During optimization, we use the reconstruction loss, the $L2$ regularizer and the coverage loss. The other extrinsics losses have a detrimental effect on patches that are outside the partial point cloud. We use 8k samples per iteration.

We obtain the partial point clouds from depth maps similar to Park *et al.* [21]. We also employ their free-space loss, which encourages the network to regress positive values for samples between the surface and the camera. We use 30% free-space samples. We consider depth maps from a fixed and from a per-scene random viewpoint. For shape completion, we report the F-score between the full groundtruth mesh and the reconstructed mesh. Similar to Park *et al.* [21], we also compute the mesh accuracy for shape completion. It is the 90th percentile of shortest distances from the surface samples of the reconstructed shape to surface samples of the full groundtruth. Table 5 shows how, due to local refinement on the patch level, we outperform the baselines everywhere.

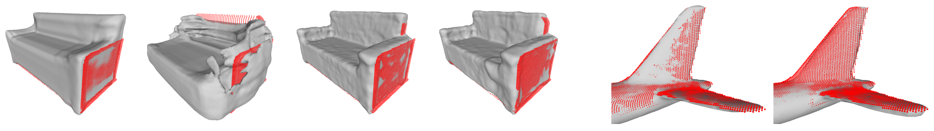
Fig. 7. Shape Completion. (Sofa) from left to right: Baseline, DeepSDF, ours unrefined, ours refined. (Airplane) from left to right: Ours unrefined, ours refined.

Table 5. Partial Point Cloud Completion from Depth Maps. We complete depth maps from a fixed camera viewpoint and from per-scene random viewpoints.

	sofas fixed		sofas random		airplanes fixed		airplanes random	
	acc.	F-score	acc.	F-score	acc.	F-score	acc.	F-score
baseline	0.094	43.0	0.092	42.7	0.069	58.1	0.066	58.7
DeepSDF-based baseline	0.106	33.6	0.101	39.5	0.066	56.9	0.065	55.5
ours	0.091	48.1	0.077	49.2	0.058	60.5	0.056	59.4
ours+refined	**0.052**	**53.6**	**0.053**	**52.4**	**0.041**	**67.7**	**0.043**	**65.8**

4.4 Articulated Deformation

Our patch-level representation can model some articulated deformations by *only* modifying the patch extrinsics, without needing to adapt the patch latent codes. Given a template surface and patch extrinsics for this template, we first encode it into patch latent codes. After manipulating the patch extrinsics, we can obtain an articulated surface with our smooth blending from Eq. 11, as Fig. 8 demonstrates.

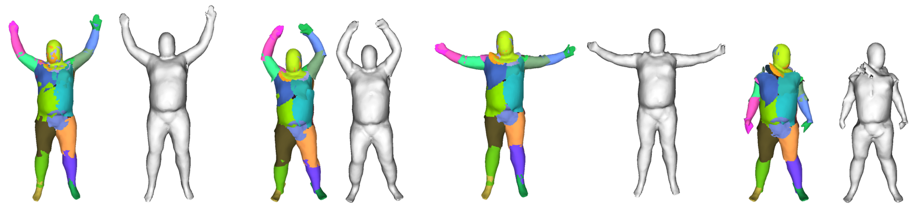

Fig. 8. Articulated Motion. We encode a template shape into patch latent codes (first pair). We then modify the patch extrinsics, while keeping the patch latent codes fixed, leading to non-rigid deformations (middle two pairs). The last pair shows a failure case due to large non-rigid deformations away from the template. Note that the colored patches move rigidly across poses while the mixture deforms non-rigidly.

5 Concluding Remarks

Limitations. We sample the SDF using DeepSDF's sampling strategy, which might limit the level of detail. Generalizability at test time requires optimizing patch latent codes and extrinsics, a problem shared with other auto-decoders. We fit the reduced test set in 71 min due to batching, one object in 10 min.
Conclusion. We have presented a mid-level geometry representation based on patches. This representation leverages the similarities of objects at patch level leading to a highly generalizable neural shape representation. For example, we show that our representation, trained on one object category can also represent other categories. We hope that our representation will enable a large variety of applications that go far beyond shape interpolation and point cloud completion.
Acknowledgements. This work was supported by the ERC Consolidator Grant 4DReply (770784), and an Oculus research grant.

References

1. Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. In: Computer Vision and Pattern Recognition (CVPR) (2020)
2. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic FAUST: Registering human bodies in motion. In: Computer Vision and Pattern Recognition (CVPR) (2017)
3. 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)
4. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Computer Vision and Pattern Recognition (CVPR) (2019)
5. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In: European Conference on Computer Vision (ECCV) (2016)
6. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH (1996)
7. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: Cvxnets: Learnable convex decomposition. In: Advances in Neural Information Processing Systems Workshops (2019)
8. Deng, B., Lewis, J., Jeruzalski, T., Pons-Moll, G., Hinton, G., Norouzi, M., Tagliasacchi, A.: Nasa: Neural articulated shape approximation (2020)
9. Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Learning elementary structures for 3D shape generation and matching. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
10. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Local deep implicit functions for 3d shape. In: Computer Vision and Pattern Recognition (CVPR) (2020)
11. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: International Conference on Computer Vision (ICCV) (2019)
12. Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: A papier-mache approach to learning 3D surface generation. In: Computer Vision and Pattern Recognition (CVPR) (2018)
13. Kato, H., Ushiku, Y., Harada, T.: Neural 3D mesh renderer. In: Computer Vision and Pattern Recognition (CVPR) (2018)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
15. Liu, S., Saito, S., Chen, W., Li, H.: Learning to infer implicit surfaces without 3D supervision. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
16. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. In: Conference on Computer Graphics and Interactive Techniques (1987)
17. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3D reconstruction in function space. In: Computer Vision and Pattern Recognition (CVPR) (2019)
18. Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Implicit surface representations as layers in neural networks. In: International Conference on Computer Vision (ICCV) (2019)

19. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4D reconstruction by learning particle dynamics. In: International Conference on Computer Vision (CVPR) (2019)
20. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.P.: Multi-level partition of unity implicits. In: ACM Transactions on Graphics (TOG) (2003)
21. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: Computer Vision and Pattern Recognition (CVPR) (2019)
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (NeurIPS) (2017)
24. Riegler, G., Osman Ulusoy, A., Geiger, A.: OctNet: Learning deep 3D representations at high resolutions. In: Computer Vision and Pattern Recognition (CVPR) (2017)
25. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: International Conference on Computer Vision (ICCV) (2019)
26. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems (NeurIPS) (2016)
27. Shimada, S., Golyanik, V., Tretschk, E., Stricker, D., Theobalt, C.: DispVoxNets: Non-rigid point set alignment with supervised learning proxies. In: International Conference on 3D Vision (3DV) (2019)
28. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3D-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
29. Stutz, D., Geiger, A.: Learning 3D shape completion under weak supervision. In: International Journal of Computer Vision (IJCV) (2018)
30. Tretschk, E., Tewari, A., Zollhöfer, M., Golyanik, V., Theobalt, C.: DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects. European Conference on Computer Vision (ECCV) (2020)
31. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: Computer Vision and Pattern Recognition (CVPR) (2017)
32. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2Mesh: Generating 3D mesh models from single RGB images. In: European Conference on Computer Vision (ECCV) (2018)
33. Williams, F., Parent-Levesque, J., Nowrouzezahrai, D., Panozzo, D., Moo Yi, K., Tagliasacchi, A.: Voronoinet: General functional approximators with local support. In: Computer Vision and Pattern Recognition Workshops (CVPRW) (2020)