EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Self-supervised Learning of Fine-to-Coarse Cuboid Shape Abstraction

Gregor Kobsik, Morten Henkel, Yanjiang He, Victor Czech, Tim Elsner, Isaak Lim, and Leif Kobbelt

Visual Computing Institute, RWTH Aachen University, Germany

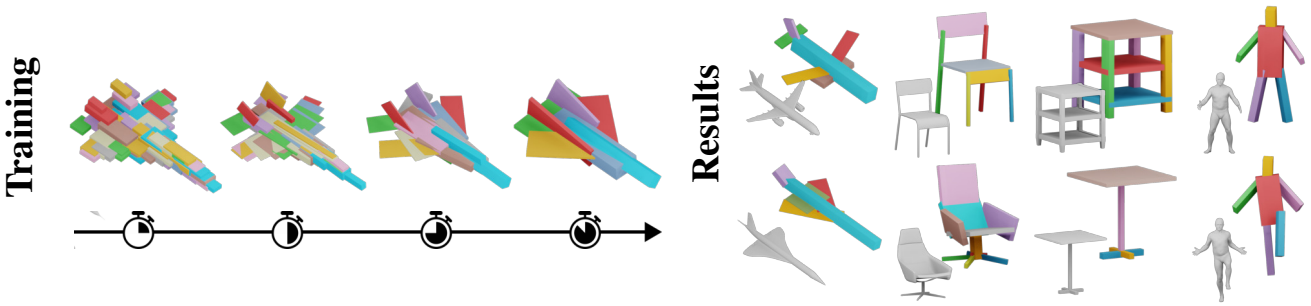


Figure 1: *Left: During training the number of primitives decreases from hundreds of cuboids to only a few. Colors are based on a global cuboid index. Right: At inference we obtain a structurally consistent abstraction of the shapes. Colors indicate distinct parts per category.*

Abstract

The abstraction of 3D objects with simple geometric primitives like cuboids allows us to infer structural information from complex geometry. It is important for 3D shape understanding, structural analysis and geometric modeling.

We introduce a novel fine-to-coarse self-supervised learning approach to abstract collections of 3D shapes. Our architectural design allows us to reduce the number of primitives from hundreds (fine reconstruction) to only a few (coarse abstraction) during training. This allows our network to optimize the reconstruction error and adhere to a user-specified number of primitives per shape while simultaneously learning a consistent structure across the whole collection of data. We achieve this through our abstraction loss formulation which increasingly penalizes redundant primitives. Furthermore, we introduce a reconstruction loss formulation to account not only for surface approximation but also volume preservation. Combining both contributions allows us to represent 3D shapes more precisely with fewer cuboid primitives than previous work.

We evaluate our method on collections of man-made and humanoid shapes comparing with previous state-of-the-art learning methods on commonly used benchmarks. Our results confirm an improvement over previous cuboid-based shape abstraction techniques. Furthermore, we demonstrate our cuboid abstraction in downstream tasks like clustering, retrieval, and partial symmetry detection.

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis;***

1. Introduction

The decomposition of 3D objects into their underlying structural elements is a longstanding problem in the field of computer graphics and computer vision. Real-world objects can often be decomposed into geometrically distinct parts. For example, a chair can be decomposed into a seat, a backrest and four legs, while a human consists of a head, a torso, as well as two arms and legs (cmp. Figure 1). Obtaining the underlying structure of objects allows us to argue about 3D shapes by analyzing the relationships between their components and automatically classify collections of them into different sub-categories, e.g. chairs without a backrest or tables with a single leg support. The desired abstraction should adhere to two target objectives. It should be *expressive* to represent the object as truthfully as possible, but also *compact* to abstract the general structure of the object [SZTL19]. Both objectives are non-trivial to

satisfy simultaneously. An optimal solution to a compact representation is most often contrary to an expressive one and vice versa.

While this is challenging to automate, humans tend to decompose objects into their components naturally [Bie87]. Much research has been invested into learning the decomposition of 3D objects given manual annotations, either with structural information [MGY*19, YKC*16] or semantic labels [CFG*15, QSMG17, QLP*22]. However, annotating data, especially in 3D, is a time consuming process.

Self-supervised techniques focus on the geometric properties of the object without human annotations [TSG*17, SZTL19, YC21]. In this paper we propose a self-supervised cuboid-based shape abstraction method. In contrast to the previous methods, we do not optimize our model with a fixed number of primitives, but initially use an extensive number of cuboid primitives to capture all details with a very fine-granular reconstruction, satisfying the expressiveness requirement first. Then, we gradually trade reconstruction quality of our model for compactness by reducing the amount of available primitives during training. This forces our model to learn a compact abstraction of the target shape using a prescribed number of primitives by discarding or merging redundant ones while approximating the shape as faithfully as possible. During training individual primitives cover increasingly larger areas of the shape continuously improving the fitting to their geometric parts. Ultimately, resulting in each cuboid representing a single distinct part of the object.

We choose cuboids as our representative primitive as it is one of the most basic 3D shapes. Even though superquadrics [PUG19] or implicit parts [CYF*19] allow for a more precise decomposition of the object, they introduce complex geometry per primitive, which is contrary to a compact representation. Cuboids are simple and compact, offer the least number of degrees of freedom, and thus are widely used in other structure-based algorithmic and learning approaches [MWZ*14, MGY*19, LPG24].

Our key contributions are:

- Novel training scheme for self-supervised shape abstraction with a fine-to-coarse loss functions, which allows discarding or merging redundant primitives.
- Simple user control over the abstraction quality by defining a target number of primitives.
- State-of-the-art performance on the structural shape reconstruction benchmarks for cuboid-based methods.
- Applying our results to downstream task like co-segmentation, editing, clustering, retrieval, or partial symmetry detection.

2. Related Work

2.1. Algorithmic Primitive Fitting

Primitive fitting and shape abstraction techniques have a long tradition in 3D shape analysis [SB90, LA04]. We refer to [RMG18] for an in-depth survey. They allow extracting structural information of a single shape to obtain e.g. a label-agnostic segmentation of the shape, but correspondences between abstracted primitives within a collection are often ignored. They miss out on the extraction of global knowledge from a dataset and need to be re-run on unseen samples [CJB03, LLL*20]. More recently, [LWRC22] proposed a probabilistic approach using Expectation-Maximization with global switching (EMS) for robust superquadric recovery from point clouds, demonstrating improved accuracy and outlier handling.

2.2. Supervised Structural Learning

Some research has been performed on supervised learning for structured representation of collections of 3D shapes [LXC*17, GYW*19, MGY*19, LPG24], including recent work on sequential and generative part-based methods [WZX*20, YHZ*25]. These works use the underlying structure to either interpolate between different shapes using the learned latent space or to generate novel 3D shapes following the learned structure of the data. Conditioning a generative model on an abstract representation leads to more precise and structure-preserving results. Furthermore, it allows to prescribe the final shape in an intuitively controllable manner. Unfortunately, this data cannot be easily obtained and all approaches rely on annotated datasets. These methods can benefit greatly from an automatic cuboid abstraction of 3D shapes. PQ-Net [WZX*20] represents 3D shapes as a sequence of parts, using a Seq2Seq autoencoder to capture both structural relationships and fine geometric details, enabling shape interpolation and generation. More recently, PrimitiveAnything [YHZ*25] reformulates shape abstraction as a primitive assembly generation task using an auto-regressive transformer trained on large-scale human-crafted abstractions, producing assemblies that align with human visual cognition.

2.3. Self-supervised Shape Abstraction

To extract the structure of geometric objects one cannot always depend on annotated data. Self-supervised techniques focus on geometric features of a shape, learn a latent representation, and finally extract an approximate reconstruction of the input using a limited number of primitives. One defining part of each method is the selection of the representative primitive.

Cuboids. The simplest choice of a volumetric primitive is a cuboid [TSG*17, SZTL19, YC21]. They allow intuitive interpretability of their parameters and straightforward processing, but sacrifice reconstruction quality per primitive due to the limited expressiveness of the shape. [TSG*17] showed that they are still a valid choice for shape abstraction. [SZTL19] continued the line of work and extended it to hierarchical compositions of cuboids. They fit cuboids at multiple levels and use a cuboid selection module to obtain the optimal abstraction. It is a coarse-to-fine approach that considers first cuboids at the coarsest level and subdivides them if necessary. In an ablation study using a multi-stage training scheme and freezing networks, they surpass a fine-to-coarse setup. In contrast, we show that a fine-to-coarse approach starting with a large number of primitives and discarding or merging redundant ones is able to provide superior results. [YC21] propose to couple shape abstraction together with segmentation. They introduce two decoders, where the first one learns the abstraction and the second one learns to associate cuboids with the nearest points and couple them by their loss formulation. Furthermore, they observe that shape abstraction methods suffer often from degenerate primitives reconstructing only the surface of the shape and not its volume. To

resolve this problem [YC21] introduce a loss formulation incorporating surface normals of the input shape as well as the normals of the cuboid surface. We propose to use a simpler approach to resolve this issue and do not consider only the surface of the shape but also enforce volume preservation of the shape with our loss directly.

Superquadrics. An extension to cuboids provides the class of superquadrics. By incorporating only two more parameters they are able to extend the primitive types by ellipsoids, spheres, cylinders, octahedra and interpolations in between. They offer the advantage of a continuous, differentiable parametrized representation, but are rather challenging to edit intuitively or sample uniformly [PF95, Fer18]. Nevertheless, multiple methods [PUG19, PGG20, LWTY24] used them to abstract 3D shapes improving the reconstruction quality while using a similar or lower number of primitives than previous work. SuperDec [FSG*25] introduced a transformer-based architecture trained on ShapeNet that decomposes point clouds into superquadric primitives, demonstrating strong generalization to real-world scenes.

Free form. The most general class of primitives can be characterized by implicit parts or deformable templates. They learn to deform a simple sphere mesh [PKGF21], cuboids and cylinders [SZYC23] or to generate an implicit function [CCZZ24, NLXZ22]. The methods mostly rely on a branched autoencoder architecture [CYF*19] where the decoder consists of multiple sub-networks learning a representation of one commonly occurring part each. They offer an expressive representation (e.g., good reconstruction quality) at the cost of compactness, as each primitive is represented by a complex mesh and not a simple parametric object.

3. Cuboid Shape Abstraction

In the following section we introduce our learning-based fine-to-coarse shape abstraction approach. Given a surface point cloud X_S and a volume point cloud X_V our goal is to train a neural network that predicts a set of cuboid primitives $\{C_i\}_{i=1,\dots,M}$ which approximates the corresponding 3D shape as closely as possible. Each cuboid is identified by its unique index m and parametrized by a rotation represented as an unit quaternion $r \in \mathbb{R}^4$, a translation $t \in \mathbb{R}^3$, scaling $s \in \mathbb{R}^3$ as well its existence probability $\gamma \in [0,1]$. During inference time we only require a surface point cloud X_S as input to our network.

3.1. Architecture Design

Our method uses a simple and flexible architecture. It consists of a point cloud encoder, two transformer blocks and finally a cuboid prediction head (cmp. Figure 2). Due to the nature of transformers operating on sets of tokens it allows us to use a dynamic number of latent features, which can either represent the input shape or the cuboid primitives. With this architecture we do not need to specify a fixed number of primitives a-priori like [TSG*17, SZTL19, YC21, PUG19, LWTY24, CYF*19] and can even discard redundant primitives to speedup the training and inference times. The encoder part of the architecture is inspired by [ZNW22] and the cuboid decoder by [YC21]. We provide a brief description of our architecture below and a detailed explanation in the appendix.

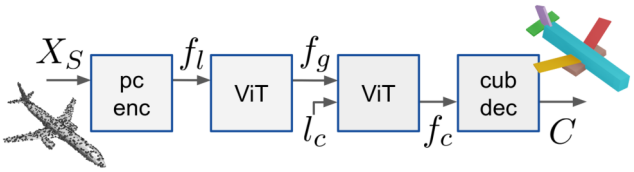


Figure 2: *A surface point cloud X_S is processed by a point cloud encoder to obtain local features f_l. These features exchange information between each other throught a ViT to obtain global features f_g. Next, they are concatenated with learnable cuboid latents l_c and passed throught a second ViT to obtain cuboid features f_c. These features are decoded into a set of cuboid primitives C which approximate X_S.*

Local shape features. The input to our network is a point cloud $X_S \in \mathbb{R}^{2048\times 3}$ sampled from a surface mesh. First, we select a subset of N points using furthest point sampling. Next, for each selected sample we compute the K nearest neighbors to extract a point cloud patch, which is finally embedded into the latent space using a small PointNet to obtain a local shape feature $f_l \in \mathbb{R}^{N\times 128}$. We use a relatively low value of $N = 128$ and $K = 32$ in all our experiments.

Global shape features. To obtain global shape features $f_g \in \mathbb{R}^{N\times 128}$ we pass f_l through a small Vision Transformer (ViT) [Dos20]. The self-attention modules exchange information between local tokens to extract global context about the shape.

Abstract cuboid features. We compute the abstract cuboid features $f_c \in \mathbb{R}^{M\times 128}$ in an auto-decoder fashion conditioned on f_g. At the beginning of the training we define learnable cuboid latents $l_c \in \mathbb{R}^{M\times 128}$ as random vectors and optimize them with the weights of our network jointly. Next, we concatenate $[f_g, l_c]$ and pass it through the second ViT block to allow each cuboid token to attend to the global tokens. To additionally allow exchange of information among the cuboid tokens we use self-attention layers. From the output sequence we keep tokens at positions of l_c to obtain the abstract cuboid features f_c. It is important to note that during inference, the learnable cuboid latents l_c are fixed. They act as learned structural templates that capture common geometric configurations across the dataset, while the network architecture predicts instance-specific attributes (scale, rotation, translation) to align these templates with the input point cloud.

We empirically obtained better results using two small ViTs computing f_g first, rather than concatenating $[f_l, l_c]$ directly and using one large ViT. Details are analyzed in Section 4.6.

Predicted cuboid primitives. Each f_c^m corresponds directly to a cuboid primitive C_m. The parameters $p_m = [r_m, t_m, s_m, \gamma_m]$ are computed by fully connected layers each of which shares its parameters across all cuboids.

3.2. Loss Formulation

We formulate the training of our network as a gradual transition between a shape reconstruction task and a shape abstraction task by continually reducing the number of used primitives. In the beginning we fit an excessive number of cuboids to the input shape which

allows to tightly approximate the target shape with a minimal reconstruction error. At the end of the training process the number of primitives has been drastically reduced, thus the remaining cuboids form an abstraction of the shape. The reconstruction loss $\mathcal{L}_{rec}$ tries to reconstruct the shape with the available primitives as precisely as possible, while the abstraction loss $\mathcal{L}_{abs}$ modulates the descriptive power of our representation by continually decreasing the number of available primitives. Both losses are coupled by the existence probability γ of each cuboid. This fine-to-coarse approach allows us to overcome local minima introduced through suboptimal initialization of the neural network in contrast to directly optimizing a low, but fixed number of primitives. Our loss formulation is geometrically motivated and thus allows to extract structural information based purely on geometric features without any annotations.

Reconstruction loss

We use a Chamfer distance based loss formulation as our reconstruction loss. For the sake of completeness, we recall Equation (1) - (4) from [PUG19]. To compute the loss from the primitives to the target $P = \{p_j^m\}_{j=1..J,m=1..M} \rightarrow X = \{x_i\}_{i=1,...,I}$, we compute the closest neighbor x of p_j^m in P and its distance Δ_j^m. Next, we average the distance across all points of all cuboids as

$$\mathcal{L}_{P\rightarrow X}(P,X) = \sum_{m=1}^{M} \frac{S_m}{J} \sum_{j=1}^{J} \Delta_j^m \gamma_m \tag{1}$$

with

$$\Delta_j^m = \min_{x\in X} \|x - p_j^m\|_2 \tag{2}$$

and weight its influence by its size S_m. Similarly, the loss from the target point cloud to the primitives $X \rightarrow P$ is computed as

$$\mathcal{L}_{X\rightarrow P}(X,P) = \sum_{i=1}^{I} \sum_{m=1}^{M} \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1} (1-\gamma_{\bar{m}}), \tag{3}$$

where $\gamma_{\bar{m}}$ is a shorthand notation to denote that there exists a primitive closer than primitive m and

$$\Delta_i^m = \min_{p\in P^m} \|x_i - p\|_2. \tag{4}$$

Note that the distances Δ^m are weighted with the existence probability γ_m as only primitives that exist should contribute to the reconstruction error.

In contrast to [PUG19, YC21, LWTY24] we do not consider only the surface of the shape but also aim to preserve its spatial volume explicitly. Given that the above loss formulation depends only on a target point cloud $X = \{x_i\}$ and a sampled point cloud $P^m = \{p_j^m\}$ of each predicted primitive, we extend it to model volume preservation. Furthermore, we do not use a fixed number of samples per primitive as [PUG19], but increase it inversely proportional to the number of remaining primitives. This allows us to keep a constant sample density throughout the training while dynamically adjusting the primitive count.

To compute the surface loss $\mathcal{L}_{surf}$ we sample the surface of each primitive uniformly to obtain P_S and compare it with the target surface point cloud X_S:

$$\mathcal{L}_{surf} = \alpha\mathcal{L}_{P\rightarrow X}(P_S,X_S) + \beta\mathcal{L}_{X\rightarrow P}(X_S,P_S). \tag{5}$$

To compute the volume loss $\mathcal{L}_{vol}$ we sample the volume of each primitive uniformly to obtain P_V and compare it with the target volume point cloud X_V. This loss is defined as

$$\mathcal{L}_{vol} = \alpha\mathcal{L}_{P\rightarrow X}(P_V,X_V) + \beta\mathcal{L}_{X\rightarrow P}(X_V,P_V). \tag{6}$$

The target volume point clouds X_V are generated as a preprocessing step by performing rejection sampling within the watertight surface meshes of the ShapeNet objects. This internal sampling is only required during training to provide the necessary supervision for volume preservation. We weight both losses with default values $\alpha = 1.2$ and $\beta = 0.8$ used in [PUG19], respectively. Furthermore, we compute the box size S_m to weight the loss either as the surface area or its volume. We use the same number of samples for the surface and volume and weight both losses with λ_{vol} and λ_{surf}. The resulting reconstruction loss is given by

$$\mathcal{L}_{rec} = \lambda_{vol}\mathcal{L}_{vol} + \lambda_{surf}\mathcal{L}_{surf}. \tag{7}$$

Abstraction loss

During training we optimize the abstraction loss $\mathcal{L}_{abs}$ starting with a high number of primitives and progressively reduce their count. Pruning a single primitive affects some portions of the shape and will force other neighboring primitives to approximate the affected region.

To control the number of primitives we define a target function $\Gamma(\phi) \in \mathbb{R}$. In the simplest case it is formulated as a linear interpolation

$$\Gamma(\phi) = \kappa_{max} - (\kappa_{max} - \kappa_{min}) \cdot \phi \tag{8}$$

between the maximum κ_{max} and minimum κ_{min} number of primitives depending on the progress of the training $\phi = \frac{epoch_{cur}}{epoch_{max}}$. We analyze other target functions (cmp. Equation (11)) in Section 4.6. To follow $\Gamma(\phi)$ closely we use the binary cross entropy as the primitive abstraction loss

$$\mathcal{L}_{abs} = \sum_i -t_i \log\gamma_i - (1-t_i)\log(1-\gamma_i), \tag{9}$$

where $t_i = 1$ for $\Gamma(\phi)$ cuboids with the largest existence probability γ_i, otherwise $t_i = 0$. Specifically, in Equation (9) we implicitly sort the M predicted cuboids by their existence probabilities in descending order, and set the target $t_i = 1$ only for the top $\Gamma(\phi)$ primitives.

Final loss. The final loss used to optimize the cuboid prediction network is a sum of the previously defined losses

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{abs}\mathcal{L}_{abs}. \tag{10}$$

Additionally, we restrict the number of processed cuboids by masking out l_c^m with $\gamma_c^m < 0.01$ in every prediction at the end of each epoch. This speeds up the training and inference times.

3.3. Merging primitives

During training the reconstruction loss $\mathcal{L}_{rec}$ approximates the target shape, while the abstraction loss $\mathcal{L}_{abs}$ reduces the number of used primitives. These are contradicting objectives. As more primitives allow for a better reconstruction quality, $\mathcal{L}_{rec}$ is prone to maximizing the number of used primitives and pursues a different

optimum than $\mathcal{L}_{abs}$. Especially, when two cuboids try to approximate the shape of a cylinder they tend to overlap and rotate by 45° along the rotation symmetry axis, e.g. the corpus of an airplane or a round tabletop of a table. The initial approximation can often be simplified by merging overlapping cuboids to obtain structurally equivalent representation using less primitives without a significant degradation in reconstruction quality. We merge such cuboids at inference time as a post-processing step. Implementation details are provided in the appendix.

4. Experimental Evaluation

Datasets. We use common datasets to benchmark cuboid shape abstraction methods. For all experiments we use three categories from the ShapeNet dataset [CFG*15]: plane (3640), chair (5929), table (7555). Furthermore, we use a subset of the DFAUST dataset [BRPMB17] including humans (4179) for the reconstruction benchmark. We use only every tenth frame from DFAUST to exclude repetitive poses and match the size of the ShapeNet classes. Similar to previous data, we split this dataset randomly with a ratio of 4:1 between the train and the test set. Similar to previous work, we train a single model for each category.

Reference methods. We compare our approach with two state-of-the-art cuboid-based shape abstraction methods: [SZTL19] (HCA) and [YC21] (CAS). Furthermore, we compare with [SZYC23] (DPF$_{PPM}$) using their primitive prediction module to compute cuboid primitives. Whenever applicable we use the published model weights, otherwise we train the method five times for every category using the standard hyper-parameters and report the best performing run in terms of Chamfer distance. For DPF$_{PPM}$ we reduce the batch size to 8 to train on our GPUs.

4.1. Structured Shape Reconstruction

For the reconstruction benchmark we report three metrics to evaluate the quality of the abstraction. Chamfer distance (CD) is used to measure the surface reconstruction quality. We uniformly sample 2048 points on the ground-truth shape surface and the predicted cuboid surface. To measure how well the abstraction does approximate the spatial volume of the original shape we compute Intersection over Union (IoU) on a uniformly sampled grid with the size of 128^3. Furthermore, we report the mean number of used cuboid primitives (Num). Using an extensive number of primitives allows to easily adhere to the shape surface and volume as every primitive needs to approximate only a small portion of the shape. The reconstruction becomes much more challenging with a limited number of primitives. We regress our network to limit the mean number of cuboids within the typical range of previous work. All metrics are reported in Table 1. We provide qualitative samples in Figure 3.

Cuboids. We see that our method achieves the best scores for CD and IoU on almost all classes while using the least or second least number of cuboids on average. [SZTL19] are able to abstract the data with a lower number of primitives at the cost of a suboptimal reconstruction performance. In cases where we match or undercut their number of primitives, we are able to obtain a better reconstruction quality measured in terms of CD and IoU. Comparing with [YC21] we obtain a similar CD using substantially less primitives. In all cases we are able to significantly improve upon the volume preservation. This can be attributed to the chosen loss function, which in [YC21] focuses on the reconstruction of the surface and their normal direction, but does not account for the shape volume. Starting the reconstruction with an extensive amount of primitives allows us to capture fine details of the shape first and thus more easily overcome a suboptimal initialization leading to local minima. Preserving those features during the optimization of the network is easier than discovering them with a limited number of primitives.

We provide additional qualitative samples in Figure 8 and 9. Our reconstruction adheres faithfully to the geometry of the ground truth shape. Especially, for more complex shapes like fighter jets,

		Num↓	CD↓	IoU%↑	Prim. Type
plane	BAE	**3.69**	0.040	30.9	free form
	RIM	5.82	0.039	30.9	
	DAE	7.33	**0.025**	**59.8**	
	SQR	15.0	0.038	39.5	superquadrics
	EMS	8.29	0.037	43.8	
	SDec	**3.97**	**0.030**	**45.3**	
	HCA	7.14	0.040	36.2	cuboids
	CAS	10.64	0.029	38.1	
	DPF$_{PPM}$	15.93	0.035	55.4	
	Ours	**6.03**	**0.026**	**56.0**	
chair	BAE	**3.24**	0.049	37.1	free form
	RIM	7.06	**0.046**	36.2	
	DAE	6.17	0.047	**38.7**	
	SQR	12.0	0.063	30.2	superquadrics
	EMS	**7.40**	0.078	26.9	
	SDec	8.20	**0.046**	**40.7**	
	HCA	**6.38**	0.054	39.3	cuboids
	CAS	9.77	**0.036**	49.5	
	DPF$_{PPM}$	11.42	0.056	44.3	
	Ours	8.37	**0.036**	**54.9**	
table	BAE	**1.99**	0.053	34.4	free form
	RIM	4.97	0.050	34.0	
	DAE	5.09	**0.046**	**39.7**	
	SQR	12.0	0.070	23.8	superquadrics
	EMS	**6.71**	0.066	27.2	
	SDec	7.12	**0.052**	**27.9**	
	HCA	**4.44**	0.058	29.7	cuboids
	CAS	7.64	0.044	37.1	
	DPF$_{PPM}$	9.45	0.056	42.3	
	Ours	5.67	**0.035**	**45.1**	
human	CAS	7.50	**0.029**	32.0	cuboids
	DPF$_{PPM}$	16.00	0.040	44.4	
	Ours	**6.02**	0.032	**58.8**	

Table 1: *Quantitative evaluation of shape reconstruction on the test set. We report Chamfer distance (CD), Intersection of Union (IoU) and mean number of primitives (Num). Best highlighted in bold.*

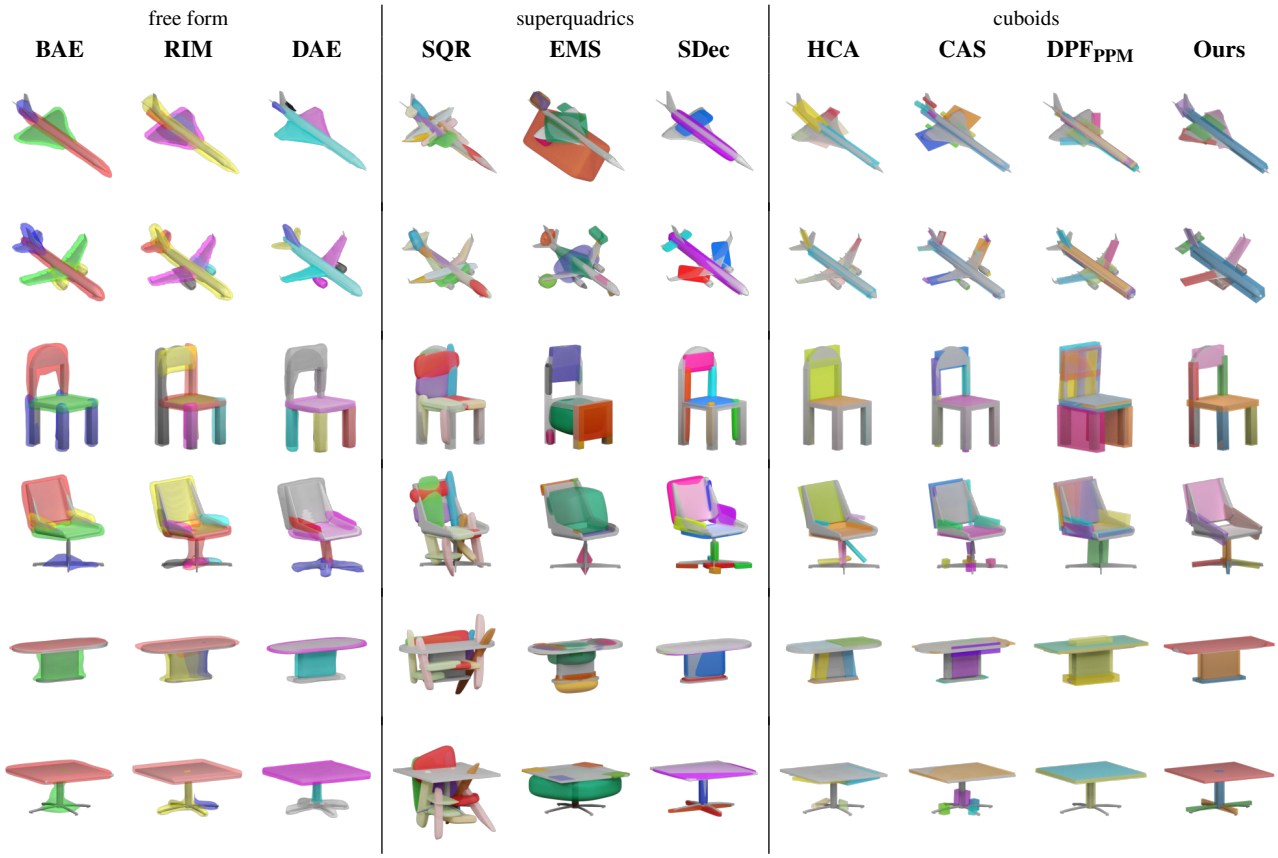


Figure 3: *Qualitative comparison with nine baseline methods: free form (BAE [CYF*19], RIM [NLXZ22], DAE [CCZZ24]), superquadrics (SQR [PUG19], EMS [LWRC22], SDec [FSG*25]), and cuboids (HCA [SZTL19], CAS [YC21], DPF$_{PPM}$ [SZYC23]). Colors indicate distinct parts.*

office chairs, and one legged tables we are able to reconstruct the shape more accurately than the state of the art. This is directly reflected by our CD and IoU scores. [YC21] try to overcome this issue by over-segmenting these challenging shapes. This leads to degenerate primitives, e.g. chair and table legs or barely recognizable human feet (cmp. Figure 9). Inspecting the table and human class more closely, we observe that other methods [SZTL19, YC21] struggle with joints between distinct primitives. The cuboids are often disconnected. By directly modeling volume preservation in our volume loss function $\mathcal{L}_{vol}$ we are able to overcome this problem. We observe that [SZYC23] struggle to extract a meaningful abstraction and over-partition the shapes. We attribute this to a reduced batch size during the training of their model. Unfortunately, no pre-trained model weights are released.

Superquadrics. We compare against superquadric-based methods including SQR [PUG19], EMS [LWRC22], and SuperDec [FSG*25]. Superquadrics offer greater expressivity than cuboids through two additional shape parameters, enabling smoother approximations of curved geometry. As shown in Table 1, SuperDec achieves competitive reconstruction quality with few primitives, particularly on the plane category. Notably, SuperDec trains a single model across all object categories, whereas our method trains one model per category. Despite this, our cuboid-based method achieves the highest IoU across all categories while maintaining comparable or better Chamfer distance. This demonstrates that the fine-to-coarse training strategy and volume preservation loss can compensate for the simpler primitive type, achieving strong reconstruction quality with highly interpretable and compact cuboid representations.

Free form. We additionally compare against free-form decomposition methods: BAE [CYF*19], RIM [NLXZ22], and DAE [CCZZ24]. These methods generate implicit or deformable part representations, offering high expressivity but at the cost of interpretability and compactness. As shown in Table 1, DAE achieves the best Chamfer distance and IoU among free-form methods, particularly excelling on the plane category (IoU: 59.8%). However, while methods like BAE [CYF*19] and RIM [NLXZ22] demonstrate remarkable capabilities in part decomposition and semantic segmentation of 3D shapes, providing a valid and often interpretable structural breakdown, they diverge from the core objective of shape abstraction. Their reliance on generating highly detailed part sets, often characterized by an excessive number of vertices and faces, pushes their output closer to a mere decomposition or dense reconstruction rather than a simplified, abstract representa-

	mAcc%↑			mIoU%↑		
	plane	**chair**	**table**	**plane**	**chair**	**table**
HCA	61.0	63.3	52.2	41.4	48.4	36.3
CAS	84.5	**91.3**	92.6	**74.9**	**84.8**	85.9
DPF_{PPM}	78.9	83.4	86.6	62.9	74.7	76.5
Ours	**85.5**	82.0	**93.6**	66.3	73.7	**87.1**

Table 2: *Quantitative evaluation of co-segmentation performance on the test set. We report mean per-label Accuracy (mAcc) and Intersection over Union (mIoU). Best highlighted in bold.*

tion (see Figure 3). True shape abstraction, by definition, aims for a parsimonious representation that captures essential geometric and topological features while significantly reducing complexity. In this regard, RIM and BAE, despite their merits in structured shape reconstruction, do not fundamentally achieve the simplification inherent to an abstract shape understanding.

It is important to note that direct numerical comparison across primitive types requires careful interpretation. Free-form and superquadric methods inherently have higher expressivity per primitive, while cuboids offer maximum interpretability and compactness. Furthermore, some methods operate on voxelized input instead of point clouds, and we evaluate all methods against high-resolution ground truth shapes. Despite these differences, our method achieves the highest IoU among all cuboid methods and remains competitive with more expressive representations.

4.2. Shape Co-Segmentation

To demonstrate the structural consistency of the generated abstractions with semantic annotations we evaluate our method on the co-segmentation task defined in [CTZ20]. We compute the mean per-label Accuracy (mAcc) and Intersection over Union (mIoU) as the evaluation criterion, using 4 parts for plane (body, tail, wing, engine), 4 parts for chair (back, seat, leg, arm) and 2 parts for table (top, support). To obtain a semantic mapping from class labels to primitive indices, we use all annotated shapes from the training set and assign each primitive to a label via a voting scheme. Next, we propagate the primitive class labels to point clouds of the test set via nearest neighbor and evaluate the segmentation quality. Results are reported in Table 2. Given that all presented metrics are computed by us, they may exhibit slight variations compared to previous literature as a consequence of our random point cloud sampling procedure.

Our method performs best on the table, second best on the plane and third on the chair category. [YC21] model their abstraction through a segmentation module and use a higher number of cuboids, allowing them to segment the shape with finer details. The same holds true for [SZYC23]. This is most obvious in comparison to [SZTL19] who use the least number of primitives resulting in poor performance on this benchmark. Even though we use a lower number of primitives than [YC21] our scores are similar. The reported metrics demonstrate that our model is able to recover geometric parts aligned with semantics of the shapes.

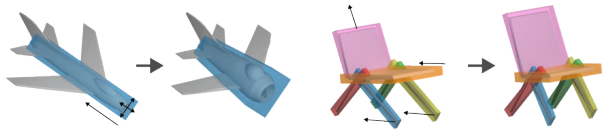


Figure 4: *We apply our cuboid primitives together with cage-based deformation to edit meshes. Black arrows indicate directions of cage deformation.*

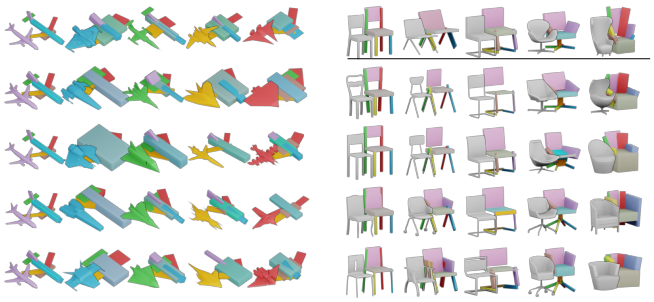


Figure 5: *Left: k-means clustering ($k = 5$) on top of our cuboid parameters $[r,t,s,\gamma]$. Each column represents a cluster. Right: Retrieval of the nearest neighbors from a collection of shapes. Top row shows the query cuboids.*

4.3. Shape editing

Drawing upon our novel cuboid abstraction method, which efficiently decomposes complex 3D shapes into a set of constituent cuboid primitives, we establish a sparse yet effective cage for deformation. These automatically derived cuboids serve as the handles of our deformation system. By manipulating the vertices of these cuboid primitives – translating, rotating, or scaling them as demonstrated in Figure 4 where an airplane fuselage is scaled or chair components are repositioned – we induce corresponding deformations on the underlying high-resolution 3D mesh. This cage-based approach leverages the structural understanding provided by our cuboid abstraction, allowing for intuitive and localized or global control over the shape's geometry while preserving its essential features, making complex edits significantly more manageable than direct mesh manipulation.

4.4. Shape Clustering and Retrieval

We apply the output of our network to the downstream task of shape clustering and retrieval. First, we concatenate the cuboid parameters $[r,t,s]$ into a single vector and multiply them with their binary existence probability γ, to zero out non-existent parts. Next, we perform k-means clustering ($k = 5$) and compute the pairwise Euclidean distances of these vectors. The results are presented in Figure 5. Based solely on the cuboid abstraction we can discover distinct plane subcategories like passenger airplanes (purple), spacecrafts (blue) or jet fighters (green). The retrieval results show that our abstract representation can be used to search for structurally similar shapes in collections of 3D models.

Figure 6: *Using our cuboid abstraction to decompose shapes into its geometric parts enables us to perform partial symmetry detection. Colors indicate groups of symmetric parts.*

4.5. Partial Symmetry Detection

Furthermore, we apply the predicted cuboid abstraction to the task of partial symmetry detection. Given a geometrically motivated abstraction of 3D shapes we are able to decompose them into their parts. Results are visualized in Figure 6 with colors indicating groups of symmetric parts. We are able to automatically extract meaningful partial symmetries based on our cuboid shape abstraction. Please refer to supplementary materials for implementation details.

4.6. Ablation Study and Analysis

We perform an ablation study of our architecture (Table 3) without primitive merging. We present the results as examples on the airplane and chair class. We train each configuration five times and report the best run in terms of Chamfer distance (CD).

First, we evaluate the performance using different abstraction loss weights λ_{abs}. We observe that setting a larger λ_{abs} leads to degradation of reconstruction quality as the network puts too much emphasis on the abstraction and not enough on the reconstruction of the shape at the beginning of the training leading to suboptimal starting configuration. In contrast, setting a smaller λ_{abs} leads to an improved reconstruction quality at the cost of compactness. The network focuses mainly on minimizing the reconstruction error using more primitives.

Second, we use only surface $\mathcal{L}_{surf}$ or only volume loss $\mathcal{L}_{vol}$ as reconstruction loss $\mathcal{L}_{rec}$. As shown in Table 3, leaving out $\mathcal{L}_{vol}$ results in a significant performance drop, particularly in terms of IoU which falls to 0.0% for planes. In this configuration, the network minimizes the surface-to-point Chamfer distance by flattening the initially excessive number of cuboids into thin, two-dimensional sheets or lines. While these degenerated cuboids can approximate the shape's surface, they fail to represent its three-dimensional volume. In most cases the network does not recover from this state and eventually diverges (Figure 7). Contrary, leaving out $\mathcal{L}_{surf}$ does not lead to degenerated or failed training, but the cuboids reconstruct the surface less accurately. These findings confirm that a combination of surface reconstruction and volume preservation is essential for capturing the solid nature of the abstracted parts.

Figure 7: *The cuboid primitives degrade into thin lines when $\mathcal{L}_{vol}$ is omitted from the training.*

Third, we evaluate different cuboid target functions

$$\Gamma(\phi) = \begin{cases} \kappa_{max} - (\kappa_{max} - \kappa_{min}) \cdot \phi, & \text{if linear} \\ \kappa_{max} \cdot (\phi + 1)^{-4}, & \text{if exp} \\ \kappa_{min} + (\kappa_{max} - \kappa_{min}) \cdot \frac{1}{2}(\cos(\pi\phi) + 1), & \text{if cosine} \\ \kappa_{min} + (\kappa_{max} - \kappa_{min}) \cdot (\cos(\frac{\pi}{2} + \frac{\pi}{2}\phi) + 1). & \text{if half-cos} \end{cases} \tag{11}$$

We observe that the linear and the exponential function do not adhere to the desired number of primitives at the end of the training. Furthermore, we observe minor oscillations in the set of selected active primitives at the end of training, and the choice of $\Gamma(\phi)$ directly impacts the stability of this process. In the case of the linear function, the constant reduction rate leaves very limited time for the network to resolve these oscillations in the existence probabilities γ of cuboids near the detection threshold (0.5). This leads to persistent "flickering" in the final selection and suboptimal abstractions. In contrast, the cosine scheme slows the rate of primitive reduction as it approaches κ_{min}, providing a crucial stabilization period. This allows training oscillations to settle into a stable state where the network consistently identifies the most geometrically relevant primitives, leading to the best performance in terms of CD and IoU.

Fourth, we perform an ablation of our architecture and the cuboid latents l_c. In the first case we directly concatenate l_c with the local features. In the second case we do not use l_c at all and define our cuboid features f_c as all output tokens of the ViT. In both cases we consequently use only one large ViT with twice the number of layers. We observe that the training without using l_c explicitly diverges. In the case of $[p_c, f_l]$ only 20% of the runs converged, in these cases the results were similar to our baseline. We argue that our architecture design helps the network to learn a global description of the shape first. Using learnable cuboid latents l_c allows the network to learn a template of the whole collection which it modifies based on the input conditions. This enables us to learn a structurally consistent and geometrically meaningful abstraction across the whole dataset.

Fifth, we conducted an ablation study where training commenced directly with the target number of cuboids, setting $\kappa_{max} = \kappa_{min}$. In this configuration, our model operates with a fixed primitive budget from the very beginning, entirely bypassing our proposed fine-to-coarse training scheme. This effectively reduces the learning process to a singular optimization task of primitive fitting across a collection of shapes, devoid of hierarchical guidance. While quantitative metrics indicate that the model can still generate meaningful cuboid abstractions, their overall quality consistently falls short compared to those produced by our full approach. We attribute this performance degradation to the absence of progressive refinement. Without the ability to first establish a refined

	plane			chair		
	Num↓	**CD↓**	**IoU%↑**	**Num↓**	**CD↓**	**IoU%↑**
$\lambda_{abs} = 1e\text{-}2$	7.00	0.026	58.6	10.00	0.037	58.1
$\lambda_{abs} = 1e\text{-}4$	10.00	0.023	62.6	9.00	0.040	44.7
$\mathcal{L}_{rec} = \mathcal{L}_{surf}$	17.3	0.058	0.0	10.00	0.030	52.5
$\mathcal{L}_{rec} = \mathcal{L}_{vol}$	7.00	0.032	51.3	11.40	0.036	57.8
$\Gamma(\phi) = \text{linear}$	8.64	0.029	52.1	15.71	0.035	60.3
$\Gamma(\phi) = \exp$	7.98	0.026	58.9	10.04	0.039	57.9
$\Gamma(\phi) = \text{cosine}$	7.00	0.025	60.3	13.77	0.036	58.5
w/o l_c	37.32	0.028	28.8	—	—	—
$[p_c, f_l]$	7.00	0.026	57.5	10.40	0.038	57.9
$\kappa_{max} = \kappa_{min}$	6.00	0.026	57.9	10.00	0.039	46.0
w/o PP	7.00	0.024	61.1	10.00	0.035	58.8

Table 3: *Ablation study using different configuration of abstraction loss weight λ_{abs}, reconstruction loss $\mathcal{L}_{rec}$, cuboid target function $\Gamma(\phi)$, and architecture. The baseline model (w/o PP) uses $\lambda_{abs} = 1e\text{-}3$, $\mathcal{L}_{rec} = \mathcal{L}_{surf} + \mathcal{L}_{vol}$, $\Gamma(\phi) = \text{half-cos}$ and $[l_c, f_g]$. All metrics evaluated on the plane and chair class without the post-processing step.*

θ_{merge}	plane			chair		
	Num↓	**CD↓**	**IoU%↑**	**Num↓**	**CD↓**	**IoU%↑**
1.0	5.33	0.031	48.9	7.03	0.040	52.3
1.1	5.75	0.028	53.5	7.49	0.038	53.7
1.2	6.03	0.026	56.0	7.79	0.038	54.5
1.3	6.22	0.026	58.1	8.07	0.037	55.1
1.4	6.35	0.025	58.6	8.37	0.036	54.9
1.5	6.54	0.025	58.8	8.78	0.036	55.5
w/o	7.00	0.024	61.1	10.00	0.035	58.8

Table 4: *Comparison of reconstruction quality using different merge threshold θ_{merge} values. We also report values without using post processing (w/o). All metrics evaluated on the plane and chair class.*

representation of the global shape structure, the model is compelled to allocate primitives to minimize immediate reconstruction error. This often leads to a sub-optimal distribution of cuboids, where primitives are prematurely consumed by prominent global features, hindering the model's capacity to efficiently learn comprehensive, fine-grained details. We provide more details and qualitative samples in the supplementary material (Table 6).

Last, we analyze the influence of post-processing on the cuboid reconstruction quality. We report metrics in Table 4 evaluated on the plane and chair class using varying merge thresholds θ_{merge} and no post-processing at all w/o. Using a smaller value for θ_{merge} leads to merging of more cuboids. Using less primitives results naturally in a less accurate reconstruction. Especially, when we merge only highly overlapping cuboids (large θ_{merge}) the reconstruction quality degrades only marginally. This can be attributed to structurally redundant cuboids. Overlapping cuboids can more easily satisfy the reconstruction loss $\mathcal{L}_{rec}$ when approximating non-square geometry, but they are structurally redundant from an abstraction point of view and should be merged.

4.7. Limitations and Future Work

While our method currently assumes axis-aligned input shapes, it demonstrates reasonable robustness to minor misalignments. Future work could incorporate an auxiliary network to predict the global canonical orientation. This allows the network to learn the position and shape of common parts, but does not ensure that the network is aware of the relationships between them. Introducing rotational-invariance to the prediction of primitives would enable the network to be more approachable in real-world scenarios.

Another line of work could extend our method from cuboids to super-quadrics by predicting two additional parameters or deformable templates by extending the decoder network further. It is also possible to exchange the decoder network to predict implicit functions instead of cuboids. Although hybrid-approaches combining learning-based and classical shape analysis techniques are a valid choice, formulating primitive merging in terms of a loss function would enable to predict the abstraction in a single inference step without additional post-processing. Furthermore, it would be highly interesting to see an extension of this work towards 3D shape generative models conditioned on automatically predicted cuboid primitives instead of manually annotated ones.

5. Conclusion

In this paper we propose a novel fine-to-coarse self-supervised method for cuboid shape abstraction. Our loss formulation allows to approximate a collection of shapes with an extensive number of primitives first and continually reduce their count to form an abstraction second. We show that preserving distinct geometric details is easier than learning them using a limited number of primitives from the beginning. We compare with previous work on common benchmarks to demonstrate state-of-the-art performance. Furthermore, we apply our cuboid abstraction to downstream task, like shape editing, clustering, retrieval or partial symmetry detection.

References

[Bie87] BIEDERMAN I.: Recognition-by-components: a theory of human image understanding. *Psychological review 94* (1987). 2

[BM92] BESL P. J., MCKAY N. D.: Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures* (1992), vol. 1611, Spie. 14

[BRPMB17] BOGO F., ROMERO J., PONS-MOLL G., BLACK M. J.: Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017). 5

[CCZZ24] CHEN Z., CHEN Q., ZHOU H., ZHANG H.: Dae-net: Deforming auto-encoder for fine-grained shape co-segmentation. In *ACM SIGGRAPH 2024 Conference Papers* (2024). 3, 6

[CFG*15] CHANG A. X., FUNKHOUSER T., GUIBAS L., HANRAHAN P., HUANG Q., LI Z., SAVARESE S., SAVVA M., SONG S., SU H., XIAO J., YI L., YU F.: *ShapeNet: An Information-Rich 3D Model Repository*. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2, 5

[CJB03] CHEVALIER L., JAILLET F., BASKURT A.: Segmentation and superquadric modeling of 3d objects. 2

[CTZ20] CHEN Z., TAGLIASACCHI A., ZHANG H.: Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 7

[CYF*19] CHEN Z., YIN K., FISHER M., CHAUDHURI S., ZHANG H.: Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019). 2, 3, 6

[Dos20] DOSOVITSKIY A.: An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020). 3

[Fer18] FERREIRA P.: Sampling superquadric point clouds with normals. *arXiv preprint arXiv:1802.05176* (2018). 3

[FSG*25] FEDELE E., SUN B., GUIBAS L., POLLEFEYS M., ENGELMANN F.: SuperDec: 3D Scene Decomposition with Superquadric Primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2025). 3, 6

[GYW*19] GAO L., YANG J., WU T., YUAN Y.-J., FU H., LAI Y.-K., ZHANG H.: Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG) 38*, 6 (2019). 2

[LA04] LIEN J.-M., AMATO N. M.: Approximate convex decomposition. In *Proceedings of the twentieth annual symposium on Computational geometry* (2004). 2

[LLL*20] LIN C., LIU L., LI C., KOBBELT L., WANG B., XIN S., WANG W.: Seg-mat: 3d shape segmentation using medial axis transform. *IEEE transactions on visualization and computer graphics 28*, 6 (2020). 2

[LPG24] LI S., PASCHALIDOU D., GUIBAS L.: Pasta: Controllable part-aware shape generation with autoregressive transformers. *arXiv preprint arXiv:2407.13677* (2024). 2

[LWRC22] LIU W., WU Y., RUAN S., CHIRIKJIAN G. S.: Robust and accurate superquadric recovery: A probabilistic approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022). 2, 6

[LWTY24] LI J., WANG H., TAN J., YUAN J.: Shared latent membership enables joint shape abstraction and segmentation with deformable superquadrics. *IEEE Transactions on Image Processing* (2024). 3, 4

[LXC*17] LI J., XU K., CHAUDHURI S., YUMER E., ZHANG H., GUIBAS L.: Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG) 36*, 4 (2017). 2

[MGY*19] MO K., GUERRERO P., YI L., SU H., WONKA P., MITRA N. J., GUIBAS L. J.: Structurenet: hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG) 38*, 6 (2019). 2

[MWZ*14] MITRA N. J., WAND M., ZHANG H., COHEN-OR D., KIM V., HUANG Q.-X.: Structure-aware shape processing. In *ACM SIGGRAPH 2014 Courses*. 2014. 2

[NLXZ22] NIU C., LI M., XU K., ZHANG H.: Rim-net: Recursive implicit fields for unsupervised learning of hierarchical shape structures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022). 3, 6

[PF95] PILU M., FISHER R. B.: Equal-distance sampling of superellipse models. 3

[PGG20] PASCHALIDOU D., GOOL L. V., GEIGER A.: Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 3

[PKGF21] PASCHALIDOU D., KATHAROPOULOS A., GEIGER A., FIDLER S.: Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 3

[PUG19] PASCHALIDOU D., ULUSOY A. O., GEIGER A.: Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019). 2, 3, 4, 6

[QLP*22] QIAN G., LI Y., PENG H., MAI J., HAMMOUD H., ELHOSEINY M., GHANEM B.: Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems 35* (2022). 2

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017). 2

[RMG18] RODRIGUES R. S., MORGADO J. F., GOMES A. J.: Part-based mesh segmentation: a survey. In *Computer Graphics Forum* (2018), vol. 37. 2

[SB90] SOLINA F., BAJCSY R.: Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE transactions on pattern analysis and machine intelligence 12*, 2 (1990). 2

[SZTL19] SUN C.-Y., ZOU Q.-F., TONG X., LIU Y.: Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Transactions on Graphics (TOG) 38*, 6 (2019). 1, 2, 3, 5, 6, 7, 11, 12, 14

[SZYC23] SHUAI Q., ZHANG C., YANG K., CHEN X.: Dpf-net: Combining explicit shape priors in deformable primitive field for unsupervised structural reconstruction of 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023). 3, 5, 6, 7, 11, 12

[TSG*17] TULSIANI S., SU H., GUIBAS L. J., EFROS A. A., MALIK J.: Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017). 2, 3

[WZX*20] 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* (2020). 2

[YC21] YANG K., CHEN X.: Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. *ACM Transactions On Graphics (TOG) 40*, 4 (2021). 2, 3, 4, 5, 6, 7, 11, 12, 14

[YHZ*25] YE J., HE Y., ZHOU Y., ZHU Y., XIAO K., LIU Y.-J., YANG W., HAN X.: Primitiveanything: Human-crafted 3d primitive assembly generation with auto-regressive transformer, 2025. 2

[YKC*16] 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)* (2016). 2

[ZNW22] ZHANG B., NIESSNER M., WONKA P.: 3dilg: Irregular latent grids for 3d generative modeling. *Advances in Neural Information Processing Systems 35* (2022). 3

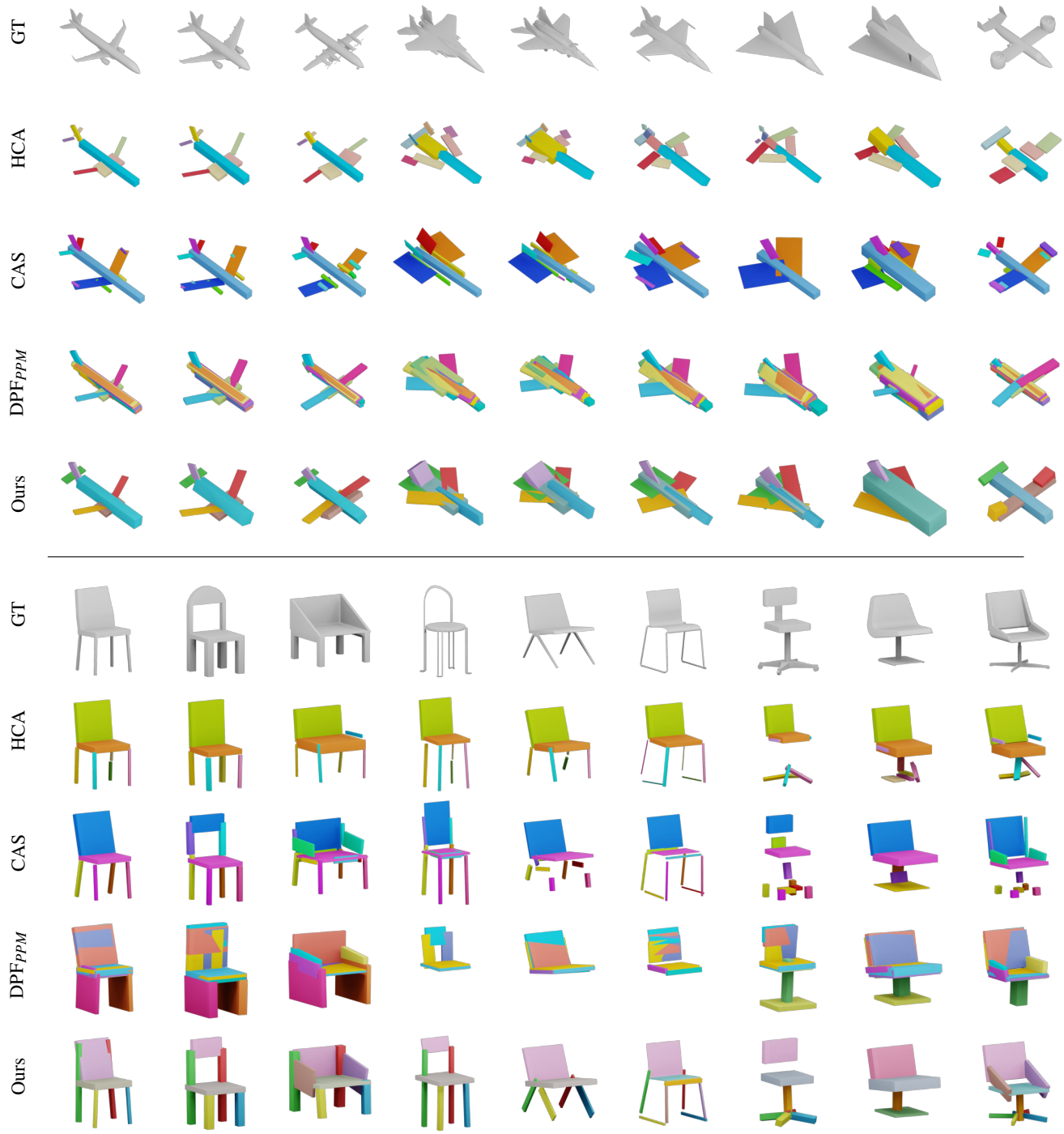


Figure 8: *Qualitative samples of the plane and chair class. We visualize the ground truth (GT) shape and the cuboid abstraction of [SZTL19] (HCA), [YC21] (CAS), [SZYC23] (DPF$_{PPM}$) and Ours. Note, our abstraction preserves the geometry of the ground truth shape more closely using less cuboid primitives. Colors indicate distinc parts per category.*

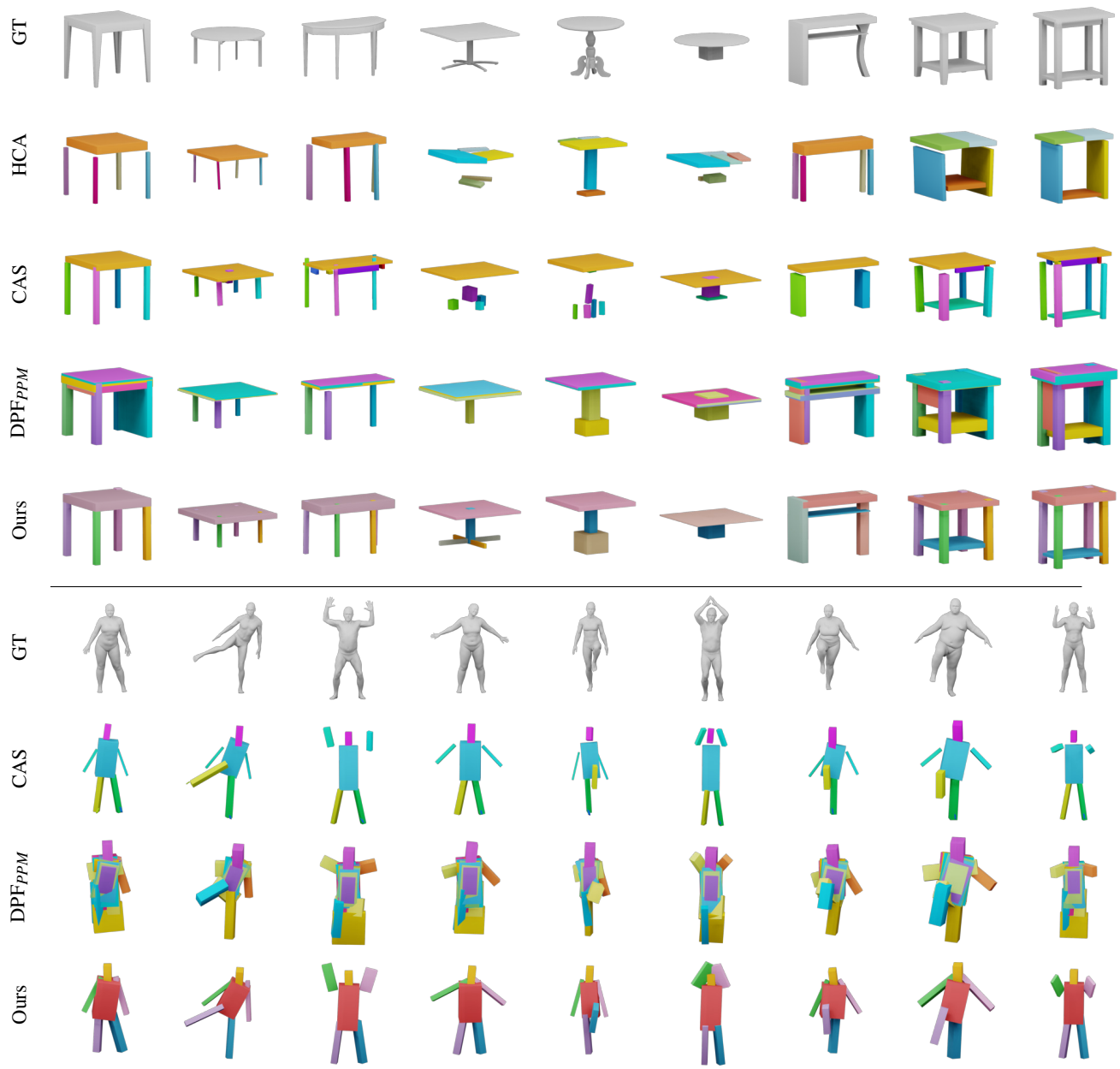


Figure 9: *Qualitative samples of the table and human class. We visualize the ground truth (GT) shape and the cuboid abstraction of [SZTL19] (HCA), [YC21] (CAS), [SZYC23] (DPF$_{PPM}$) and Ours. Note, our abstraction preserves the geometry of the ground truth shape more closely using less cuboid primitives. Colors indicate distinc parts per category.*

Appendix A: Technical Details and Additional Evaluations

We release our code at: https://github.com/GregorKobsik/Fine-to-Coarse-Cuboid-Abstraction.

Architecture

In the following section we outline the architecture in more detail.

Point cloud encoder. The first component of our network is a point cloud encoder. It can be viewed in terms of a sophisticated token embedding for point cloud data. First, we select a subset of N points using furthest point sampling ($N = 128$). For each selected point, we query the K nearest neighbors ($K = 32$) and compute their distance vectors with respect to the input point. These relative position vectors are projected onto a basis, and the resulting embeddings are expanded using sinusoidal functions before being concatenated with the original relative features. Next, a local neural network is applied to these enhanced edge features to learn local patterns. We use a single linear layer with weight normalization and *ReLU* activation. The point cloud features are transformed to a dimension of 256. Subsequently, a max-operation aggregates the features at the destination nodes by taking the maximum feature vector from all incoming edges. Finally, a second neural network processes these aggregated features. It is another single linear layer with weight normalization and *ReLU* activation, that reduces the dimensionality of the features to 128. The output of this component represents the local shape features $f_l \in \mathbb{R}^{128\times128}$ at the downsampled point locations, effectively learning and aggregating local geometric information from the input point cloud.

Token encoder. The second component of our network is a token encoder, implemented as a Vision Transformer (ViT) to enable the integration of information across the previously established local feature representations. This transformer architecture consists of 6 cascaded processing blocks. Within each block, a multi-head self-attention mechanism, utilizing 4 attention heads, is first applied to model global relationships and dependencies within the input sequence. Following the attention mechanism, a standard multi-layer perceptron with a *GELU* non-linearity further refines the feature representations. To ensure stable training dynamics, a pre-layer normalization is applied to the features at the input of each block. Furthermore, we integrate drop path layers, implementing stochastic depth, where individual blocks are probabilistically skipped during training. The dropout probability is linearly scaled with the depth of the transformer, ranging from 0 to 0.1, to improve the model's generalization capabilities. Consequently, this token encoder transforms the initial local shape features f_l into a set of global shape features $f_g \in \mathbb{R}^{128\times128}$, where each token now encapsulates a richer, more globally informed representation of the input.

Token decoder. The third component of our architecture is a token decoder, instantiated as a Vision Transformer (ViT) with an identical structural configuration to the token encoder detailed previously. A key distinction at this stage is the introduction of learnable cuboid latents, denoted as $l_c \in \mathbb{R}^{128\times128}$. These latents are concatenated with the previously computed global shape features, yielding a combined feature tensor $[f_g, l_c] \in \mathbb{R}^{256\times128}$, which serves as the input to the token decoder. Following processing by the transformer, the resulting output tokens are partitioned. We selectively retain the latter half of these tokens, designated as the abstract cuboid features f_c, for subsequent processing. The discarded initial half of the tokens can be interpreted as an internal memory mechanism within the transformer, facilitating the representation and propagation of global shape information. Conversely, the retained second half of the tokens undergoes transformation within the decoder to yield latent features that abstractly describe the input shape in terms of latent cuboid primitives. It is pertinent to note that, to optimize training efficiency, the number of learnable cuboid latent tokens is incrementally reduced during the training.

Cuboid primitives decoder. The final component of our network is a cuboid primitive decoder, which processes latent tokens from the preceding transformer stage to predict parameters defining the geometry and existence of multiple cuboids. Specifically, it regresses the scale, rotation (represented as quaternions subsequently converted to rotation matrices), translation, and an existence probability for each potential cuboid. The scale, rotation, and translation parameters are each predicted by a distinct linear layer. A sigmoid activation function constrains the predicted scale parameters to the range $[0,1]$. For rotation, the predicted quaternion components are normalized to unit length before being transformed into a 3×3 rotation matrix. Translation parameters are mapped to the range $[-1,1]$ using a *tanh* activation function. In contrast, the existence probability is determined by a two-layer sub-network employing a *LeakyReLU* activation between the linear layers and a final sigmoid activation to output a probability value in $[0,1]$. These predicted parameters facilitate the generation of 3D cuboid geometry by transforming a canonical cube representation. This methodology enables a flexible and learnable abstraction of 3D shapes into a collection of cuboidal primitives, and the system is designed to accommodate a variable number of active cuboids, thereby adapting to diverse scene complexities.

Hyper-Parameters

We train all models on a single RTX 2080Ti with 11GB of VRAM. If not otherwise noted, our models are trained for 1000 epochs with a batch size of 16 using AdamW with a learning rate of $1e$-3 and a cosine annealing scheduler with warm-up for the first 5 epochs. We set $\lambda_{vol} = 1e1$, $\lambda_{surf} = 1e0$ and $\lambda_{abs} = 1e$-3. Our model uses two ViTs with 6 layers and an embedding size of 128. We decided to use a half-cosine scheme for the cuboid target function $\Gamma(\phi)$ with $\kappa_{max} = 128$ and $\kappa_{min} = 7$ to closely constrain the final number of cuboids to previous work. For chairs we use $\kappa_{min} = 10$. The merge threshold θ_{merge} is set to $1.2, 1.4, 1.0$ and 1.0 for the plane, chair, table and human class, respectively.

Merging primitives.

Cuboids that overlap completely or in part can be detected by computing the ratio V^{ratio} of the sum of their volumes and the volume of their bounding box. The ratio $V^{ratio} > 1$ if they are overlapping along an axis, or $V^{ratio} \approx 1$ if they are touching each other along one face. In both cases they can be merged into a single cuboid.

In the following we consider only cuboids with existence $\gamma > 0.5$. To check which cuboids should be merged we first compute the

oriented bounding box BB_{ij} for every existing cuboid pair C_i, C_j. Next, we compute the volume ratio

$$V_{ij}^{ratio} = \frac{V(C_i)+V(C_j)}{V(BB_{ij})} \tag{12}$$

and select the pair with the largest ratio. If the ratio is above a threshold $V_{ij}^{ratio} > \theta_{merge}$, we substitute the parameters of C_i with the parameters of BB_{ij} and set the existence probability $\hat{\gamma}_i = 1, \hat{\gamma}_j = 0$. We repeat this process, until all valid cuboids are merged, also considering the updated C_i as a candidate. Inaccurate alignment of thin cuboids by just a few degrees can lead to relatively large bounding boxes, preventing the merge operation. To counteract this, we estimate the volume ratio V^{ratio} using cuboids with an increased size by a small fraction ϵ_{size}. In this fashion, we transform the initial set of cuboids C into a merged set of cuboids $\hat{C}$ representing our cuboid shape abstraction. We keep track of the ancestors of $\hat{C}$ by using a binary encoding that defines which cuboids of the initial predicted set C were merged together.

Using a binary encoding to track ancestors of merged cuboids introduces novel cuboid indices. In the worst case this can lead to 2^n unique indices, where $n = \kappa_{min}$. Re-indexing these indices by clustering geometrically similar primitives based on the cuboid parameters $[r,t,s,\gamma]$ could improve the performance on the co-segmentation benchmark. We leave this to future work.

Partial Symmetry Detection

We compute the partial symmetry by sampling a surface point cloud X_S of each shape and extract points $X_m \subseteq X_S$ enclosed by each cuboid C_m first. To ensure that each cuboid encloses its corresponding surface points we add a small epsilon ϵ_{size} to their size. Next, we register pairs of point clouds X_i, X_j on top of each other using the iterative closest points algorithm [BM92] to compute a symmetry matrix M^{sym}. We mark $M_{i,j}^{sym}$ as symmetric if the Chamfer distance $\Delta_{i,j}$ after the registration is below a merge threshold $\Delta_{i,j} < \theta_{sym}$. Finally, we extract symmetry groups by computing connected components of the symmetry matrix M^{sym}. We use $\theta_{sym} = 5e\text{-}4$, except for the table class where we set $\theta_{sym} = 1e\text{-}3$.

Inference and Training Times

We report training and inference times in Table 5. Our model can be trained on a single NVIDIA RTX 2080Ti GPU within 19 hours on the airplane class. During inference we can process 428 shapes per second on this consumer GPU. Given an enterprise GPU like the NVIDIA H100 we can process up to 3481 shapes per second.

Additional Qualitative Samples

Please refer to the supplementary materials to view the cuboid abstractions of [SZTL19, YC21] and Ours evaluated on the test set. We provide renderings of all abstractions as images (*.png-files) as well as the original cuboid data (*.ply/*.obj-files). Furthermore, we provide a video which visualizes our fine-to-coarse training process on an exemplary shape. For this we rendered an abstraction after every training epoch and concatenated these images into a video file.

	RTX 2080Ti 11GB	H100 96GB HBM2e
Training ↓	19.14 h	–
Inference ↑	428.2 it/s	3481.6 it/s

Table 5: *Evaluation of the inference and training times on a consumer GPU (RTX 2080Ti) and an enterprise GPU (H100). We report the training time in hours on a single GPU for the airplane class as well as the number of processed shapes per second during inference.*

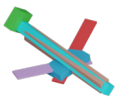
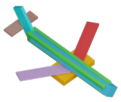
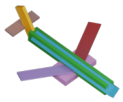
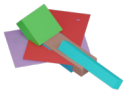
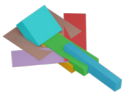
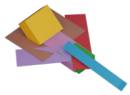
	plane		
	$k_{max} = 7$	$k_{max} = 16$	$k_{max} = 128$
Num↓	6.00	7.00	7.00
CD↓	0.026	0.024	0.024
IoU%↑	57.9	60.5	61.1
			
			
	chair		
	$k_{max} = 10$	$k_{max} = 16$	$k_{max} = 128$
Num↓	10.0	10.00	10.00
CD↓	0.039	0.033	0.035
IoU%↑	46.0	54.9	58.8
			
			

Table 6: *Ablation study on the effect of k_{max} parameter.*