Disentangling Levels of Detail in Implicit Shape Generation with Explicit Bounding Proxies

Caleb N. Trotz
caleb_trotz@brown.edu
Brown University
Providence, Rhode Island, USA

R. Kenny Jones
russell_jones@brown.edu
Brown University
Providence, Rhode Island, USA

Rana Hanocka
ranahanocka@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Daniel Ritchie
daniel_ritchie@brown.edu
Brown University
Providence, Rhode Island, USA

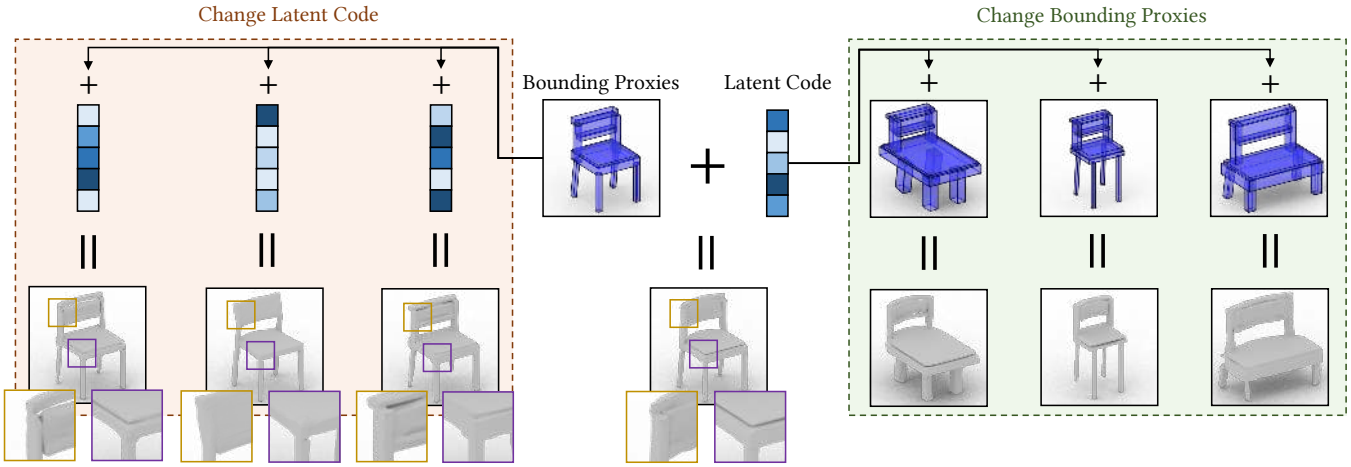

Figure 1: We present CageSDF, a neural implicit shape model which disentangles high-level object shape from low-level surface geometry. The generation of a shape is driven by a latent code and a set of bounding proxies (center). Modifying the latent code changes the low-level surface geometry while leaving the overall shape unchanged (left); modifying the bounding proxies changes the overall shape while keeping the geometric surface details constant (right). We achieve this disentanglement via a carefully-designed, self-supervised training scheme.

ABSTRACT

Considerable effort and enthusiasm has been directed towards building deep generative models of 3D shapes. Existing techniques can be divided into two categories: unstructured and structured models. Unstructured models capture high-level global shape attributes well, but lack user-driven inputs and control. By contrast, structured part-based generative model synthesis enables local editing, yet the coarse proxies are a poor approximation for surface geometry. In this work, we present an ongoing generative modeling project. This modeling project began with CageSDF, a model that achieves the best of both worlds by conditioning an implicit surface generator on an arrangement of bounding cuboid proxies. CageSDF enables user-driven control at the high-level shape structure by manipulating cuboid proxies, and at the surface detail level by modifying the generator's latent code. Key to achieving this disentanglement is a carefully-designed, self-supervised training procedure, which guarantees the model's latent code governs only the low-level surface geometry. We demonstrate that CageSDF is effective at generating novel shapes and provides both high and low level control. We lastly explore extensions designed to improve the visual quality and quantitative efficacy of this model for upcoming conference submissions.

Authors' addresses: Caleb N. Trotz, caleb_trotz@brown.edu, Brown University, Providence, Rhode Island, USA, 02912; R. Kenny Jones, russell_jones@brown.edu, Brown University, Providence, Rhode Island, USA, 02912; Rana Hanocka, ranahanocka@uchicago.edu, University of Chicago, Chicago, Illinois, USA, 60637; Daniel Ritchie, daniel_ritchie@brown.edu, Brown University, Providence, Rhode Island, USA, 02912.

0730-0301/2022/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; **Neural networks**.

KEYWORDS

shape analysis, deep generative models, implicit surface models

ACM Reference Format:
Caleb N. Trotz, R. Kenny Jones, Rana Hanocka, and Daniel Ritchie. 2022. Disentangling Levels of Detail in Implicit Shape Generation with Explicit Bounding Proxies. *ACM Trans. Graph.* 1, 1 (April 2022), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACKNOWLEDGEMENTS

There is a long list of folks I would like to thank for making this work possible; here are a few who I want to shout out in particular. To Daniel – thank you so much for taking me on as your advisee. Your guidance each week has been invaluable throughout our first and (soon, hopefully) second submission cycles. You have put in long hours to help me make the figures featured in this lovely paper, as well as come up with new plans to make our little project bigger and better. I learned from you that research is a marathon, not a sprint, and I appreciate that lesson very much.

To Spike, thank you for accidentally talking me into joining this wonderful and strange university back in 2018. Your openness and poignant advice have been two of the most valuable aspects of my undergraduate career here. Thank you also for helping me expand my geometric horizons and elucidating even the most abstruse concepts with wit and elan. If I possess even a tenth of your mathematical acumen at some point, I will be supremely content. To Bena, your support for my off-beat project ideas and your lucid pedagogical style have been key to my development as a mathematician and computer scientist. You made topology, mind bending as it can be, one of the highlights of my math coursework here. To Shriram, thank you so very much for putting up with me as an insufferable first year. Your humbling presence and words of wisdom helped me grow both personally and academically – as did teaching alongside you.

Kenny, I appreciate your insights into this weird alchemical science we call deep learning so much, in addition to your tireless work making our work sound entirely engaging on paper. It is no small feat to pull together documents like this on short time scales, but you truly made it happen with your work. Rana, your input has been critical to this project from its earliest stages back in 2020. I have learned so much from your work, and I am incredibly grateful to have the chance to call you a collaborator on this project.

Lastly, I would be remiss not to thank my friends and family, who have been indispensable in supporting me throughout my career here. Seiji, you are one of the few people I can count on to understand my ideas thoroughly and completely, and with very little need for elaboration or explanation. Getting to receive and share counsel with you has been one of the more rewarding aspects of my research these past two years. Penina, Nesya, Rachel, Aidan: your open doors and caring ears have made my work possible. I lean on you all as unwavering supporters in my journey. Gabe and Zach, your support as my older siblings has helped me realize what sort of person I want to be, and I can't thank you enough for that. Eliana, your artwork and words of encouragement have kept me going. Dad, Mom, Cynthia, Ed: thank you for giving me the space to be the son, student, and scientist I need to be, and for always answering when I call. Last but not least, Nana and Grandpa: without your support none of this would have happened - I thank you dearly. To all my friends and family, I appreciate you greatly and love you very much.

1 INTRODUCTION

Virtual 3D shape models are a critical resource in many domains, including industrial and product design, architecture, urban planning, gaming, animation, and virtual reality. But it is notoriously difficult to create new 3D models from scratch: each new model requires a significant time investment from a trained 3D artist. Graphics researchers have introduced a host of computation-assisted techniques in the pursuit of making 3D modeling easier: sketch-based interfaces [Cordier et al. 2016], part recombination systems [Chaudhuri et al. 2013, 2011; Funkhouser et al. 2004], and more. Of late, considerable effort and enthusiasm has been directed toward *generative models* of shapes, and deep generative models in particular. In theory, such models have the potential to synthesize a wide variety of 3D shapes at scale, suggest interesting new possibilities for design, and enable rapid exploration of shape spaces.

Many different types of deep generative shape models have been proposed, leveraging different underlying shape representations. These can be divided roughly into two categories: unstructured and structured models. Unstructured generative models synthesize entire shapes as volumetric occupancy grids [Wu et al. 2016], point clouds [Achlioptas et al. 2018] or implicit surfaces [Chen and Zhang 2019; Park et al. 2019]. Of these, implicit surface models currently produce the highest-quality output geometry, capturing complex surfaces and shapes of varying genus. While these models allow global exploration of the learned shape manifold via a latent code, local, more directed manipulations are less well-supported. In contrast, structured generative models incorporate knowledge of an object's part structure into their generative process, often generating proxy geometry (e.g. cuboids) for each part [Jones et al. 2020, 2021; Li et al. 2017; Mo et al. 2019]. The part proxies allow part-aware local editing, especially if the generated shape representation contains information about connections, symmetries, or other parametric relationships between parts [Jones et al. 2020, 2021]. However, cuboid proxies are a poor approximation for real object surface geometry. Some structured generative models include a step which converts these proxies into surface geometry, but they have limitations: the geometry is represented either as low-quality voxels or point clouds [Li et al. 2017; Mo et al. 2019] or genus-zero meshes [Gao et al. 2019; Yang et al. 2020]. Furthermore, cuboid proxies might not be useful as low-level geometric features if given to the model as conditioning in the absence of broader information about part structure, including the parametric relationships mentioned above. Rather than deciding which part-structural features are relevant to the task, some methods [Hertz et al. 2022] learn *contextualized* part representations that encode part-structural information alongside explicit per-part information.

In this paper, we present a new class of structure-aware generative shape model that achieves the best of both worlds. Specifically, we design a novel implicit shape generative model which is conditioned on an arrangement of cuboid proxies. The cuboid proxies can be provided manually, or they can be the output of another generative model [Jones et al. 2020, 2021; Mo et al. 2019]. A user

can manipulate the cuboid proxies to change the high-level shape of the object, or they can change the model's latent code to vary the low-level surface geometry (e.g. a flat vs. curved chair back); see examples in Fig. 1. In other words, our model effectively disentangles the high-level and low-level details of the shape. The cuboid proxies act as a sort of bounding control cage for the implicit surface, so we call this model CageSDF. Learning this disentangled model requires a carefully-designed training procedure to guarantee that the model's latent code governs only the low-level surface geometry. We accomplish this disentanglement via a form of self-supervision: using the cuboid proxies to create deformations of each training shape and forcing the training procedure to use the same latent code for all deformations.

We evaluate CageSDF on its ability to reconstruct existing shapes and generate high-quality novel shapes for multiple shape categories. We compare our method to a recent unstructured implicit surface generative model, showing that its outputs are on par with or higher quality. We also show that CageSDF successfully disentangles high-level and low-level shape details, and we demonstrate the necessity of our novel training process in enabling this disentanglement. Finally, we show how one can swap latent codes from one shape to another to perform a type of geometric style transfer while preserving high-level shape structure.

In summary, our contributions are:

- A new implicit surface generative model conditioned on a control cage formed from bounding proxies.
- A novel training procedure for this model which disentangles high-level shape from low-level surface geometry, making the model's latent code responsible solely for the latter.

2 RELATED WORK

Neural implicit shape models. CageSDF is a neural implicit shape model: a neural network which takes as input a point in space and returns as output an indication of how far that point is from the shape's surface. Multiple similar architectures for this class of function have been proposed, including DeepSDF [Park et al. 2019], IM-Net [Chen and Zhang 2019], Deep Level Sets [Michalkiewicz et al. 2019], and Occupancy Networks [Mescheder et al. 2019]. All of these methods use a training procedure in which the network is tasked with either classifying whether the input point is inside or outside of the surface, or with regressing the signed distance to the surface. Our CageSDF implementation is based on DeepSDF, though in principle its key ideas could be integrated into any of these base architectures. These works have shown that the distribution of point samples used during training is critical to the quality of the learned model. One recent paper proposes an importance-sampling-based approach, in which training samples are drawn with probability proportional to their distance to the ground truth surface [Davies et al. 2021]. We use a variant of this sampling method, tailored to our cuboid-conditioned approach.

Structured implicit shape models. The wave of initial work on neural implicit shape models inspired follow-up work which, like our approach, attempts to add structure to the implicit surface generation process. BSPNet [Chen et al. 2019] and CVXNet [Deng et al. 2020] generate an output distance field via predicting convex primitives as unions of half-spaces. This approach is better suited for modeling shapes with sharp edges, but this form of structuring does not enable editing or level-of-detail disentanglement. Another line of work learns to represent and reconstruct shapes via multiple implicit distance fields, where the spatial extent of each field is governed by a Gaussian [Genova et al. 2020, 2019]. These methods focus entirely on improving reconstruction accuracy, not editing, manipulation, or novel shape generation. A similar approach has also been proposed for learning implicit representations of large-scale scenes [Jiang et al. 2020]. The most related piece of prior work to ours in this space is DualSDF [Hao et al. 2020], which learns two coupled SDFs for a shape, one at a fine level of detail and the other at a coarse level of detail. Like our approach, the user can edit the coarse level representation and see the results propagated to the finer level. However, DualSDF's 'sphere cloud' representation for the coarse level of detail is a less-natural proxy for editing than our cuboids (especially for manufactured shapes), and their approach does not produce a disentangled latent space that allows changing fine detail while keeping coarse shape unchanged.

Learning to refine coarse geometry. As mentioned in Section 1, some structure-aware shape generative models work by first generating cuboid part proxies and then refining those proxies into final surface geometry. StructureNet [Mo et al. 2019] and ShapeAssembly [Jones et al. 2020] generate a point cloud within each cuboid, but these point clouds are a poor quality representation of surface geometry. GRASS [Li et al. 2017] generates a volumetric occupancy grid within each part cuboid, and voxel-based representations require prohibitively large memory footprint to represent surface details. SDM-Net [Gao et al. 2019] and DSM-Net [Yang et al. 2020] take a different approach, instead converting each part cuboid into a high-resolution mesh and then deforming this mesh toward a more detailed surface shape. While this approach produces a smooth, continuous surface representation, it cannot represent non-genus-zero parts. Like our work, DSM-Net also implements a disentangled latent space, though the disentangled factors are different from ours: "structure" (the part connectivity graph of the shape) vs. "style" (all aspects of part geometry).

Outside of shape generative modeling, there has been other work on learning to refine coarse surfaces into more detailed ones. Neural subdivision [Liu et al. 2020] and deep geometric texture synthesis [Hertz et al. 2020] can produce beautiful, detailed refinements of a coarse input shape, but they require an explicitly-provided style exemplar and can only synthesize stationary patterns (not semantic details). D^2IM-Net [Li and Zhang 2021] copies surface details from an input image, but it is only applicable in image-based reconstruction settings and only adds details to shape regions which are visible in the input image. Most recently, DECOR-GAN [Chen et al. 2021] proposed a method for refining a coarse voxel grid shape into a higher-resolution, detailed one guided by a style exemplar. Similar to us, they demonstrate the ability to swap styles between shapes. This approach was designed for 'upsampling' the outputs of low-res generative models, and its coarse voxel input is a poor proxy representation for editing and manipulation.

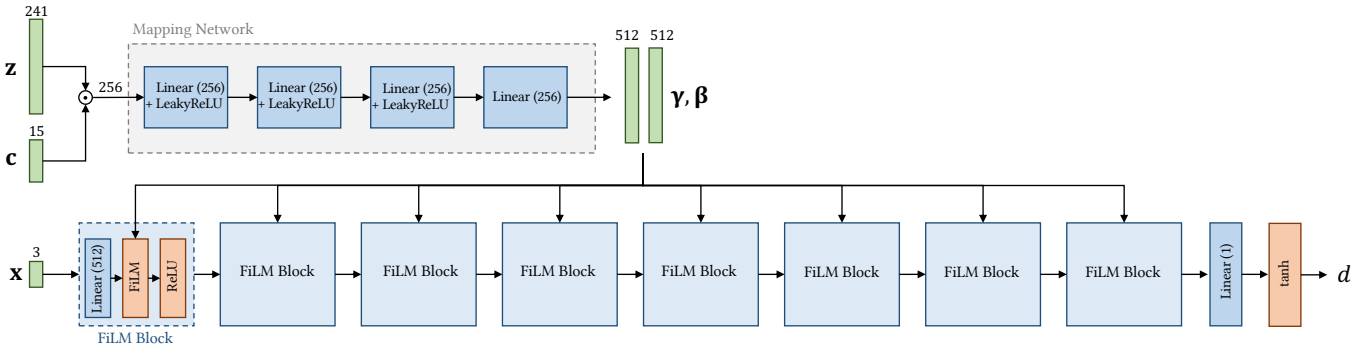

Figure 2: Network architecture overview. The input to our network is a point x which is mapped to a signed distance value d, conditioned on a latent code z and one of the cuboids c. The low-level surface geometry is effectively controlled by the FiLM blocks. We use blue to denote modules with learnable parameters, orange for fixed function modules, and green for data tensors on which the modules operate.

3 METHOD

Our goal is to learn a parameterized function f_θ which can take as input a point in space $\mathbf{x}$, a set of cuboids C, and a latent code $\mathbf{z}$, and produce as output a signed distance value d. The surface implicitly defined by this distance field should be tightly bounded by the cuboids C, with geometric deviations within the shape of the cuboids being governed by the latent code $\mathbf{z}$.

In this section, we describe our approach for solving the problem defined above. We first introduce our representation for f_θ, the CageSDF architecture (Section 3.1). We then describe how we create appropriate training data for this network: the datasets we use (Section 3.2), our self supervised training scheme which encourages disentangled levels of detail (Section 3.3), and our point sampling approach (Section 3.4). Finally, we discuss our training procedure (Section 3.5) and how to use the trained network to produce shape geometry (Section 3.6).

3.1 CageSDF Network Architecture

Fig. 2 illustrates the CageSDF architecture. It is based on DeepSDF [Park et al. 2019], and like that architecture, its backbone is a fully-connected, multi-layer perceptron (MLP) which maps an input point $\mathbf{x}$ to a signed distance value d. The network is further conditioned on a latent code $\mathbf{z}$ and one of the cuboids $\mathbf{c} \in C$ which contains the input point $\mathbf{x}$. This cuboid is represented as 15-dimensional vector (center, principal axis directions, and principal axis scales). These conditioning inputs should modulate how the network processes input points. Thus, we inject these inputs into the network's data flow via featurewise linear modulation (FiLM) [Perez et al. 2018], scaling and shifting the network's intermediate outputs (similar in spirit to the adaptive instance normalization layers in StyleGAN [Karras et al. 2018]). Specifically, $\mathbf{z}$ and $\mathbf{c}$ are concatenated and then processed via a nonlinear mapping network to produce a set of elementwise scale parameters γ and shift parameters β. FiLM was originally developed for convolutional networks but has recently been shown effective when incorporated in fully connected networks for implicit function representation [Chan et al. 2020].

3.2 Datasets

In this paper, we train CageSDF on three categories of shapes: chairs, tables, and vases. Training the network requires, for each category, a dataset of shapes with bounding cuboids for their parts. For the chair and table categories, we use the ShapeAssembly dataset [Jones et al. 2020], as the cuboids are produced by programs which provide parametric editing capabilities. For vases, we use the StructureNet dataset [Mo et al. 2019], which pairs part-segmented shapes with minimum volume oriented bounding boxes for each part. We note that though StructureNet provides semantic labels for each part, our model does not require part labels and does not use them.

The ShapeAssembly cuboids, while derived from those provided by StructureNet, are modified slightly when coerced into program form and are not guaranteed to bound their respective part geometries. However, our model requires this property. Thus, we design an optimization procedure which modifies the program parameters such that the bounding property holds (see Appendix for details).

3.3 Disentangling Levels of Detail

We wish for the surface that our network f_θ generates to be tightly bounded by the cuboids, and for geometric deviation from these cuboids to be governed by $\mathbf{z}$. Trained naively, there is no reason we should expect the network architecture described in Section 3.1 to do this. The network is free to learn to base its outputs on the cuboids alone (while ignoring the latent code), the latent code alone (while ignoring the cuboids), or some inscrutable combination of latent code and cuboids. Therefore, we must actively take steps to force the network to learn the representation we desire.

Ideally, if we had access to a dataset in which multiple shapes were bounded by the same set of cuboids C, this problem would disappear: the network would have to learn to use the latent code to disambiguate between the different shapes. Real-world datasets are not structured so conveniently, but we can approximate this desired setup with self-supervised data augmentation. While it is not obvious to how synthetically produce shapes that share the same bounding cuboids but should have different latent codes, one can do the opposite: produce multiple shapes which have different

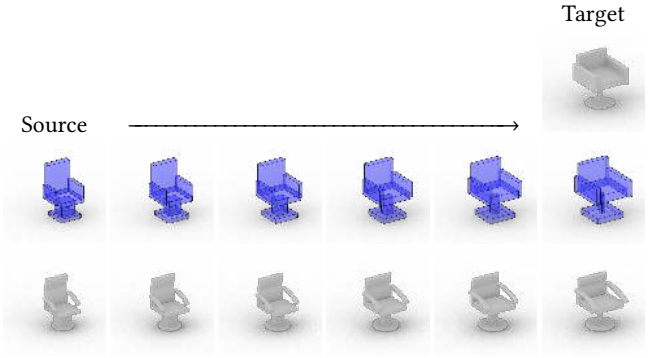

Figure 3: Illustration of our deformation-based self-supervision for disentanglement. The Source shape's cuboids C are interpolated toward the Target shape's cuboids C^T, and the deformation is transferred to the underlying Source shape via trilinear interpolation.

bounding cuboids but should share the same latent code. Specifically, given a shape $\mathcal{S}$ bounded by cuboids C, we create a set of tuples $\{(\mathcal{S}', C')\}$ by deforming C to produce C' and interpolating the deformation to $\mathcal{S}$ to create $\mathcal{S}'$. The $\mathcal{S}'$ have the same low-level surface geometry as $\mathcal{S}$, but their corresponding cuboids C' are different from C. At training time, we will force all the $\mathcal{S}'$ to be reconstructed with the same latent code. This procedure is similar in spirit to a recent method for disentangling shape and appearance in image generative models [Ojha et al. 2021].

Using random deformations to produce the C' may introduce non-semantic distortions in the resulting shapes $\mathcal{S}'$. To ensure that our deformed shapes remain near the manifold of plausible shapes, we instead construct deformations by interpolating the cuboids C toward another set of cuboids from the dataset C^T (Figure 3). In selecting C^T, we must take care that it is a structurally-compatible interpolation target for C. For ShapeAssembly cuboids, we select C^T from the set of programs which are equivalent to C up to continuous parameters. For StructureNet cuboids, we select the set of instances with part hierarchies which are equivalent up to continuous parameters.

To produce the deformed shapes $\mathcal{S}'$ from deformed cuboids C', the cuboid deformations are transferred to the source mesh for $\mathcal{S}$ by trilinear interpolation. If a mesh vertex is contained by multiple cuboids, its new position is determined by the average of the positions dictated by each of those cuboids.

For each shape in our training set, we select an interpolation target and produce an interpolation sequence of length six; from these, we hold out the middle two for testing. We denote the jth deformation of the ith shape in a dataset as $(\mathcal{S}_{ij}, C_{ij})$. All told, this self-supervised augmentation procedure results in a training dataset of 7,600 chairs, 9,800 tables, and 1,100 vases.

3.4 Point Sampling

Given a deformation-augmented dataset $\{(\mathcal{S}_{ij}, C_{ij})\}$, the final data preparation step is to sample (point, cuboid, SDF) tuples $(\mathbf{x}, \mathbf{c}, d)$ to use as training data. In order to properly sample the details in each shape, we use the importance sampling strategy proposed in [Davies et al. 2021] to sample 250,000 points from each shape. However, this may leave certain cuboids (*e.g.,* small or low frequency regions) without any point samples. To alleviate this, we additionally sample 25,000 points uniformly across each shape. For each sampled point $\mathbf{x}_{ij}^k$, we assign it to the cuboid $\mathbf{c}_{ij}^k$ which most contains it (i.e. the cuboid with respect to which it has the minimum SDF). We denote the set of tuples which constitutes our training set as $\{(\mathbf{x}_{ij}^k, \mathbf{c}_{ij}^k, d_{ij}^k)\}$, where k indexes the combined set of points for all cuboids in each C_{ij}.

3.5 Training Procedure

Given a training set of tuples $\{(\mathbf{x}_{ij}^k, \mathbf{c}_{ij}^k, d_{ij}^k)\}$, we train the network f_θ using an autodecoder setup, as in DeepSDF [Park et al. 2019]. Specifically, we assign an optimizable latent code variable $\mathbf{z}_i$ to each shape, and we minimize the difference between the ground truth SDF value d_{ij}^k and the network's predicted value:

$$\min_{\theta, \mathbf{z}_i} \frac{1}{NM} \sum_{i,j=1}^{N,M} \frac{1}{P_{ij}} \sum_{k=1}^{P_{ij}} |f_\theta(\mathbf{x}_{ij}^k, \mathbf{z}_i, \mathbf{c}_{ij}^k) - d_{ij}^k| + \frac{\lambda}{N} \sum_{i=1}^{N} ||\mathbf{z}_i||_2^2$$

In the above, N is the number of training shapes, M is the number of deformations per shape, and P_{ij} is the number of training points for the deformed shape $(\mathcal{S}_{ij}, C_{ij})$. The second term is a regularizer which encourages the optimized latent codes to have a small norm; this serves a similar role as the KL divergence loss in a variational autoencoder [Kingma and Welling 2014]. As in DeepSDF, we set $\lambda = 10^{-2}$.

We minimize this loss function using stochastic gradient descent with a batch size of 64. We use the Adam optimizer [Kingma and Ba 2014] with an initial learning rate of 10^{-4}, decaying by a factor of two every 500 epochs. We train the network for 1000 epochs.

3.6 Using the Trained Model

Once trained, we can use the network to produce new output geometry using a process similar to that of DeepSDF: given a latent code, we form a regular volumetric grid of points in space, evaluate the network at all points, and then pass the resulting discrete signed distance field to marching cubes to extract the zero isosurface [Lorensen and Cline 1987]. In our case, evaluating the network at each point also requires an input cuboid. For grid points that fall inside exactly one cuboid, that cuboid is used as input. For points that fall inside more than one cuboid, we assign them to the cuboid which they are most inside (as measured by SDF). Grid points that fall outside of all cuboids have their SDF set to an outside value (0.1, in our implementation).

At this point, we have left open the question of how to obtain an input latent code for the network. In the next section, we evaluate multiple latent code sampling options.

4 RESULTS & EVALUATION

We perform a variety of qualitative and quantitative experiments in order to demonstrate the effectiveness of our technique. Experiments were run on a single Tesla V100 GPU, and took between 1-4.5 days to train (where larger datasets require more training time).

Table 1: Quantitative results on reconstructing shapes. We report the mean over training set. The proxy cuboids guide CageSDF's reconstructions, producing more accurate results than DeepSDF.

Category	Method	Chamfer Distance ⇓
Chair	DeepSDF	0.000845
	CageSDF	**0.000447**
Table	DeepSDF	0.0259
	CageSDF	**0.0247**
Vase	DeepSDF	0.0228
	CageSDF	**0.0135**

4.1 Reconstruction

As a preliminary test, we evaluate CageSDF's ability to accurately reconstruct the shapes in its training set. Here, we compare against DeepSDF as a baseline. We measure reconstruction accuracy using Chamfer distance. Table 1 shows the results of this experiment, and Fig. 4 shows some qualitative examples. As expected, having the 'scaffolding' of the cuboid proxies improves CageSDF's ability to reconstruct its training shapes, especially in the presence of more complex part arrangements (e.g. the legs of the tables shown in the middle rows).

4.2 Disentanglement

CageSDF is designed to disentangle high-level shape from low-level surface geometry. Here we evaluate how well it meets that goal. Specifically, we evaluate how well the model reconstructs held-out shapes from the deformation interpolation sequences described in Section 3.3, i.e. shapes with deformed cuboids but the same latent code. If the model reconstructs these well, then it has disentangled these factors successfully. By contrast, we would expect an entangled model to vary the low-level surface geometry as well as the high-level shape, which would result in poor reconstruction accuracy. We compare CageSDF with an ablation that does not use the self-supervised training augmentation from Section 3.3.

Table 2 shows the results of this experiment, and Fig. 5 shows a qualitative comparison. Trained with our self-supervised disentanglement augmentation, CageSDF does a good job of reconstructing deformation sequences of shapes that have different bounding cuboids but the same latent code (including steps of the sequence that were not seen at training time, shown as highlighted columns in Fig. 5). Without the self-supervised disentanglement procedure, the network does considerably worse: incorrectly changing the low-level surface geometry of the shape (middle) or omitting geometry entirely (bottom). Furthermore, our self-supervised training also produces a significant increase in reconstruction accuracy.

4.3 Latent Code Swapping

Here, we demonstrate the value of swapping one shape's latent code for that of another shape. Since changing the latent code governs only the low-level surface geometry of the shape while keeping the high-level shape the same, this swapping can be viewed as a sort of "style transfer." Fig. 6 shows multiple examples of swaps applied

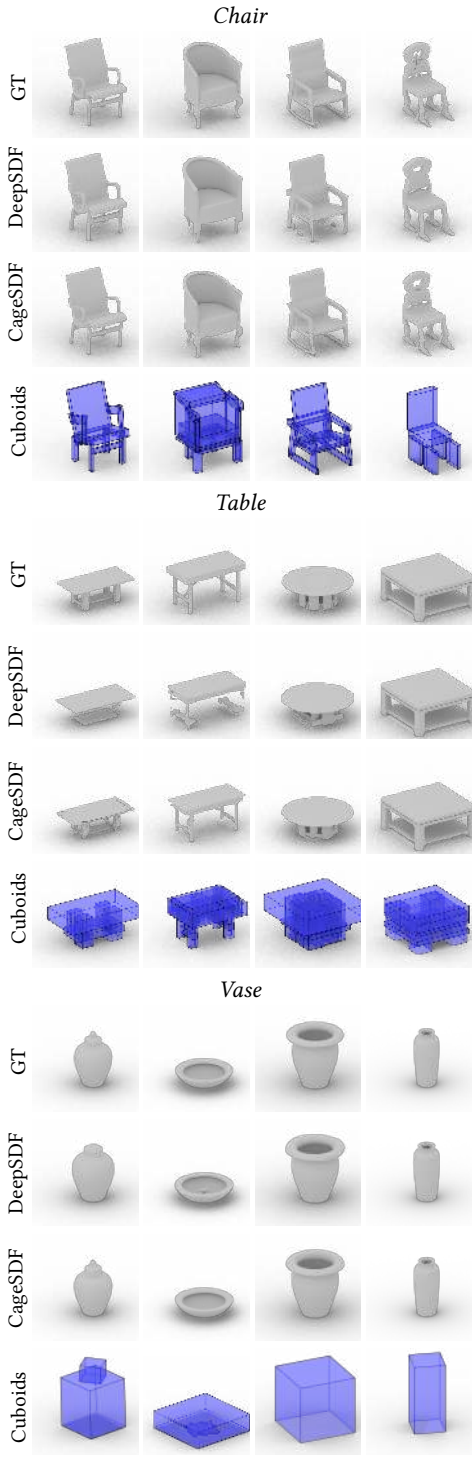

Figure 4: Qualitative comparison between different methods on reconstructing training shapes. CageSDF can rely on its proxy cuboids to handle structurally-complex regions (e.g. table legs, middle rows).

Table 2: Evaluating how well CageSDF disentangles high-level from low-level shape details, with and without our deformation-based self-supervised training scheme. Evaluation is based on reconstruction accuracy for held-out steps of deformation interpolation sequences from the training data. Our self-supervised training scheme improves reconstruction accuracy significantly.

Condition	Chamfer Distance ⇓
with augmentation	**0.000617**
w/out augmentation	0.00274

to different shapes. Observe how the high-level shape defined in "Cuboids A" is preserved, while incorporating surface detail from the latent code of "Shape B". Note how the chair in the first row obtains a curved back, the legs of the shape in the third row become ornately decorated, the table top sides become sloped in the last row.

4.4 Novel Shape Generation

Finally, we evaluate how well CageSDF works in the context of novel shape generation. In this experiment, we compare four different methods:

- **DeepSDF**: Generating novel shapes by training a DeepSDF model and sampling latent codes from an isotropic Gaussian fit to the codes of the training data.
- **ShapeAssembly + CageSDF (⫫)**: Generating cuboids using ShapeAssembly[Jones et al. 2020], and then conditioning CageSDF on those cuboids. The CageSDF latent code is drawn from a Gaussian kernel density estimator (KDE) constructed over the codes for the training set. We use Silverman's rule of thumb to set the kernel bandwidth [Silverman 1986]. The ⫫ symbol indicates that the latent code is sampled independently of the cuboids.
- **ShapeAssembly + CageSDF (⫫̸)**: The same as the above, except the weights of each Gaussian component of the KDE are set based on the similarity of that component's corresponding training set cuboids C_i with the input ShapeAssembly cuboids $\hat{C}$. Specifically, the weights are computed as $\text{softmax}([-d(\hat{C}, C_1), -d(\hat{C}, C_2), \ldots, -d(\hat{C}, C_n)]$, where d is Chamfer distance. The ⫫̸ symbol indicates that the latent code is sampled in a manner dependent on the cuboids: while a set of cuboids may admit multiple types of low-level surface geometry, there can be correlation between them.

We evaluate the performance of these different models using the following metrics:

- **Frechet Distance (FD):** The Frechet Distance [Heusel et al. 2017] between a set of generated shapes and a held-out validation set of shapes, as measured in the feature space of a PointNet [Qi et al. 2017] classifier trained on ShapeNetCore [Chang et al. 2015].
- **Generalization (Gen):** The average Chamfer distance between a generated shape and its nearest neighbor in the training set.

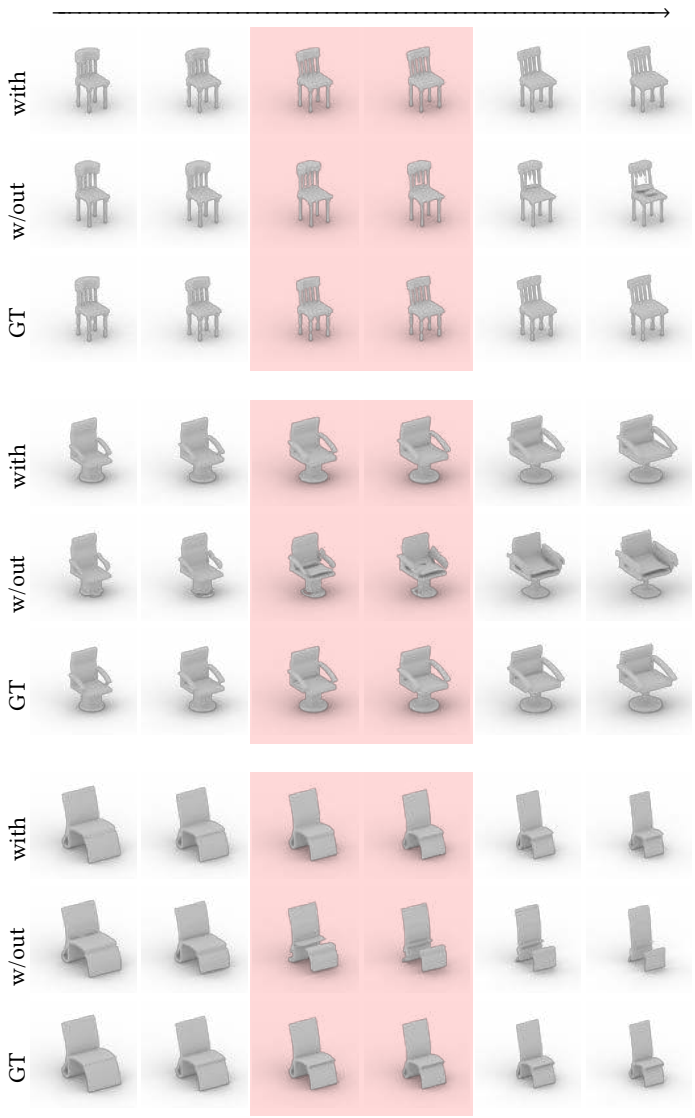

Figure 5: Qualitative examples for the disentanglement experiment described in Section 4.2. Training with self-supervised deformation augmentation allows CageSDF to better reconstruct shapes with different bounding proxies, but with the same latent code. The highlighted columns correspond to shapes that were not seen at training time.

- **Coverage (Cov):** The average Chamfer distance between a shape in the validation set and its nearest neighbor in a set of generated shapes.
- **Variety (Var):** The average Chamfer distance between a shape in a generated set of shapes and its nearest neighbor in the same set of generated shapes.

The results of this experiment are shown in Table 3, and we show qualitative comparisons in Fig. 7. Numerically, the methods which use CageSDF achieve the best Frechet Distance with respect to the validation set, and they also cover the validation set better. DeepSDF

Figure 6: Examples of swapping latent codes from one shape to another. Observe that the high-level shape defined by "Cuboids A" is preserved, while incorporating surface detail from "Shape B". Note the curved chair back in the top row, the legs of the chair in the third row, and how the vase becomes less tapered and acquires a rounded lip in the last row.

Table 3: Quantitatively evaluating different methods' novel shape generation abilities. Our technique achieves the best Frechet Distance (FD) and Coverage (Cov) in the validation set. DeepSDF produces more generalization and variety, but the qualitative results in Fig. 7 are poor (i.e. over-generalization). We do not include raw ShapeAssembly results, as they provide only structural approximations to the true shape distributions.

Category	Method	FD ⇓	Gen ⇑	Cov ⇓	Var ⇑
Chair	DeepSDF	489.4	**0.125**	0.106	**0.087**
	ShapeAssembly + CageSDF (⫫)	57.4	0.093	0.101	0.073
	ShapeAssembly + CageSDF (⫰)	**55.0**	0.094	**0.100**	0.073
Table	DeepSDF	2121.8	**0.237**	0.134	**0.098**
	ShapeAssembly + CageSDF (⫫)	**563.9**	0.162	0.117	0.082
	ShapeAssembly + CageSDF (⫰)	572.3	0.164	**0.114**	0.081

produces more generalization and variety; however, inspecting the qualitative results, it is clear that this additional variety/generality manifests as more poor-quality outputs. We note that for CageSDF, the dependent latent code sampling method usually produces better results than the independent one, especially for tables.

4.5 Failure Cases

CageSDF also exhibits some failure cases, which are illustrated in Fig. 8. First, though CageSDF often generates symmetric geometry given symmetric input cuboids, it is not guaranteed to do so exactly (left). Second, the final surface inherits artifacts present in the input bounding cuboids. For instance, 'floating' cuboids can lead to disconnected parts (middle). When cuboids come from a generative model of shape structure, a poor-quality output from that model hamstrings CageSDF, which is constrained to follow these cuboids (right).

5 CONCLUSION & FUTURE WORK

We presented CageSDF, a technique for generating implicit 3D shapes based on an arrangement of bounding cuboid proxies. A considerable advantage of our technique is the ability to incorporate user-driven control at both the high-level shape structure (by manipulating cuboid proxies), and at the surface detail level (by modifying the generator's latent code). The key to achieving this disentanglement between high-level and low-level shape details is a self-supervised training procedure designed to ensure the model's latent code governs only the low-level surface geometry. We showed that CageSDF performs on par or better than DeepSDF in terms of training shape reconstruction quality and novel shape generation. We also demonstrated CageSDF's editing facilities via cuboid manipulation and latent code swapping.

A limitation of our technique is its requirement of shapes with per-part cuboid annotations, whereas unstructured alternatives, such as DeepSDF, can be trained on unannotated shape collections. There is a trade-off between enabling user-directed control and requiring more richly-annotated input data, which we plan to explore more in the future. One interesting direction to explore for

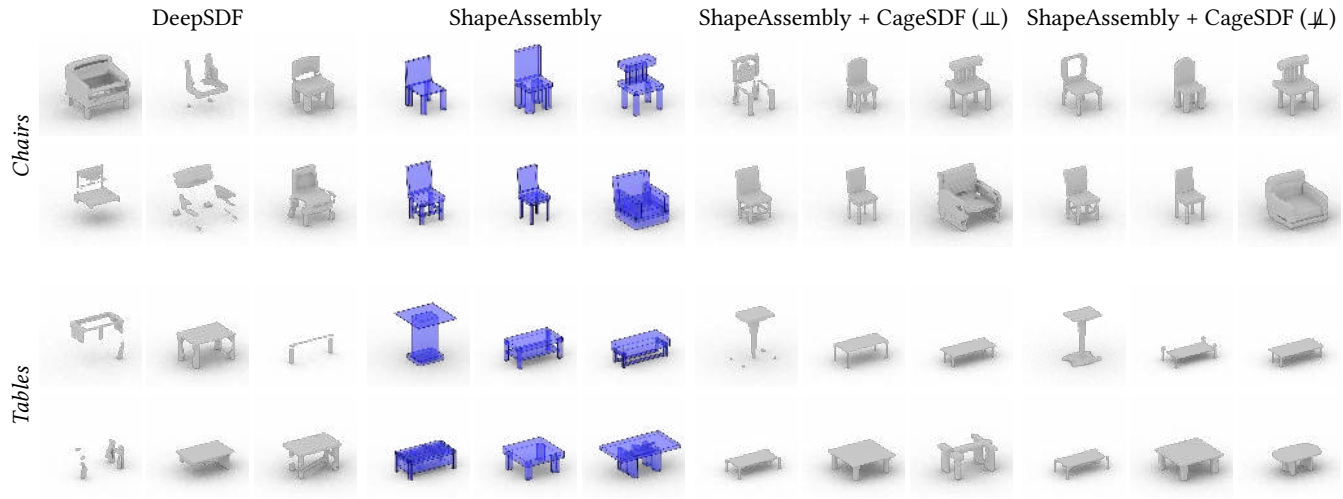

Figure 7: Quantitative comparison of different methods on the task of novel shape generation. DeepSDF produces poor output geometries. While ShapeAssembly cuboids alone are a weak approximation of an object's surface, combining them with CageSDF can produce good-quality output surfaces. Sampling a latent code dependent on these cuboids (⫰) helps keep CageSDF from manifesting the kinds of errors that DeepSDF produces.

Figure 8: Failure cases of our method. Left: failure to enforce bilateral symmetry; middle: disconnected leg geometry due to disconnected proxy cuboids; right: implausible output due to poor quality cuboids generated by ShapeAssembly.

eliminating this trade-off would be to leverage recent work on unsupervised abstraction of shapes into consistent cuboids [Sun et al. 2019; Yang and Chen 2021].

There are multiple open avenues for future work, some of which we are currently developing in preparation for future conference submissions. Our formulation of CageSDF uses a single, global latent code to govern low-level surface geometry across the entire shape. In our developing formulaion, we obtain more spatially-localized control by instead learning a collection (or hierarchy) of latent codes to govern the geometry for different cuboids. This is inspired by SPAGHETTI [Hertz et al. 2022], a Transformer-based architecture that harnesses self-attention [Vaswani et al. 2017] to contextualize each part-level encoding with global information and information about other parts within the same shape. As mentioned in Section 3.1, while our implementation is based on DeepSDF, the ideas behind CageSDF are in principle applicable to any implicit surface generative model. As such, our developing formulation uses occupancy probability prediction networks, initially explored in [Mescheder et al. 2019]. These are simpler to predict and more globally consistent, as we can combine occupancy values predicted by different part encodings without losing information using a max operator. In contrast, part-wise signed distances might not be consistent when predicted on local surface geometry (e.g. if a part surface not currently being predicted is closer to the query point in question than the part surface to which we are predicting signed distance). Exploring integration with other such models could also be fruitful, especially recent models which impose interpretable structure onto the generated distance field [Yu et al. 2021]. It could also be promising to investigate conditioning the implicit surface model on more structural information beyond the raw part proxies: where parts attach to one another, where symmetries exist between parts, semantic labels of parts, etc.

FUNDING

This work is supported by NSF CAREER Award #1941808 given to Daniel Ritchie. I thank NSF for helping fund stipends and procure hardware for this project.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning Representations and Generative Models for 3D Point Clouds. In *International Conference on Machine Learning (ICML)*.

Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. 2020. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. In *arXiv*.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012* (2015).

Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. 2013. AttribIt: Content Creation with Semantic Attributes. *ACM Symposium on User Interface Software and Technology (UIST)* (Oct. 2013).

Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. 2011. Probabilistic Reasoning for Assembly-Based 3D Modeling. *ACM Trans. Graph.* 30, 4, Article 35 (July 2011), 10 pages.

Zhiqin Chen, Vladimir G. Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. 2021. DECOR-GAN: 3D Shape Detailization by Conditional Refinement. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2019. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. arXiv:1911.06971 [cs.CV]
Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Frederic Cordier, Yotam Gingold, Even Entem, Marie-Paule Cani, and Karan Singh. 2016. Sketch-based Modeling. In *Eurographics 2016*. The Eurographics Association, Lisbonne, Portugal. https://doi.org/10.2312/egt.20161035
Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. 2021. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. arXiv:2009.09808 [cs.GR]
Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. CvxNet: Learnable Convex Decomposition. (June 2020).
Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by Example. In *ACM SIGGRAPH 2004 Papers*.
Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao (Richard) Zhang. 2019. SDM-NET: Deep Generative Network for Structured Deformable Mesh. In *SIGGRAPH Asia*.
Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. 2020. Local Deep Implicit Functions for 3D Shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4857–4866.
Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7154–7164.
Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. 2020. DualSDF: Semantic Shape Manipulation using a Two-Level Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep Geometric Texture Synthesis. *ACM Trans. Graph.* 39, 4, Article 108 (2020). https://doi.org/10.1145/3386569.3392471
Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. https://doi.org/10.48550/ARXIV.2201.13168
Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*.
Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. 2020. Local Implicit Grid Representations for 3D Scenes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.
R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. 2020. ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis. *ACM Transactions on Graphics (TOG), Siggraph Asia 2020* 39, 6 (2020), Article 234.
R. Kenny Jones, David Charatan, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. 2021. ShapeMOD: Macro Operation Discovery for 3D Shape Programs. In *SIGGRAPH 2021*.
Tero Karras, Samuli Laine, and Timo Aila. 2018. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv:1812.04948 [cs.NE]
Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).
Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*.
Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 52.
Manyi Li and Hao Zhang. 2021. D^2IM-Net: Learning Detail Disentangled Implicit Fields from Single Images. In *CVPR 2021*.
Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. 2020. Neural Subdivision. *ACM Trans. Graph.* 39, 4, Article 124 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392418
William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 163–169. https://doi.org/10.1145/37401.37422
Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders P. Eriksson. 2019. Deep Level Sets: Implicit Surface Representations for 3D Shape Inference. *CoRR* abs/1901.06802 (2019).
Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. 2019. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. In *SIGGRAPH Asia*.
Utkarsh Ojha, Krishna Kumar Singh, and Yong Jae Lee. 2021. Generating Furry Cars: Disentangling Object Shape and Appearance across Multiple Domains. In *International Conference on Learning Representations*.
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *AAAI 2018*.
Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 652–660.
B. W. Silverman. 1986. *Density estimation for statistics and data analysis*. Chapman and Hall London ; New York. 175 pages : pages.
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 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356529
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*.
Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Lin Gao. 2020. DSM-Net: Disentangled Structured Mesh Net for Controllable Generation of Fine Geometry. arXiv:2008.05440 [cs.GR]
Kaizhi Yang and Xuejin Chen. 2021. Unsupervised Learning for Cuboid Shape Abstraction via Joint Segmentation from Point Clouds. In *SIGGRAPH 2021*.
Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2021. CAPRI-Net: Learning Compact CAD Shapes with Adaptive Primitive Assembly. arXiv:2104.05652 [cs.CV]

APPENDIX: SHAPEASSEMBLY CUBOID FITTING

Here we describe our procedure (referenced in Section 3.2) for optimizing the parameters of a program in the ShapeAssembly dataset such that its output leaf cuboids are proper bounding volumes of their respective part geometries. Each ShapeAssembly dataset program was computed from a shape in the PartNet dataset. For each leaf part $\mathcal{P}$ in the shape, we sample $N_{\mathbf{x}} = 10,000$ points $\mathbf{x}$ on its surface. We then optimize the parameters θ of the ShapeAssembly program P such that for all leaf parts $\mathcal{P}_i$, these points fall within their corresponding leaf cuboid $\mathsf{P}(\theta)_i$ while keeping that leaf cuboid close to the minimum volume oriented bounding box $\hat{\mathbf{c}}_i$ of the part $\mathcal{P}_i$. This optimization is performed by minimizing the following loss function:

$$\min_{\theta} \frac{1}{N_{\mathcal{P}} N_{\mathbf{x}}} \sum_{i=1}^{N_{\mathcal{P}}} \sum_{j=1}^{N_{\mathbf{x}}} \max(\text{SDF}(\mathsf{P}(\theta)_i, \mathbf{x}_{ij}) + 0.01, 0) + \frac{1}{N_{\mathcal{P}}} \sum_{i=1}^{N_{\mathcal{P}}} ||\mathsf{P}(\theta)_i - \hat{\mathbf{c}}_i||_1$$

where $N_{\mathcal{P}}$ denotes the number of leaf parts in the shape, and $\text{SDF}(\mathbf{c}, \mathbf{x})$ is the signed distance of the point $\mathbf{x}$ to the cuboid $\mathbf{c}$. The first term tries to make the cuboids tight bounding volumes for their respective points (penalizing any point-to-cuboid signed distance values larger than -0.01). The second term is a regularizer that tries to keep the optimized cuboids as similar to minimum volume bounding cuboids as possible. The trade-off between these two terms is necessary because the minimum volume bounding cuboids may not be expressible as the output of a valid ShapeAssembly program. We minimize this loss using the Adam optimizer [Kingma and Ba 2014] with learning rate 10^{-3} for 2000 iterations.