Learning Adaptive Hierarchical Cuboid Abstractions of 3D Shape Collections

CHUN-YU SUN*, Tsinghua University and Microsoft Research Asia
QIAN-FANG ZOU*, University of Science and Technology of China and Microsoft Research Asia
XIN TONG and YANG LIU†, Microsoft Research Asia

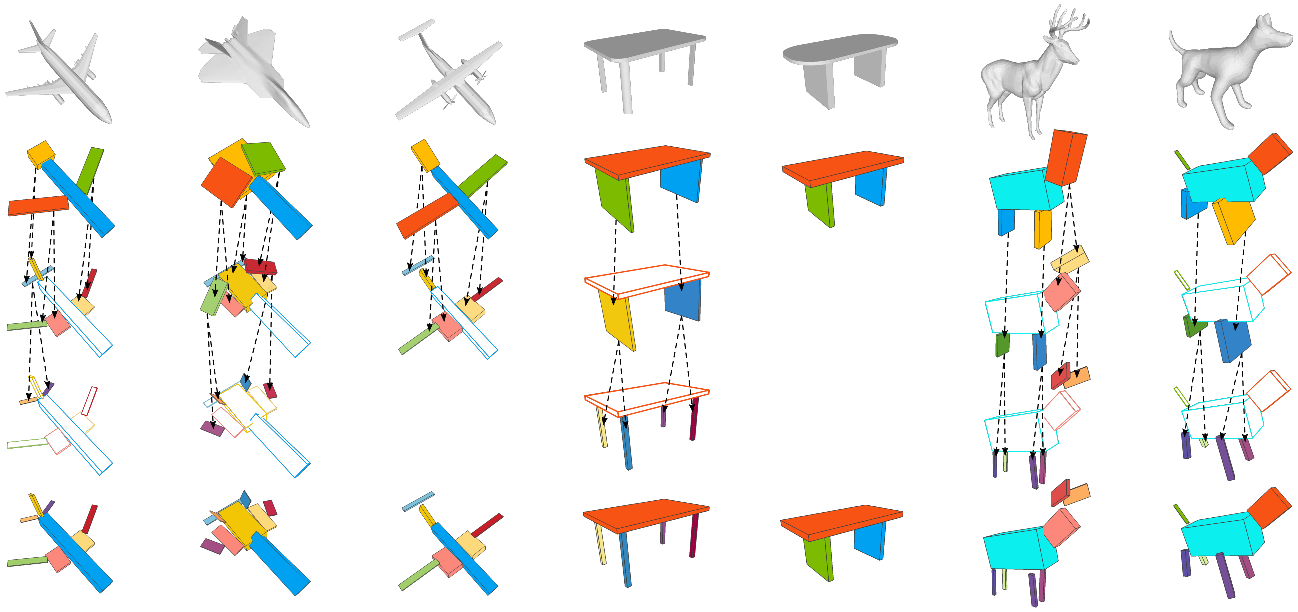

Fig. 1. Adaptive hierarchical cuboid abstractions predicted by our network for 3D shapes from airplane, table and animal categories. Our neural network was learned from a class of 3D shapes in an unsupervised manner. From top to bottom: the input 3D shape, the predicted cuboids at the first, second and third level of the adaptive cuboid abstraction tree, the adaptive cuboid abstraction of the 3D shape. The black lines reveal the hierarchical relationship of cuboids between the adjacent levels which abstracts how the shape part decomposes. Cuboids are color-coded by their corresponding indices defined in the network. From the coded-colors, we can find the consistent correspondence of cuboids between different shapes in a class. At the second and third levels of the adaptive cuboid abstraction tree, we also render the cuboids from the higher level in wire-frame for better illustration.

Abstracting man-made 3D objects as assemblies of primitives, *i.e.*, shape abstraction, is an important task in 3D shape understanding and analysis. In this paper, we propose an unsupervised learning method for automatically constructing compact and expressive shape abstractions of 3D objects in a class. The key idea of our approach is an adaptive hierarchical cuboid representation that abstracts a 3D shape with a set of parametric cuboids adaptively selected from a hierarchical and multi-level cuboid representation shared by all objects in the class. The adaptive hierarchical cuboid abstraction offers a compact representation for modeling the variant shape structures and their coherence at different abstraction levels. Based on this representation, we design a convolutional neural network (CNN) for predicting the parameters of each cuboid in the hierarchical cuboid representation and the adaptive selection mask of cuboids for each input 3D shape. For training the CNN from an unlabeled 3D shape collection, we propose a set of novel loss functions to maximize the approximation quality and compactness of the adaptive hierarchical cuboid abstraction and present a progressive training scheme to refine the cuboid parameters and the cuboid selection mask effectively.

We evaluate the effectiveness of our approach on various 3D shape collections and demonstrate its advantages over the existing cuboid abstraction approach. We also illustrate applications of the resulting adaptive cuboid representations in various shape analysis and manipulation tasks.

CCS Concepts: • **Computing methodologies** → **Shape analysis**.

Additional Key Words and Phrases: shape abstraction, adaptive hierarchical cuboid representation

ACM Reference Format:
Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. 2019. Learning Adaptive Hierarchical Cuboid Abstractions of 3D Shape Collections. *ACM Trans. Graph.* 38, 6, Article 241 (November 2019), 13 pages. https://doi.org/10.1145/3355089.3356529

[*] Both authors were intern students at Microsoft Research Asia.
[†] Corresponding author

0730-0301/2019/11-ART241 $15.00
https://doi.org/10.1145/3355089.3356529

1 INTRODUCTION

> ABSTRACTION IS REAL, PROBABLY MORE REAL THAN NATURE.
>
> — JOSEF ALBERS, 1966

Real world objects, especially man-made objects, often exhibit strong structures and are composed of parts with simple shapes. This abstract shape and structure information can well characterize a 3D object and plays an important role in many graphics and vision applications [Fu et al. 2016; Hu et al. 2018; Mitra et al. 2014]. Shape abstraction represents the shape and structure of 3D objects with assemblies and relations of simple primitive instances. An ideal shape abstraction should be *expressive* and *compact* so that it can represent various 3D shapes with as few as possible primitive instances. Moreover, the layout of the primitive instances should be consistent with the *structure* of the 3D object so that it can easily reveal the common structure and sub-structure shared by different objects. Although humans can easily recognize and create good shape abstractions of 3D objects, developing an automatic 3D shape abstraction algorithm is still a challenging task.

Shape approximation methods [Cohen-Steiner et al. 2004; Li et al. 2011; Lu et al. 2007; Yan et al. 2006; Zou et al. 2017] fit a 3D shape with a set of 3D primitives via expensive non-linear optimization or learned recurrent networks. Although these methods achieve good expressiveness and compactness, they all ignore the structure information of 3D shapes. Shape co-segmentation and structure co-analysis methods [Mitra et al. 2014; Xu et al. 2015] assume a fine level shape abstraction (*i.e.* , semantic parts of over-segmentation patches) of each 3D shape is known and infer the common structure shared by a family of 3D shapes via clustering or deep neural networks trained from annotated 3D shapes. However, obtaining the semantic parts of 3D shapes or their structure-aligned segmentation patches is not easy. Recently, Tulsiani *et al.* [2017] present an unsupervised method for constructing consistent cuboid abstractions for a family of 3D shapes. However, their method tends to generate oversimplified abstractions and cannot handle large structure variations well due to the single level of abstraction representation.

In this paper, we propose an unsupervised method for abstracting a class of unlabeled 3D shapes with various structures. Our method is based on two key observations. First, although 3D objects in a class have various structures, they still share a common structure in a higher abstraction level. Second, the different parts of a 3D shape may exhibit different levels of structure details. Based on these two observations, we propose an adaptive hierarchical cuboid representation for shape abstraction. We model the common structures of 3D shapes in multiple abstraction levels with a hierarchical relationship and abstract each shape by selecting leaf cuboids adaptively from the cuboid hierarchy. The hierarchy of cuboids represents the inclusion relationship between the cuboids at adjacent hierarchical levels. Each level of the cuboid hierarchy models the 3D shapes with a specific number of parametric cuboids, where the layout of cuboids at each level and the layout of adaptive cuboids are consistent for shapes sharing similar structures. The size, position, and orientation of the cuboids are optimized to fit each 3D shape.

The adaptive hierarchical cuboid representation provides a compact and expressive abstraction for 3D shapes with various structures. The cuboid hierarchy well models the coherence of the shape structure in different abstraction levels, while the adaptive cuboid abstraction provides a compact representation for 3D shapes with various geometry and structure. However, generating this abstraction from unlabeled 3D shape collections is non-trivial because both the parameters of cuboids and their adaptive hierarchy are unknown. To overcome this challenge, we start from an initial candidate cuboid hierarchy with a fixed yet redundant number of cuboids in each level and design a convolutional neural network (CNN) that encodes input shapes and decodes the parameters of cuboids in the cuboid hierarchy and the cuboid selection mask for constructing the adaptive cuboid abstraction. A set of loss functions is designed to improve the geometric similarity between the resulting cuboids to the input 3D shape, as well as the hierarchical inclusion relationship of the cuboids, and the compactness and completeness of the selection mask. A novel progressive training scheme is proposed to predict the cuboid hierarchy and the selection mask gradually and results in an adaptive cuboid abstraction with a non-fixed number of cuboids.

With the help of the adaptive hierarchical cuboid representation, our method successfully trains the network by exploiting the intrinsic structural coherence of 3D shapes at different abstraction levels. Also, the hierarchy constraint at different levels also makes our training more robust to shape and structure variations. The resulting network not only predicts a compact abstraction of each 3D shape but also infers the hierarchical structures shared by the whole shape collection.

We evaluate the performance of our method on four man-made shape collections (airplane, chair, desk, four-legged animal) and demonstrate its advantage over other state-of-the-art shape abstraction approaches in terms of abstraction expressiveness and compactness, and illustrate the abstraction consistency found by our method. We also demonstrate the application of the learned adaptive hierarchical cuboid representation on a set of shape understanding and manipulation tasks, including structure classification, abstraction-aware shape deformation, interpolation, and retrieval. As an obvious limitation, our method cannot offer non-geometry semantics to the abstraction because our method considers the geometric approximation quality of the cuboid abstraction only.

2 RELATED WORK

Our work is related to shape approximation methods that are based on surface or volumetric primitives. It is also related to shape co-segmentation and shape structure analysis if we regard the segmented surface patches or semantic parts of 3D shapes as primitives. In this section, we first review the literature related to our work and then briefly discuss the shape abstraction works based on other representations.

Shape abstraction by geometric optimization. Variational shape approximation techniques [Cohen-Steiner et al. 2004; Yan et al. 2006] fit the 3D shape with simple surface patches, where the surface patches are progressively updated and inserted to minimize the approximation error. Bradshaw *et al.* [2004] construct a hierarchical sphere tree by refining a medial axis approximation to a sphere tree progressively. Zou *et al.* [2017] train a recurrent network with 3D shapes and their cuboid representations computed by geometric optimization for inferring cuboids representation from depth images. Wu *et al.* [2018] and Du *et al.* [2018] develop two different strategies for deriving the CSG tree of a 3D shape. All these methods achieve relatively good expressiveness and compactness by optimizing both the number of primitive instances and their parameters. However, they all ignore the structure information of the 3D shapes. Yumer *et al.* [2012] present a spectrum extraction method for generating consistent abstraction of a 3D shape collection. However, their results do not explicitly encode the structure information of 3D shapes. Different from these methods, our method learns an adaptive shape abstraction from a family of 3D shapes. Our shape abstraction results not only deliver a compact approximation for each 3D shape but also characterize the hierarchical structure of the whole shape family.

Shape co-segmentation. A set of methods have been developed for generating consistent segmentation of 3D shape collections. Unsupervised co-segmentation methods start from an over-segmentation of 3D shapes and cluster the segmented patches based on either the features of the segmented patches [Golovinskiy and Funkhouser 2009; Hu et al. 2012; Sidi et al. 2011] or high-level features learned from patch features [Shu et al. 2016]. Huang *et al.* [2011] formulate joint segmentation as an integer quadratic programming problem and segment 3D shapes in a heterogeneous shape database. From a shape abstraction perspective, all these methods assume that a fine-level abstraction of each shape (*i.e.*, over-segmentation) is known and derive a consistent layout of primitive groups for all 3D shapes. Instead, our method infers the abstraction of each shape (parameters of the cuboids) and a consistent layout of cuboids for all 3D shapes simultaneously. Supervised co-segmentation methods [Kalogerakis et al. 2017, 2010; Mo et al. 2019; Yu et al. 2019] train deep neural networks from annotated 3D shape collections for generating consistent segmentation of 3D shapes. Different from these supervised methods that require annotated data for training, our method learns shape abstraction from unlabeled 3D shapes.

Shape structure co-analysis. Several methods assume that the semantic parts of each 3D shape are known and analyze the consistent hierarchical structure of these parts shared by all 3D shapes. Kaick *et al.* [2013] present a method for selecting a representative hierarchical structure of parts for a 3D shape collection from the part hierarchy of each 3D shape. Fish *et al.* [2014] analyze the geometric configurations of a family of co-segmented 3D shapes and represent their structure with geometric distributions. Recently, Yi *et al.* [2017] learn hierarchical shape segmentation and labeling from online repositories, in which the part information and scene graphs are provided. Li *et al.* [2017] also represent the 3D shape with a hierarchical oriented bounding boxes (OBB) tree. They train a supervised recursive neural network to predict the hierarchical tree structure. Their recursive neural network is later used by Niu *et al.* [2018] for predicting structure from image input directly.

Different from these methods that assume that the shape abstraction (*i.e.*, parts) of each object is known, our method derives the primitives for each shape and their abstraction structure from an unlabeled shape collection.

Unsupervised 3D shape abstraction. Tulsiani *et al.* [2017] propose a convolutional neural network for predicting a consistent cuboid representation for a 3D shape collection. The network is trained with unlabeled 3D shapes by measuring the distance between the cuboids and the corresponding 3D shapes. After that, a reinforcement learning model is learned and applied to remove unnecessary cuboids. Because the shape is abstracted by a set of cuboids in a same level, without any hierarchy, their method cannot well address structure variations, often results in over-simplified abstractions as shown in Figure 11(c). In contrast, our adaptive hierarchical cuboid representation can better handle shapes with various structures.

Representations for shape abstraction. Besides the volumetric and surface-based primitives described above, a set of curve based representations, such as curve skeletons [Cornea et al. 2007; Hassouna and Farag 2009], curve networks [De Goes et al. 2011; Gori et al. 2017; Mehra et al. 2009], Eulerian wires [Lira et al. 2018], have also been used for approximating 3D shapes. For man-made objects, planar patches and quadric patches [Cohen-Steiner et al. 2004; Li et al. 2011; Wu and Kobbelt 2005; Yan et al. 2006] are also used for abstracting 3D shapes. We take parametric cuboids as the output of the shape abstraction network because it provides a compact yet expressive representation for our learning task.

3 ADAPTIVE HIERARCHICAL CUBOID REPRESENTATION

In this section, we present our adaptive hierarchical cuboid representations for shape abstraction. We first define *cuboid abstraction* (Section 3.1) and bring the hierarchy into *multi-level cuboid abstraction* (Section 3.2), then we introduce *adaptive cuboid abstraction* built upon the hierarchical cuboid abstraction as the optimal abstraction for the input shape (Section 3.3).

3.1 Cuboid abstraction

Similar to the work of [Tulsiani et al. 2017; Zou et al. 2017], we use a set of cuboids as an abstraction form of 3D shapes. A cuboid c_i is parameterized as a scaled unit cube centered at the original point under a rigid transformation: $h_i = (\mathbf{s}_i, \mathbf{r}_i, \mathbf{t}_i)$. $\mathbf{s}_i \in \mathbb{R}^3$ is the scaling factors along the three coordinate axes, $\mathbf{r}_i \in \mathbb{R}^4$ is the unit quaternion representing the rotation, and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation vector. We say a set of cuboids $\{c_1, \cdots, c_n\} := C$ is an *abstraction* of a 3D shape S if the union of these cuboids is a *good* approximation to S. Here a *good* approximation is measured in the following aspects:

- **Volume coverage**: the volume difference between the shape volume and the union of all the cuboids should be small;
- **Surface coverage**: the surface area difference between S and the surface regions of all the cuboids should be small;
- **Mutual exclusion**: any two of cuboids has the least overlap so that each of them can represent different parts of S.

Besides the above criterions, the abstraction should also present the symmetric relationship between some cuboids if the shape structure possesses some types of symmetry at the part level.

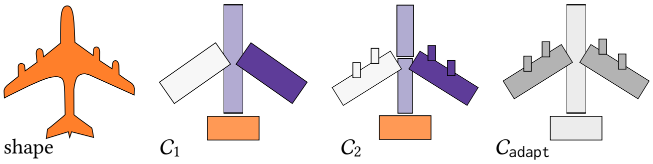

Fig. 2. Illustration of adaptive hierarchical cuboid representation (2-level).

3.2 Hierarchical cuboid abstraction

For a good 3D abstraction, the number of cuboids should vary according to the complexity of the input shape. However, too few cuboids cannot capture small shape details and too many cuboids might yield an over-decomposition. Inspired by the fact that many shapes can be decomposed into meaningful parts in a multi-level and hierarchical way, we introduce *hierarchy* to the cuboid abstraction.

For a 3D shape S, a collection of cuboid abstractions: $C_1(S) = \{c_1^{(1)}, \cdots, c_{n_1}^{(1)}\}, C_2(S) = \{c_1^{(2)}, \cdots, c_{n_2}^{(2)}\}, \cdots, C_K(S) = \{c_1^{(K)}, \cdots, c_{n_K}^{(K)}\}$, $0 < n_1 < n_2 < \cdots < n_K$, is called a *hierarchical cuboid abstraction* of S if the following hierarchical relationship exists: for any cuboid $c_l^{(k)} \in C_k$, there exists a unique $c_o^{(k-1)} \in C_{k-1}$ as its parent and $c_l^{(k)}$ can be contained by $c_o^{(k-1)}$ approximately. This relationship is denoted by $c_l^{(k)} \subseteq c_o^{(k-1)}$. A set of evaluation criterions of hierarchical cuboid representation is as follows:

- **Abstraction quality**: each cuboid abstraction C_k is a good approximation to $\mathcal{S}$;
- **Hierarchical coverage**: the part of $\mathcal{S}$ approximated by the children cuboids can be also approximated by their parent cuboid.

With the inclusion relationship between cuboids, the hierarchical cuboid representation induces a tree structure whose root is a virtual node connecting to every cuboid of C_1. C_k is called the *k-th level abstraction* of S.

For two 3D shapes A and B, if the tree structures of their hierarchical cuboid representations are the same, we say their hierarchical cuboid abstractions *consistent*; if A has a subtree structure that is same to a subtree of B, we say their hierarchical abstractions are *consistent* in a sub-level. A good hierarchical cuboid representation should offer consistent tree structures to the shapes whose geometric structures are similar.

3.3 Adaptive cuboid abstraction

The cuboids at higher hierarchical levels provide a coarse 3D abstraction and can well capture big parts of the shape while the cuboids at lower levels can better represent small shape parts, but may over-decompose the target shape. A good selection of cuboids from different levels can well approximate the shape while keeping the number of cuboids small. A constructive way to get such a selection is to visit the abstraction tree from top to bottom and prune the subtree if the root cuboid of the subtree can approximate well the part of the shape that its children cuboids approximate. The leaf cuboids on the pruned tree form an *adaptive abstraction* of the input shape, denoted by $C_{\text{adapt}}(S) = \{c_1^\star, \ldots, c_m^\star\}$. Here m is the number of picked cuboids and it characterizes the *compactness* of an adaptive abstraction.

In Figure 2 we illustrate the concept of adaptive cuboid representation in 2D. C_1 and C_2 are two-level hierarchical cuboid abstractions of the input shape. Cuboids in C_2 are colored using the color of their parent cuboids in C_1. C_{adapt} is the adaptive cuboid abstraction and we distinguish the level of cuboids with different colors (cuboids from C_2 are darker). More real examples of 3D abstractions can be found in Figure 1 and Section 5.

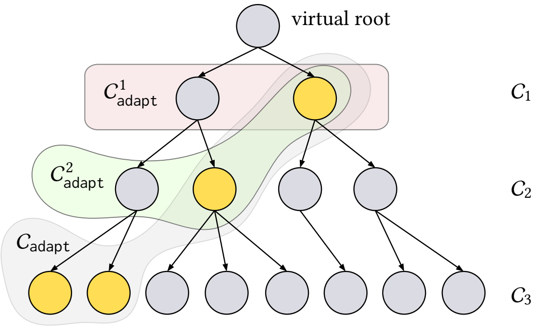

Fig. 3. Illustration of the tree structure of a 3-layer hierarchical cuboid abstraction. For this particular example, C^1_{adapt} is coincident with C_1.

The cuboids in $C_{\text{adapt}}(S)$ and their ancestor nodes form a subtree of the hierarchical cuboid abstraction tree, denoted by Tree(C_{adapt}). The *consistency* defined in Section 3.2 also stands for Tree(C_{adapt}). On Tree(C_{adapt}), we can also build a series of adaptive cuboid abstractions by restricting the depth level of cuboids at a certain level k: $C^k_{\text{adapt}}, k = 1, \ldots, K$, called *k-level adaptive cuboid abstraction*. The construction C^k_{adapt} is simple. By removing all sub-trees whose level is below k from Tree(C_{adapt}), the leaf nodes at this pruned tree form C^k_{adapt}. By construction, we have $C^K_{\text{adapt}} = C_{\text{adapt}}$, and the union of C^k_{adapt}s is Tree(C_{adapt}). In Figure 3, we illustrate the construction of k-level adaptive cuboid abstraction on a three-layer hierarchical abstraction. The yellow nodes represent the cuboids of C_{adapt}, the nodes inside the cyan region form C^2_{adapt} and the node inside the pink region form C^1_{adapt}.

The quality of cuboid abstractions is evaluated under the following conceptual criteria (their realizations are provided as the loss functions in Section 4 for network training).

- **Abstraction quality**: C_{adapt} is a *good* approximation to $\mathcal{S}$.
- **Hierarchical consistency**: $C^k_{\text{adapt}}, k = 1, \ldots, K-1$ are all *good* approximations to $\mathcal{S}$ so that Tree(C_{adapt}) can provide a hierarchical cuboid decomposition to $\mathcal{S}$.
- **Abstraction compactness**: m is as small as possible.

4 ALGORITHM

4.1 Overview

We design a convolutional neural network to learn the adaptive hierarchical cuboid abstractions from a category-specified shape collection without supervision. The network structure (Section 4.2) encodes an input 3D shape and outputs a hierarchical cuboid abstraction and a cuboid selection mask with which a set of cuboids are picked from the hierarchical cuboid abstraction to form the adaptive cuboid abstraction. Two decoder branches — *cuboid prediction module* and *cuboid selection module*, are designed to generate the above outputs. To overcome the difficulty of no supervision on the

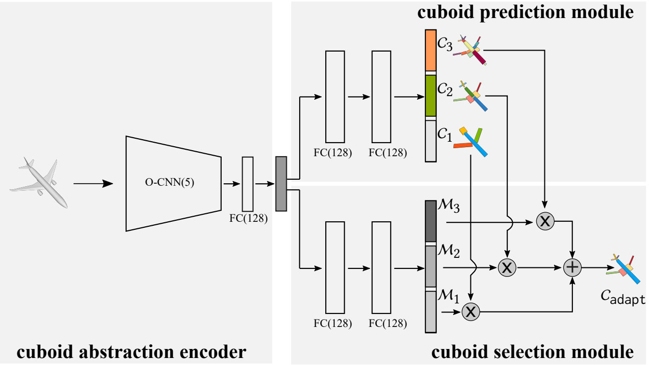

Fig. 4. The three-level cuboid abstraction network structure. Left: the 3D encoder. Upper-right: the cuboid prediction module that infers the parameters for all the cuboids. Lower-right: the selection prediction module that picks the cuboids from each level to form the optimal cuboid abstraction.

cuboid parameters and the selection of cuboids, we design a set of novel loss functions that realize the evaluation criteria of adaptive hierarchical cuboid abstractions (Section 4.3), and we also propose a progressive training scheme to train the network robustly and effectively as follows.

- **Initial training**. We train the encoder and the cuboid prediction module from scratch to generate a multi-level cuboid abstraction with hierarchy.
- **Cuboid selection training**. We fix the encoder and the cuboid prediction module and use the current hierarchical abstraction output to train the cuboid selection module to obtain adaptive cuboid abstractions.
- **Cuboid prediction training**. With the fixed encoder and the selected cuboids, we update the cuboid prediction module to improve the cuboid parameters. A set of loss functions defined on the adaptive abstraction are added to encourage the selected cuboids to be a good approximation to the input shape.
- **Alternative training**. We update the cuboid selection module and the cuboid prediction module alteratively. The alternative training progressively refines the selection mask and the parameters of the selected cuboids. Finally, we fine-tune the whole network with a small learning rate.

4.2 Network structure

Our network has a simple encoder-decoder structure as shown in Figure 4. It includes a 3D encoder and two decoder branches: the cuboid prediction module and the cuboid selection module.

3D encoder. The 3D encoder takes a 3D shape as input and outputs a 128-dimensional vector as the latent code. We employ the octree-based convolution neural network (O-CNN) [Wang et al. 2017] to construct this 3D encoder. O-CNN is a 3D convolution neural network that utilizes the sparsity of 3D surfaces in the volumetric space to achieve high efficiency both on memory consumption and computational cost. The encoder structure of O-CNN(d) is as follows:

$$\text{d-depth octree} \rightarrow \mathrm{U}_d \rightarrow \mathrm{U}_{d-1} \rightarrow \cdots \rightarrow \mathrm{U}_2.$$

U_l is the basic unit "convolution + BN + ReLU + pooling" at the l-th depth octants. The channel number of the feature map for U_l is $2^{\max(1,9-l)}$ and the convolution kernel size is 3. The convolution and pooling operations are designed for the octree structure. In our implementation, we use O-CNN(5). A fully-connected layer with a tanh activation is added after O-CNN(5). Experimentally we found that removing the BN layer helps predicate the cuboid abstraction with a smaller approximation error and more diverse structures, thus we do not use the BN layer in our implementation.

The input surface S is discretized as a set of points with normal information, denoted by $\mathcal{P}(S)$. In our implementation, we sample 5000 points uniformly from the input surface.

Cuboid prediction module. This module decodes the latent code as the cuboid parameters of all the cuboids by two fully-connected layers. The dimension of the output vector is $\sum_{k=1}^{K} 10\, n_k$ and each 10-dimensional subvector corresponds to the parameters of a cuboid. Here n_k, the number of cuboids at each level, is predefined by the user. In our implementation, we set $K = 3$.

Cuboid selection module. This module decodes the latent code as a $\sum_{k=1}^{K} n_k$-dimensional vector in which each entry is the probability value for the corresponding cuboid being selected for the adaptive abstraction, where 1 indicates that a cuboid is *selected*. This decoder is also constructed by two fully-connected layers. By rounding all the entries of this vector, we obtain a cuboid selection mask. With this mask, an adaptive cuboid abstraction can be extracted from the predicted hierarchical cuboid abstraction.

4.3 Loss functions

From the network output, we can construct a series of abstractions:

- Cuboid abstractions at each hierarchical level: $C_k, k = 1, \ldots, K$;
- Adaptive cuboid abstraction and its k-level adaptive cuboid abstractions: $C_{\texttt{adapt}}$ and $C^k_{\texttt{adapt}}, k = 1, \ldots, K-1$.

We design a set of loss functions to evaluate their quality according the evaluation criteria introduced in Section 3. We categorize the loss functions into four types: (1) the approximation losses that encourage the cuboid abstraction to be a good approximation to the input shape; (2) the hierarchical loss that helps build the hierarchical relationship between adjacent abstraction levels; (3) regularization losses for improving the aesthetics of the abstraction and avoiding some degeneracy; (4) losses for the validity and sparsity of the selection mask.

4.3.1 *Geometry approximation losses.*

Volume coverage loss. To faithfully approximate the input shape, the union of cuboids of an abstraction C should contain the input shape S. To be specific, each point of $\mathcal{P}(S)$ is expected to be inside one of the cuboids. We define a distance function $\mathrm{dis}_{\mathrm{in}}(\mathbf{x}, c)$ to measure the coverage of a point $\mathbf{x} \in \mathbb{R}^3$ by a cuboid c:

$$\mathrm{dis}_{\mathrm{in}}(\mathbf{x}, c) = \begin{cases} 0, & \text{if } \mathbf{x} \in \Omega_{\texttt{vol}}(c); \\ \min_{\mathbf{q} \in \Omega_{\texttt{surf}}(c)} \|\mathbf{x} - \mathbf{q}\|, & \texttt{otherwise}, \end{cases}$$

where $\Omega_{\texttt{vol}}(c)$ and $\Omega_{\texttt{surf}}(c)$ denote the interior region of c and the boundary surface of c, respectively.

By summing all the point coverage values, a volume coverage loss for C is defined as follows:

$$L_{\mathtt{vol}}(C) := \frac{\sum_{\mathbf{p}\in\mathcal{P}(S)} \min_{c\in C} \mathrm{dis}_{\mathtt{in}}(\mathbf{p}, c)^2}{\sum_{\mathbf{p}\in\mathcal{P}(S)} 1}.$$

Surface coverage loss. We define a loss function to penalize the surface shape difference between S and the cuboid surface. On the boundary of each cuboid, we uniformly sample n_s points, and define the surface coverage loss as the summation of the squared shortest distances from these samples to S:

$$L_{\mathtt{surf}}(C) := \frac{\sum_{c\in C} \sum_{\mathbf{p}\in\Omega_{\mathtt{surf}}(c)} \mathrm{dis}(\mathbf{p}, S)^2}{\sum_{c\in C} \sum_{\mathbf{p}\in\Omega_{\mathtt{surf}}(c)} 1},$$

where $\mathrm{dis}(\mathbf{p}, S)$ is the shortest Euclidean distance from $\mathbf{p}$ to S. In our implementation, n_s is set to 96 for cuboids at the coarsest level and 26 for others.

Mutex loss. The overlap region between cuboids should be as small as possible so that the abstraction is compact and each cuboid represents a unique part of the shape. Because the intersected volume of two cuboids is not differentiable, we use an alternative way to penalize cuboid intersection.

A distance function $\mathrm{dis}_{\mathrm{out}}(\mathbf{x}, c)$ is defined to measure how far a point is away from a cuboid c:

$$\mathrm{dis}_{\mathrm{out}}(\mathbf{x}, c) = \begin{cases} 0, & \text{if } \mathbf{x} \notin \Omega_{\mathtt{vol}}(c); \\ \min_{\mathbf{q}\in\Omega_{\mathtt{surf}}(c)} \|\mathbf{x} - \mathbf{q}\|, & \mathtt{otherwise.} \end{cases}$$

For any two cuboids c_i and c_j, if they have no overlap, we have $\mathrm{dis}_{\mathrm{out}}(\mathbf{p}, c_i) = 0, \forall \mathbf{p} \in \Omega_{\mathtt{vol}}(c_j)$ and $\mathrm{dis}_{\mathrm{out}}(\mathbf{q}, c_j) = 0, \forall \mathbf{q} \in \Omega_{\mathtt{vol}}(c_i)$. Based on this fact, we sample n_m points uniformly from the interior and boundary region of each cuboid, and penalize the distance from these points to other cuboids by the following mutex loss function:

$$L_{\mathtt{mutex}}(C) := \frac{\sum_{c_i\in C} \sum_{\mathbf{p}\in c_i} \sum_{j\neq i} \mathrm{dis}_{\mathrm{out}}(\mathbf{p}, c_j)^2}{\sum_{c_i\in C} \sum_{\mathbf{p}\in c_i} \sum_{j\neq i} 1},$$

where $\mathbf{q} \in c_i$ is a sample point. n_m is set to 27 in our implementation.

Remark. The sample points from the interior and boundary of cuboids used in the loss functions are parameterized by the linear combinations of corner points of the cuboid, thus these losses are differentiable and friendly to network training.

4.3.2 Hierarchy loss.

Hierarchical coverage loss. For any abstraction $C_k, k < K$, we utilize its cuboid inclusion relationship between C_k and C_{k+1} to design its hierarchical volume coverage loss. We divide the points of $\mathcal{P}(S)$ into n_{k+1} groups, $\mathcal{P}_1(S), \ldots, \mathcal{P}_{n_{k+1}}(S)$, by assigning each point to its closest cuboid in C_{k+1}, where the closeness is defined by $\mathrm{dis}_{\mathtt{in}}$. Here for simplicity, we assume that $\mathcal{P}_l(S)$ is associated with $c_l^{(k+1)}$, $\forall l$. Intuitively, each point group should be contained by a cuboid of C_k if $c_l^{(k+1)}$ is contained by this cuboid. We utilize this inherent hierarchy to define the hierarchical coverage loss for $C_k, k < K$:

$$L_{\mathtt{H}}(C_k) := \frac{\sum_{j=1}^{n_{k+1}} \sum_{\mathbf{p}\in\mathcal{P}_j(S)} \mathrm{dis}_{\mathtt{in}}(\mathbf{p}, c_j^\star)^2}{\sum_{j=1}^{n_{k+1}} \sum_{\mathbf{p}\in\mathcal{P}_j(S)} 1},$$

where $c_j^\star = \arg\min_{c\in C_k} \sum_{\mathbf{p}\in\mathcal{P}_j(S)} \mathrm{dis}_{\mathtt{in}}(\mathbf{p}, c)^2$.

Hierarchical relationship. Since $c_j^\star$ contains the shape region associated with $c_j^{(k+1)}$, we set $c_j^\star$ as the parent cuboid of $c_j^{(k+1)}$. In this way, the hierarchical relationship of $C_1, \ldots, C_K$ can be built.

4.3.3 Regularization losses.

Average area loss. To avoid degenerate cuboids predicted by the network, we penalize the difference between the area of each cuboid from the average area of cuboids. The average loss functions are defined as

$$L_{\mathtt{avearea}}(C) := \frac{\sum_{c\in C} \left(|\Omega_{\mathtt{surf}}(c)| - \frac{\sum_{c\in C} |\Omega_{\mathtt{surf}}(c)|}{\sum_{c\in C} 1} \right)^2}{\sum_{c\in C} 1},$$

where $|\Omega_{\mathtt{surf}}(c)|$ is the surface area of the cuboid c.

Bilateral symmetry loss. Notice that many man-made shapes possess symmetry structures, we design a symmetry loss to encourage the cuboid abstraction to have this property. Currently, we consider bilateral symmetry only and assume that the shapes in the dataset are pre-aligned so that the bilateral symmetry plane is the XY-plane. For a cuboid c, we sample n_b points from its interior and boundary and denote the set of the bilateral symmetry copies of these points by $Q(c)$. If there exists a cuboid c_b (c_b can be c itself) that is the bilateral symmetry copy of c, we have $\mathrm{dis}_{\mathtt{in}}(\mathbf{q}, c_b) = 0, \forall \mathbf{q} \in Q(c)$. Based on this observation, the bilateral symmetry loss is defined as:

$$L_{\mathtt{sym}}(C) = \frac{\sum_{c\in C} \sum_{\mathbf{p}\in Q(c)} \mathrm{dis}_{\mathtt{in}}(\mathbf{p}, c^\star)^2}{\sum_{c\in C} \sum_{\mathbf{p}\in Q(c)} 1},$$

where $c^\star = \arg\min_{c_j\in C} \sum_{\mathbf{p}\in Q(c)} \mathrm{dis}_{\mathtt{in}}(\mathbf{p}, c_j)^2$, and n_b is set to 27.

Cuboid alignment loss. We find that many parts of man-made shapes are nearly axis-aligned like of the armrest of a chair and the wings of an airplane. We add an axis-aligned alignment loss to let the cuboid align with the coordinate axes to improve the aesthetics of the cuboid abstraction.

$$L_{\mathtt{align}}(C) := \frac{\sum_{c\in C} \|\mathbf{r}(c) - \mathbf{r}_I\|^2}{\sum_{c\in C} 1},$$

where $\mathbf{r}(c)$ is the quaternion parameter of c and $\mathbf{r}_I = (1, 0, 0, 0)^T$ corresponds to the identity transformation.

4.3.4 Losses for cuboid selection.

Tree completion loss. The cuboid selection module outputs a set of selection probability values for all the cuboids in the hierarchical cuboid abstraction. We denote $\mathrm{Prob}(c)$ as the selection probability for cuboid c. A simple rounding of all the $\mathrm{Prob}(c)$s yield a set of selected cuboids, however, this selection may violate the tree structure of adaptive cuboid abstraction. We design a tree completion loss to penalize the invalid selection.

$$L_{\mathtt{complete}} := \frac{\sum_{c\in C_K} |p_h(c) - 1|^2}{\sum_{c\in C_K} 1},$$

where $p_h(c)$ is the summation of the selection probability values of c and all its ancestor cuboids (except the virtual root). For a valid adaptive abstraction, $p_h(c), c \in C_k$ should be 1 because there is only one selected cuboid on the path from c to its ancestor at C_1.

Mask sparsity loss. A smaller number of the selected cuboids make the adaptive cuboid abstraction more compact. We define the sparseness loss as

$$L_{\texttt{sparse}} := \frac{\sum_{k=1}^{K}\sum_{c\in C_k}\text{Prob}(c)}{\sum_{k=1}^{K}\sum_{c\in C_k} 1}.$$

Approximation Loss. The selected cuboids should be also a good cuboid abstraction to the input surface. We modify the volume and surface coverage losses to estimate the approximation quality of all the cuboids with the consideration of the probability of each cuboid appearing in the adaptive abstraction:

$$L^{\star}_{\texttt{vol}} := \frac{\sum_{k=1}^{K}\sum_{\mathbf{p}\in\mathcal{P}(S)}\text{Prob}(c^{(k)}(\mathbf{p}))\cdot\min_{c\in C_k}\text{dis}_{\texttt{in}}(\mathbf{p},c)^2}{\sum_{k=1}^{K}\sum_{\mathbf{p}\in\mathcal{P}(S)}\text{Prob}(c^{(k)}(\mathbf{p}))};$$

$$L^{\star}_{\texttt{surf}} := \frac{\sum_{k=1}^{K}\sum_{c\in C_k}\sum_{\mathbf{p}\in\Omega_{\texttt{surf}}(c)}\text{Prob}(c)\cdot\text{dis}(\mathbf{p},S)^2}{\sum_{k=1}^{K}\sum_{c\in C_k}\sum_{\mathbf{p}\in\Omega_{\texttt{surf}}(c)}\text{Prob}(c)},$$

where $c^{(k)}(\mathbf{p}) = \arg\min_{c\in C_k}\text{dis}_{\texttt{in}}(\mathbf{p},c)^2$.

We define the approximation loss by simply summing the above two terms and normalizing the sum by the average coverage losses on the coarsest and finest level, per shape:

$$L_{\texttt{approx}} := \frac{L^{\star}_{\texttt{vol}} + L^{\star}_{\texttt{surf}}}{(L_{\texttt{vol}}(C_1) + L_{\texttt{surf}}(C_1) + L_{\texttt{vol}}(C_K) + L_{\texttt{surf}}(C_K))/2}.$$

This loss measures the approximation quality of the adaptive cuboid abstraction. It is designed with the consideration of the probability of each cuboid appearing in the adaptive abstraction.

4.4 Network training

We train our network using a progressive training scheme as introduced in Section 4.1.

- **Initial training**. We first train the network to generate the hierarchical cuboid abstraction. The loss function $L_{\texttt{pred}}$ of each training shape is a combination of the geometry approximation losses, hierarchical loss and regularization losses:

$$L_{\texttt{pred}} := \sum_{k=1}^{K}(L_{\texttt{vol}}(C_k) + L_{\texttt{surf}}(C_k) + L_{\texttt{mutex}}(C_k)) + \sum_{k=1}^{K-1}L_{\texttt{H}}(C_k) + \sum_{k=1}^{K}\left(\alpha_{\texttt{sy}}L_{\texttt{sym}}(C_k) + \alpha_{\texttt{aa}}L_{\texttt{avearea}}(C_k) + \alpha_{\texttt{al}}L_{\texttt{align}}(C_k)\right).$$

  We set $\alpha_{\texttt{sym}} = 0.1, \alpha_{\texttt{avearea}} = 5, \alpha_{\texttt{align}} = 0.001$ at the beginning of the training. The regularization is useful at the first few epochs to reach a reasonable good cuboid layout. After 500 epochs, $\alpha_{\texttt{sym}}$ and $\alpha_{\texttt{align}}$ reduces by half every 100 epochs.

- **Cuboid selection training**. By fixing the encoder and the cuboid prediction decoder, the hierarchical abstractions of the training dataset are also known. We use these hierarchical abstractions to train the cuboid selection module. The loss function of each training shape is formulated as:

$$L_{\texttt{mask}} := L_{\texttt{complete}} + \lambda_{\texttt{approx}}L_{\texttt{approx}} + \lambda_{\texttt{sparse}}L_{\texttt{sparse}},$$

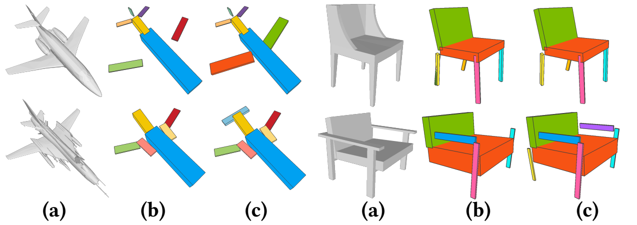

Fig. 5. Adaptive abstraction correction. (a) the input shape; (b) an invalid adaptive abstraction chosen from the worst cases of our test data; (c) the valid adaptive abstraction after correction.

  where $\lambda_{\texttt{approx}}$ and $\lambda_{\texttt{sparse}}$ balance the approximation error and the sparsity of the selection mask. We set $\lambda_{\texttt{approx}} = 0.1$ and $\lambda_{\texttt{sparse}} = 0.4$ in our implementation. After the training, the cuboid selection module can predict the selection mask. However, the network cannot guarantee that $L_{\texttt{complete}}$ vanishes always, so the resulting adaptive abstraction can be invalid. We propose a simple way to fix the issue: (1) we first round the probability values to 0 or 1, and obtain a set of selected cuboids; (2) we examine the cuboids of the hierarchical abstraction from level 1 to $K-1$; if the selection probability of a cuboid is 1, we set the probability of all its descendant cuboids to 0; (2) if for a cuboid $c \in C_k$, $p_h(c^{(K)})$ vanishes, we pick the nearest ancestor cuboid of $c^{(K)}$ which has no descendant cuboid with the selected label, and label it *selected*. This strategy guarantees that the resulting adaptive abstraction is valid. Figure 5 shows some correction results.

- **Cuboid prediction training** With the trained cuboid selection module and the correction operation, the adaptive cuboid abstractions of the training dataset can be computed. We fix the encoder and the cuboid selection decoder and train the cuboid prediction module. Since the adaptive cuboid abstraction is available at the current stage, we add the approximation losses of it and its k-level adaptive cuboid abstractions into the loss function $L_{\texttt{pred}}$ to refine the cuboid parameters, especially for the cuboids on the subtree of $C_{\texttt{adapt}}$.

$$L^{\star}_{\texttt{pred}} := \beta\sum_{k=1}^{K}\left(L_{\texttt{vol}}(C^k_{\texttt{adapt}}) + L_{\texttt{surf}}(C^k_{\texttt{adapt}}) + L_{\texttt{mutex}}(C^k_{\texttt{adapt}})\right) + L_{\texttt{pred}}.$$

  β is initialized as 1.

- **Alternative training**. We iterate the cuboid selection training and the cuboid prediction training. β gets doubled after each round of alternative training. we execute a few rounds of alternative training, and fine-tune the whole network. The loss used in the fine-tuning is: $L^{\star}_{\texttt{pred}} + 0.02\,L_{\texttt{mask}}$.

5 EXPERIMENTS AND APPLICATIONS

5.1 Experiment setup and evaluation metrics

We conduct a series of experiments to evaluate our method. The network is implemented in the TensorFlow framework [Abadi et al. 2015] and the experiments were run on a desktop computer with an Intel Core I7-6850K CPU (3.6GHz) and a GeForce GTX 1080 Ti GPU (11 GB memory). Our implementation is available at https://isunchy.github.io/projects/cuboid_abstraction.html.

Dataset. The network is trained separately on four man-made shape categories: airplane (3640 shapes), chair (5929 shapes), table (7555 shapes) and four-legged animal (122 shapes). These datasets are obtained from ShapeNet [Chang et al. 2015] and [Tulsiani et al. 2017]. We partition the data set into the ratio of 4:1 for training and test. All the shapes are prealigned and scaled within a unit box.

Abstraction and network parameters. We specify the number of cuboids at each level (#C_1, #C_2, #C_3) with respect to the shape complexity: (4, 8, 16) for airplane, (8, 16, 32) for chair, (3, 6, 12) for table and (5, 10, 20) for animal. These ad-hoc cuboid numbers for each category is set experimentally to obtain better visual pleasing and more semantically consistent results. The weights of loss function uses the default values as stated in Section 4.

Network training. The cuboid prediction network at the first stage was trained with the Adam optimizer (lr=0.001) and converged after 1000 epochs. The mask selection network (200 epochs) and the hierarchical cuboid prediction network (200 epochs) were alternatively trained, and they converged after 5 rounds. In the final fine-tuning stage, the learning rate of parameters in the encoder is set to 10^{-5} and the network converged after 200 epochs. The mini-batch size is 32 and the training time for the four categories is 16, 24, 14, 20 hours, respectively.

Evaluation metric. We compute the Chamfer distance between the input shape surface and the boundary surface of the set of cuboids to evaluate the fitting quality of the network output. The metric is denoted by D_{cd} and scaled by 1000 for better viewing.

Visualization of abstraction consistency and hierarchy. The learned cuboid abstraction exhibits a consistent correspondence. The cuboids of different shapes with the same index usually correspond to a similar part of the target shape. For instance, in our learned results for the airplane category, the cuboid with index 2 in C_1 corresponds to the left wing, and the cuboid with index 6 in C_3 represents the right tail wing. This index-dependent consistency is also observed in other learning works [Liu et al. 2018; Tulsiani et al. 2017]. To illustrate this consistency, we specify a unique color to the index of cuboids in each level abstraction. In total, there are $\sum_i^K n_i$ colors. This color coding helps us find consistent correspondence across different shapes in the same category.

5.2 Network evaluation and ablation study

We conducted a set of experiments to validate the efficacy of our training scheme and the essential of proposed loss functions.

The effect of progressive training. Our training scheme can gradually improve the fitting quality and select more appropriate cuboids as the adaptive abstraction. Figure 6 illustrates the predicted results for three shapes in the training dataset at different training stages. The first row shows the cuboids at three different layers after the initial training, and the adaptive cuboid representation obtained by training the mask selection module. The 2nd, 3rd rows show the outputs after the third and the fifth alternative training of the mask selection and the cuboid prediction. The 4th row is the final result after fine-tuning the whole network. We use WP-1, WP-2, WP-3, WP-4 to denote these four watch points. We can clearly see that the alternative training improves the adaptive cuboid selection and cuboid parameters: the tail and back of the airplane are better characterized by cuboids and the selection mask on the tail is gradually updated, the orientation of the cuboid at the chair seat region fits the input better, and the cuboids corresponding to the leg and the desktop of the table are aligned to each other more closely. The metric evaluation reported in Table 1 also shows that the alternative training improves the approximation quality gradually.

Table 1. Quality statistics of the staged training results on the four shape categories. The number in the table is the average $D_{\text{cd}}(C_{\text{adapt}})$ over all the shapes. We separate the statistics results of the training and test set in the form of "training(test)".

Category	WP-1	WP-2	WP-3	WP-4
Airplane	10.06(18.84)	9.52(18.87)	9.35(18.70)	**9.32(18.68)**
Chair	22.49(26.68)	20.76(25.75)	20.61(25.70)	**18.92(24.79)**
Table	19.84(28.13)	19.37(27.81)	19.25(27.71)	**18.30(27.58)**
Animal	19.90(20.10)	19.12(20.03)	19.03(19.88)	**18.85(19.06)**

Table 2. Quality statistics of results from the network using different loss combinations. We present the average $D_{\text{cd}}(C_{\text{adapt}})$ of the training and test sets as"training(test)".

Category	Net I	Net II	Net III	Net IV	Net V
Airplane	**9.32(18.68)**	10.18(18.73)	11.11(19.03)	11.60(19.38)	10.98(18.81)
Chair	**18.92(24.79)**	19.15(24.85)	21.44(26.31)	19.58(25.65)	19.72(25.37)
Table	**18.30(27.58)**	20.32(28.51)	19.96(28.10)	22.66(30.06)	21.56(29.80)
Animal	**18.85(19.06)**	20.76(21.18)	18.91(19.08)	19.07(19.11)	19.12(19.15)

Ablation study of loss terms. Every loss plays its own role in training the cuboid prediction network. We test a set of networks to prove the essential of these loss functions.

- Net I: use all the loss functions;
- Net II: remove the hierarchical coverage loss;
- Net III: remove the mutex loss;
- Net IV: remove the average area loss;
- Net V: remove the alignment loss and bilateral symmetry loss.

We train these networks on the airplane category and compare their result quality. From the quality statistics (Table 2) and the visualization of the cuboid abstraction (Figure 7), We found that (1) with all the loss functions, the network yields the adaptive abstractions with the smallest error than other networks; (2) without the hierarchical coverage loss, the results have a worse approximation error and more inconsistent hierarchy than (1); (3) without the mutex loss, the predicted cuboids overlap each other and the adaptive abstraction contains more cuboids; (4) Without the average area loss, many degenerate cuboids appear in C_3, and they are easy to be picked into the adaptive abstraction and damage the abstraction expressiveness; (5) Without the regularization of the alignment and bilateral symmetry losses, the orientation and layout of cuboids are unpleasant and the structure of the shape is not well captured by the cuboids.

Bottom-up hierarchical training. For predicting hierarchical structures, a common practice is to train the network in a top-down or bottom-up manner, where the network is trained stage by stage and

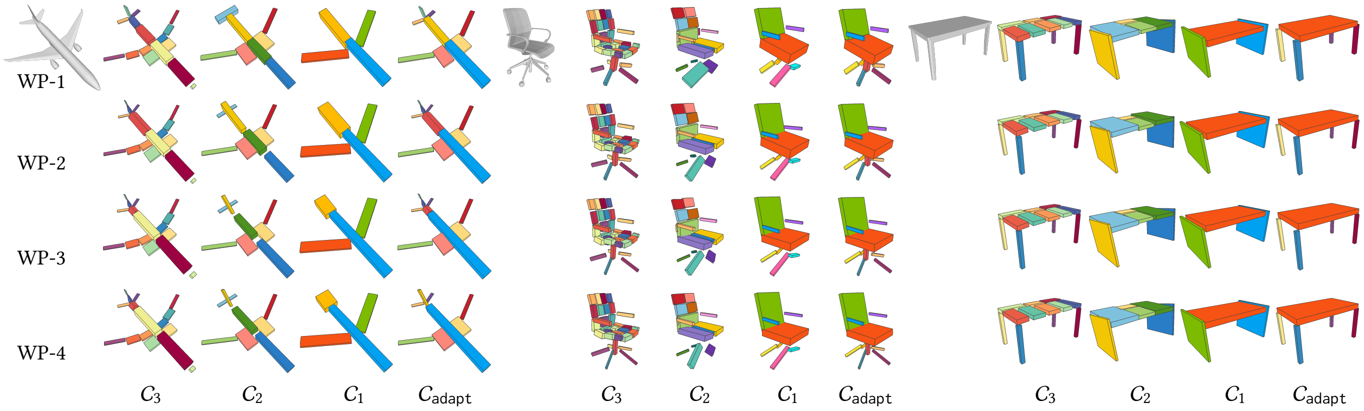

Fig. 6. Visualization of hierarchical and adaptive cuboid abstraction at different training stages.

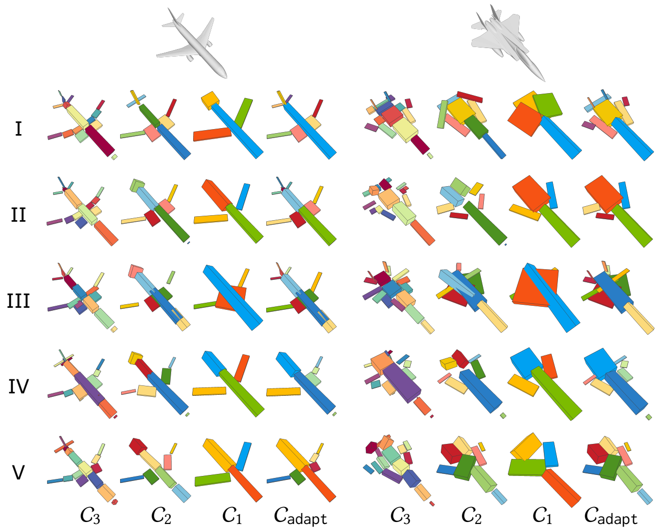

Fig. 7. Ablation study of loss terms. The two input shapes are chosen from the training set of the airplane category. More visual comparisons on other categories are provided in the supplement material.

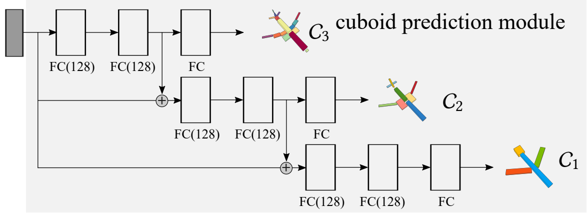

Fig. 8. The bottom-up hierarchical network for three-layer cuboid prediction (Decoder part).

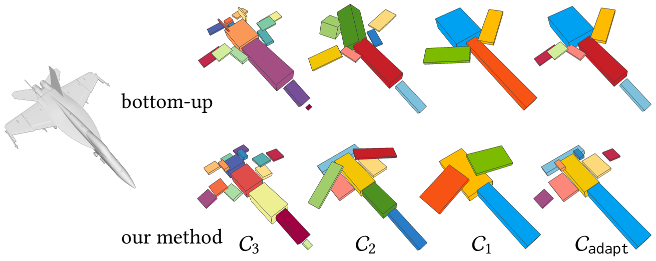

Fig. 9. Visualization of results of the bottom-up hierarchical network on the airplane category. The results of our network is also shown for comparison. The average Chamfer metrics on the training and test sets are 11.43 and 19.48, respectively.

the subnetworks used in the previous stages are frozen when the later stages are in training progress. We experimented a kind of these training schemes and found it is no better and even worse than our current training scheme. The experiment is set up as follows: the network (illustrated in Figure 8) uses the same encoder as ours, but trains the decoder branch for the finest level of cuboids first; then freezes the encoder and the decoder branch of C_3 to train the decoder branch of C_2, and so on in a bottom-up way. After training this cuboid prediction module, we use its output to train the mask selection module. The downside of this bottom-up training is that the encoder and decoder part for the finer layers is not updated in the later stage and the error would accumulate. Compared with our network output from the initial training, this bottom-up trained network predicts less visual-pleasing results with larger approximation error as tested on the airplane category (see Figure 9), thus it is essential to update the decoders of each layer simultaneously as done in our network.

5.3 Experiments and comparisons

The quality statistics of our method on the four shape categories are collected in Table 3. Figure 1 and Figure 10 show a collection of adaptive cuboid abstractions predicted by our network.

We compare our results with the work of [Tulsiani et al. 2017] which is an unsupervised learning approach for cuboid abstraction. Since Tulsiani *et al.*'s network uses all the shapes in the dataset as the training data, we retrain Tulsiani *et al.*'s network on our training sets and evaluate the result quality on both the training and test set. From Table 3, we can clearly see that the quality of our adaptive abstraction approximates the input shape much better.

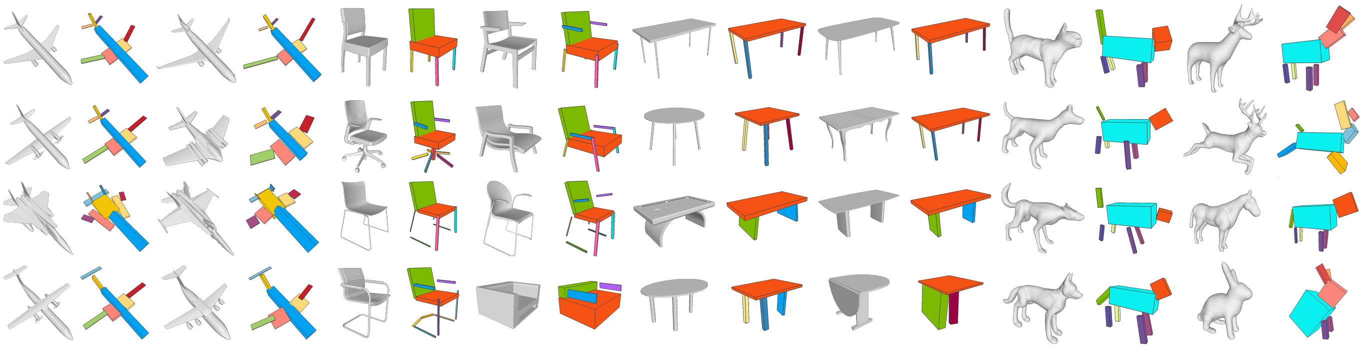

Fig. 10. Various adaptive hierarchical cuboid abstractions predicted by our network for 3D shapes from four categories.

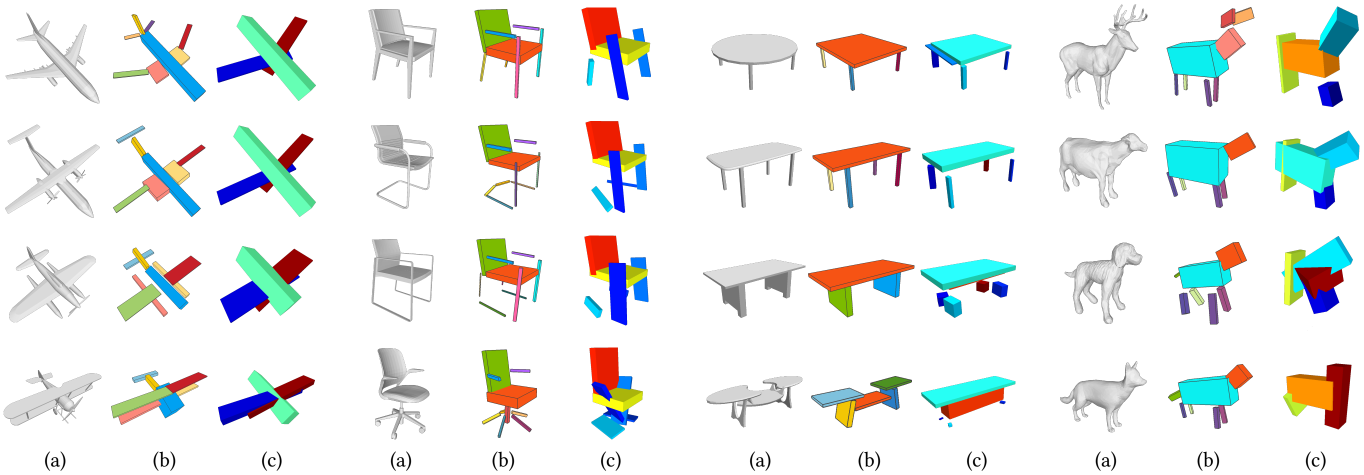

Fig. 11. Comparison of abstraction results between our method and [Tulsiani et al. 2017]. (a) the input shape, (b) our adaptive abstraction; (c) the result of [Tulsiani et al. 2017]. Cuboids are color-coded by their corresponding indices at each hierarchical layer defined in our network or Tulsiani *et al.* 's network. It is clear to see that our result approximates the input more faithfully.

Table 3. Quality statistics of 3D abstraction results. The numbers in the table are the average metrics over all the shapes. We present the statistical results of the training and test set as "training(test)". The result quality of [Tulsiani et al. 2017] is provided for comparison.

Category	$\#C_{\texttt{adapt}}$	$D_{\mathrm{cd}}(C_{\texttt{adapt}})$	$D_{\mathrm{cd}}(\widetilde{C})$	r	N_{cor}
Airplane	7.93	9.32(18.68)	18.50(21.22)	1.08%	1.16
Chair	6.77	18.92(24.79)	25.32(27.34)	7.83%	1.64
Table	4.59	18.30(27.58)	26.97(29.71)	1.95%	1.23
Animal	7.40	18.85(19.06)	35.04(36.58)	3.09%	1.00

The advantage of our method is also verified by the visualization in Figure 11. Our adaptive abstraction provides more meaningful structures and captures important and small components.

We also evaluate how many corrections occurred in our training data by measuring the correction ratio r — the number of shapes with correction over the total number of shapes, and N_{cor} — the average number of corrected nodes in the corrected trees. We can see that most results do not need any correction. The correction ratio of the Chair category is little higher due to the large structure variations, however, the number of corrected nodes is still small.

Table 4. Quality statistics of results of using the cuboid configuration (5, 10, 20) in the network. We present the average $D_{\mathrm{cd}}(C_{\texttt{adapt}})$ of the training and test sets as"training(test)".

Category	default	(5, 10, 20)
Airplane	9.32(18.68)	10.27(18.93)
Chair	18.92(24.79)	20.44(25.26)
Table	18.30(27.58)	18.86(27.81)

Choices of cuboid number. The cuboid numbers in the three levels are fixed and serve as an initial cuboid candidate set in our network. The cuboid number in the adaptive abstraction is different from these fixed numbers (see the average cuboid number statistics — $\#C_{\texttt{adapt}}$ in Table 3). To test the sensitivity of our method to this initial number of cuboids, we reused the cuboid configuration of the Animal category — (5, 10, 20) for the other three categories. The statistical results in Table 4 show that the predicted abstractions have slightly worse approximation errors than our defaulting setting on the training data, but there is no much difference on the test data. The visual results are presented in the supplemental material.

Table 5. Statistics of the number of classes in the 3-level classification. # Class-1, # Class-2, # Class-3 are the number of classes from the coarsest level to the finest level.

Category	# Class-1	# Class-2	# Class-3
Airplane	1	20	41
Chair	32	138	163
Table	5	43	61
Animal	4	38	41

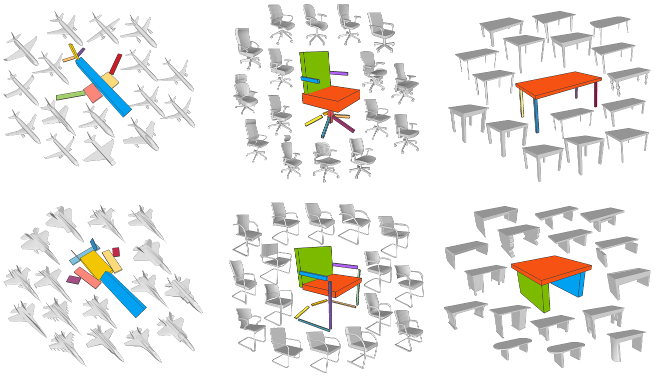

Fig. 12. Shape classification by using adaptive cuboid abstraction structure. Six classes with randomly sampled shapes from airplane, chair and table category are visualized. Shapes in each group have the same adaptive cuboid tree structure which is rendered in center.

5.4 Applications

We utilize the learned adaptive hierarchical cuboid abstraction and the latent code for different shape analysis and editing tasks.

Structure-aware classification. Shapes in the same category usually possess similar geometric structure though there exist structural variations. As the cuboid abstraction can characterize structural variations and provide a hierarchical structure, we use the selection mask vector (whose entry is 0 or 1) to classify the shapes of a collection in a multi-level. We first classify the shapes by using the selection mask vector of $C^1_{\texttt{adapt}}$ to obtain a coarse level of classification, then each class is subdivided into sub-classes by using the selection mask of $C^2_{\texttt{adapt}}$, finally, we use the selection mask of $C_{\texttt{adapt}}$ to get the third-level classification. In Table 5 we count the number of classes of the 3-level classification on the training sets of the four shape categories. In Figure 12 we visualize six common classes of $C_{\texttt{adapt}}$ from the airplane, chair, and table category. We can see that the adaptive abstraction helps group structure-similar shapes. In Figure 13(left) we visualize the distribution of the number of adaptive abstractions in each class by pie charts. The area of the slice represents the ratio of the number of adaptive abstractions in each class. The abstraction-based classification also helps distinguish shapes with rare geometric structures. Figure 13(right) illustrates some of classes which contain single shape only.

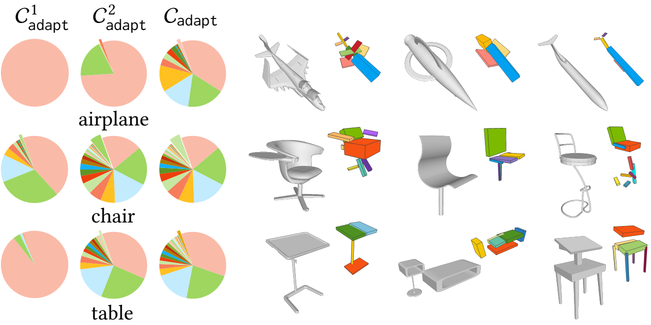

Fig. 13. Left: the distribution of the number of adaptive abstractions in each class. We group the classes whose number of elements are less than 10 to a single piece for better visualization (see the slightly translated pieces). Right: shape classes that have one instance only.

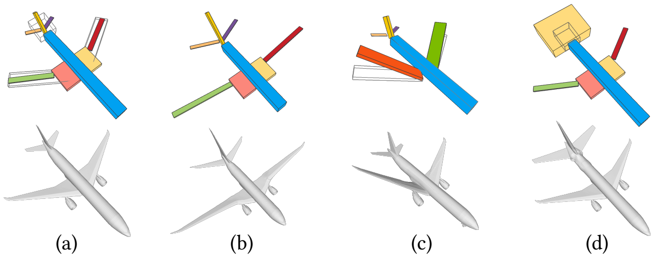

Fig. 14. Cuboid-based shape deformation. (a) The input shape and its adaptive abstraction. The parent cuboids of some selected cuboids are rendered as frames. (b) the deformation result by editing the cuboids of the adaptive abstraction; (c) & (d): the deformation results by editing some selected parent cuboids. The initial selected cuboids are rendered in wire-frame.

Cuboid-based shape deformation. The cuboid abstraction provides a set of handles for editing the shape geometry. Any change of the parameters of a cuboid introduces an affine transformation with respect to itself. This transformation can be applied to the local region controlled by the cuboid and passed to its descendant cuboids. For a 3D shape, assume that its adaptive abstraction has m cuboids, we divide the surface into m groups by associating the surface point to its nearest cuboid. For any cuboid with changed parameters, we can apply the affine transformation to the corresponding point group. However, this transformation would yield non-smooth transitions around part-adjacent regions. Thus for each surface point, we blend the transformations of its nearest two cuboids with the weight of the inverse distance from the point to those cuboids, and deform the point with the blended transformation. In Figure 14 we provide deformation results for an airplane model with the editing on the cuboids of its adaptive abstraction and their parent cuboids.

Abstraction-aware shape retrieval from 3D input. 3D retrieval is an important task in 3D analysis. Besides retrieving a shape with the least geometry difference from the dataset, the structural similarity between the retrieved shape and the 3D query is also important. We utilize the latent code of the adaptive cuboid abstraction network for this task. For a 3D query, we use its 3D latent code as the key for searching. As for comparison, we also use the latent code trained from a 3D autoencoder for this task. We choose the O-CNN-based 3D autoencoder [Wang et al. 2017, 2018] as the competitor, in which the octree depth is 5. We run a retrieval benchmark for these metrics

Table 6. Approximation quality of the top-k retrieved shapes on three categories. The average Chamfer distance between the k-th retrieved shape and the query is measured.

category	method	1st	5th	10th
Airplane	our method	18.6	21.0	22.0
	autoencoder	18.9	21.4	22.6
Chair	our method	24.9	29.2	30.5
	autoencoder	23.1	28.3	30.7
Table	our method	27.9	32.2	34.3
	autoencoder	26.2	32.0	34.1

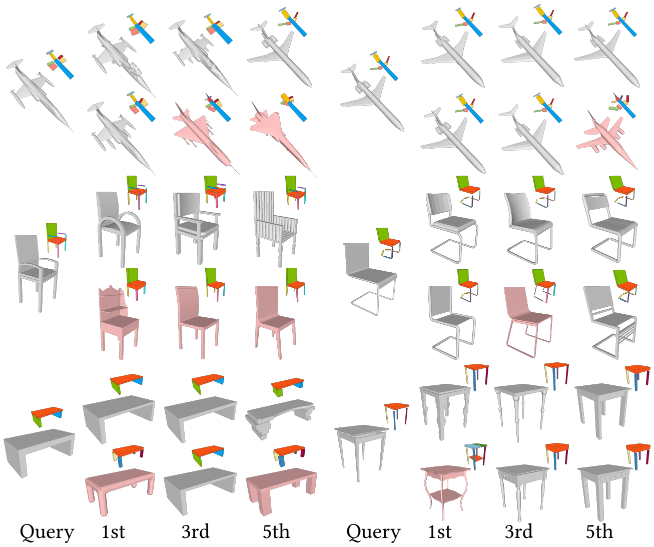

Fig. 15. Comparison of 3D shape retrieval using our abstraction latent code and 3D autoencoder's latent code. The query shape is on the left. The 1st, 3rd, 5th retrieved shapes and their adaptive cuboid abstractions are visualized on the right. The results of using our method (upper row) and autoencoder latent code (lower row) are shown side by side. The retrieved shapes from our method have the similar adaptive abstraction layout and their geometric structures are more consistent to the query than using the autoencoder approach. Structure-dissimilar shapes are highlighted in red.

on the airplane, chair and table datasets: for each shape in the test set, we query Top-10 shapes from the training set. The statistical results of Table 6 show that both approaches find shapes with a similar level of geometric error, but our method is capable of finding more structure-similar shapes as shown in Figure 15 because our latent code characterizes both geometry and abstraction similarity.

Abstraction-aware interpolation. Shape interpolation in the latent space is popular in the learning-based shape generation works. However, many works did not consider the shape structure validity, and the interpolated shapes are often incomplete or less meaningful to humans. With our network, we can decode the interpolated latent code to the adaptive abstraction whose abstraction structure is more meaningful. Figure 16 shows the interpolated adaptive abstractions for two input shapes. We find that the transition of abstractions is smooth in most cases. We also use the interpolated abstraction to retrieve the most similar shape from the dataset, and deform the shape accordingly, for the shape generation purpose.

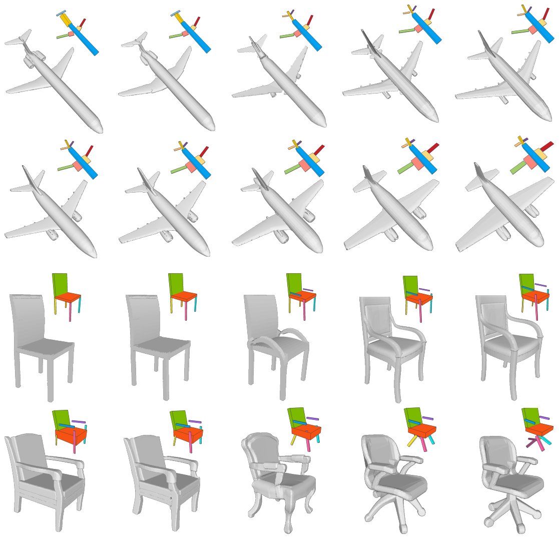

Fig. 16. Abstraction-aware shape interpolation. For the two 3D shapes on at the two ends of each row, we interpolate their latent codes linearly and illustrate the predicted adaptive cuboid abstractions. We also retrieve the most similar 3D shapes from the dataset using the interpolated latent code and deform the retrieved shape by our cuboid-based deformation as the interpolated shapes.

6 CONCLUSION

We have presented an adaptive hierarchical cuboid representation for 3D shape abstraction and introduce an unsupervised learning approach to extract this representation from unlabeled shape data. We evaluate our approach on four shape datasets and demonstrate its superiority over the existing method. We also utilize the learned abstraction successfully for different shape analysis tasks and achieve convincing results. We believe that the resulting abstractions could benefit many shape analysis and manipulation tasks, such as functional prediction and structural-aware shape manipulation. Also, our learning approach could be used for generating shape abstractions of online unlabeled shape repositories.

Limitations. Some small components of shapes like chair wheels are not captured by our learned adaptive cuboid abstraction. It is because the volume and surface coverage losses are not high on these small components where the network unlikely places small cuboids. A possible solution is to first detect feature regions by primitive fitting or unsupervised segmentation and then penalize the coverage loss on those regions with a large weight.

A few directions of our research are worth for future study.

Multi-level abstraction. In our work we only experiment with three-level cuboid abstraction. For highly-structured like bicycles, it is essential to use more levels of abstraction to capture the variant structures. It would be interesting to test our algorithm on more complicated shape categories.

More geometry relationships. Currently we only use the inclusion relationship and a weak bilateral-symmetry. Other geometry relationships like rotational and translation symmetry [Li et al. 2017] are not considered. How to explore other geometry relationships under the unsupervised setting is an interest research direction.

Non-static shapes. Because the shapes in our training dataset mostly have similar poses, it is hard to abstract two shape parts that have clear semantic meaning by two cuboids if these parts cannot be distinguished from geometry. If the shape data comes with animation or motion sequences, it will help learn a more structure-aware and part-consistent abstraction.

Small sample learning. As the usability of shape abstraction is determined by users, it would be better to use a few of labelled data to supervise the learning of shape abstraction or intervene the learning by correcting the imperfect prediction in an active learning manner.

Explicit consistency. In our method, the abstraction consistency is not explicitly formulated as there is no groundtruth. How to quantize it and use it as a loss function is worth studying.

Beyond cuboid abstraction. Currently we assume that the union of the cuboids in the abstraction approximates the input shape. A promising extension is to introduce more types of Boolean operators to enhance the abstraction representation, like intersection and difference for mimicking CSG operations. Another direct extension is to support more geometric primitives, like ellipsoids, cylinders, and cones to improve the abstraction compactness and expressiveness.

ACKNOWLEDGMENTS

We thank the authors of [Chang et al. 2015] and [Tulsiani et al. 2017] for sharing data and the anonymous reviewers for their valuable suggestions and feedback.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, and etal. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Gareth Bradshaw and Carol O'Sullivan. 2004. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.* 23, 1 (2004), 1–26.

Angel X. Chang, Thomas Funkhouser, and etal. 2015. ShapeNet: an information-rich 3D model repository. arXiv:1512.03012 [cs.GR].

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational Shape Approximation. *ACM Trans. Graph. (SIGGRAPH)* 23, 3 (2004), 905–914.

Nicu D. Cornea, Deborah Silver, and Patrick Min. 2007. Curve-skeleton properties, applications, and algorithms. *IEEE. T. Vis. Comput. Gr.* 13, 3 (2007), 530–548.

Fernando De Goes, Siome Goldenstein, Mathieu Desbrun, and Luiz Velho. 2011. Exoskeleton: Curve network abstraction for 3D shapes. *Comput. Graph.* 35, 1 (2011), 112–121.

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. InverseCSG: Automatic conversion of 3D models to CSG trees. *ACM Trans. Graph. (SIGGRAPH ASIA)* (2018).

Noa Fish, Melinos Averkiou, Oliver van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J. Mitra. 2014. Meta-representation of shape families. *ACM Trans. Graph. (SIGGRAPH)* (2014), 34:1–34:11.

Qiang Fu, Xiaowu Chen, Xiaoyu Su, Jia Li, and Hongbo Fu. 2016. Structure-adaptive shape editing for man-made objects. *Comput. Graph. Forum (EG)* 35, 2 (2016), 27–36.

Aleksey Golovinskiy and Thomas Funkhouser. 2009. Consistent segmentation of 3D models. *Computers and Graphics* 33, 3 (2009), 262–269.

Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. FlowRep: Descriptive curve networks for free-form design shapes. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 59:1–59:14.

M. S. Hassouna and A. A. Farag. 2009. Variational curve skeletons using gradient vector flow. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 12 (2009), 2257–2274.

Ruizhen Hu, Lubin Fan, and Ligang Liu. 2012. Co-segmentation of 3D shapes via subspace clustering. *Comput. Graph. Forum (SGP)* 31, 5 (2012), 1703–1713.

R. Hu, M. Savva, and O. van Kaick. 2018. Functionality representations and applications for shape analysis. *Comput. Graph. Forum* 37, 2 (2018), 603–624.

Qixing Huang, Vladlen Koltun, and Leonidas Guibas. 2011. Joint shape segmentation with linear programming. *ACM Trans. Graph. (SIGGRAPH ASIA)* 30, 6 (2011), 125:1–125:12.

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 2017. 3D shape segmentation with projective convolutional networks. In *Computer Vision and Pattern Recognition (CVPR).*

Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. 2010. Learning 3D mesh segmentation and labeling. *ACM Trans. Graph. (SIGGRAPH)* 29, 4 (2010), 102:1–102:12.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative recursive autoencoders for shape structures. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 52:1–52:14.

Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. 2011. GlobFit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph. (SIGGRAPH)* 30, 4 (2011), 52:1–52:12.

Wallace Lira, Chi-Wing Fu, and Hao Zhang. 2018. Fabricable Eulerian wires for 3D shape abstraction. *ACM Trans. Graph. (SIGGRAPH ASIA)* (2018), 240:1–240:13.

Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. 2018. PlaneNet: piece-wise planar reconstruction from a single RGB image. In *Computer Vision and Pattern Recognition (CVPR).*

Lin Lu, Yi-King Choi, Wenping Wang, and Myung-Soo Kim. 2007. Variational 3D shape segmentation for bounding volume computation. *Comput. Graph. Forum* 26, 3 (2007), 329–338.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J. Mitra. 2009. Abstraction of man-made shapes. *ACM Trans. Graph. (SIGGRAPH ASIA)* 28, 5 (2009), 137:1–137:10.

Niloy Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. 2014. Structure-aware shape processing. In *SIGGRAPH 2014 Courses.* 1:1–1:22.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. 2019. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Computer Vision and Pattern Recognition (CVPR).*

Chengjie Niu, Jun Li, and Kai Xu. 2018. Im2Struct: Recovering 3D shape structure from a single RGB image. In *Computer Vision and Pattern Recognition (CVPR).*

Zhenyu Shu, Chengwu Qi, Shiqing Xin, Chao Hu, Li Wang, Yu Zhang, and Ligang Liu. 2016. Unsupervised 3D shape segmentation and co-segmentation via deep learning. *Comput. Aided Geom. Des.* 43 (2016), 39 – 52.

Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph. (SIGGRAPH ASIA)* 30, 6 (2011), 126:1–126:10.

Shubham Tulsiani, , Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. 2017. Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Recognition (CVPR).*

Oliver van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. 2013. Co-hierarchical analysis of shape structures. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013), 69:1–69:10.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 72:1–72:11.

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Trans. Graph. (SIGGRAPH ASIA)* (2018).

Jianhua Wu and Leif Kobbelt. 2005. Structure recovery via hybrid variational surface approximation. *Comput. Graph. Forum (EG)* 24, 3 (2005), 277–284.

Q. Wu, K. Xu, and J. Wang. 2018. Constructing 3D CSG models from 3D raw point clouds. *Comput. Graph. Forum* 37, 5 (2018), 221–232.

Kai Xu, Vladimir G. Kim, Qixing Huang, and Evangelos Kalogerakis. 2015. Data-Driven Shape Analysis and Processing. *Comput. Graph. Forum* 36, 1 (2015), 101–132.

Dong-Ming Yan, Yang Liu, and Wenping Wang. 2006. Quadric Surface Extraction by Variational Shape Approximation. In *Geometric Modeling and Processing - GMP.* 73–86.

Li Yi, Leonidas Guibas, Aaron Hertzmann, Vladimir G. Kim, Hao Su, and Ersin Yumer. 2017. Learning hierarchical shape segmentation and labeling from online repositories. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 70:1–70:12.

Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. 2019. PartNet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Computer Vision and Pattern Recognition (CVPR).*

Mehmet Ersin Yumer and Levent Burak Kara. 2012. Co-abstraction of shape collections. *ACM Trans. Graph. (SIGGRAPH ASIA)* 31, 6 (2012), 166:1–166:11.

Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 2017. 3D-PRNN: Generating shape primitives with recurrent neural networks. In *International Conference on Computer Vision (ICCV).*