Learning Shape Abstraction by Cropping Positive Cuboid Primitives with Negative Ones

Xi Zhao[1*], Haodong Li[1] and Haoran Wang[1]

[1*]School of Computer Science and Technology, Xi'an Jiaotong University, No.28 Xianning West Road, Xi'an, 710049, Shaanxi, P.R. China.

*Corresponding author(s). E-mail(s): xi.zhao@mail.xjtu.edu.cn;
Contributing authors: greyishsong@stu.xjtu.edu.cn; wanghr_xjtu@foxmail.com;

Abstract

High-quality 3D model abstraction is needed in many graphics or 3D vision tasks to improve the rendering efficiency, increase transmission speed or reduce space occupation. Traditional simplification algorithms for 3D models rely heavily on the mesh topology and ignore objects' overall structure during optimization. Learning-based methods are then proposed to form an end-to-end regression system for abstraction. However, existing learning-based methods have difficulty representing shapes with hollow or concave structures. We propose a self-supervised learning-based abstraction method for 3D meshes to solve this problem. Our system predicts the positive and negative primitives, where positive primitives are to match the inside part of the shape, and negative primitives represent the hollow region of the shape. More specifically, the bool difference between positive primitives and the object is fed to a network using Iteration Error Feedback (IEF) mechanism to predict the negative primitives, which crop the positive primitives to create hollow or concave structures. In addition, we design a new separation loss to prevent a negative primitive from overlapping the object too much. We evaluate the proposed method on the ShapeNetCore dataset by Chamfer Distance (CD) and Intersection over Union (IoU). The results show that our positive-negative abstraction schema outperforms the baselines.

Keywords: Shape Abstraction, Primitive Fitting, Iterative Error Feedback

1 Introduction

The acquisition of 3D models has become more accessible nowadays, which significantly promotes the application of 3D data. Processing high-quality 3D models is pretty time-consuming, so how to convert models to be more light-weighted and effective representations has become a critical problem. One of the most popular ways of shape abstraction is representing 3D objects with a set of primitives, which can approximate the original shape with simple shape elements while keeping geometrical structures. Such abstraction can be used for many tasks related to modeling, shape analysis, reconstruction, and adaptive rendering.

3D shape abstraction methods mainly contain two categories: geometrical optimization-based methods and learning-based methods. The mesh optimization algorithm [1] introduces edge-collapse operation. Garland et al. propose a simplification algorithm guided by Quadric Error Metrics (QEM) [2], using the edge-collapse operation to reduce edges and faces. Later, many varieties of the QEM-based method have been proposed, but all of them lack the usage of global

information. The co-abstraction method [3] hierarchically generates a spectrum of abstractions for each object and constructs different levels of abstraction based on co-analysis. Both edge-collapse and co-abstraction methods can generate simplified meshes. However, the cost of optimization limits the efficiency of these methods. Learning-based abstraction methods [4, 5] perform better in speed, for they only predict a set of primitives by neural networks to fit the shape. Still, sometimes the primitives have difficulty matching the original shape precisely because the regression results lack accuracy with just one-step prediction. Zhao et al. [6] further propose a method to improve the results by iteratively refining the matching accuracy with the Iterative Error Feedback (IEF) mechanism.

Although IEF-based network [6] improved the result accuracy, we notice that it usually ignores hollow regions of the shape and tends to cover such regions with large cuboids. However, the hollow regions may be crucial for structure and style representation (for example, the hollow region under the chair's seat in Fig. 1). Therefore, considering such a structure in the abstraction results is necessary. A simple solution is to use more fine-grained cuboids during abstraction, but controlling the size and number of cuboids is not trivial during the regression process.

In this paper, we follow the work [6] to use cuboids as primitives for shape abstraction. To deal with hollow regions that are hard to be described by single cuboids, we propose to use *negative primitive* during primitive prediction. We call the primitives predicted by the previous learning-based methods the *positive primitives*, representing objects' overall shape, and the primitives remove the hollow region from the positive ones *negative*. More specifically, after predicting the positive primitives, we form a negative volume, which is the region that belongs to the positive primitives but is out of the original shape, and feed it to an IEF-based network to predict negative primitives. In this way, our system can crop positive primitives with negative primitives to form better abstraction results. Finally, we test our approach with public 3D model datasets. The results show that our method improves the abstraction results quantitatively and qualitatively compared with baseline cuboid abstraction methods.

Our main contributions are as follows:

- We propose a positive-negative schema, using negative primitives to crop assembly results and form better local geometries of the shape.
- We designed a novel separation loss for negative primitive prediction.
- We demonstrate that our approach performs better than the baseline methods, whether measured by IoU (12%) or CD (23%).

2 Related Works

2.1 Learning-based 3D Shape Abstraction Methods

Learning-based methods have shown effectiveness for shape abstraction tasks. Such methods typically feed networks with unstructured 3D data and estimate a list of primitives.

Voxel-based approaches convert 3D shapes to discrete, regular grids as the neural network input. Maturana et al. introduce VoxNet [7], the prior voxel-based approaches, for object classification and scene segmentation tasks. An unsupervised learning framework [4] uses similar techniques to extract features from voxel. Tulsiani et al. design a novel loss function measuring the difference between points sampled from abstraction results and the origin 3D model for unsupervised learning. Based on this framework, Sun et al. develop an adaptive hierarchical prediction module [5] to choose primitives from several levels of details according to a set of selection loss functions. Paschalidou et al. use superquadrics as primitives [8] to provide more expressive 3D scene parses.

Other approaches directly process unstructured data, such as point clouds or meshes. Yang et al. [9] generate cuboid abstraction similar to [4, 5] through jointly predicting cuboid allocation as part segmentation and cuboids shapes. They improve abstraction accuracy by enforcing the consistency between segmentation and abstraction. Li et al. [10] introduce a Supervised Primitive Fitting Network (SPFN), which integrates PointNet++ [11] for point cloud segmentation before primitive fitting. SPFN adjusts the segmentation strategy dynamically for a better-fitting result. Some other segmentation models such as HPNet [12], ParSeNet [13], CPFN [14] or PrimitiveNet [15] introduces different methods to generate a set

of essential surface patches. Guo et al. construct 3D line segments to match 2D ones from multi-view images [16] to reconstruct polygonal surfaces (mesh). Bridal et al. introduced an approach without segmentation [17] to fit industrial products from a point cloud. Primitives like planes, cylinders, cones, and spheres commonly form industrial products. This approach generates all kinds of primitives simultaneously, provides several candidate results, and selects the best through voting. Wu et al. [18] used primitives extracted by Globfit [19] to combine objects, performing well on complicated geometrical structures. However, such a process relies on multiple pre-requests, restricting its application to more datasets.

2.2 Applications on Stylization

The primitive representation of the 3D model can be a tool for stylization. Liu et al. [20] present a 3D stylization algorithm turning an input 3D model into the style of a cube. They use the as-grid-as-possible energy with ℓ_1-regularization on rotated surface normals to capture cubic-style sculptures. Huang et al. [21] construct polycube maps by minimizing ℓ_1-norm of the mesh normals, then using the polycube maps for hexahedral re-meshing and quadrangulation. The process of Physical Primitive Decomposition (PPD) [22] tries to understand an object through its components, considering both the geometrical and physical behaviors of the primitives. 3DStyleNet [23] predicts primitives (ellipsoids) for model segmentation, the segmentation result is used to transfer geometric style from the source to the target.

2.3 Positive and Negative Volume

Positive and negative volumes is a traditional technique that can be applied to many 3D shape processes.

Yumer et al. [3] propose the co-abstraction method, which progressively generates a spectrum of abstractions using positive and negative sub-volumes. A positive sub-volume will be added to the current abstraction result, and a negative sub-volume will be subtracted from it. The sub-volumes can be arbitrary shapes that are generated by an optimization process.

Chen et al. [24] propose an approach roughly simulating the pipeline of human visual perception on indoor objects. They try a coarse-to-fine scheme of abstraction and progressively approximate objects with constructive solid geometry (CSG) operations. The abstraction is represented as a CSG tree, whose non-leaf node represents a boolean operation of primitives, and the leaf node represents a single primitive.

We use positive and negative volumes from different aspects. The novelty of our method is that instead of directly using the optimization process to generate positive and negative volumes, our approach applies neural networks to predict them and use them for shape representation and approximation.

3 Approach

3.1 Overview

We designed a positive-negative schema for primitive prediction. The network predicts a set of cuboids (positive primitives) first, which are cropped by another group of cuboids (negative primitives); unlike the abstraction methods that only combine primitives, our approach also removes the volume occupied by negative primitives to represent local concave details. The entire pipeline of our method is shown in Fig. 1. We design a positive prediction module to predict positive cuboids and a negative prediction module for negative cuboids. We then apply the cropping operation to the two types of cuboids and generate the final results. The input of the positive prediction module is the voxelized shape. The input of the negative prediction module is the voxelized difference representing the concave region of the original shape. In practice, we perform a two-stage training: the positive prediction module is trained first without the negative prediction module, then the negative prediction module is trained with the positive prediction module fixed. The cropping operation is a difference operation that applies bool minus between two 3D objects.

3.2 Primitive Prediction

Both the positive and negative prediction modules are primitive prediction model, which works in an iterative manner inspired by [6], consisting of an initial prediction stage and an iterative refinement stage.

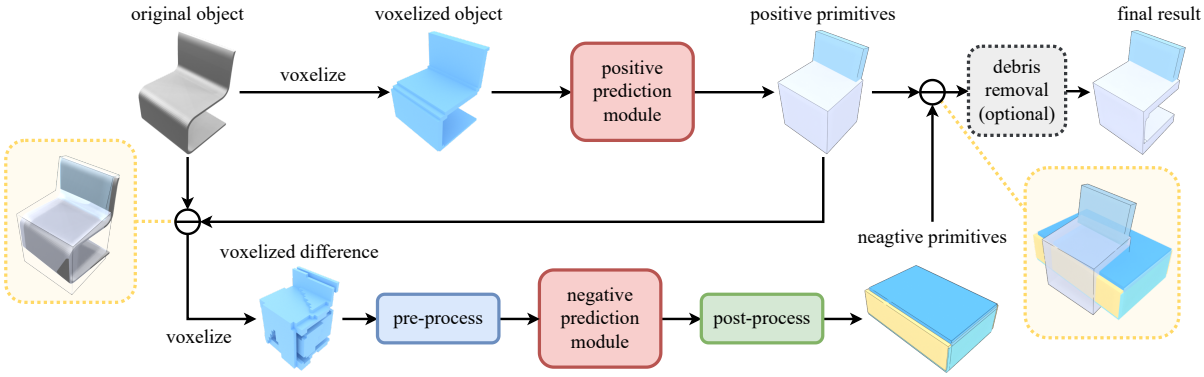

Fig. 1 The pipeline of our method. First, we predict positive primitives, then negative primitives. Then we crop the positive primitives with the negative ones. Each $\ominus$ indicates a difference operation (i.e. bool minus between two 3D objects). The difference operation on the right side is the cropping operation.

The initial prediction stage takes a voxelized object as input and uses a 3D-CNN as encoder, then intermediate fully-connected (FC) layers, and finally, FC layers as decoders to generate the output (shown in Fig. 2). The output of this network is a $10N$ dimensional vector containing information on N cuboids. Each cuboid C_i is represented by transformations applied to a unit cube as a vector $\mathbf{c}_i = (\mathbf{s}_i, \mathbf{r}_i, \mathbf{t}_i)$, where $\mathbf{s}_i \in \mathbb{R}^3$ represents scale factors, $\mathbf{r}_i \in \mathbb{R}^4$ represents a rotation (as a quaternion), $\mathbf{t}_i \in \mathbb{R}^3$ represents a translation.

The refinement stage works iteratively. In each iteration, the network takes its last step result as input (shown in Fig. 3). First, a shared encoder encodes the input object with the previous result to a feature vector. Then, intermediate layers and decoders generate a $10N$ dimensional residual vector, which will be added to the previous result to update the primitive configurations. In the first iteration, the output of the initial prediction stage is used as the last result.

The positive and negative prediction modules use the same two-step network because essentially both modules try to fit cuboids to a volume: the positive prediction module aims to fit cuboids to the inside region of the shape. In contrast, the negative prediction module seeks to find the concave part located at the shape's outside space. Because of the different goals, the two prediction modules must have different input and loss definitions. The positive prediction module takes the voxelized original object as the input. The negative prediction module takes the voxelized difference between positive primitives and the original object as input. Fig. 1 shows an example of the voxelized difference. It fits this difference with cuboid primitives.

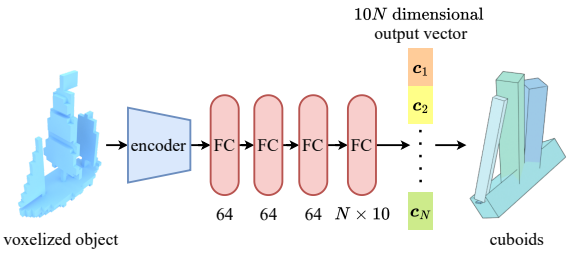

Fig. 2 The initial prediction stage, shared by positive and negative prediction modules.

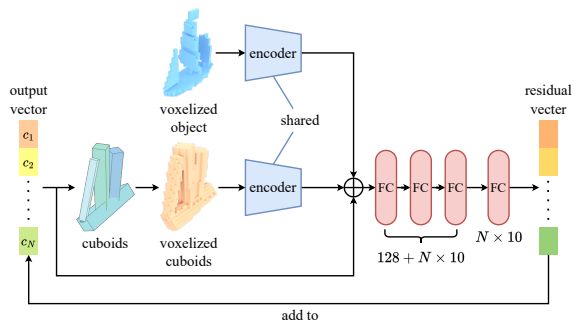

Fig. 3 The refinement stage, shared by positive and negative prediction modules. $\oplus$ indicates vector concatenation.

3.3 Data Processing for Negative Primitive Prediction

There are two steps of difference operations related to negative primitives. First, we compute the difference between the positive primitives (high-level representation) and the original object (low-level representation) to feed to the negative primitive module. Second, we subtract the difference between the positive and negative primitives to improve the local geometry. Both operations can lead to noisy shapes because the positive primitives, the original shapes, and the negative primitives are not perfectly coincident. Therefore, we apply a pre-process to adjust the input difference volume during the former step and a post-process to avoid less cropping (and over cropping) during the latter step.

Pre-process

Directly feeding the difference between positive primitives and the original object to the negative prediction module may cause geometric problems. Because the prediction module uses volumetric representation for input, the negative primitives may exceed the boundary of positive primitives, resulting in perforations and bulges. We apply a pre-process on the input to solve such a problem by removing voxels near the object S. Assume $\mathrm{Vol}(C^+)$ denotes the volumetric representation of all positive primitives C^+, the difference is $V_\Delta = \mathrm{Vol}(C^+) - \mathrm{Vol}(S)$. Before being fed into the negative prediction module, any voxel v at the boundary satisfying $d(v, S) < \theta_d$ (defined by Eq. 1) will be removed from V_Δ.

We use Signed Distance Field (SDF) to accelerate the computation of Euclidean distance, the SDF used in the negative prediction module is also computed in the pre-process. For each $v \in V_\Delta$, if C^+ is nearer than S, we sample several points close to v from C^+, otherwise, we sample from S but move them along the normal vector by θ_d.

Post-process

To avoid less cropping or over cropping (e.g. perforations), we apply a post-process before calculating the difference $C^+ - C^-$. For each negative primitive, if it overlaps the original object too much or has a narrow gap with the object, we try to make it tangent to the object by scaling it slightly. All scales will not exceed the limit of θ_d. We compute the scale factors by sampling points on the surface of S and measuring the distance from each sampled point to each negative primitive. If any sampled point can be found inside a primitive, we shrink it. We discard a primitive if any sampled point inside it is far from its surface. If there is no sampled point inside a primitive, we expand it. After being scaled, a primitive will also be discarded if it still overlaps too much with the object or is far from the object.

Non-Maximum Suppression (NMS) is applied to the remaining primitives to remove the redundant ones. Less negative primitives produce a less noisy shape after cropping and can be represented by a smaller set of parameters.

3.4 Debris Removal

Debris removal is a process after the final cropping operation. Sometimes one positive primitive will be cropped into several fragments, some fragments may be quite small compared with others. If a fragment is too small to form any meaningful structure, it visually appears as debris. By simply removing fragments with a small surface area, we can remove debris in most cases. For example, we remove fragments whose surface area less than 1% of the total, the results are illustrated in Fig. 4.

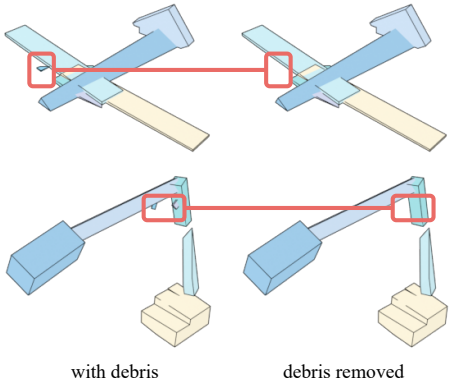

Fig. 4 Removing debris from the results (the threshold of removal is set to 1% of the total surface area in our experiments)

3.5 Loss Functions for Primitive Prediction

3.5.1 Loss for Initial Prediction Stage

We use fitting loss and symmetry loss to train the initial prediction network as Zhao et al. [6]. Fitting loss, inspired by the Chamfer Distance (CD), penalizes the inconsistency between N predicted primitives $C = \bigcup_{i=1}^{N} C_i$ and the input object S. First, we define the distance between a point p and a shape S:

$$d(p, S) = \begin{cases} 0 & \text{if } p \in S \\ \min_{q \in S} \|p - q\| & \text{else} \end{cases} \tag{1}$$

Based on this definition we have L^1_{fit} estimates how much the input object covers the predicted primitives and L^2_{fit} estimates the reverse coverage rate:

$$\begin{aligned} L^1_{\text{fit}}(S, C) &= \sum_{p \in C} d(p, S)^2 \\ L^2_{\text{fit}}(C, S) &= \sum_{p \in S} d(p, C)^2 \end{aligned} \tag{2}$$

The bidirectional fitting loss prevents primitives from over expanding or shrinking. Because of the inherent symmetry of man-made objects, we use symmetry loss to penalize the dis-symmetry primitives:

$$L_{\text{sym}}(C) = \sum_{i=1}^{N} \min_{1 \le j \le N} \text{CD}(C_i, \text{mirror}(C_j)) \tag{3}$$

where CD denotes Chamfer Distance between cuboids and the function mirror produces the mirror operation according to the xOy plane.

Finally, the loss function used to train the initial prediction network is defined as:

$$L_{\text{init}} = L^1_{\text{fit}} + L^2_{\text{fit}} + L_{\text{sym}} \tag{4}$$

3.5.2 Loss for Refinement Stage

The refinement network uses the fitting loss and a modified version of the symmetry loss. When it tries to adjust a primitive $C_i \subseteq C$ with its mirror instance $\text{mirror}(C_i) \subseteq C$, it will always be penalized by L_{sym} in Equation 3. Such a penalty may disturb the refinement network's optimization of primitives. So a conditional symmetry loss is used to solve this problem: [6]

$$\begin{aligned} L_{\text{con-sym}}(C) &= \sum_{i=1}^{N} f(C_i) \min_{1 \le j \le N} \text{CD}(C_i, \text{mirror}(C_j)) \\ f(C_i) &= \begin{cases} 1 & \text{if } \text{IoU}(C_i, \text{mirror}(C_i)) \le \alpha \\ 0 & \text{else} \end{cases} \end{aligned} \tag{5}$$

Beside, a regularization loss is used to constrain the scaling, rotation or translation within each iteration:

$$L_{\text{R}} = L_{\text{s}} + L_{\text{r}} + L_{\text{t}} \tag{6}$$

where L_{s} denotes scale constraint as

$$L_{\text{s}} = \sum_{i=1}^{N} \left(\max\{m_{\text{s}}(C_i) - \theta_{\text{s}}, 0\} \right)^2 \tag{7}$$

The θ_{s} is the threshold of scaling. L_{r} and L_{t} for rotation and translation constraint are defined similarly with thresholds θ_{r} and θ_{t}. In practice we set $\theta_{\text{s}}, \theta_{\text{r}}, \theta_{\text{t}}$ to $0.02, 0.02, 20°$.

The loss function used to train the refinement network can be concluded as:

$$L_{\text{refine}} = L^1_{\text{fit}} + L^2_{\text{fit}} + L_{\text{con-sym}} + L_{\text{R}} \tag{8}$$

3.5.3 Separation Loss for Negative Prediction Module

Generally, we perform the bool difference (i.e. subtraction) operation on positive primitives C^+ and negative primitives C^- as $C^+ - C^-$ to obtain the final result. In practice, a negative primitive may overlap the original object, cropping some of S by mistake. To avoid over-cropping, $C^- = \bigcup_{i=1}^{N^-} C^-_i$ is preferred to separate from the original object. So we design a separation loss:

$$L_{\text{sep}}(S, C^-) = \sum_{i=1}^{N^-} \sum_{p \in P(s)} \left(\theta_d - \min\left\{ d(p, C^-_i), \theta_d \right\} \right) \tag{9}$$

where $P(S)$ denotes uniformly sampled points in $\text{int}\, S$ (i.e. the interior of S). For a negative primitive C^-_i separating from S, the separation loss is roughly zero. θ_d is the threshold same as section

3.3. If C_i^- overlaps S, some points in $P(S)$ will be closer to C_i^-, then $d(p, C_i^-) < \theta_d$, the L_{sep} increases.

We use L_{sep} to train the negative prediction module in both initialization (L_{init}) and refinement (L_{refine}) stage.

4 Experiments

4.1 Dataset

We use seven categories of models in the ShapeNetCore dataset [25] (chairs, airplanes, buses, benches, sofas, ships and lamps are used) and animal models collected by Sun et al. [5] to train and test our network. Two-thirds of models of each category are used for training and others for testing. Each model is normalized and with its center aligned to $(0, 0, 0)$, then voxelized to 32^3 voxels.

4.2 Training Details

Our network is implemented using PyTorch and trained for 12 hours on a computer with Core i9-10900X and RTX 3080 (10 GB memory). It takes the network 0.12 seconds on average to predict primitives for each object and 2.3 seconds to crop the positive primitives. We use the Adam optimizer to train the network, with a learning rate of 0.001, β_1 of 0.9, and β_2 of 0.999. When training the refinement stage of a prediction module, it inherits the weights of the encoder from the initial prediction stage. The encoder's learning rate is set to 0.0001 (1/10 of other layers). Each module is trained with a batch size of 32 for 150 epochs.

4.3 Evaluation and Comparison

To evaluate our approach, we compare our method with works proposed by Tulsiani et al. [4] (LEARNING for short) and Sun et al. [5] (HIERARCHICAL for short). We also compare our results with the result of Zhao et al. [6]. As this work generates the positive primitives only, we denote it as POSPRIM for short.

Qualitative evaluation

Fig. 5 illustrates some qualitative results of our method. Our network shows the ability to generalize across different categories and different shapes. The difference between a concave object and positive primitives is usually convex. So fitting the difference is more straightforward than a concave, complicated shape. For objects which are difficult to fit with positive primitives, such as chairs having unusual legs or backs, our approach usually gives good abstraction through cropping. Other examples like non-symmetric sofas, ships with sails, or hollow buses also prove the effectiveness of our approach.

Quantitative evaluation

To evaluate our method, we use two metrics to measure the quality of abstraction results. Chamfer Distance (CD) is calculated by sampling points from abstractions and objects and performs better when measuring surface differences. Intersection over Union (IoU) measures the difference of spacial occupancy. With both metrics, we can compare different methods more accurately. We show the CD and IoU for different methods in Table 1 and Table 2. We can see from these tables that our method outperforms the baselines.

We also show the results of different methods in Fig. 6. From the figure, we can see that POSPRIM tends to expand primitives to cover the surface of the original object, leaving huge differences between the concave region of the shapes and primitives—for example, the lower part of the chair example. With the IEF mechanism, our approach reaches higher precision than LEARNING. HIERARCHY uses more primitives to fit the original object. Still, its mutex loss penalizing cuboid intersection may lead to unnecessary sparsity (i.e., in the plane example, the cuboids representing fuselage and tail are far from each other), resulting in a good IoU with undesirable visual quality.

4.4 Ablation Study

To prove the efficiency of different modules within our network, we compare our network with the following versions:

- **Without pre-process**: our network without pre-process for negative prediction.
- **Without post-process**: our network without post-process before cropping operation.
- **Without L_{sep}**: our network without the separation loss.

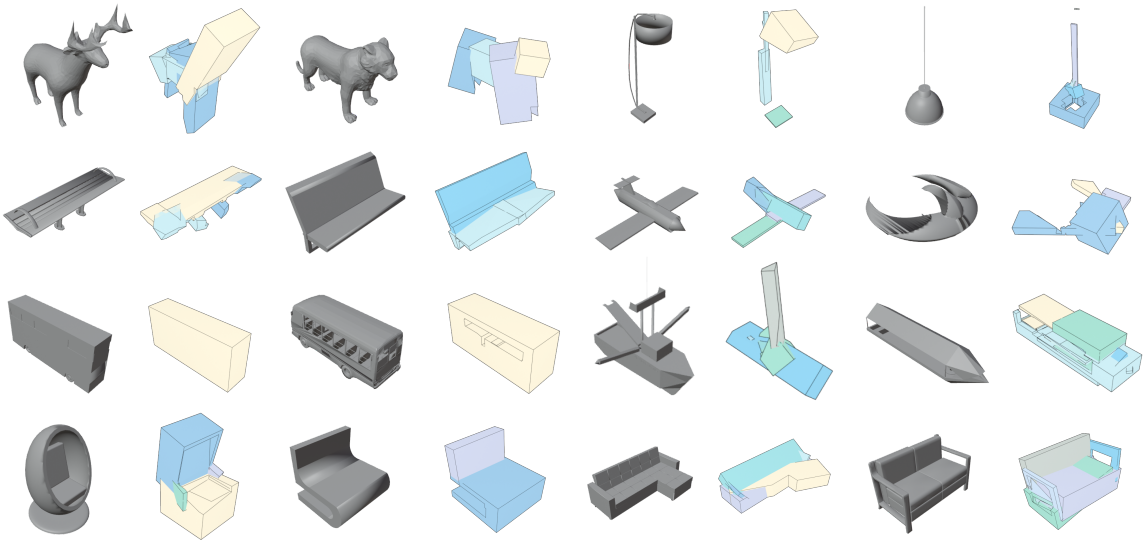

Fig. 5 Abstraction results of our method for each category in the dataset.

Table 1 Average IoU on the test set, higher is better.

IoU (%)	Airplane	Chair	Bench	Lamp	Sofa	Bus	Ship	Animal	Average
LEARNING	47.49	47.12	38.45	32.78	62.52	65.26	48.52	23.54	45.71
HIERARCHY (C_{adapt})	50.99	51.79	36.54	34.56	59.99	66.30	55.18	32.05	48.42
HIERARCHY (C_1)	49.23	50.24	36.18	34.47	58.18	62.56	54.20	31.48	47.07
POSPRIM	23.21	33.21	30.16	21.65	29.06	25.57	25.28	21.56	26.21
OURS	**56.31**	**58.18**	**40.98**	**36.19**	**66.95**	**79.49**	**59.24**	**46.32**	**55.54**

- **POSPRIM**: The learning based abstraction without considering negative primitives.

In Table. 3 and Table. 4 we show average CD and IoU for our ablation study. Our method has the highest IoU and the lowest CD values for all classes, proving the advantage of our method's final design.

We test a network that does not perform pre-processing on $S - C^+$ (mentioned in section 3.3). This network directly feeds the oversized volume $\text{Vol}(S - C^+)$ to the negative prediction module. From Fig. 7 we can see that the network without pre-process produces oversize negative primitives, so the bodies of the plane and the ship are over-cropped. Over-cropping usually creates too many perforations or holes in the abstraction result.

We also test a network which does not perform post-process (mentioned in section 3.3) on negative primitives before cropping. The post-process tries to make negative primitives tangent to the original object. By doing so we reduce the possibility of less-cropping. Fig. 8 shows abstractions encountering less cropping because of undersized negative primitives. Without the post-process, a slice is reserved under the chair's lags even if it should be cropped. Also, half of the bench remains solid, but the real bench is hollow.

We test a modified version of our network that removes the separation loss L_{sep}. The separation loss encourages a negative primitive to separate from the original object, therefore limiting its size. Without the separation loss, the negative primitives tend to be oversize and envelop $S - C^+$, then discarded because of overlapping too much with the original object. A lamp shown in Fig. 9 has not been cropped because all negative primitives are discarded.

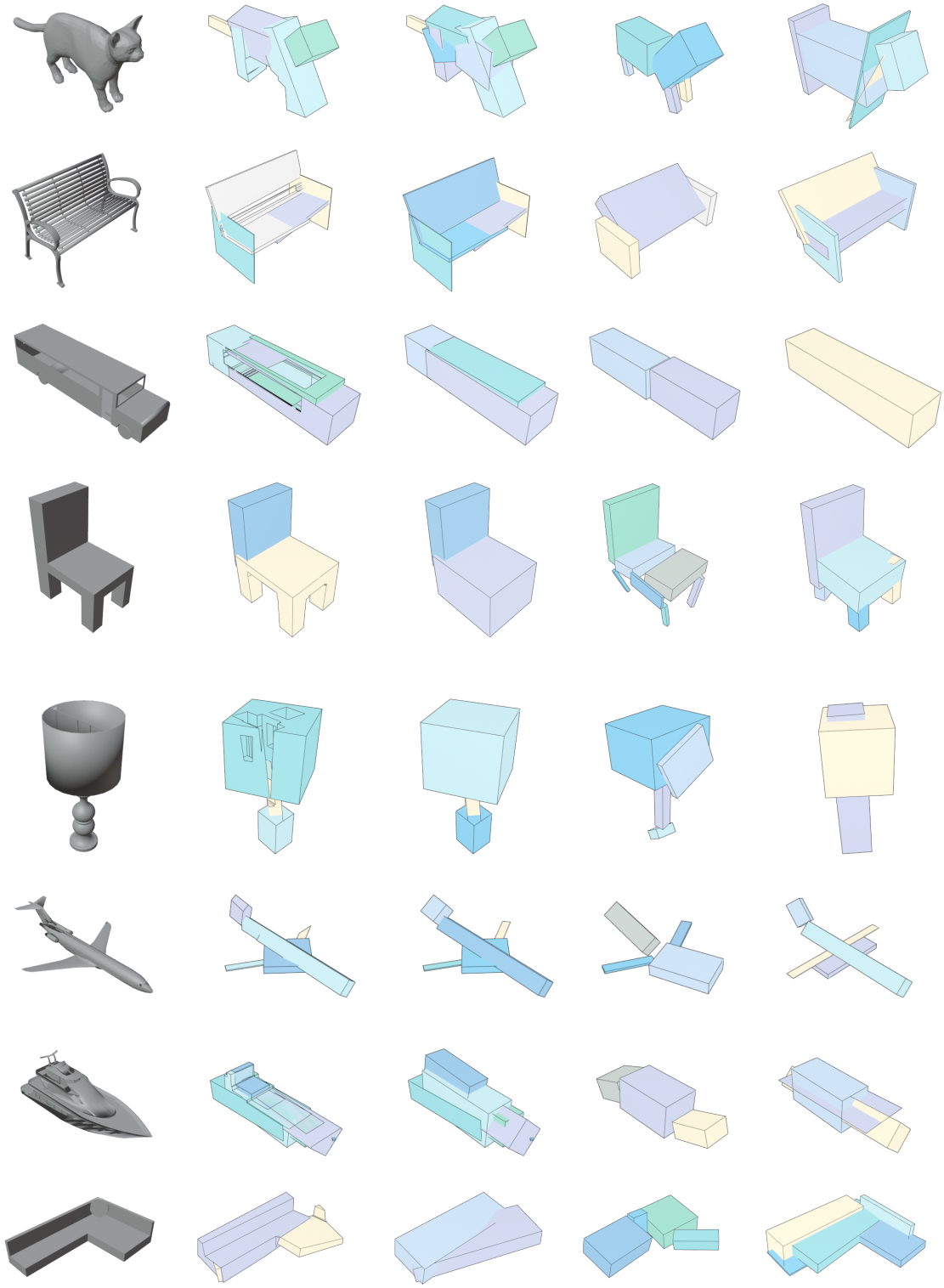

Fig. 6 The original objects and abstraction results by our method, POSPRIM [6], HIERARCHY [5] and LEARNING [4]

Table 2 Average CD on the test set (scaled by 1000), lower is better.

CD	Airplane	Chair	Bench	Lamp	Sofa	Bus	Ship	Aminal	Average
LEARNING	21.25	28.02	29.93	36.98	23.76	19.99	22.93	31.20	26.75
HIERARCHY (C_{adapt})	18.30	24.32	26.11	34.88	24.74	18.84	22.06	29.51	24.85
HIERARCHY (C_1)	20.44	26.28	32.02	37.37	24.91	18.84	22.96	29.55	26.55
POSPRIM	16.46	24.54	23.72	30.65	20.87	18.54	21.49	28.14	23.05
OURS	**14.61**	**19.61**	**19.03**	**22.97**	**18.28**	**16.24**	**15.78**	**25.14**	**18.96**

Table 3 Average CD of each ablation study per category (scaled by 1000), lower is better.

CD	Airplane	Chair	Bench	Lamp	Sofa	Bus	Ship	Animal	Average
Without pre-process	15.95	28.61	19.69	28.04	20.15	19.24	16.30	39.21	23.40
Without post-process	15.64	24.60	33.19	28.26	24.86	22.64	19.44	41.61	26.28
Without L_{sep}	22.99	20.63	19.40	27.99	20.12	19.21	15.86	32.98	22.40
POSPRIM	16.46	24.54	23.72	30.65	20.87	18.54	21.49	28.14	23.05
OURS	**14.61**	**19.61**	**19.03**	**22.97**	**18.28**	**16.24**	**15.78**	**25.14**	**18.96**

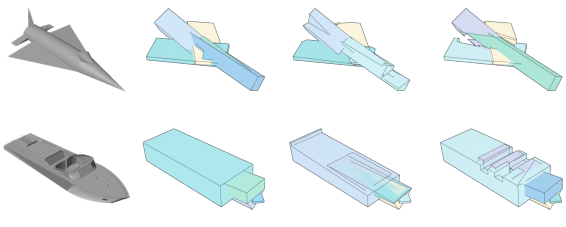

Fig. 7 The original object, POSPRIM, OURS and OURS without pre-process

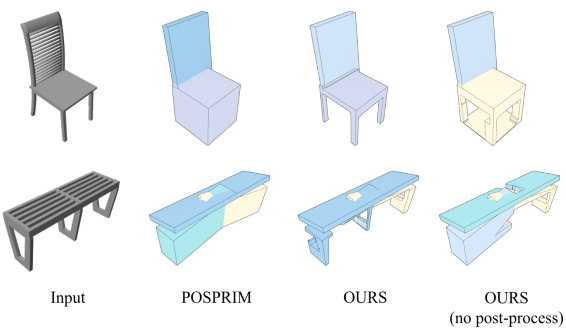

Fig. 8 The original object, POSPRIM, OURS and OURS without post-process

5 Conclusion and Discussions

Shape abstraction can provide compact and useful representations for 3D models. Although learning-based methods are quite popular recently, previous works are not good at shapes with concave regions. This paper introduces a positive-negative schema for shape abstraction to deal with such a problem. We build a network to predict positive and negative primitives, then crop the positive primitives with negative ones. Compared with the baseline methods, our approach can use fewer primitives to describe 3D shapes more accurately. The qualitative and quantitative (CD and IoU) results tested on the ShapeNet dataset show that our approach outperforms the baselines on this test set. Furthermore, other networks can apply our positive-negative schema, which may benefit future research.

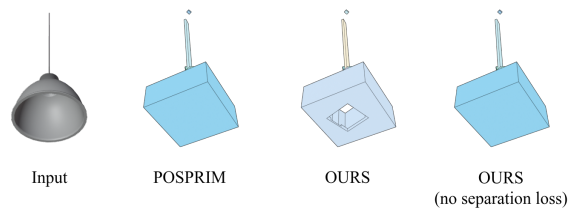

Fig. 9 The original object, POSPRIM, OURS and OURS without separation loss

Our method has its limitations. First, the cropping operation may split a positive primitive into several segments, creating unnatural disconnected components. Second, although the negative primitives lead to better geometric accuracy, we only use them once. In the future, we will explore how to iteratively use the negative primitive and make abstractions with different levels of detail. Finally,

Table 4 Average IoU of each ablation study per category, higher is better.

IoU (%)	Airplane	Chair	Bench	Lamp	Sofa	Bus	Ship	Animal	Average
Without pre-process	47.56	48.81	36.26	34.86	63.83	64.48	52.81	25.81	46.80
Without post-process	49.21	47.56	27.20	21.88	40.68	49.51	31.64	22.81	36.31
Without L_{sep}	42.67	46.97	37.76	31.71	63.11	64.39	50.21	14.10	43.87
POSPRIM	23.21	33.21	30.16	21.65	29.06	25.57	25.28	21.56	26.21
OURS	**56.31**	**58.18**	**40.98**	**36.19**	**66.95**	**79.49**	**59.24**	**46.32**	**55.54**

although cuboids can represent most objects well, they are naturally unsuitable for curved shapes like most lamps. How to consider other types of primitives in our system is another direction that is worth exploring.

References

[1] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '93, pp. 19–26. Association for Computing Machinery, New York, NY, USA (1993). https://doi.org/10.1145/166117.166119

[2] Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, pp. 209–216 (1997)

[3] Yumer, M.E., Kara, L.B.: Co-abstraction of shape collections. ACM Trans. Graph. **31**(6), 1–11 (2012). https://doi.org/10.1145/2366145.2366185

[4] 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 (CVPR), pp. 2635–2643 (2017)

[5] Sun, C.-Y., Zou, Q.-F., Tong, X., Liu, Y.: Learning adaptive hierarchical cuboid abstractions of 3d shape collections. ACM Trans. Graph. **38**(6), 1–13 (2019). https://doi.org/10.1145/3355089.3356529

[6] Zhao, X., Wang, H., Zhang, B., Yang, Y.-J., Hu, R.: Learning cuboid abstraction of 3d shapes via iterative error feedback. Computer-Aided Design **141**, 103092 (2021). https://doi.org/10.1016/j.cad.2021.103092

[7] Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928 (2015). https://doi.org/10.1109/IROS.2015.7353481

[8] 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 (CVPR), pp. 10344–10353 (2019)

[9] Yang, K., Chen, X.: Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. ACM Trans. Graph. **40**(4), 1–11 (2021). https://doi.org/10.1145/3450626.3459873

[10] Li, L., Sung, M., Dubrovina, A., Yi, L., Guibas, L.J.: Supervised fitting of geometric primitives to 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2652–2660 (2019)

[11] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5099–5108. Curran Associates, Inc., ??? (2017)

[12] Yan, S., Yang, Z., Ma, C., Huang, H., Vouga, E., Huang, Q.: Hpnet: Deep primitive segmentation using hybrid representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2753–2762 (2021)

[13] Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., Měch, R.: Parsenet: A parametric surface fitting network for 3d point clouds. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) Computer Vision – ECCV 2020, pp. 261–276. Springer, Cham (2020)

[14] Lê, E.-T., Sung, M., Ceylan, D., Mech, R., Boubekeur, T., Mitra, N.J.: Cpfn: Cascaded primitive fitting networks for high-resolution point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 7457–7466 (2021)

[15] Huang, J., Zhang, Y., Sun, M.: Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 15343–15353 (2021)

[16] Guo, J., Liu, Y., Song, X., Liu, H., Zhang, X., Cheng, Z.: Line-based 3d building abstraction and polygonal surface reconstruction from images. IEEE Transactions on Visualization and Computer Graphics, 1–15 (2022). https://doi.org/10.1109/TVCG.2022.3230369

[17] Birdal, T., Busam, B., Navab, N., Ilic, S., Sturm, P.: A minimalist approach to type-agnostic detection of quadrics in point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3530–3540 (2018)

[18] Wu, Q., Xu, K., Wang, J.: Constructing 3d csg models from 3d raw point clouds. Computer Graphics Forum **37**(5), 221–232 (2018). https://doi.org/10.1111/cgf.13504

[19] Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: Consistently fitting primitives by discovering global relations. In: ACM SIGGRAPH 2011 Papers. SIGGRAPH '11, pp. 1–12. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1964921.1964947

[20] liu, H.-T.D., Jacobson, A.: Cubic stylization. ACM Trans. Graph. **38**(6), 1–10 (2019). https://doi.org/10.1145/3355089.3356495

[21] Huang, J., Jiang, T., Shi, Z., Tong, Y., Bao, H., Desbrun, M.: ℓ1-based construction of polycube maps from complex shapes. ACM Trans. Graph. **33**(3), 1–11 (2014). https://doi.org/10.1145/2602141

[22] Liu, Z., Freeman, W.T., Tenenbaum, J.B., Wu, J.: Physical primitive decomposition. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018)

[23] Yin, K., Gao, J., Shugrina, M., Khamis, S., Fidler, S.: 3dstylenet: Creating 3d shapes with geometric and texture style variations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 12456–12465 (2021)

[24] Chen, X., Tang, J., Li, C.: Progressive 3d shape abstraction via hierarchical csg tree. In: Second International Workshop on Pattern Recognition, vol. 10443, pp. 205–210 (2017). SPIE

[25] 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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)