CvxNets:
Learnable Convex Decomposition

Boyang Deng
Google Research

Kyle Genova
Google Research

Soroosh Yazdani
Google Hardware

Sofien Bouaziz
Google Hardware

Geoffrey Hinton
Google Research

Andrea Tagliasacchi
Google Research

Abstract

Any solid object can be decomposed by a collection of convex polytopes (in short, convexes). When a small number of convexes are used, such a decomposition can be thought of as a piece-wise approximation of the geometry. Such a decomposition is fundamental to real-time physics simulation in computer graphics. Convex objects also have the property of being simultaneously an explicit and implicit representation: one can interpret it *explicitly* as a mesh derived by computing the vertices of a convex hull, or *implicitly* as the collection of half-space constraints or support functions. Their implicit representation makes them particularly well suited for neural network training, as they abstract away from the topology of the geometry they need to represent. We introduce a network architecture to represent a low dimensional family of convexes. This family is automatically derived via an auto-encoding process. We investigate the application of the network to automatic shape abstraction and reconstruction.

1 Introduction

While images admit a standard representation in the form of a scalar function uniformly discretized on a grid, the curse of dimensionality has prevented the effective usage of analogous representations for learning 3D geometry. Voxel representations have shown some promise at low resolution [4, 9, 15, 26], while hierarchical representations have attempted to reduce the memory footprint required for training [18, 20, 25], but at the significant cost of complex implementations. Rather than representing the *volume* occupied by 3D objects, one can resort to modeling its *surface* via a collection of points [1, 8], polygons [14, 17, 24], or surface patches [11]. Alternatively, one might follow Cezanne's advice and "treat nature by means of the cylinder, the sphere, the cone, everything brought into proper perspective", and think to approximate 3D geometry as *geons* [2] – collections of simple to interpret geometric primitives [23, 27], and their composition [19, 10]. Hence, one might rightfully start wondering *"why do so many representations of 3D data exist, and why would one be more advantageous than the other?"* One observation is that multiple equivalent repre-

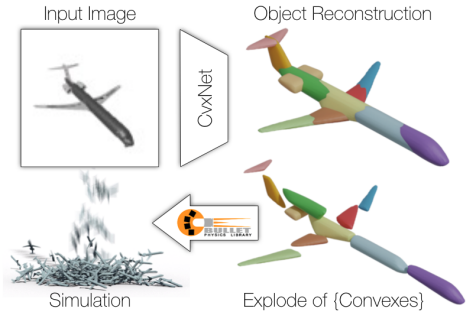

Figure 1: Our method reconstruct a 3D object from a input image as a collection of convex hulls. We visualize the explode of these convexes, which can then be readily used for physics simulation [7], as well as other downstream applications.

Workshop on Perception as Generative Reasoning, NeurIPS 2019, Vancouver, Canada.

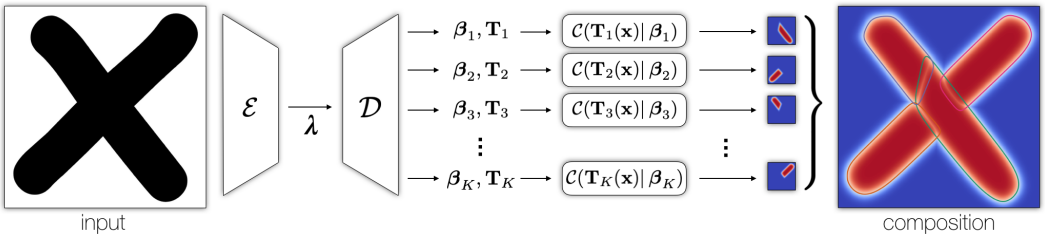

Figure 2: Our network approximates input geometry as a *composition* of convex elements. Note that it *does not* prescribe how the final image is generated, but merely output the shape $\{\boldsymbol{\beta}_k\}$ and pose $\{\mathbf{T}_k\}$ parameters of the abstraction.

sentations of 3D geometry exist because real-world applications need to perform different *operations* and queries on this data [3, Ch.1]. It's generally acknowledged that, in computer graphics, primitives enable highly efficient collision detection [21] and resolution [22]. In computer vision and robotics, part-based models provide a natural decomposition of an object into its semantic components. This creates a representation useful to reason about extent, support, mass, contact, ... quantities that are key in order to describe the scene, and eventually design action plans [13, 12].

Contributions In this paper, we propose a novel representation for geometry based on primitive decomposition. The representation is parsimonious, as we *approximate* geometry via a *small number* of *convex* elements, while we seek for their low-dimensional representation to be automatically inferred from data – without any human supervision. More specifically, inspired by recent works [23, 10, 16] we train our pipeline in a self-supervised manner: predicting the primitive configuration and their parameters by checking whether the reconstructed geometry matches the one of the target.

2 Method – CvxNets

Our object is represented via an indicator $\mathcal{O} : \mathbb{R}^3 \rightarrow [0, 1]$, and with $\partial\mathcal{O} = \{\mathbf{x} \in \mathbb{R}^3 | \mathcal{O}(x) = 0.5\}$ we indicate the surface of the object. The indicator function is defined such that $\{\mathbf{x} \in \mathbb{R}^3 \mid \mathcal{O}(x) = 0\}$ defines the outside of the object and $\{\mathbf{x} \in \mathbb{R}^3 \mid \mathcal{O}(x) = 1\}$ the inside. Given an input (e.g. image, point cloud, voxel grid) an encoder estimates the parameters $\{\boldsymbol{\beta}_k\}$ of our template representation $\hat{\mathcal{O}}(\cdot)$ with K primitives (indexed by k). We then evaluate the template at random sample points $\mathbf{x}$, and our training loss ensures $\hat{\mathcal{O}}(\mathbf{x}) \approx \mathcal{O}(\mathbf{x})$. The pipeline is illustrated in Figure 2.

Differentiable convex indicator We define a decoder that given a collection of (unordered) half-space constraints constructs the indicator function of a *single* convex object; such a function can be evaluated at any point $\mathbf{x} \in \mathbb{R}^3$. We define $\mathcal{H}_h(\mathbf{x}) = \mathbf{n}_h \cdot \mathbf{x} + \mathbf{d_h}$ as the signed distance of the point $\mathbf{x}$ from the h-th plane with normal $\mathbf{n}_h$ and offset $\mathbf{d_h}$. Given a sufficiently large number H of half-planes the signed distance function of any convex object can be approximated by taking the intersection ($\max$ operator) of the signed distance functions of the planes. To facilitate gradient learning, instead of maximum, we use the smooth maximum function LogSumExp and define the *approximate* signed distance function:

$$\Phi(\mathbf{x}) = \text{LogSumExp}\{\delta\mathcal{H}_h(\mathbf{x})\}, \tag{1}$$

Note this is an approximate SDF, as the property $\|\nabla\Phi(\mathbf{x})\| = 1$ is not necessarily satisfied $\forall\mathbf{x}$. We then convert the signed distance function to an indicator function $\mathcal{C} : \mathbb{R}^3 \rightarrow [0, 1]$:

$$\mathcal{C}(\mathbf{x}|\boldsymbol{\beta}) = \text{Sigmoid}(-\sigma\Phi(\mathbf{x})), \tag{2}$$

We denote the collection of hyperplane parameters as $\mathbf{h} = \{(\mathbf{n}_h, \mathbf{d}_h)\}$, and the overall set of parameters for a convex as $\boldsymbol{\beta} = [\mathbf{h}, \delta, \sigma]$. We treat σ as a hyperparameter, and consider the rest as the learnable parameters. As illustrated in Figure 3, the parameter δ controls the smoothness of the generated convex, while σ controls the sharpness of the transition of the indicator function. In summary, given a collection of hyperplane parameters, this differentiable module generates a *function* that can be evaluated at any position $\mathbf{x}$.

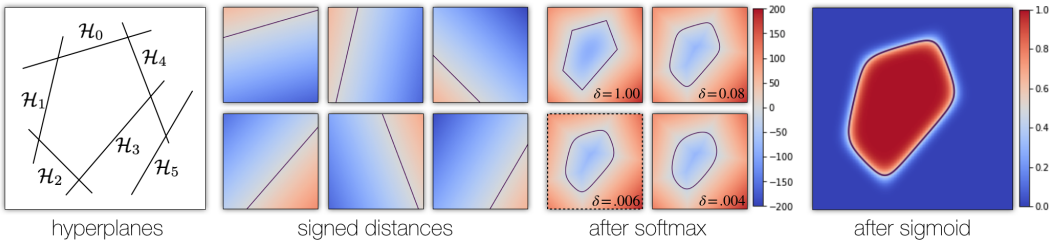

Figure 3: An example of how a collection of hyperplane parameters for an image specifies the indicator *function* of a convex object. The soft-max allows gradients to propagate through all hyperplanes and allows for the generation of *smooth* convex, while the sigmoid parameter controls the *slope* of the transition in the generated indicator. Note that our soft-max function is a LogSumExp.

Convex encoder/decoder While a sufficiently large set of hyperplanes can represent any convex object, one may ask whether it would be possible to discover some form of correlation between their parameters. Towards this goal, we employ the bottleneck auto-encoder architecture. Given an input, the encoder $\mathcal{E}$ derives a latent representation λ from the input Then, a decoder $\mathcal{D}$ derives the collection of hyperplane parameters. While in theory permuting the H hyperplanes generates the same convex, the decoder $\mathcal{D}$ correlates a particular hyperplane with a corresponding orientation.

Multi convex decomposition Having a learnable pipeline for a single convex object, we can now generalize the expressivity of our model by representing generic *non-convex* objects as *compositions* of convex elements [21]. To achieve this task an encoder $\mathcal{E}$ outputs a low dimensional latent representation of all K convexes $\boldsymbol{\lambda}$ that $\mathcal{D}$ decodes into a *collection* of K parameter tuples. Each tuple (indexed by k) is comprised of a shape code $\boldsymbol{\beta}_k$, and corresponding transformation $\mathbf{T}_k(\mathbf{x}) = \mathbf{x} + \mathbf{c}_k$ that transforms the point from the world coordinate to the local coordinate. $\mathbf{c}_k$ is the predicted translation vector (Figure 2).

Training losses First, we want the indicator function of our object $\mathcal{O}$ to be well approximated:

$$\mathcal{L}_{\text{approx}}(\omega) = \mathbb{E}_{\mathbf{x}\sim\mathbb{R}^3} \|\hat{\mathcal{O}}(\mathbf{x}) - \mathcal{O}(\mathbf{x})\|^2, \tag{3}$$

where $\hat{\mathcal{O}}(\mathbf{x}) = \max_k\{\mathcal{C}_k(\mathbf{x})\}$, and $\mathcal{C}_k(\mathbf{x}) = \mathcal{C}(\mathbf{T}_k(\mathbf{x})|\boldsymbol{\beta}_k)$. The application of the $\max$ operator produces a perfect union of indicator functions. We couple the approximation loss with a small set of auxiliary losses that enforce the desired properties of our decomposition:

$$\mathcal{L}_{\text{decomp}}(\omega) = \mathbb{E}_{\mathbf{x}\sim\mathbb{R}^3} \|\text{relu}(\sum_k\{\mathcal{C}_k(\mathbf{x})\} - \tau)\|^2 \quad \text{no overlap between convexes} \tag{4}$$

$$\mathcal{L}_{\text{loc}}(\omega) = \tfrac{1}{K} \sum_{\mathbf{x}\in\mathcal{N}_k^1} \|\mathbf{c}_k - x\|^2 \quad \text{“runaway” convexes are pulled to } \mathcal{O} \tag{5}$$

$$\mathcal{L}_{\text{guide}}(\omega) = \tfrac{1}{K} \sum_k \tfrac{1}{N} \sum_{\mathbf{x}\in\mathcal{N}_k^N} \|\mathcal{C}_k(\mathbf{x}) - \mathcal{O}(\mathbf{x})\|^2 \quad \text{each represents at least N samples} \tag{6}$$

$$\mathcal{L}_{\text{unique}}(\omega) = \tfrac{1}{H} \sum_h \|d_h\|^2 \quad \text{convexes have a unique parameterization} \tag{7}$$

3 Experiments

We use the ShapeNet [5] dataset in our experiments. We use the same voxelization, image renderings, and the train/test split as in Choy et. al. [6] and the same data split as [16]. Moreover, we use the same multi-view depth renderings as [10] for our {Depth}-to-3D experiments. We quantitatively compare our method to a number of self-supervised algorithms with different characteristics. First, we consider VP [23] that learns a parsimonious approximation of the input via (the union of) oriented boxes and SIF [10] that represents solid geometry as an iso-level of a sum of weighted Gaussians We also select OccNet [16] and AtlasNet [11], from the class of techniques that *directly* learn *non-interpretable* representations of implicit functions; in contrast to the previous methods, these solutions do not provide any form of shape decomposition.

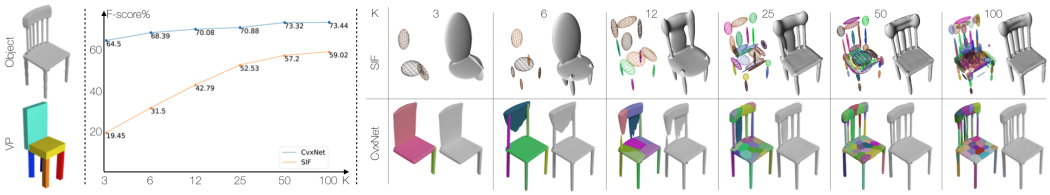

Figure 4: **Analysis of accuracy vs. # primitives** – (left) The ground truth object to be reconstructed and the single shape-abstraction generated by VP [23]. (middle) Quantitative evaluation (ShapeNet/Multi) of abstraction performance with an increase number of primitives – the closer the curve is to the top-left, the better. (right) A qualitative visualization of the primitives and corresponding reconstructions.

Category	IoU			Chamfer-L_1			F-Score		
	OccNet	SIF	Ours	OccNet	SIF	Ours	OccNet	SIF	Ours
airplane	0.728	0.662	**0.739**	0.031	0.029	**0.025**	79.52	71.40	**84.68**
bench	**0.655**	0.533	0.631	**0.041**	0.058	0.043	71.98	58.35	**77.68**
cabinet	**0.848**	0.783	0.830	0.138	**0.039**	0.048	71.31	59.26	**76.09**
car	**0.830**	0.772	0.826	0.071	**0.022**	0.031	69.64	56.58	**77.75**
chair	**0.696**	0.572	0.681	0.124	**0.102**	0.115	63.14	42.37	**65.39**
display	**0.763**	0.693	0.762	0.087	**0.049**	0.065	63.76	56.26	**71.41**
lamp	**0.538**	0.417	0.494	0.678	**0.216**	0.352	**51.60**	35.01	51.37
speaker	**0.806**	0.742	0.784	0.440	**0.067**	0.112	58.09	47.39	**60.24**
rifle	0.666	0.604	**0.684**	0.033	0.028	**0.023**	78.52	70.01	**83.63**
sofa	**0.836**	0.760	0.828	0.052	0.039	**0.036**	69.66	55.22	**75.44**
table	**0.699**	0.572	0.660	0.152	**0.112**	0.121	68.80	55.66	**71.73**
phone	**0.885**	0.831	0.869	0.022	0.024	**0.018**	85.60	81.82	**89.28**
vessel	**0.719**	0.643	0.708	0.070	**0.041**	0.052	66.48	54.15	**70.77**
mean	**0.744**	0.660	0.731	0.149	**0.064**	0.080	69.08	59.02	**73.49**

{Depth}-to-3D

Category	IoU			Chamfer-L_1				F-Score			
	OccNet	SIF	Ours	AtlasNet	OccNet	SIF	Ours	AtlasNet	OccNet	SIF	Ours
airplane	0.571	0.530	**0.598**	0.104	0.147	**0.065**	0.093	67.24	62.87	52.81	**68.16**
bench	**0.485**	0.333	0.461	0.138	0.155	**0.131**	0.133	54.50	**56.91**	37.31	54.64
cabinet	**0.733**	0.648	0.709	0.175	0.167	**0.102**	0.160	46.43	**61.79**	31.68	46.09
car	**0.737**	0.657	0.675	0.141	0.159	**0.056**	0.103	51.51	**56.91**	37.66	47.33
chair	**0.501**	0.389	0.491	0.209	0.228	**0.192**	0.337	38.89	**42.41**	26.90	38.49
display	0.471	0.491	**0.576**	**0.198**	0.278	0.208	0.223	**42.79**	38.96	27.22	40.69
lamp	**0.371**	0.260	0.311	**0.305**	0.479	0.454	0.795	33.04	**38.35**	20.59	31.41
speaker	**0.647**	0.577	0.620	**0.245**	0.300	0.253	0.462	35.75	**42.48**	22.42	29.45
rifle	0.474	0.463	**0.515**	0.115	0.141	**0.069**~	0.106	**64.22**	56.52	53.20	63.74
sofa	**0.680**	0.606	0.677	0.177	0.194	**0.146**	0.164	43.46	**48.62**	30.94	42.11
table	**0.506**	0.372	0.473	**0.190**	**0.189**	0.264	0.358	44.93	**58.49**	30.78	48.10
phone	**0.720**	0.658	0.719	0.128	0.140	0.095	**0.083**	58.85	**66.09**	45.61	59.64
vessel	0.530	0.502	**0.552**	0.151	0.218	**0.108**	0.173	**49.87**	42.37	36.04	45.88
mean	**0.571**	0.499	0.567	0.175	0.215	**0.165**	0.245	48.57	**51.75**	34.86	47.36

RGB-to-3D

Table 1: **Reconstruction performance on ShapeNet/Multi** – We evaluate our method against AtlasNet [11], OccNet [16] and SIF [10]. We provide in input either (left) a collection of depth maps or (right) a single color image. For AtlasNet [11], note that IoU cannot be measured as the meshes are not watertight. We omit VP [23], as it only produces a very rough shape decomposition.

Abstraction As our convex decomposition is learnt on a shape collection, the convex element produced by our decoder are in natural correspondence – e.g. we expect the same k-th convex to represent the leg of a chair in the chairs dataset. We evaluate our shape abstraction capabilities by varying the number of components and evaluate the trade-off between representation parsimony and reconstruction accuracy; we visualize this via pareto-optimal curves in the plot of Figure 4. We compare with SIF [10], and note that thanks to the generalized shape space of our model, our curve dominates theirs regardless of the number of primitives chosen.

Reconstruction We quantitatively evaluate the reconstruction performance against a number of state-of-the-art methods given inputs as multiple depth map images ({Depth}-to-3D) and a single color image (RGB-to-3D); see Table 1. We find that CvxNet is: ① consistently better than other *part decomposition* methods (SIF, VP, and SQ) which share the common goal of learning *shape elements*; ② in general comparable to the state-of-the-art reconstruction methods; ③ significantly better than the leading technique (OccNet [16]) when evaluated in terms of F-score, and tested on multi-view depth input. Note that SIF [10] first trains for the template parameters on ({Depth}-to-3D) with a reconstruction loss, and then trains the RGB-to-3D image encoder with a parameter regression loss; conversely, our method trains both encoder and decoder of the RGB-to-3D task from *scratch*.

4 Conclusions

We propose a differentiable representation of convex primitives that is amenable to learning, and whose inference result is *directly usable* in graphics/physics pipelines; see Figure 1. Our self-supervised technique provides more detailed reconstructions than very recently proposed part-based techniques (SIF [10] in Figure 4), and even consistently beats the leading reconstruction technique on multi-view input (OccNet [16] in Table 1).

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. 2018.

[2] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 1987.

[3] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[5] 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*, 2015.

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Comp. Vision*. Springer, 2016.

[7] Erwin Coumans and Yunfei Bai. PyBullet, a python module for physics simulation for games, robotics and machine learning. pybullet.org, 2016–2019.

[8] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[9] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *International Conference on 3D Vision (3DV)*, 2017.

[10] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. *arXiv preprint arXiv:1904.06447*, 2019.

[11] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mache approach to learning 3d surface generation. In *Proc. of Comp. Vision and Pattern Recognition (CVPR)*, 2018.

[12] Eric Heiden, David Millard, and Gaurav Sukhatme. Real2sim transfer using differentiable physics. Workshop on Closing the Reality Gap in Sim2real Transfer for Robotic Manipulation, 2019.

[13] Eric Heiden, David Millard, Hejia Zhang, and Gaurav S Sukhatme. Interactive differentiable simulation. *arXiv preprint arXiv:1905.10706*, 2019.

[14] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. of the European Conf. on Comp. Vision*, 2018.

[15] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. of Comp. Vision and Pattern Recognition (CVPR)*, 2018.

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.

[17] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[19] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proc. of Comp. Vision and Pattern Recognition (CVPR)*, 2018.

[20] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[21] Daniel Thul, Sohyeon Jeong, Marc Pollefeys, et al. Approximate convex decomposition and transfer for animated meshes. In *SIGGRAPH Asia 2018 Technical Papers*, page 226. ACM, 2018.

[22] Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)*, 2016.

[23] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proc. of the European Conf. on Comp. Vision*, 2018.

[25] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive o-cnn: a patch-based deep representation of 3d shapes. In *SIGGRAPH Asia 2018 Technical Papers*, page 217. ACM, 2018.

[26] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[27] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.