Neural Part Priors: Learning to Optimize Part-Based Object Completion in RGB-D Scans

Alexey Bokhovkin and Angela Dai

Technical University of Munich

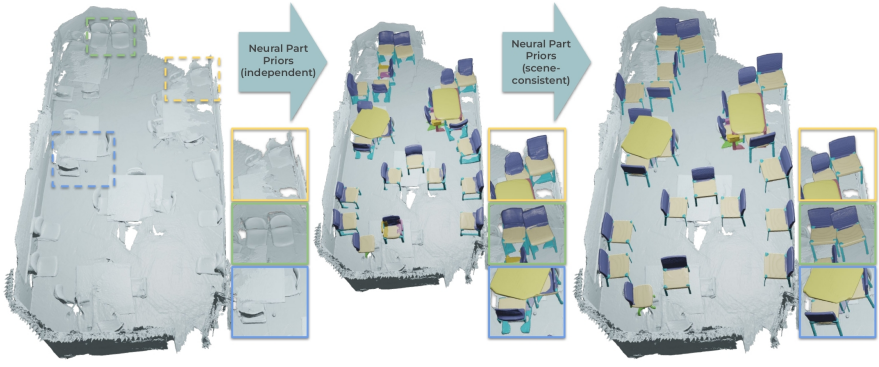


Fig. 1: Our Neural Part Priors learn latent spaces of object part geometries, which we can use to fit to partial, real-world RGB-D scans of a scene to understand its complete object part decompositions. Objects are detected as bounding boxes in the input scan, and for each detected object we estimate its semantic part labels and map them into the latent part space. We can then optimize the part representations to fit to the input scan; furthermore, rather than independently optimizing each object, we optimize for consistency with similar objects in the same scene, producing globally-consistent, complete part decompositions.

Abstract. 3D object recognition has seen significant advances in recent years, showing impressive performance on real-world 3D scan benchmarks, but lacking in object part reasoning, which is fundamental to higher-level scene understanding such as inter-object similarities or object functionality. Thus, we propose to leverage large-scale synthetic datasets of 3D shapes annotated with part information to learn Neural Part Priors (NPPs), optimizable spaces characterizing geometric part priors. Crucially, we can optimize over the learned part priors in order to fit to real-world scanned 3D scenes at test time, enabling robust part decomposition of the real objects in these scenes that also estimates the complete geometry of the object while fitting accurately to the observed real geometry. Moreover, this enables global optimization over geometrically similar detected objects in a scene, which often share strong geometric commonalities, enabling scene-consistent part decompositions. Experiments on the ScanNet dataset demonstrate that NPPs significantly outperforms state of the art in part decomposition and object completion in real-world scenes.

Project page: alexeybokhovkin.github.io/neural-part-priors/

Keywords: 3D semantic scene understanding, part segmentation, 3D reconstruction

1 Introduction

With the introduction of commodity RGB-D sensors (e.g., Microsoft Kinect, Intel RealSense, etc.), remarkable progress has been made in reconstruction and tracking to construct 3D models of real-world environments [33,47,6,35,11]. This has enabled construction of large-scale datasets of real-world 3D scanned environments [8,4], enabling significant advances in 3D semantic segmentation [10,20,7] and 3D semantic instance segmentation [23,15,21]. Such 3D object recognition is very promising, but recognition at the level of objects remains limited for many higher-level scene understanding and in particular lacks the finer-grained knowledge required for interacting with and manipulating objects (e.g., a table surface can afford support for other objects, but not the table legs).

Simultaneously, significant work has been done towards part segmentation on 3D shapes, where synthetic datasets are available with part annotations for supervision [32]. Recently, Bokhovkin et al. [3] proposed to bridge two tasks together to predict part decompositions of objects in real-world 3D scenes, leveraging part information from synthetic CAD models aligned to real-world scanned scenes [1,5,8]. This made an important step towards part-based understanding in 3D scenes, but remained limited on resolution due to a dense volumetric representation, relied heavily on synthetic priors that did not match precisely to real observations, and made independent predictions for each object without considering common object similarities in scenes.

We thus propose to learn Neural Part Priors (NPPs), and learn geometric part priors from synthetic data encoded into latent spaces that can be optimized over at inference to fit precisely to objects in real-world scanned scenes. Our NPPs leverage the representation power of neural implicit functions to learn spaces for representing the geometry of the parts of each class category. A shape can then be represented by a set of latent codes for each of its parts, where each code decodes to predict the respective part segmentation and signed distance field representation of the part geometry. Importantly, this enables joint optimization over all parts of a shape by traversing through the part latent space to find the set of parts that best explain a shape observation. As repeated objects often appear in a scene under different partial observation patterns, resulting in inconsistent predictions when made independently for each object, we further optimize for part consistency between similar objects detected in a scene to produce scene-consistent part decompositions.

To fit to real-world 3D scan data, we first perform object detection and estimate the part types for each detected object. We can then optimize jointly over the part codes for each shape to fit to the observed scan geometry; we leverage a predicted part segmentation of the detected object, and optimize jointly

across the parts of each shape such that each part matches the segmentation, and their union fits to the object. This joint optimization across parts produces a high-resolution part decomposition whose union represents the complete shape while fitting precisely to the observed real geometry. Furthermore, this optimization at inference time allows leveraging global scene information to inform our optimized part decompositions; in particular, we consider objects of the same predicted class with similar estimated geometry, and optimize them jointly, enabling more robust and scene-consistent part decompositions.

In summary, we present the following contributions:

- We propose to learn optimizable part priors for 3D shapes, encoding part segmentation and part geometry into a latent space for the part types of each class category.
- Our learned, optimizable part priors enable test-time optimization over the latent spaces to fit to partial, cluttered object geometry in real-world scanned scenes, resulting in robust and precise semantic part completion.
- We additionally propose a scene-consistent optimization, jointly optimizing over similar objects to enabling globally-consistent part decompositions for repeated object instances in a scene.

2 Related Works

3D Object Detection and Instance Segmentation. 3D semantic scene understanding has seen rapid progress in recent years, with large-scale 3D datasets [8,4,16] and developments in 3D deep learning showing significant advances in object-level understanding of 3D scenes. Various methods explore learning on different 3D representations for 3D object detection and 3D instance segmentation, including volumetric grids [23,44], point clouds [39,50,27,38,34], sparse voxel representations [15,21], and multi-view hybrid approaches [23,38]. These approaches have achieved impressive performance in detecting and segmenting objects in real-world observations of 3D scenes, but do not consider lower-level object part information that is requisite for many vision and robotics tasks, particularly those involving object interaction and manipulation.

Recently, Bokhovkin et al. [3] proposed an approach to estimate part decompositions of objects in RGB-D scans, leveraging structural semantic part type prediction in combination with a pre-computed set of geometric part priors. Due to the use of dense volumetric part priors, the part reasoning is limited to coarse resolutions, and often does not precisely match the input observed geometry. We also address the task of semantic part prediction and completion for objects in real-world 3D scans, but leverage a learned, structured latent space representing neural part priors, enabling part reasoning at high resolutions which optimizing to fit accurately to the observed scan geometry.

3D Scan Completion. As real-world 3D reconstructions are very often incomplete due to the complexity of the scene geometry and occlusions, various

approaches have been developed to predict complete shape or scene geometry from partial observations. Earlier works focused on voxel-based scan completion for shapes [49,13], with more recent works tacking the challenge of generating complete geometry from partial observations of large-scale scenes [43,12,9,14], but without considering individual object instances. Several recent works propose to detect objects in an RGB-D scan and estimate the complete object geometries, leveraging voxel [24,3] or point [53,34] representations. Our approach to predicting part decompositions of objects inherently provides object completion as the union of the predicted parts; in contrast to previous approaches that estimate object completion in RGB-D scans, we propose to characterize object parts as learned implicit priors, enabling test-time traversal of the latent space to fit accurately to observed input scan geometry.

Part Segmentation of 3D Shapes. Part segmentation for 3D shapes has been well-studied in shape analysis, typically focusing on understanding collections of synthetic shapes. Various methods have been developed for unsupervised part segmentation by finding a consistent segmentation across a set of shapes [19,26,42,25,30]. Recent deep learning based approaches have leveraged datasets of shapes with part annotations to learn part segmentation on new shapes [28,52,22]. In particular, approaches that learn part sequences and hierarchies to capture part structures have shown effective part segmentation for shapes [46,45,51,32,31,48]. These approaches target single-object scenarios, whereas we construct a set of learned part priors that can be optimized to fit to real-world, noisy, incomplete scan geometry.

Neural Implicit Representations of 3D Shapes. Recently, we have seen significant advances in generative shape modeling with learned neural implicit representations that can represent continuous implicit surface representations, without ties to an explicit grid structure. Notably, DeepSDF [36] proposed an MLP-based network that predicts the SDF value for a given 3D location in space, conditioned on a latent shape code, which demonstrated effective modeling of 3D shapes while traversing the learned shape space. Such implicit representations have also been leveraged in hybrid approaches coupling explicit geometric locations with local implicit descriptions of geometry for shapes [18,17] as well as scenes [37], without semantic meaning to the local decompositions. We propose to leverage the representation power of such learned continuous implicit surfaces to characterize semantic object parts that can be jointly optimized together to fit all parts of an object to a partial scan observation.

3 Method

3.1 Overview

We introduce Neural Part Priors (NPPs) to represent learned spaces of geometric object part priors, that enable joint part segmentation and completion of objects in real-world, incomplete RGB-D scans. From an input 3D scan $\mathcal{S}$, we first

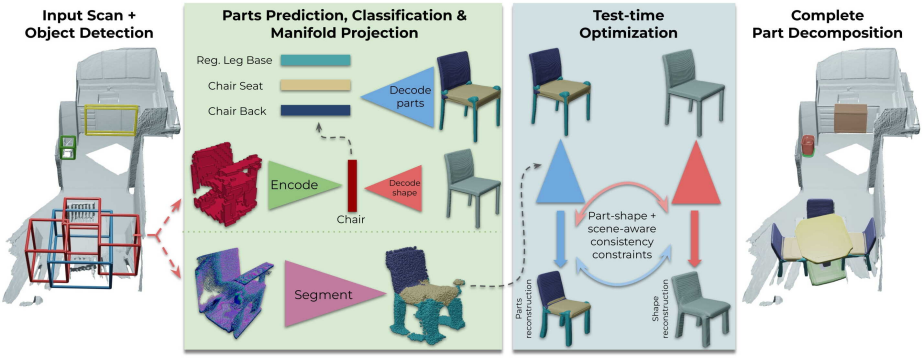


Fig. 2: Method overview. From an input scan, we first detect 3D bounding boxes for objects. For each object, we predict their semantic part structure as a set of part labels and latent codes for each part. These latent codes map into the space of neural part priors, along with a full shape code used to regularize the shape structure. We then refine these codes at test time by optimizing to fit to the observed input geometry along with inter-object consistency between similar detected objects, producing effective part decompositions reflecting complete objects with scene consistency.

detect objects $\mathcal{O} = \{o_i\}$ in the scan characterized by their bounding boxes and orientations, then for each object we predict its part decomposition into a part class categories and their corresponding complete geometry. This enables holistic reasoning about each object in the scene, and prediction of complete geometry in unobserved regions in the scan. Since captured real-world scene geometry tends to contain significant incompleteness or noise, we propose to model our geometric part priors based on complete, clean synthetic object part data, represented as a learned latent space over implicit part geometry functions. This enables optimization at test time over the latent space of parts to fit to real geometry observations, enabling part-based object completion while precisely representing real object geometry. Furthermore, rather than only considering each object independently, we observe that repeated objects often occur in scenes under different partial observations, leading to inconsistent independent predictions; we thus jointly optimize across similar objects in a scene to produce scene-consistent part decompositions. An overview of our approach is shown in Fig. 2.

Our NPPs spaces characterize object part geometry as signed distance fields (SDFs), trained on part annotations for shapes. For each detected object $o_i \in \mathcal{O}$ in an input scan, we predict its semantic parts as the set of part categories that compose the object along with initial estimates of their corresponding latent codes in the learned part space. We then optimize for refined object pose alignment, and then for the part latent codes to fit to the observed geometry of each o_i. For objects of the same class category and with similar predicted shape geometry by chamfer distance, we jointly constrain them together in this

optimization. This results in scene-consistent, high-fidelity characterizations of both object part semantics and complete object geometry.

3.2 Object Detection

From input 3D scan $\mathcal{S}$, we first detect objects in the scene, leveraging a state-of-the-art 3D object detection backbone from MLCVNet [50]. MLCVNet interprets $\mathcal{S}$ as a point cloud and proposes objects through voting [39] at multiple resolutions, providing an output set of axis-aligned bounding boxes for each detected object o_i. We extract the truncated signed distance field D_i for each o_i at 4mm resolution to use for test-time optimization. We then aim to characterize shape properties for o_i to be used for rotation estimation and test-time optimization, and interpret D_i as a 32^3 occupancy grid which is input to a 3D convolutional object encoder to produce the object's shape descriptor $\mathbf{s}_i \in \mathbb{R}^{256}$.

Initial Rotation Estimation. From $\mathbf{s}_i$, we use a 2-layer MLP to additionally predict an initial rotation estimate of the object as r_i^{init} around the up (gravity) vector of $\mathcal{S}$. We note that the up vectors of an RGB-D scan can be reliably estimated with IMU and/or floor estimation techniques [8]. The rotation estimation is treated as a classification problem across $n_r = 12$ bins of discretized angles ($\{0°, 30°, \ldots, 330°\}$), using a cross entropy loss. We use the estimated rotation r_i^{init} to resample D_i to approximate the canonical object orientation, from which we use to optimize for the final rotation r_i and the object part latent codes.

3.3 Learned Space of Neural Part Priors

We first learn a set of latent part spaces for each class category, where each part space represents all part types for the particular object category. To this end, we employ a function f_p characterized as an MLP to predict the implicit signed distance representation for each part geometry of the class category. In addition to the latent part space, we additionally train a proxy shape function f_s as an MLP that learns full shape geometry as implicit signed distances, which will serve as additional regularization during the part optimization. Both f_p and f_s are trained in auto-decoder fashion following DeepSDF [36]. Then each train shape part is embedded into a part latent space by optimizing for its code $\mathbf{z}_k^{\text{p}} \in \mathbb{R}^{256}$ such that f_p conditioned on this code and the part type maps a point $\mathbf{x} \in \mathbb{R}^3$ in the canonical space to SDF value d of the part geometry:

$$f_p : \mathbb{R}^3 \times \mathbb{R}^{256} \times \mathbb{Z}_2^{N_c} \rightarrow \mathbb{R}, \qquad f_p(\mathbf{x}, \mathbf{z}_k^{\text{p}}, \mathbb{1}_{\text{part}}) = d. \tag{1}$$

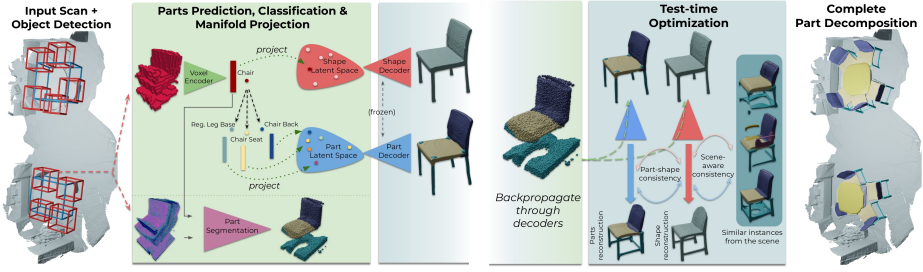
where $\mathbb{1}_{\text{part}} \in \mathbb{Z}_2^{N_c}$ is a one-hot encoding of the part type for a maximum of N_c parts. The shape space is trained analogously for each class category where $\mathbf{z}_i^{\text{s}} \in \mathbb{R}^{256}$ represents a shape latent code in the space:

$$f_s : \mathbb{R}^3 \times \mathbb{R}^{256} \rightarrow \mathbb{R}, \qquad f_p(\mathbf{x}, \mathbf{z}_i^{\text{s}}) = d. \tag{2}$$

We train these latent spaces of part and shape priors on the synthetic PartNet [32] dataset to characterize a space of complete parts and shapes. To learn the latent spaces, we minimize the reconstruction error over all train shape parts, while optimizing for latent codes $\{\mathbf{z}_k^{\mathrm{p}}\}$ and weights of f_p. We use an ℓ_1 reconstruction loss with ℓ_2 regularization on the latent codes:

$$L = \sum_{j=1}^{N_p} |f_p(\mathbf{x}_j, \mathbf{z}_k^{\mathrm{p}}, \mathbb{1}_{\mathrm{part}}) - D^{\mathrm{gt}}(\mathbf{x}_j)|_1 + ||\mathbf{z}_k^{\mathrm{p}}||_2^2 \tag{3}$$

for N_p points near the surface. We train f_s analogously.



(a) Projection into part and shape spaces. (b) Joint optimization to fit to input scan.

Fig. 3: (a) Projection into the part and shape latent spaces along with part segmentation from input scan geometry. (b) Optimization at test time to fit to observed scan geometry while maintaining inter-part consistency within a shape and inter-shape consistency for geometrically similar objects.

3.4 Optimizing for Part Decompositions in Real Scenes

Once we have learned our latent space of parts, we can traverse them at inference time to find the part-based decomposition of an object that best fits to its real-world observed geometry in a scene. In particular, since real-world observations are typically incomplete, we can optimize for complete part decompositions based on strong priors given by the trained latent spaces. This allows for effective regularization by synthetic part characteristics (clean, complete) while fitting precisely to real observed geometry.

To guide this optimization for a detected object box o characterized by its shape feature $\mathbf{s}$, we first predict its high-level decomposition into a set of semantic part types $\{(c_k, \mathbf{p}_k)\}$, where $\mathbf{p}_k \in \mathbb{R}^{256}$ is a part feature descriptor and c_k the part class label. Note that we discard the i object suffix here for simplicity. We then use this semantic part information to initialize the part optimization.

To obtain the semantic part type predictions, we employ a message-passing graph neural network that implicitly infers part relations to predict the set of component part types. Similar to [31], from the shape feature $\mathbf{s}$ we use an MLP to predict at most $N_c = 10$ parts. For each potential part k, we predict its

probability of existence, its part label c_k, and its corresponding feature vector $\mathbf{p}_k$, with additional proxy losses on part adjacency probabilities between each potential pair of parts to learn structural part information. This produces the semantic description of the set of parts for the object $\{(c_k, \mathbf{p}_k)\}$, from the parts predicted with part existence probability > 0.5.

Projection to the Latent Part Space. We then learn a projection mapping from the part features $\{\mathbf{p}_k\}$ to the learned latent part space based on synthetic part priors, using a small MLP to predict $\{\tilde{\mathbf{z}}_k^{\mathrm{p}}\}$, as shown in Fig. 3(a). This helps to provide a close initial estimate in the latent part space in order to initialize optimization over these part codes to fit precisely to the observed object geometry. We additionally project the shape code $\mathbf{s}$ analogously to the learned latent shape space with a small shape projection MLP to predict $\{\tilde{\mathbf{z}}^{\mathrm{s}}\}$, which we use to help regularize the part code optimization to remain globally consistent over the shape. Both of these projection MLPs are trained using MSE losses against the optimized train codes of the latent spaces.

Part Segmentation Estimation. In addition to our projection initialization, we estimate part segmentation $\{D^{\mathrm{p}}\}_{\mathrm{p}=1}^{N_{parts}}$ for the input object TSDF D over the full volume, representing part SDF geometry in the regions predicted as corresponding to the part p, where part segmentation regions cover the entire shape, including unobserved regions. This is used to guide part geometry predictions when optimizing at test time to fit to real observed input geometry. For each point $\mathbf{x} \in \mathbb{R}^3$ which has distance $< d_{trunc} = 0.16$m from the input object TSDF D, we classify it to one of the predicted parts $\{(c_k, \mathbf{p}_k)\}$ or background using a small PointNet-based[40] network. This segmentation prediction takes as input the corresponding shape feature $\mathbf{s}$, the initial estimated rotation r_i^{init}, and the 3D coordinates of $\mathbf{x}$, and is trained with a cross-entropy loss.

3.5 Joint Part Optimization

To obtain the final part decompositions, we traverse over the learned latent part space to fit to the observed input scan geometry, as shown in Fig. 3(b) From the initial estimated part codes $\{\tilde{\mathbf{z}}_k^{\mathrm{p}}\}$ and shape code $\{\tilde{\mathbf{z}}^{\mathrm{s}}\}$, their decoded part SDFs should match to each of $\{D^{\mathrm{p}}\}_{\mathrm{p}=1}^{N_{parts}}$. Since the part and shape latent spaces have been trained in the canonical shape space, we optimize for a refined rotation prediction r from r^{init} using iterative closest points [2,41] between the sampled points near D and the initial shape estimate from projection $\tilde{\mathbf{z}}^{\mathrm{s}}$. We use N_i sampled points near the observed input surface D (near being SDF values < 0.025m) for rotation refinement, with N the number of points not predicted as background during part segmentation.

While the predicted projected part and shape codes $\{\tilde{\mathbf{z}}_k^{\mathrm{p}}\}, \{\tilde{\mathbf{z}}_k^{\mathrm{s}}\}$ can provide a good initial estimate of the part decomposition of the complete shape, they represent synthetic part and shape priors that often do not fit the observed real input geometry. We thus optimize for part decompositions that best fit the input

observations by minimizing the energy:

$$L = \sum_k L_{\text{part}} + L_{\text{shape}} + w_{\text{cons}} L_{\text{cons}}, \tag{4}$$

where L_{part} denotes the part reconstruction loss, L_{shape} a proxy shape reconstruction loss, L_{cons} a regularization to encourage global part consistency within the estimated shape, and w_{cons} is a consistency weight.

L_{part} is an ℓ_1 loss on part reconstruction:

$$L_{\text{part}} = \sum_{\text{p}=1}^{N_{parts}} \sum_{N^{\text{p}}} w_{trunc} |f_p(\mathbf{z}_k^{\text{p}}) - T_r(D^{\text{p}})| + ||\mathbf{z}_k^{\text{p}}||_2^2, \tag{5}$$

where N^{p} is the number of points classified to part p and w_{trunc} gives a fixed greater weight for near-surface points ($< d_{trunc} = 0.16$m).

L_{shape} is a proxy ℓ_1 loss on shape reconstruction:

$$L_{\text{shape}} = \sum_N w_{trunc} |f_s(\mathbf{z}^{\text{s}}) - T_r(D)| + ||\mathbf{z}^{\text{s}}||_2^2. \tag{6}$$

Finally, L_{cons} encourages all parts to reconstruct a shape similar to the optimized shape:

$$L_{\text{cons}} = \sum_N |f_p(\mathbf{z}_k^{\text{p}}) - f_s(\mathbf{z}^{\text{s}})|, \tag{7}$$

where $f_s(\mathbf{z}^{\text{s}})$ is frozen for L_{cons}. This allows for reconstructed parts to join together smoothly without boundary artifacts to holistically reconstruct a shape.

This produces a final optimized set of parts for each object in the scene, where parts both fit precisely to observed input geometry and represent the complete geometry of each part, even in unobserved regions. The final part geometries can be extracted from the SDFs with Marching Cubes [29] to obtain a surface mesh representation of the semantic part completion.

Scene-Consistent Optimization. While this formulation enables joint part optimization within an object, we observe that multiple objects within a scene often tend to correlate with each other. In particular, scenes often contain repeated instances of objects which are then observed from different views, thus frequently resulting in inconsistent optimized part decompositions when considered as independent objects. Thus, we propose a scene-consistent optimization between similar predicted objects, where objects in a scene are considered similar if their predicted class category is the same and the chamfer distance between their decoded shapes from $\tilde{\mathbf{z}}_k^{\text{s}}$ is $< \tau_s$.

For a set of N_{sim} similar objects in a scene, we collect together their predicted part segmentations and observed input SDF geometry in the canonical orientation based on $T_r(D^{\text{p}})$ to provide a holistic set of constraints across different partial observations to produce $\{D_i\}_{i=1}^{N_{sim}}$. The $\{D_i\}_{i=1}^{N_{sim}}$ are then aggregated to form $\mathbf{D}'$ by sampling a set of N_{avg} points near the surfaces of $\{D_i\}_{i=1}^{N_{sim}}$ where

	Chamfer Distance – Accuracy (↓)								Chamfer Distance – Completion (↓)							
Method	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg
SG-NN[9] + MLCVNet[50] + PointGroup[27]	0.047	0.110	0.146	0.173	0.350	0.051	0.146	0.083	0.054	0.141	0.123	0.192	0.382	0.045	0.156	0.089
MLCVNet[50] + StructureNet[31]	0.024	0.074	0.104	0.166	0.424	0.039	0.138	0.061	0.028	0.129	0.118	0.154	0.352	0.037	0.136	0.067
Bokhovkin et al.[3]	0.029	0.073	0.099	0.168	0.244	0.036	0.108	0.056	0.031	0.095	**0.108**	**0.151**	0.236	0.038	0.110	0.059
Ours	**0.014**	**0.054**	**0.096**	**0.141**	**0.199**	**0.031**	**0.089**	**0.041**	**0.018**	**0.076**	0.110	0.155	**0.195**	**0.029**	**0.097**	**0.046**

Table 1: Evaluation of semantic part completion on Scan2CAD [1] in comparison to state-of-the-art part segmentation [27,31] and semantic part completion [3]. Our optimizable part priors produce more accurate part decompositions.

N_{avg} is the average number of points across the N_{sim} objects, and each point is assigned the minimum SDF value within its 30-point local neighborhood (to help ensure that objects do not grow thicker in size from small-scale misalignments). Based on $\mathbf{D}'$, we then optimize for the part decompositions following Eq. 4.

3.6 Implementation Details

We first train our latent part and shape spaces on per-category on the synthetic PartNet [32] dataset. We then train the projection mapping into the learned part and shape spaces as well as the part segmentation. This is first pre-trained on synthetic PartNet data using virtually scanned incomplete inputs to take advantage of the large amount of synthetic data. To apply to real-world observations, we then fine-tune the projections and part segmentation on ScanNet [8] data using MLCVNet [50] detections on train scenes.

For test-time optimization, we optimize for part and shape codes using an Adam optimizer with learning rate of 3e-4 for 600 iterations. The learning rate is decreased by factor of 10 after 300 iterations. To enable more flexibility to capture input details, we enable optimization of the decoder weights for parts and shape after 400 iterations. Optimization for each part takes $\approx$ 90 seconds. For further implementation and training details, we refer to the supplemental.

4 Results

We evaluate our Neural Part Priors for semantic part completion on real-world RGB-D scans from ScanNet [8]. We use the official train/val/test split of 1045/156/312 scans. In order to evaluate part segmentation in these real-world scenes, we

	Chamfer Distance (↓)								IoU (↑)							
Method	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg
SG-NN[9] + MLCVNet[50] + PointGroup[27]	0.056	0.121	0.110	0.161	0.406	0.034	0.148	0.088	0.257	0.390	**0.300**	0.229	0.306	**0.488**	0.328	0.293
MLCVNet[50] + StructureNet[31]	0.025	0.101	**0.092**	**0.090**	0.359	0.041	0.118	0.059	0.480	0.356	0.195	0.331	0.267	0.342	0.329	0.413
Bokhovkin et al.[3]	0.027	0.059	0.095	0.102	**0.207**	0.031	0.087	0.049	**0.548**	0.542	0.224	0.328	0.442	0.375	0.409	**0.490**
Ours	**0.017**	**0.045**	0.098	**0.090**	0.219	**0.028**	**0.083**	**0.041**	0.503	**0.611**	0.213	**0.388**	**0.445**	0.335	**0.416**	0.473

Table 2: Evaluation of part segmentation on Scan2CAD [1]. We evaluate part segmentation only on observed scan geometry, in comparison with state-of-the-art part segmentation [27,31] and semantic part completion [3].

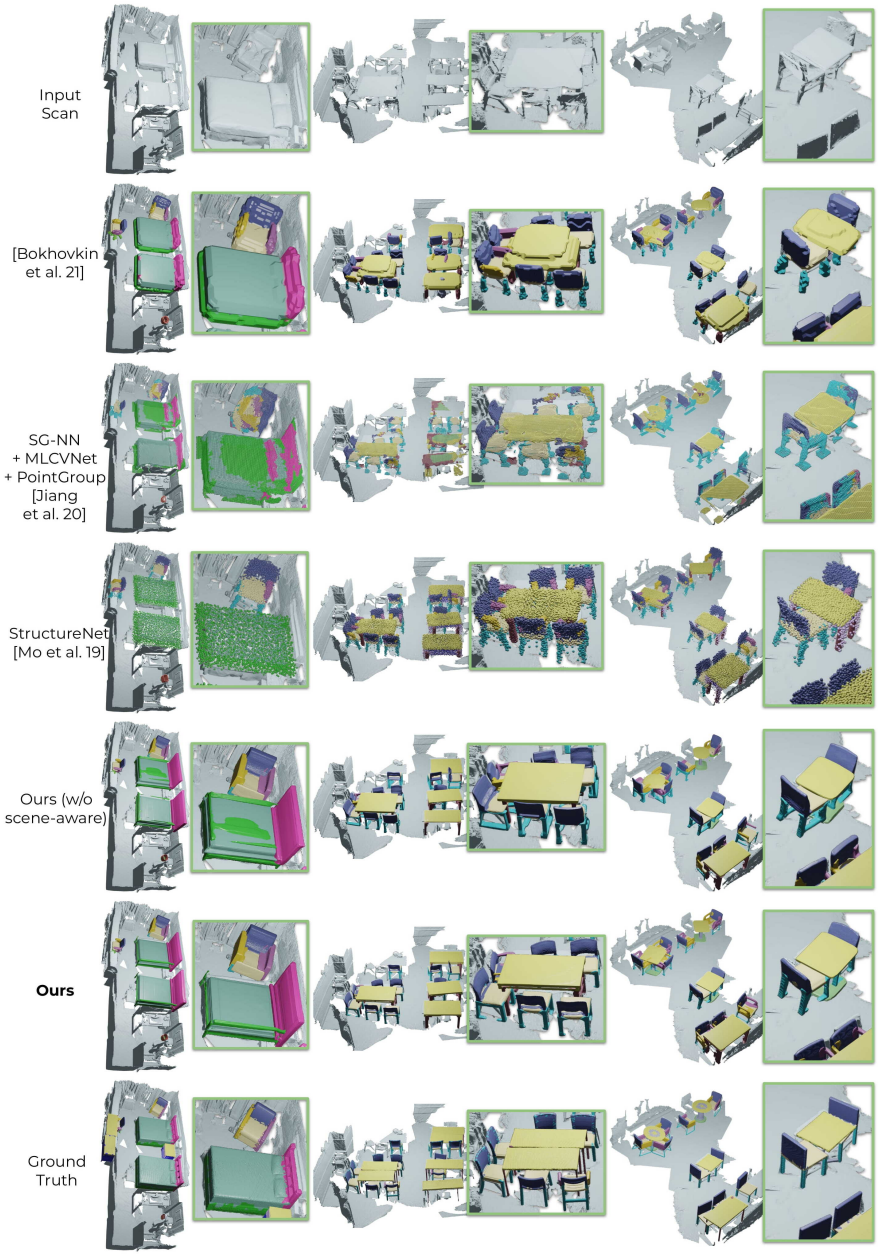


Fig. 4: Qualitative comparison of NPPs with point [27,31] and voxel-based [3] state of the art on ScanNet scans with Scan2CAD+PartNet ground truth. Our joint optimization across part priors enables more consistent, accurate part decompositions.

use the Scan2CAD [1] annotations of CAD model alignments from ShapeNet [5] to these scenes, and the PartNet [32] part annotations for the ShapeNet objects. To construct our part latent space, we train on PartNet and train a projection from train ScanNet objects to their PartNet annotations; we also train all baselines on the same ScanNet+PartNet data. We consider 6 major object class categories representing the majority of parts, comprising a total of 28 part types.

All methods are provided the same MLCVNet bounding boxes, which contain at least 40% of the closest annotated shape from Scan2CAD[1] dataset.

Evaluation Metrics. Since PartNet shapes aligned to real-world ScanNet geometry do not have precise geometric alignments, due to inexact synthetic-real associations, our evaluation metrics aim to characterize the quality of part decompositions that fit well to the real-world geometry, as well as accurately represent complete object geometry. We evaluate *accuracy* of the part segmentation with respect to the ScanNet object, and object *completion* with respect to the complete object geometry from PartNet.

Accuracy measures part segmentation predictions with respect to PartNet labels projected onto ScanNet object geometry, as a single-sided chamfer distance from the partial ScanNet object to the predicted part decomposition (as we lack complete real-world geometry available for evaluation in the other direction).

Completion evaluates a bi-directional Chamfer distance between the predicted part decomposition against the PartNet labeled shape.

For evaluation, we sample $10,000$ points per part from predicted and ground-truth mesh surfaces, which are then transformed to the coordinate space of ScanNet. We consider all evaluation metrics across each part category corresponding to an object class, and evaluate class average over the categories, and instance average over part instances. Each shape instance is evaluated over a union of predicted and ground-truth parts, and then summed to represent evaluation for this shape. If any predicted or ground-truth part does not correspond to the other, we use the center of the respective shape for evaluation.

To evaluate part segmentation of only the observed input geometry without considering completion, we use Chamfer distance and IoU. As ground truth, we project part labels from aligned PartNet shapes onto the ScanNet mesh surface. Predicted mesh part labels are projected to ScanNet to obtain predicted part segmentation. We similarly use $10,000$ sampled points per part for these metrics.

Comparison to state of the art. Tab. 1 shows a comparison to state of the art on semantic part completion for real-world ScanNet scans, with qualitative results shown in Fig. 4. We compare with Bokhovkin et al. [3], which leverages coarse 32^3 pre-computed geometric priors for semantic part completion as well as state-of-the-art part segmentation approach StructureNet [31]. Since part segmentation can also be cast as an instance segmentation approach, we also compare with PointGroup [27]. Each of these methods use the same object detection results from MLCVNet [50]; since PointGroup does not predict any geometric completion, we provide additional scan completion from SG-NN [9]. We note that since Bokhovkin et al. [3] predicts solid part geometry as 32^3

voxels, we use Marching Cubes to extract a surface with which to evaluate. In contrast to these approaches which estimate part decompositions directly and independently, our NPPs enable joint optimization over all parts to fit precisely to the observed scan geometry, resulting in improved accuracy and completion performance. Several additional qualitative results are shown in Fig. 5.

Part segmentation on 3D scans. We also evaluate part segmentation in Tab. 2, which considers segmentation of only the observed scan geometry, without any geometric completion. We intersect each method's part predictions with the original scan geometry to evaluate this part segmentation, in comparison with Bokhovkin et al. [3], StructureNet [31], and PointGroup [27]. By jointly optimizing over the parts of an object to fit to its observed scan geometry, our approach improves notably in part segmentation performance.

What is the effect of scene-consistent optimization? Tab. 3 and Fig. 4 show the effect of scene-consistent optimization. This improves results over *w/o Scene Consistency*, with somewhat stronger effect on categories that more often have repeated instances (e.g., chair, table, bed in offices, classrooms, hotel rooms). We see a much more noticeable effect in qualitative effect, with much more consistent part decompositions for similar objects in a scene, even when seen under fairly different partial views. This more holistic optimization enables more consistent reasoning about objects and their parts in a scene.

What is the impact of test-time optimization and projection initialization? We consider our approach without using a learned projection mapping to the latent part space as initialization for test-time optimization to fit to the observed scan geometry, and also evaluate only the projection mapping without any test-time optimization, shown in Tab. 3. The initial projection helps significantly to obtain a good initialization for test-time optimization (*w/o Projection Map*), while the projection results provide a good estimate but imprecise fit to the observed scene geometry (*w/o Test-Time Opt*). By leveraging both, we can achieve the best representation of the input scan as its part decomposition.

Effect of synthetic pre-training. We also evaluate the effect of synthetic pre-training of the part segmentation and projection mappings to the part and shape latent spaces in Tab. 3 (*w/o Synthetic Pretrain*). Here, the additional quantity and diversity of data helps to avoid overfitting to more limited real data.

	Chamfer Distance (↓) – Accuracy								Chamfer Distance (↓) – Completion							
Method	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg	chair	table	cab.	bkshlf	bed	bin	class avg	inst avg
w/o Projection Map.	0.032	0.156	0.140	0.249	0.352	0.029	0.160	0.081	0.026	0.144	0.137	0.192	0.309	0.025	0.139	0.070
w/o Synthetic Pretrain	0.026	0.079	0.116	0.158	**0.152**	0.035	0.094	0.052	0.029	0.097	0.132	0.206	0.234	0.033	0.122	0.061
w/o Test-Time Opt.	0.019	0.060	0.105	0.145	0.200	**0.023**	0.093	0.047	0.022	0.085	0.127	0.183	0.198	**0.024**	0.107	0.052
w/o Scene Consistency	0.016	0.056	**0.096**	**0.138**	0.207	0.032	0.091	0.043	0.019	0.079	0.111	**0.149**	0.206	0.030	0.099	0.047
Ours	**0.014**	**0.054**	**0.096**	0.141	0.199	0.031	**0.089**	**0.041**	**0.018**	**0.076**	**0.110**	0.155	**0.195**	0.029	**0.097**	**0.046**

Table 3: Ablation study evaluating semantic part completion on Scan2CAD [1]. We show the effect of our projection mapping initialization, and test-time optimization to fit to the for our design decisions. Overall, using both results in the best performance in RGB-D scan part decompositions.

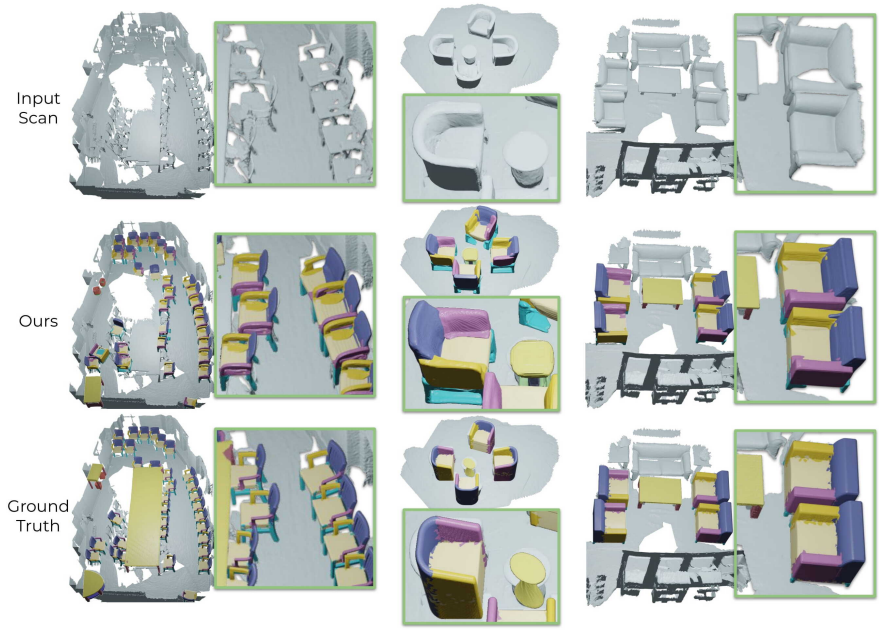


Fig. 5: Additional qualitative results on ScanNet[8] with Scan2CAD[1] and PartNet[32] targets, showing our consistent, complete part decompositions.

Limitations. While our NPPs shows strong promise towards accurate, high resolution characterization of semantic parts in real-world scenes required for finer-grained semantic scene understanding, various limitations remain. Our part latent space is trained in its canonically oriented space, and while we can optimize for shape orientations with ICP, a joint optimization or an equivariant formulation can potentially resolve sensitivity to misaligned orientations. Finally, our scene-consistent optimization for similar shapes in a scene makes an important step towards holistic scene reasoning, but does not consider higher-level, stylistic similarity that is often shared across different objects in the same scene (e.g., matching furniture set for desk, shelves, chair) which could provide notable insight towards comprehensive scene understanding.

5 Conclusion

We have presented Neural Part Priors, which introduces learned, optimizable part priors for fitting complete part decompositions to objects detected in real-world RGB-D scans. We learn a latent part space over all object parts, characterized with learned neural implicit functions. This allows for traversing over the part space at test time to jointly optimize across all parts of an object such that it fits to the observed scan geometry while maintaining consistency with any similar detected objects in the scan. This results in improved part segmentation

as well as completion in noisy, incomplete real-world RGB-D scans. We hope that this help to open up further avenues towards holistic part-based reasoning in real-world environments.

Acknowledgements

This project is funded by the Bavarian State Ministry of Science and the Arts and coordinated by the Bavarian Research Institute for Digital Transformation (bidt).

References

1. Avetisyan, A., Dahnert, M., Dai, A., Savva, M., Chang, A.X., Nießner, M.: Scan2cad: Learning CAD model alignment in RGB-D scans. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 2614–2623 (2019) 2, 10, 12, 13, 14
2. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. Spie (1992) 8
3. Bokhovkin, A., Ishimtsev, V., Bogomolov, E., Zorin, D., Artemov, A., Burnaev, E., Dai, A.: Towards part-based understanding of rgb-d scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7484–7494 (2021) 2, 3, 4, 10, 11, 12, 13, 19, 20, 21
4. Chang, A.X., Dai, A., Funkhouser, T.A., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from RGB-D data in indoor environments. In: 2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017. pp. 667–676 (2017) 2, 3
5. 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) 2, 12
6. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5556–5565 (2015) 2
7. Choy, C.B., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 3075–3084 (2019) 2
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Niessner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 2, 3, 6, 10, 14, 23
9. Dai, A., Diller, C., Nießner, M.: Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 849–858 (2020) 4, 10, 12
10. Dai, A., Nießner, M.: 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 452–468 (2018) 2

11. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG) **36**(4), 1 (2017) 2
12. Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., Nießner, M.: Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2018) 4
13. Dai, A., Ruizhongtai Qi, C., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5868–5877 (2017) 4
14. Dai, A., Siddiqui, Y., Thies, J., Valentin, J., Nießner, M.: Spsg: Self-supervised photometric scene generation from rgb-d scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1747–1756 (2021) 4
15. Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., Nießner, M.: 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9031–9040 (2020) 2, 3
16. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012) 3
17. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Local deep implicit functions for 3d shape. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4857–4866 (2020) 4
18. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7154–7164 (2019) 4
19. Golovinskiy, A., Funkhouser, T.: Consistent segmentation of 3d models. Computers & Graphics **33**(3), 262–269 (2009) 4
20. Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 9224–9232 (2018) 2
21. Han, L., Zheng, T., Xu, L., Fang, L.: Occuseg: Occupancy-aware 3d instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2940–2949 (2020) 2, 3
22. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019) 4
23. Hou, J., Dai, A., Nießner, M.: 3d-sis: 3d semantic instance segmentation of rgb-d scans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4421–4430 (2019) 2, 3
24. Hou, J., Dai, A., Nießner, M.: Revealnet: Seeing behind objects in rgb-d scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2098–2107 (2020) 4
25. Hu, R., Fan, L., Liu, L.: Co-segmentation of 3d shapes via subspace clustering. In: Computer graphics forum. vol. 31, pp. 1703–1713. Wiley Online Library (2012) 4
26. Huang, Q., Koltun, V., Guibas, L.: Joint shape segmentation with linear programming. In: Proceedings of the 2011 SIGGRAPH Asia Conference. pp. 1–12 (2011) 4

27. Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.W., Jia, J.: Pointgroup: Dual-set point grouping for 3d instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4867–4876 (2020) 3, 10, 11, 12, 13, 19, 20, 21
28. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3d shape segmentation with projective convolutional networks. In: proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3779–3788 (2017) 4
29. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987) 9
30. Luo, T., Mo, K., Huang, Z., Xu, J., Hu, S., Wang, L., Su, H.: Learning to group: A bottom-up framework for 3d part discovery in unseen categories. arXiv preprint arXiv:2002.06478 (2020) 4
31. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., Guibas, L.J.: Structurenet: Hierarchical graph networks for 3d shape generation. arXiv preprint arXiv:1908.00575 (2019) 4, 7, 10, 11, 12, 13, 19, 20, 21
32. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 909–918 (2019) 2, 4, 7, 10, 12, 14, 19
33. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE international symposium on mixed and augmented reality. pp. 127–136. IEEE (2011) 2
34. Nie, Y., Hou, J., Han, X., Nießner, M.: Rfd-net: Point scene understanding by semantic instance reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4608–4618 (2021) 3, 4
35. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG) **32**(6), 1–11 (2013) 2
36. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019) 4, 6
37. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 523–540. Springer (2020) 4
38. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3d object detection in point clouds with image votes. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4404–4413 (2020) 3
39. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9277–9286 (2019) 3, 6
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 77–85 (2017) 8, 23, 24
41. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: Proceedings third international conference on 3-D digital imaging and modeling. pp. 145–152. IEEE (2001) 8

42. Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., Cohen-Or, D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In: Proceedings of the 2011 SIGGRAPH Asia Conference. pp. 1–10 (2011) 4
43. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1746–1754 (2017) 4
44. Tung, H.Y.F., Cheng, R., Fragkiadaki, K.: Learning spatial common sense with geometry-aware recurrent networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2595–2603 (2019) 3
45. Van Kaick, O., Xu, K., Zhang, H., Wang, Y., Sun, S., Shamir, A., Cohen-Or, D.: Co-hierarchical analysis of shape structures. ACM Transactions on Graphics (TOG) **32**(4), 1–10 (2013) 4
46. Wang, Y., Xu, K., Li, J., Zhang, H., Shamir, A., Liu, L., Cheng, Z., Xiong, Y.: Symmetry hierarchy of man-made objects. In: Computer graphics forum. vol. 30, pp. 287–296. Wiley Online Library (2011) 4
47. Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., Davison, A.: Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems (2015) 2
48. Wu, R., Zhuang, Y., Xu, K., Zhang, H., Chen, B.: Pq-net: A generative part seq2seq network for 3d shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 829–838 (2020) 4
49. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1912–1920 (2015) 4
50. Xie, Q., Lai, Y.K., Wu, J., Wang, Z., Zhang, Y., Xu, K., Wang, J.: Mlcvnet: Multi-level context votenet for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10447–10456 (2020) 3, 6, 10, 12, 23
51. Yi, L., Guibas, L., Hertzmann, A., Kim, V.G., Su, H., Yumer, E.: Learning hierarchical shape segmentation and labeling from online repositories. arXiv preprint arXiv:1705.01661 (2017) 4
52. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016) 4
53. Yi, L., Zhao, W., Wang, H., Sung, M., Guibas, L.J.: Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3947–3956 (2019) 4

A Additional qualitative analysis

In Figures 6 and 7, we show additional qualitative results of comparison of NPPs to baseline methods. We can see that StructureNet [31] and Bokhovkin et al. [3] often produce incomplete and inaccurate shapes, can include inconsistent parts (e.g. predicting only one chair arm or predicting two types of legs for one chair). The advanced point cloud segmentation method PointGroup [27] is able to predict consistent part types for shapes but produces fairly noisy geometry for these parts. In addition, when comparing to baselines and NPPs without applying scene-aware constraints, we can clearly see a large amount of diversity within shapes that should be similar or identical within one scan.

B Interpolation properties of learned latent part spaces

In Figure 8, we show the interpolation capabilities of the part latent spaces that we use in NPPs to traverse during test-time optimization. Although each part space has been learned individually, their interpolations can produce consistent shapes.

C Part types per category

In Figure 9 we present the shape categories and the corresponding parts that we use in our framework. There are 6 shape categories and 28 part types in total.

D Implementation Details

We provide further implementation details; note that parameters reported here are for the 'chair' category, and other category parameter differences are specified in Table 4.

D.1 Pretrain decoders

We first train our latent part and shape spaces on the synthetic PartNet [32] dataset. This corresponds to the tasks 'Train decoder (shape)', 'Train decoder (parts)' in the Table 4. The part and shape decoders are all MLPs composed of 8 linear layers of 512 dimensions each, using ReLU nonlinearities with a final tanh for SDF output. The detailed architecture is shown in Tables 9, 10. To train the shape decoder, we use an Adam optimizer with a batch size of 24 and learning rate of 5e-5 ('lr' in the Table 4) for network weights (with a factor 0.5 ('lr factor') and decay interval of 500 epochs ('lr decay int.')) and 1e-4 for latent parameters (with a factor 0.5 and decay interval of 500 epochs), and train for 2000 epochs. For the part decoder, we extend every part latent with one-hot encoded part type and train the part decoder using an Adam optimizer with a batch size of 48 and learning rate of 5e-5 for network weights (with a decay factor of 0.5 and decay interval of 400 epochs) and 1e-4 for latent parameters (with a decay factor 0.5 and decay interval of 400 epochs), and train for 2000 epochs.

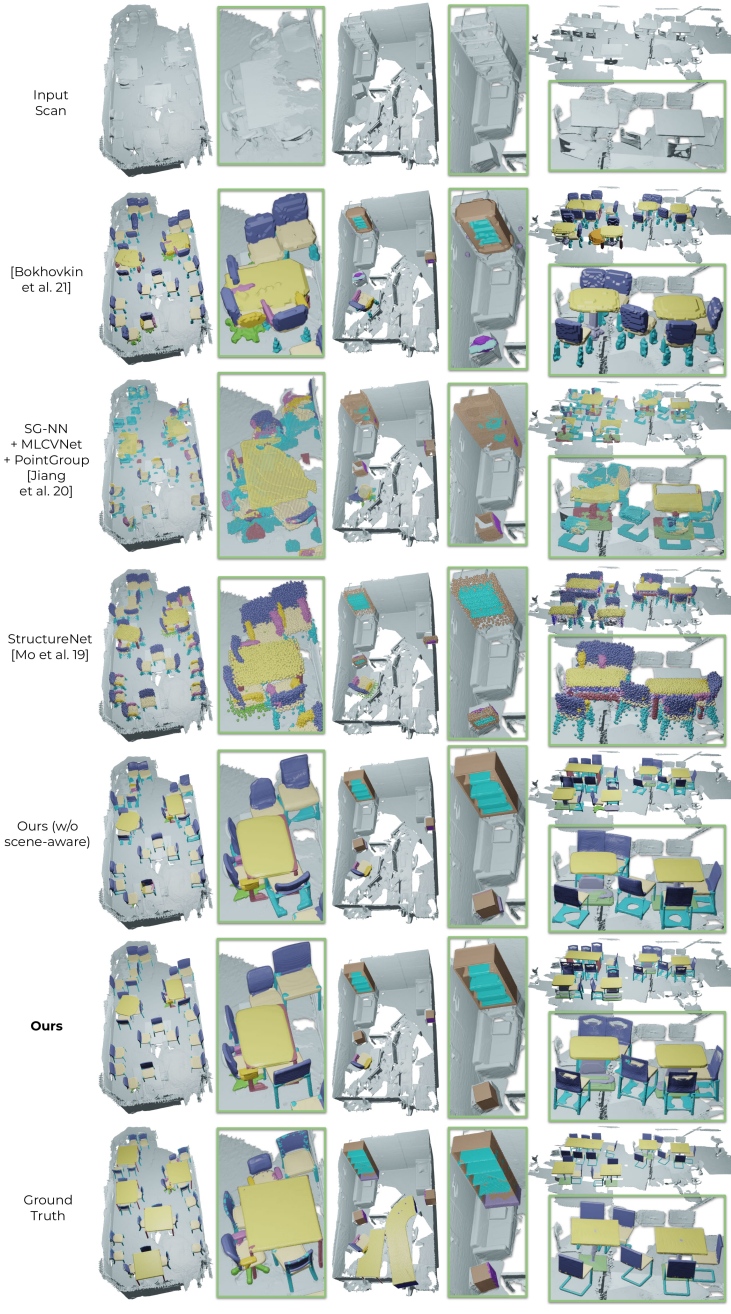


Fig. 6: Additional qualitative comparison of NPPs with point [27,31] and voxel-based [3] state of the art on ScanNet scans with Scan2CAD+PartNet ground truth.

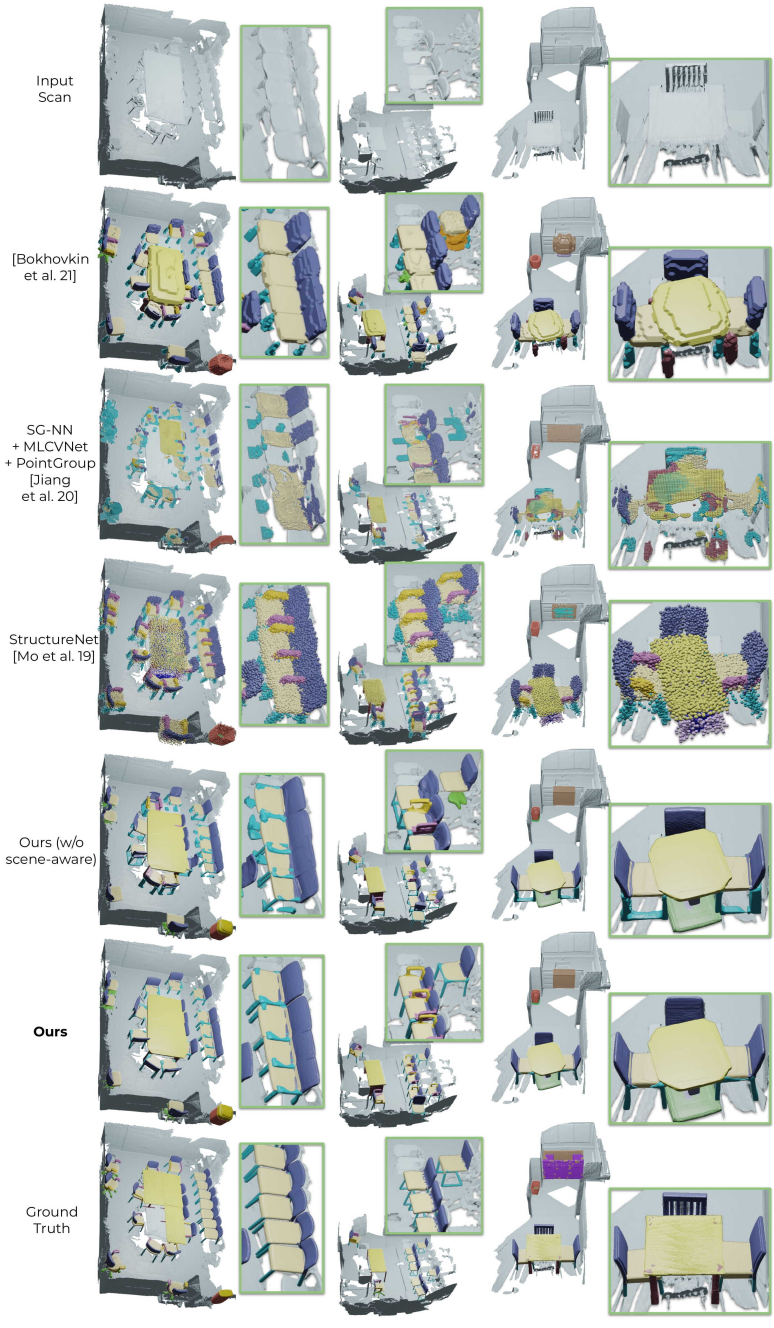


Fig. 7: Additional qualitative comparison of NPPs with point [27,31] and voxel-based [3] state of the art on ScanNet scans with Scan2CAD+PartNet ground truth.

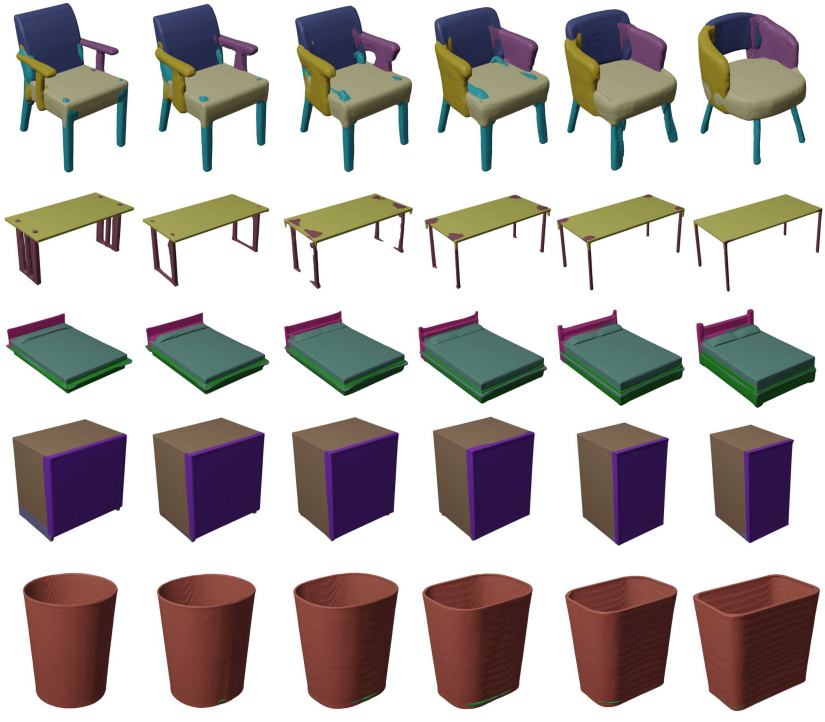


Fig. 8: Part interpolations through our learned latent part spaces for different shape classes.

Fig. 9: Part specifications per category for the parts used in our approach. Note that 'cabinet' and 'bookshelf' have the same set of parts.

D.2 Pre-training for latent projection and part segmentation

We then train the projection mapping into the learned part and shape spaces as well as the part segmentation. This part corresponds to the task 'Train projection' in the Table 4. Our model is pre-trained on synthetic PartNet data using virtually scanned incomplete inputs to take advantage of the large amount of synthetic data. We use an Adam optimizer with batch size 64 and learning rate 1e-3 ('lr' in the Table 4) decayed by half ('lr factor') every 12 epochs ('lr decay int.') for 35 epochs. We use a large and a small PointNet-based [40] network (small ('PN-small') and large ('PN-big')) to segment an input TSDF into parts and background. We refer to the Tables 7, 8 as architectures of 'PN-small' and 'PN-big' denoted in the Table 4.

D.3 Fine-tuning on ScanNet data

To apply to real-world observations, we fine-tune the projections and part segmentation on ScanNet [8] data using MLCVNet [50] detections on train scenes. We use an Adam optimizer with batch size 64, learning rate 2e-4 ('lr' in the Table 4) decayed by a factor of 0.2 ('lr factor') every 40 epochs ('lr decay int.') for 80 epochs.

D.4 Test-time optimization

For test-time optimization, we optimize for part and shape codes using an Adam optimizer with learning rate of 3e-4 ('lr' in the Table 4) for 600 iterations. The

learning rate is multiplied by factor of 0.1 ('lr factor') after 300 iterations ('lr decay int.')). This part corresponds to the task 'Test-time opt.' in the Table 4.

To enable more flexibility to capture input details, we enable optimization of the decoder weights for parts and shape after 400 iterations. We have use the first and the second linear layers of part decoder ('part dec. layers opt.') to optimize simultaneously with latent vectors optimization using Adam optimizer with learning rate of 3e-4 ('lr') for 600 iterations. The learning rate is multiplied by factor of 0.1 ('lr factor (part dec.)') after 300 iterations ('lr decay int. (part dec.)')).

In Eqs. (5), (6) we use a weight w_{trunc} for points close to and further away from the surface. We have a set $\mathcal{A}_{unif.noise}$ of points that have a distance to surface greater than $d_{trunc} = 0.16$m. Having decoded the projection of the shape $\{\tilde{\mathbf{z}}^{\mathrm{s}}\}$, we uniformly sample points around decoded shape no closer than 0.2m to the surface of this shape, and assign truncation distance d_{trunc} to these points. We also add them to the set $\mathcal{A}_{unif.noise}$. For the shape decoder and for the set $\mathcal{A}_{unif.noise}$ we set $w_{trunc} = 5.0$ ('w_{trunc} (shape unif. noise)' in the Table 4); for part decoder we set $w_{trunc} = 20.0$ ('w_{trunc} (parts unif. noise)'). Additionally, while optimizing the particular part k during test-time optimization we also use the points corresponding to other parts as noise with distance d_{trunc} and denote this set of points as $\mathcal{A}_{partnoise}$. Adding this set into optimization is necessary to decrease intersections between different part geometries after test-time optimization. We set $w_{trunc} = 5.0$ ('w_{trunc} (part noise)') for this set of points. Finally, in Eq. (4) we use an additional weight for loss consistency term, for which we set $w_{cons} = 200.0$.

To encourage geometric completeness during test-time optimization, we sample points with distances to surface from the decoded shape $\mathcal{S}$ and parts $\{\mathcal{P}_k\}$ (decoded from $\{\tilde{\mathbf{z}}^{\mathrm{s}}\}$ and $\{\tilde{\mathbf{z}}^{\mathrm{p}}_k\}$), and add them to TSDF D ('add pts. to shape' in the Table 4) or $\{D^{\mathrm{p}}\}_{\mathrm{p}=1}^{N_{parts}}$ ('add pts. to parts') to the regions where points in $\mathcal{S}$ or $\{\mathcal{P}_k\}$ are present and non-background points with distance $d < d_{trunc}$ in D and $\{D^{\mathrm{p}}\}_{\mathrm{p}=1}^{N_{parts}}$ (which we call *meaningful* points) are missing. We add only those points from $\mathcal{S}$ and $\{\mathcal{P}_k\}$ which are not closer than d_{thr} to meaningful points.

Finally, we scale the coordinates of input TSDF with a scale factor ('scale factor') to align better to the learned canonical space of synthetic shapes.

Optimization for each part takes approximately 90 seconds.

D.5 Network Architecture

We also provide an extensive information about architecture of every submodel that we use in our framework. Table 5 shows the architecture of voxel encoder that we use to encode an input occupancy grid. Table 6 shows the architecture of a module that predicts the part decomposition of an input object. The architectures of a small PointNet-like [40] network and a big PointNet-like network that we use to segment an input TSDF are shown in Tables 7, 8. Finally, we provide details about the architecture of shape and parts MLP decoders in Tables 9, 10.

Task	Parameter	Chair	Table	Cabinet	Bookshelf	Bed	Trashcan
Train decoder (shape)	# epochs	2000	2400	8000	8000	16000	16000
Train decoder (shape)	batch size	24	24	24	24	24	24
Train decoder (shape)	optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Train decoder (shape)	lr (weights)	5e-5	1e-4	1e-4	1e-4	1e-4	1e-4
Train decoder (shape)	lr factor (weights)	0.5	0.5	0.5	0.5	0.5	0.5
Train decoder (shape)	lr decay int. (weights)	500	600	1500	1500	4000	3000
Train decoder (shape)	lr (lat.)	1e-4	2e-4	2e-4	2e-4	2e-4	2e-4
Train decoder (shape)	lr factor (lat.)	0.5	0.5	0.5	0.5	0.5	0.5
Train decoder (shape)	lr decay int. (lat.)	500	600	1500	1500	4000	3000
Train decoder (parts)	# epochs	1100	1400	2000	2000	10000	10000
Train decoder (parts)	batch size	48	48	48	48	48	48
Train decoder (parts)	optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Train decoder (parts)	lr (weights)	5e-5	1e-4	1e-4	1e-4	1e-4	1e-4
Train decoder (parts)	lr factor (weights)	0.5	0.5	0.5	0.5	0.5	0.5
Train decoder (parts)	lr decay int. (weights)	400	400	800	800	3600	3600
Train decoder (parts)	lr (lat.)	1e-4	2e-4	2e-4	2e-4	2e-4	2e-4
Train decoder (parts)	lr factor (lat.)	0.5	0.5	0.5	0.5	0.5	0.5
Train decoder (parts)	lr decay int. (lat.)	400	400	800	800	3600	3600
Train projection	# epochs	35	30	60	60	250	200
Train projection	batch size	64	64	64	64	64	64
Train projection	optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Train projection	lr	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Train projection	lr factor	0.5	0.5	0.5	0.5	0.5	0.5
Train projection	lr decay int.	12	20	40	40	150	120
Train projection	segm. network	PN-small	PN-big	PN-big	PN-big	PN-small	PN-small
Fine-tune	# epochs	80	80	120	120	125	70
Fine-tune	batch size	64	64	64	64	64	64
Fine-tune	optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Fine-tune	lr	2e-4	2e-4	2e-4	2e-4	2e-4	2e-4
Fine-tune	lr factor	0.2	0.2	0.2	0.2	0.2	0.2
Fine-tune	lr decay int.	40	40	60	60	60	40
Test-time opt.	# iterations	600	600	600	600	600	600
Test-time opt.	optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Test-time opt.	lr (lat.)	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
Test-time opt.	lr factor (lat.)	0.1	0.1	0.1	0.1	0.1	0.1
Test-time opt.	lr decay int. (lat.)	300	300	300	300	300	300
Test-time opt.	shape dec. layers opt.	-	-	-	-	1,2	1,2
Test-time opt.	lr (shape dec.)	-	-	-	-	1e-4	1e-4
Test-time opt.	lr factor (shape dec.)	-	-	-	-	0.1	0.1
Test-time opt.	lr decay int. (shape dec.)	-	-	-	-	300	300
Test-time opt.	part dec. layers opt.	1,2	1,2	1,2	1,2	-	-
Test-time opt.	lr (part dec.)	3e-4	3e-4	3e-4	3e-4	-	-
Test-time opt.	lr factor (part dec.)	0.1	0.1	0.1	0.1	-	-
Test-time opt.	lr decay int. (part dec.)	300	300	300	300	-	-
Test-time opt.	w_{trunc} (shape unif. noise)	5.0	3.0	3.0	3.0	1.0	1.0
Test-time opt.	w_{trunc} (parts unif. noise)	20.0	12.0	12.0	12.0	1.0	5.0
Test-time opt.	w_{trunc} (part noise)	5.0	3.0	10.0	10.0	10.0	5.0
Test-time opt.	w_{cons}	200.0	300.0	300.0	300.0	30.0	200.0
Test-time opt.	add pts. to shape	✓	✓	✓	✓	✓	-
Test-time opt.	dist thr. (shape)	0.16m	0.16m	0.5m	0.5m	0.75m	-
Test-time opt.	add pts. to parts	-	✓	✓	✓	✓	-
Test-time opt.	dist thr. (part)	-	0.16m	0.5m	0.5m	0.75m	-
Test-time opt.	scale factor	1.0	1.2	1.4	1.4	1.1	1.1

Table 4: Hyperparameters used for training submodels used in our framework.

Encoder	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
conv0	scan occ. grid	Conv3D	(1, 32, 32, 32)	(32, 16, 16, 16)	(5, 5, 5)	(2, 2, 2)	(2, 2, 2)
gnorm0	conv0	GroupNorm	(32, 16, 16, 16)	(32, 16, 16, 16)	-	-	-
relu0	gnorm0	ReLU	(32, 16, 16, 16)	(32, 16, 16, 16)	-	-	-
pool1	relu0	MaxPooling	(32, 16, 16, 16)	(32, 8, 8, 8)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
conv1	pool1	Conv3D	(32, 8, 8, 8)	(64, 8, 8, 8)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
gnorm1	conv1	GroupNorm	(64, 8, 8, 8)	(64, 8, 8, 8)	-	-	-
relu1	gnorm1	ReLU	(64, 8, 8, 8)	(64, 8, 8, 8)	-	-	-
pool2	relu1	MaxPooling	(64, 8, 8, 8)	(64, 4, 4, 4)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
conv2	pool2	Conv3D	(64, 4, 4, 4)	(128, 2, 2, 2)	(5, 5, 5)	(2, 2, 2)	(2, 2, 2)
gnorm2	conv2	GroupNorm	(128, 2, 2, 2)	(128, 2, 2, 2)	-	-	-
relu2	gnorm2	ReLU	(128, 2, 2, 2)	(128, 2, 2, 2)	-	-	-
pool3	relu2	MaxPooling	(128, 2, 2, 2)	(128, 1, 1, 1)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
conv3	pool3	Conv3D	(128, 1, 1, 1)	(256, 1, 1, 1)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
gnorm3	conv3	GroupNorm	(256, 1, 1, 1)	(256, 1, 1, 1)	-	-	-
relu3	gnorm3	ReLU	(256, 1, 1, 1)	(256, 1, 1, 1)	-	-	-
shape feature	relu3	Flatten	(256, 1, 1, 1)	(256)	-	-	-

Table 5: Layer specification for detected object encoder.

Child decoder	Input Layer	Type	Input Size	Output Size
lin_proj	shape feature	ReLU(Linear)	256	256
node feature	lin_proj	ReLU(Linear)	256	256
lin0	node feature	Linear	256	2560
relu0	lin0	ReLU	2560	2560
reshape0	relu0	Reshape	2560	(10, 256)
node_exist	reshape0	Linear	(10, 256)	(10, 1)
concat0	(reshape0, reshape0)	Concat.	(10, 256), (10, 256)	(10, 10, 512)
lin1	concat0	Linear	(10, 10, 512)	(10, 10, 256)
relu1	lin1	ReLU	(10, 10, 256)	(10, 10, 256)
edge_exist	relu1	Linear	(10, 10, 256)	(10, 10, 1)
mp	(relu1, edge_exist, reshape0)	Mes. Passing	(10, 10, 256), (10, 10, 1), (10, 256)	(10, 768)
lin2	mp	Linear	(10, 768)	(10, 256)
relu2	lin2	ReLU	(10, 256)	(10, 256)
node_sem	relu2	Linear	(10, 256)	(10, #classes)
lin3	relu2	Linear	(10, 256)	(10, 256)
(10, child feature)	lin3	ReLU	(10, 256)	(10, 256)
lin4	node feature	ReLU(Linear)	256	256
rotation_cls	lin3	Linear	256	12

Table 6: Layer specification for decoding an object into its semantic part structure.

Pts. classifier (small)	Input Layer	Type	Input Size	Output Size
input feature	(TSDF, node feature, rotation_cls)	Concat.	(#pts, 4), 256, 12	(#pts, 272)
lin_cls_0	input feature	ReLU(Linear)	(#pts, 272)	(#pts, 128)
lin_cls_1	lin_cls_0	ReLU(Linear)	(#pts, 128)	(#pts, 128)
lin_cls_2	lin_cls_1	ReLU(Linear)	(#pts, 128)	(#pts, 128)
glob_feat_0	lin_cls_2	MaxPooling1D	(#pts, 128)	(1, 128)
glob_feat_1	glob_feat_0	Repeat	(1, 128)	(#pts, 128)
lin_cls_3	(lin_cls_1, glob_feat_1)	Concat	(#pts, 128), (#pts, 128)	(#pts, 256)
lin_cls_4	lin_cls_3	ReLU(Linear)	(#pts, 256)	(#pts, 128)
lin_cls_5	lin_cls_4	ReLU(Linear)	(#pts, 128)	(#pts, 128)
lin_cls_6	lin_cls_5	Linear	(#pts, 128)	(#pts, #classes)

Table 7: Layer specification for segmenting input TSDF using small PointNet-like network.

Pts. classifier (big)	Input Layer	Type	Input Size	Output Size
input feature	(TSDF, node feature, rotation_cls)	Concat.	(#pts, 4), 256, 12	(#pts, 272)
lin_cls_0	input feature	ReLU(Linear)	(#pts, 272)	(#pts, 256)
lin_cls_1	lin_cls_0	ReLU(Linear)	(#pts, 256)	(#pts, 128)
lin_cls_2	lin_cls_1	ReLU(Linear)	(#pts, 128)	(#pts, 128)
glob_feat_0	lin_cls_2	MaxPooling1D	(#pts, 128)	(1, 128)
glob_feat_1	glob_feat_0	Repeat	(1, 128)	(#pts, 128)
lin_cls_3	lin_cls_2	ReLU(Linear)	(#pts, 128)	(#pts, 64)
lin_cls_4	lin_cls_3	ReLU(Linear)	(#pts, 64)	(#pts, 64)
glob_feat_2	lin_cls_4	MaxPooling1D	(#pts, 64)	(1, 64)
glob_feat_3	glob_feat_2	Repeat	(1, 64)	(#pts, 64)
lin_cls_5	(lin_cls_1, glob_feat_1, glob_feat_3)	Concat	(#pts, 128), (#pts, 128), (#pts, 64)	(#pts, 320)
lin_cls_6	lin_cls_5	ReLU(Linear)	(#pts, 320)	(#pts, 128)
lin_cls_7	lin_cls_6	ReLU(Linear)	(#pts, 128)	(#pts, 64)
lin_cls_8	lin_cls_7	Linear	(#pts, 64)	(#pts, #classes)

Table 8: Layer specification for segmenting input TSDF using big PointNet-like network.

Implicit decoder	Input Layer	Type	Input Size	Output Size
lin_proj_0	node feature	ReLU(Linear)	256	512
lin_proj_1	lin_proj_0	ReLU(Linear)	512	512
lin_proj_2	lin_proj_1	ReLU(Linear)	512	512
lin_proj_3	lin_proj_2	ReLU(Linear)	512	512
lin_proj_4	lin_proj_3	Linear	512	256
lin_pts_0	(lin_proj_4, TSDF pts.)	Concat.	256, 3	259
lin_pts_1	lin_pts_0	Linear	259	512
lin_bn_1	lin_pts_1	BatchNorm	512	512
lin_relu_1	lin_bn_1	ReLU	512	512
lin_drop_1	lin_relu_1	Dropout	512	512
lin_pts_2	lin_pts_1	Linear	512	512
lin_bn_2	lin_pts_2	BatchNorm	512	512
lin_relu_2	lin_bn_2	ReLU	512	512
lin_drop_2	lin_relu_2	Dropout	512	512
lin_pts_3	lin_pts_2	Linear	512	512
lin_bn_3	lin_pts_3	BatchNorm	512	512
lin_relu_3	lin_bn_3	ReLU	512	512
lin_drop_3	lin_relu_3	Dropout	512	512
lin_pts_4	lin_pts_3	Linear	512	512 - dim(lin_pts_0)
lin_bn_4	lin_pts_4	BatchNorm	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_relu_4	lin_bn_4	ReLU	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_drop_4	lin_relu_4	Dropout	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_pts_5	(lin_pts_0, lin_drop_4)	Concat.	dim(lin_pts_0), 512 - dim(lin_pts_0)	512
lin_bn_5	lin_pts_5	BatchNorm	512	512
lin_relu_5	lin_bn_5	ReLU	512	512
lin_drop_5	lin_relu_5	Dropout	512	512
lin_pts_6	lin_pts_5	Linear	512	512
lin_bn_6	lin_pts_6	BatchNorm	512	512
lin_relu_6	lin_bn_6	ReLU	512	512
lin_drop_6	lin_relu_6	Dropout	512	512
lin_pts_7	lin_pts_6	Linear	512	512
lin_bn_7	lin_pts_7	BatchNorm	512	512
lin_relu_7	lin_bn_7	ReLU	512	512
lin_drop_7	lin_relu_7	Dropout	512	512
lin_pts_8	lin_pts_7	Linear	512	1
lin_tanh_7	lin_pts_8	Tanh	1	1

Table 9: Layer specification for implicit shape decoder.

Implicit decoder	Input Layer	Type	Input Size	Output Size
lin_proj_0	child feature	ReLU(Linear)	256	512
lin_proj_1	lin_proj_0	ReLU(Linear)	512	512
lin_proj_2	lin_proj_1	ReLU(Linear)	512	512
lin_proj_3	lin_proj_2	ReLU(Linear)	512	512
lin_proj_4	lin_proj_3	Linear	512	256
lin_pts_0	(lin_proj_4, part cls. one-hot, TSDF pts.)	Concat.	256, #parts, 3	259 + #parts
lin_pts_1	lin_pts_0	Linear	259 + #parts	512
lin_bn_1	lin_pts_1	BatchNorm	512	512
lin_relu_1	lin_bn_1	ReLU	512	512
lin_drop_1	lin_relu_1	Dropout	512	512
lin_pts_2	lin_pts_1	Linear	512	512
lin_bn_2	lin_pts_2	BatchNorm	512	512
lin_relu_2	lin_bn_2	ReLU	512	512
lin_drop_2	lin_relu_2	Dropout	512	512
lin_pts_3	lin_pts_2	Linear	512	512
lin_bn_3	lin_pts_3	BatchNorm	512	512
lin_relu_3	lin_bn_3	ReLU	512	512
lin_drop_3	lin_relu_3	Dropout	512	512
lin_pts_4	lin_pts_3	Linear	512	512 - dim(lin_pts_0)
lin_bn_4	lin_pts_4	BatchNorm	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_relu_4	lin_bn_4	ReLU	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_drop_4	lin_relu_4	Dropout	512 - dim(lin_pts_0)	512 - dim(lin_pts_0)
lin_pts_5	(lin_pts_0, lin_drop_4)	Concat.	dim(lin_pts_0), 512 - dim(lin_pts_0)	512
lin_bn_5	lin_pts_5	BatchNorm	512	512
lin_relu_5	lin_bn_5	ReLU	512	512
lin_drop_5	lin_relu_5	Dropout	512	512
lin_pts_6	lin_pts_5	Linear	512	512
lin_bn_6	lin_pts_6	BatchNorm	512	512
lin_relu_6	lin_bn_6	ReLU	512	512
lin_drop_6	lin_relu_6	Dropout	512	512
lin_pts_7	lin_pts_6	Linear	512	512
lin_bn_7	lin_pts_7	BatchNorm	512	512
lin_relu_7	lin_bn_7	ReLU	512	512
lin_drop_7	lin_relu_7	Dropout	512	512
lin_pts_8	lin_pts_7	Linear	512	1
lin_tanh_7	lin_pts_8	Tanh	1	1

Table 10: Layer specification for implicit part decoder.