CubeFit: Minimizing Cuboids and Maximizing Coverage in 3D Shape via MCTS

Chanhyeok Park*
School of Computing
KAIST
chpark1111@kaist.ac.kr

Inhwa Song*
School of Computing
KAIST
igreen0485@kaist.ac.kr

Sangmin Lee*
School of Computing
KAIST
alexlee0104@kaist.ac.kr

Abstract

Approximating / Segmenting 3D shapes without supervision is known to be a challenging task. Solving it have benefits in many downstream tasks in computer vision / graphics. In this work, we approximate/segment 3D shapes using cuboids without any supervision. In such process we define two important geometric criteria called coverage and tightness. Under such constraint we solve the optimization problem of fitting bounding boxes to the 3D shape. However, the optimization problem is not directly solvable, so we instead propose searching based MCTS approach to tighten the cuboids while having high coverage of the 3D shape. Our result show that MCTS effectively tightens the bounding boxes while maintaining almost full coverage and achieving better reconstruction results compared to SOTA neural network based learning approaches.

1 Introduction

Approximating/Segmenting 3D shapes without any supervision has been a long standing problem in vision and graphics communities. Since no standard or ground truth exist in segmenting or approximating the shapes, it is still an unsolved open problem. For these tasks, we only have to rely on humans perception to solve these tasks. There has been a line of work that tried to solve these problem like utilizing neural networks to approximate the shape [2, 4, 3, 5, 6, 1, 7] or iterative approach using lloyd sampling to minimize energy function [8]. Also, reinforcement learning approaches [9] was tried to learn the underlining human perception. In this work we define some simple geometric criteria as reward and run exploration using monte carlo tree search (MCTS) to find the optimal result that we claim it is more closer to human perception compared to other works.

Neural network-based learning approaches have shown great performance on these tasks and various unsupervised learning-based approaches were attempted to abstract or reconstruct the shape with simple primitives. They typically first define the primitives like cuboids[2, 3, 5], convexes[11, 1], and implicit fields[10, 6, 7] to abstract/reconstruct the input shape. Then, they take self-supervised approach to minimize the reconstruction loss via back-propagation. For example, Yang and Chen (CA) [5], proposed an unsupervised approach that jointly predicts the cuboid parameters and segmentation of input point clouds giving feedback (loss) to each other. Also, BSP-Net [1] learns the associations of binary space partitioning as an implicit representation to reconstruct the 3D shape.

These neural network-based approaches are good at minimizing the self-supervised loss/criteria however, these criteria must be differentiable which limits the scope of goal they can achieve. Also, they usually perform bad when unseen examples are given which is a problem when applying them to real world data. Moreover, the important point is that these outputs are not tightly fitted to the input shape due to the generalization errors of neural nets. And by the similar problem, boundaries are not clear having overlaps between boxes. These problems occur since they can only use differentiable metrics like distance based reconstruction loss.

In this work, we first initialize the bounding boxes from convex decomposition method [17]. Then, we try to refine the initialized bounding boxes to obtain better result in terms of our two geometric criteria. The first one is **full coverage** where covering the entire shape is important since it allows us to achieve more accurate reconstruction and preserve details. Second criteria is **tightness** that forces the bounding boxes to be small as possible while satisfying the **full coverage**. Having the same coverage, it is obvious that having smaller boxes result in better approximation/segmentation. With such two constraints, we propose a monte carlo tree search (MCTS) approach to obtain better result by refining the bounding boxes via discrete actions.

We quantitatively show that these refinement allows us to achieve better reconstruction results and allows us to obtain tighter result. And we also qualitatively show that our results are much better to humans in terms of shape approximation.

2 Related Work

Learning-Based Shape Abstraction. As the neural network-based learning approaches have shown powerful performance, various unsupervised learning-based approaches have attempted to abstract/reconstruct the shape with simple primitives. For primitives, cuboids [2, 3, 5], convexes [11, 1] and implicit fields [10, 6, 7] have been explored. Tulsiani et al. [2] use deep convolutional neural networks, directly predicting volumetric primitives and their translation parameters. Sun et al. [3] (HA) try to discover the adaptive hierarchical cuboid abstraction to exploit the structural coherence of 3D shapes. Furthermore, convexes and implicit fields [10, 6, 7] which are more general representations have been used as primitives to enhance the performance of reconstruction.
Efficient Search for 3D Applications. Efficient search is crucial for tackling high-dimensional search spaces since we cannot exhaustively search all the possible solutions. Continuity and extra dimensionality in 3D data make the search space extremely large, requiring dedicated algorithms for efficient searches. For this purpose beam search [12, 13], monte carlo tree search (MCTS) [14, 15, 16, 17] and learning [13, 9] approaches have been explored to solve various 3D computer vision tasks such as shape reconstruction, scene detection and convex decomposition. Sharma et al. [13] propose a neural shape parser that learns not generate CSG grammar to challenge the huge search spaces. After obtaining the results from the shape parser, they apply beam search to find the best result. Similarly, Lin et al. [9] propose a learning-based approach to reconstruct the given shape that effectively searches the large discrete action spaces. Hampali et al. [14] use MCTS to detect 3D objects in the scenes to guide the solution faster. Wei et al. [17] also utilize MCTS to overcome the limitations of one-step greedy search. In this work, we apply MCTS to effectively tackle the huge search space and obtain better result from initialization.

3 Problem Definition

In this section, we define the problem of binding 3D shapes tight with boxes. The boxes should make full coverage of the 3D shape efficiently, and allow adequate number of boxes to cover the 3D shape. We defined two criterias for defining the problem. We start by dividing this requirement into two and formulating each; (1) full coverage, (2) tightness, with using adequate number of boxes. Full coverage implies that the boxes should not leave empty space of the 3D shape. To define and represent the property of full coverage, we introduce Cov, representing the boxes' volume coverage ratio of the 3D shape,

$$Cov(B) = 1 - \frac{vol(S - \cup_i B_i)}{vol(S)} \tag{1}$$

where B is a set of boxes, S is the target 3D shape. $Cov(B)$ should be 1 in order to satisfy the full coverage requirement. The requirement of tightness implies that boxes should cover the 3D shape efficiently, meaning that the bounding boxes should minimize spare volume exceeding the volume of the target 3D shape. We define tightness by using BVS(Bounding Volume Sum) introduced by Lu et al. [8] as follows.

$$BVS(B) = \frac{\sum_{i=1}^{k} vol(B_i)}{vol(S)} \tag{2}$$

Again, B is a set of boxes, and S is the target 3D shape. While achieving these goals, we require adequate number of boxes covering the 3D shape. For instance, certain solution can be branched to another solution by dividing some box to multiple boxes, and we infringe this kind of solution by setting adequate number of boxes requirement. We formulate this requirement using Cov and BVS as follows.

$$\underset{\{B_i\}_{i=1}^{k}}{\arg\min} BVS(B) \quad s.t. \quad Cov(B) = 1 \tag{3}$$

We start from an initial bounding box by pre-processing CoACD [17]. Since initialized starting point doesn't guarantee minimization of $BVS(B)$ while insuring $Cov(B)$ is 1, we must proceed on this optimization problem based on the aforementioned requirements. We observe that we can get tighter bounding boxes by making minor adjustment operations from the initial bounding boxes.

Based on this observation, we adopt a method of modifying the bounding boxes iteratively using discrete unit actions. This approach allows us to transform a complex optimization problem to a search problem, leading us to find a sequence of actions that optimizes our objective which is to obtain tight bounding boxes.

4 Method

4.1 Configuration

In this section, we show the configuration setting of our search problem. Since, it is difficult to find an optimal solution in a continuous environment of states and actions, we will solve this problem in a discrete action space by defining discrete actions. This approximation will allow us to turn the problem into a manageable search problem. Note that we can fix these errors later by post-processing. Below we provide the detailed configure setup for MCTS.

State. Current state for each bounding box can be represented by 7 parameters. Two diagonal points of the cuboid at each end, and a rotation matrix R at time-step t. If one of the coordinates of the left vertex becomes bigger than the right vertex, we treat the bounding box as deleted by such action.

$$B_t = \{(B_{t_i})\}_{i=1}^{N} = \{(x_l, y_l, z_l, x_r, y_r, z_r, R)\}_{i=1}^{N} \tag{4}$$

Action. There are two types of actions available for each bounding box:

First, adding or subtracting a predefined unit scale to one of the parameters in $(x_l, y_l, z_l, x_r, y_r, z_r)$.

Second, changing the rotation matrix to orient the bounding box correctly based on the points it covers. The second action is necessary to correct the wrong rotations derived from poor initialization of neural networks. In total, there are $6 \times 2 + 1 = 13$ actions for modifying each bounding box.

Objective. Our goal is to minimize $BVS(B)$ while having the hard constraint of tightness ($Cov(B) = 1$). However, such hard constraint will not allow us to perform rotation actions since rotating the box may result in some loss of coverage. For such reason, we suggest to change the hard constraint of our objective function to soft constraint which redefines our objective function. We will iteratively perform discrete actions on the bounding boxes to change the scale and rotation based on the redefined objective shown below.

$$\underset{\{B_{t_i}\}_{i=1}^{N}}{\arg\min} \{BVS(B) - \alpha Cov(B)\} \tag{5}$$

4.2 Monte Carlo Tree Search (MCTS)

As we mentioned earlier, our problem has huge action space to search. Simply calculated as $(N \times 13)^T$. It could have multiple local minima that we can potentially fall into. Therefore, we decided to use MCTS, which allows us to efficiently search on very large search space by simulating multiple steps regardless of the constraints while keeping the explored results for further searches.

Tree Structure and Iteration. Each node, n, in the tree stands for the bounding boxes's state, defined by its parameters, $B_t = \{B_{t_i}\}_{i=1}^{N}$ where B_{t_i} is the i-th bounding box in timestep t. Every edge signifies an action. The root node signifies the initial bounding boxes, which the approach aims to refine tightly through a series of actions within a timestep T.

Every node has a child node that symbolizes the state of bounding boxes resulting from some set of possible actions. The tree expands exponentially as each node can perform $N \times 13$ actions as the depth (timestep) increases.

During each iteration of the tree search, a node is chosen for expansion from the root. If a node is not fully expanded, one untried action is selected at random, and the corresponding node is expanded. For nodes that are fully expanded, the upper confidence bound [19] (UCB) is calculated as

$$UCB(n) = Q(n) + c\sqrt{\frac{2\ln N(n_p)}{N(n)}} \tag{6}$$

Here, n is the current node, n_p is its parent, $Q(\cdot)$ is the value function, $N(\cdot)$ shows the number of visits, and c is an exploration hyper-parameter. Once a node is expanded, greedy actions are applied to evaluate the node. The evaluation involves calculating the score of the entire path using the last state of the bounding box since it's the best result in the path. The score is calculated as

$$BVS(B_0) - BVS(B_t) - \alpha(Cov(B_0) - Cov(B_t)) \tag{7}$$

where B_0 denotes the initial bounding boxes. Equation displays how much improvements in minimizing the objective has been made. At the end of each iteration, $Q(\cdot)$ and $N(\cdot)$ are updated by backpropagating the score and visits along the path.

5 Results

5.1 Implementation Details

We evaluate our method on ShapeNet [18] dataset using the Table, Chair data category. Note that finding an oriented bounding box is implemented with trimesh which is one of the python libraries. In our method, we used and α as 100 and c with 0.001 in all of our experiments.

5.2 Evaluation Metrics

For evaluating the tightness of our bounding boxes, we use TOV and MOV proposed by Lu et al. [20]. BVS and Cov are also used since they are the main objectives of our problem. Also, to evaluate the reconstruction performance, we adopt the commonly-used chamfer distance [21] (CD) and volumetric IoU (VIoU).

- **Total Outside Volume:**
  $\text{TOV}(B) = \frac{\text{vol}(\bigcup_i B_i \setminus S)}{\text{vol}(S)}$
- **Maximum local Outside Volume:**
  $\text{MOV}(B, \{S_i\}) = \max_i \frac{\text{vol}(B_i \setminus S)}{\text{vol}(S_i)}$
- **Volumetric Intersection over Union:**
  $\text{VIoU}(B) = \frac{\text{vol}(S \cap \bigcup_i B_i)}{\text{vol}(S \cup \bigcup_i B_i)}$

5.3 Tightness of Bounding Boxes

We evaluate the tightness of the bounding boxes with BVS, MOV [20], TOV [20] in Tab. 1. Since there isn't any baseline that achieve the same goal, we just show CA [5] which is SOTA in cuboid abstraction. However as we can see, it does not achieve full coverage so, having small BVS is meaningless. Fig. 1 shows the qualitative result. As we can see our clearly shows better results.

	Table				Chair			
Metrics	BVS↓	Cov↑	MOV↓	TOV↓	BVS↓	Cov↑	MOV↓	TOV↓
CA [5]	1.08	0.61	-	-	1.10	0.72	-	-
Initialization	2.21	**1.00**	3.95	1.04	2.09	**1.00**	4.29	0.83
MCTS	**1.72**	**1.00**	**2.82**	**0.69**	**1.64**	**1.00**	**2.96**	**0.58**

Table 1: Comparison of tightness of bounding boxes between our initialization, MCTS, and baselines.

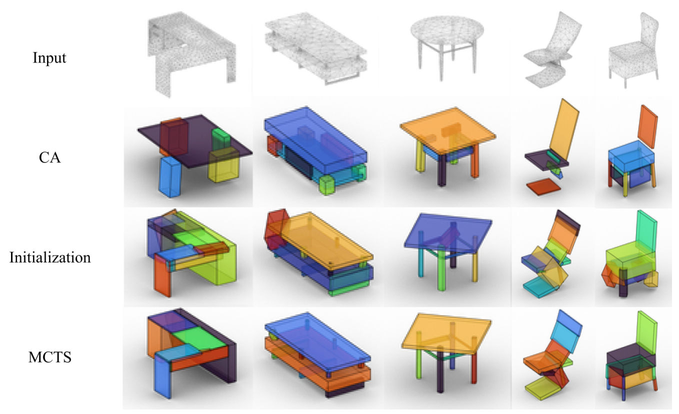

Figure 1: Comparison result of our initialized bounding boxes and

	Table		Chair	
Metrics	CD↓	VIoU↑	CD↓	VIoU↑
CA [5]	1.19	0.45	**0.812**	0.56
Initialization	0.757	0.60	0.962	0.63
MCTS	**0.641**	**0.67**	0.844	**0.69**

Table 2: Comparison of reconstruction performance between initialization, MCTS, and baseline. Note that all CD is scaled by 1000.

5.4 Shape Reconstruction

We use CD and VIoU to evaluate the reconstruction performance, where CD was calculated by uniformly sampling 4096 points from the obtained cuboids and the ShapeNet [18] mesh. VIoU was calculated by measuring the IoU between the input mesh and the merged bounding boxes. Tab. 2 shows the results. As we can see we achieve better reconstruction result than the neural network baseline. We can check that our MCTS approach effectively refines them to obtain better results.

6 Conclusion

In this work, we managed to refine the initial bounding boxes obtained from previous work to the input shape. We proposed a Monte Carlo Tree Search (MCTS) approach to obtain tighter bounding boxes considering two criteria of full coverage and tightness. Experiment results showed that our method outputs more tighter boxes while keeping the almost full coverage.

References

[1] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In CVPR, 2020.

[2] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In CVPR, 2017.

[3] Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. ACM TOG, 2019.

[4] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In CVPR, 2019.

[5] Kaizhi Yang and Xuejin Chen. Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. ACM TOG, 2021.

[6] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In CVPR, 2021.

[7] Chengjie Niu, Manyi Li, Kai Xu, and Hao Zhang. Rimnet: Recursive implicit fields for unsupervised learning of hierarchical shape structures. In CVPR, 2022.

[8] Lin Lu, Yi-King Choi, Wenping Wang, and Myung-Soo Kim. Variational 3d shape segmentation for bounding volume computation. Comput. Graph. Forum, 2007.

[9] Cheng Lin, Tingxiang Fan, Wenping Wang, and Matthias Nießner. Modeling 3d shapes by reinforcement learning. In ECCV, 2020.

[10] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In ICCV, 2019.

[11] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In CVPR, 2020.

[12] Abhinav Gupta, Martial Hebert, Takeo Kanade, and David Blei. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. NeurIPS, 2010.

[13] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In CVPR, 2018.

[14] Shreyas Hampali, Sinisa Stekovic, Sayan Deb Sarkar, Chetan S Kumar, Friedrich Fraundorfer, and Vincent Lepetit. Monte carlo scene search for 3d scene understanding. In CVPR, 2021.

[15] Sinisa Stekovic, Mahdi Rad, Friedrich Fraundorfer, and Vincent Lepetit. Montefloor: Extending mcts for reconstructing accurate large-scale floor plans. In ICCV, 2021.

[16] Sinisa Stekovic, Mahdi Rad, Alireza Moradi, Friedrich Fraundorfer, and Vincent Lepetit. Mcts with refinement for proposals selection games in scene understanding. IEEE TPAMI, 2022.

[17] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collisionaware concavity and tree search. ACM TOG, 2022.

[18] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information rich 3d model repository. arXiv preprint arXiv:1512.03012, abs/1512.03012, 2015.

[19] Levente Kocsis and Csaba Szepesvari. Bandit based monte carlo planning. In European Conference on Machine Learning, 2006.

[20] Lin Lu, Yi-King Choi, Wenping Wang, and Myung-Soo Kim. Variational 3d shape segmentation for bounding volume computation. Comput. Graph. Forum, 2007.

[21] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen Cf Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In IJCAI, 1977.