Learning Part Generation and Assembly for Structure-Aware Shape Synthesis

Jun Li, Chengjie Niu, Kai Xu[*]

National University of Defense Technology

Abstract

Learning powerful deep generative models for 3D shape synthesis is largely hindered by the difficulty in ensuring plausibility encompassing correct topology and reasonable geometry. Indeed, learning the distribution of plausible 3D shapes seems a daunting task for the holistic approaches, given the significant topological variations of 3D objects even within the same category. Enlightened by the fact that 3D shape structure is characterized as part composition and placement, we propose to model 3D shape variations with a *part-aware* deep generative network, coined as *PAGENet*. The network is composed of an array of per-part VAE-GANs, generating semantic parts composing a complete shape, followed by a part assembly module that estimates a transformation for each part to correlate and assemble them into a plausible structure. Through delegating the learning of part composition and part placement into separate networks, the difficulty of modeling structural variations of 3D shapes is greatly reduced. We demonstrate through both qualitative and quantitative evaluations that *PAGENet* generates 3D shapes with plausible, diverse and detailed structure, and show two applications, i.e., semantic shape segmentation and part-based shape editing.

1 Introduction

Learning deep generative models for 3D shape synthesis has been attracting increasing research interest lately. Despite the notable progress made since the seminal work of (Wu et al. 2016), one major challenge still demanding more attention is how to ensure the *structural correctness* of the generated shapes. Most existing generative models are structure-oblivious. They tend to generate 3D shapes in a holistic manner, without comprehending its compositional parts explicitly. Consequently, when generating 3D shapes with detailed structures, the details of complicated part structures are often blurred or even messed up (see Figure 1(c)).

To alleviate this issue, one common practice is to increase the shape resolution to better capture fine-grained details, with the cost of increased training examples and learning time. The main reason behind is that the 3D shape representation employed by those models, e.g., volume or point cloud, are oblivious to shape structures. With such representations, part information is not encoded and thus cannot be decoded either during the generation process.

In the field 3D shape analysis, a consensus about the "structure" of a shape is the combination of part composition and the relative positioning between parts (Mitra et al. 2013; Xu et al. 2016). Following that insight, we approach the

(a) Part-wise generation   (b) After assembling   (c) Holistic generation   (d) Part-based editing
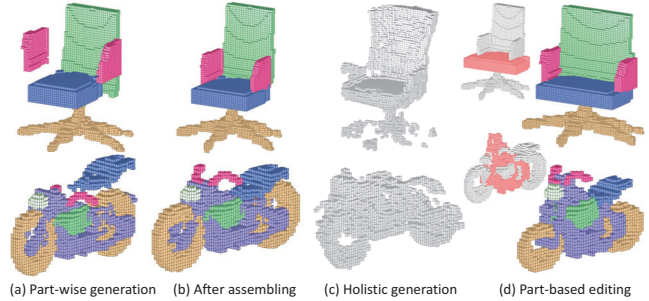
Figure 1: PAGENet first generates the semantic parts (a) and then assembles them with proper transformations, forming a structurally valid 3D shape (b) in contrast to the results by holistic generation (c). The learned model supports part-based shape editing (d): The anchor part (the pink parts in the inset) is edited by the user and all parts are automatically deformed and re-assembled into a plausible shape.

modeling of structural variations of 3D shapes through learning a *part-aware* generative network, named as PAGENet. The model is thus aware of what parts it is generating, through a semantic-part-wise generation process. On the other hand, the model should be able to preserve the relative spatial placement between the generated parts, according to some learned part placement prior. Such placement determines how adjacency semantic parts assemble together in forming a structurally valid shape.

PAGENet is composed of an array of *part generators*, each of which is a combination of variational auto-encoder (VAE) and generative adversarial network (GAN) trained for generating a specified semantic part of the target shape category, followed by a *part assembler* that estimates a transformation for each part used to assemble them into a valid shape structure (Figure 1(a,b)). In this work, PAGENet is realized in the volumetric setting; it can be easily extended to other representations such as point cloud.

Our model delegates the generation of parts and relations on separate networks, greatly reducing the modeling difficulty of 3D shape structures. 3D structure variation is modeled by the concatenation of the latent vectors of all part generators, forming a *structured latent space* of 3D shapes with different dimensions of the latent vector controlling the generation of different parts. This facilitates part-based shape editing and latent-space part reshuffling of the generated shapes with structure preservation. Figure 1(d) shows two examples of part-based editing: Based on our part assembler, edit over anchor parts can be propagated to other parts, resulting in structurally reasonable edited shapes.

---

[*]Corresponding author: Kai Xu (kevin.kai.xu@gmail.com)

Our main contributions include:

- A divide-and-conquer approach to learning a deep generative model of structure-aware shape generation.

- A part assembly module learned to correlate the generated semantic parts with their assembling transformations.

- Two applications including semantic shape segmentation and part-based structure-aware shape editing.

2   Related work

Deep generative models of 3D shapes.   Deep generative models for 3D shape generation have been developed based on various 3D representations, such as volumetric grids (Wu et al. 2016; Girdhar et al. 2016; Wang et al. 2017; Riegler, Ulusoy, and Geiger 2017), point clouds (Fan, Su, and Guibas 2016; Achlioptas et al. 2018), surface meshes (Groueix et al. 2018; Wang et al. 2018b), implicit functions (Chen and Zhang 2019; Park et al. 2019), and multi-view images (Soltani et al. 2017). Common to these works is that shape variability is modeled in a holistic, structure-oblivious fashion, which is mainly due to the limited options of deep-learning-friendly 3D shape representations.

Structure-aware 3D shape synthesis.   Research on *deep generative models* for structure-aware shape synthesis starts to gain increasing attention recently. Huang et al. (2015) propose a deep generative model based on part-based templates learned *a priori*, which is not end-to-end trainable. Li et al. (2017) propose the first deep generative model of 3D shape structures. They employ recursive neural network to achieve hierarchical encoding and decoding of parts and relations. The binary-tree-based architecture is extended to N-ary in (Mo et al. 2019). However, these models do not explicitly ensure a quality part assembly as in our method; see Section 4 for a comparison. In (Zou et al. 2017), a sequential part generation model is learned with recurrent neural networks, which produces parts as cuboids only.

Nash and Williams (2017) propose ShapeVAE to generate part-segmented 3D objects. Their model is trained using shapes with dense point correspondence. On the contrary, our model requires only part level correspondence. Wu et al. (2018) couples the synthesis of intra-part geometry and inter-part structure. Similar idea is proposed in (Balashova et al. 2018) where landmark-based structure priors are used for structure-aware shape generation. In (Wang et al. 2018a), 3D shapes are generated with part labeling based on GAN and then refined using a pre-trained part refiner. Our method takes a reverse process where we first generate parts and then their assembling transformations.

More closely related are the two concurrent works of part-based shape synthesis in (Schor et al. 2019) and (Dubrovina et al. 2019). Schor et al. (2019) train part-wise generators and a part composition network for the generation of 3D point clouds. Dubrovina et al. (2019) propose a nice decomposer-composer network to learn a factorized shape embedding space for part-based 3D shape modeling. Different from our work, however, their model is not a generative one. In their work, novel shapes are synthesized through
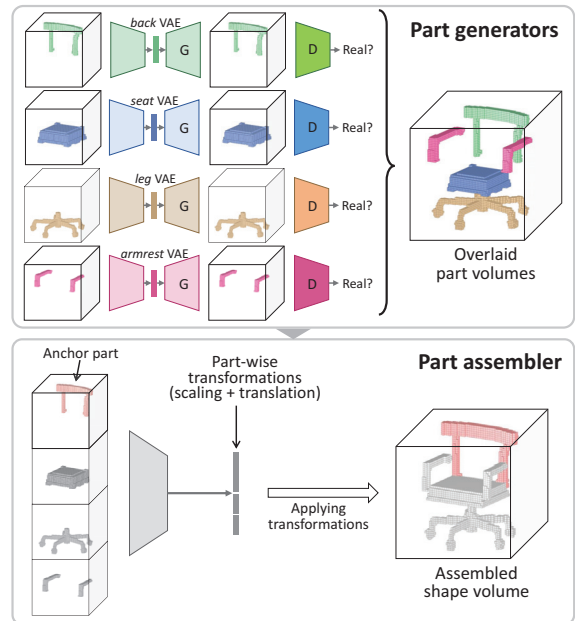

Figure 2: Top: Training the part generators (one VAE-GAN per semantic part). Since parts are generated independently, there could be mismatching and disconnection (see the overlaid volumes). Bottom: A part assembler is trained to regress a transformation (scaling + translation) for each part, which assembly the parts into a plausible shape.

randomly sampling and assembling the pre-exiting parts embedded in the factorized latent space.

3   Method
3.1   Network architecture

Our network architecture (Figure 2) is composed of two modules: a part-wise generative network and a part assembly module. The part-wise generative network contains K part generators, each for one of the K predefined semantic part labels (e.g., back, seat, leg and armrest for a chair). Each part generator is trained to generate a volume of a specific part from a random vector. Taking the generated volumes for all the parts as input, the part assembler predicts a transformation (scaling + translation) for each part, to assemble the parts into a complete shape with proper part scaling and inter-part connection.

3.2   Part generators

For each semantic part, we train a generative network of 3D volumes, which is a combination of variational auto-encoder (VAE) and generative adversarial network (GAN), or VAE-GAN. The VAE part comprises of an encoder and a decoder of 3D volumes with a resolution of $64 \times 64 \times 64$. The dimension of the latent vector is 50. Similar to (Wu et al. 2016), the encoder consists of five volumetric fully convolutional layers with a kernel size of $4 \times 4 \times 4$ and a stride of 2. Batch normalization and ReLU layers are inserted between convolutional layers. The decoder / generator simply reverses the
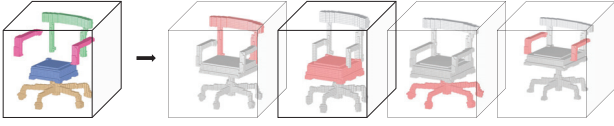
Figure 3: Reasonable assembly result can be produced with different parts serving as anchor (highlighted in red color).
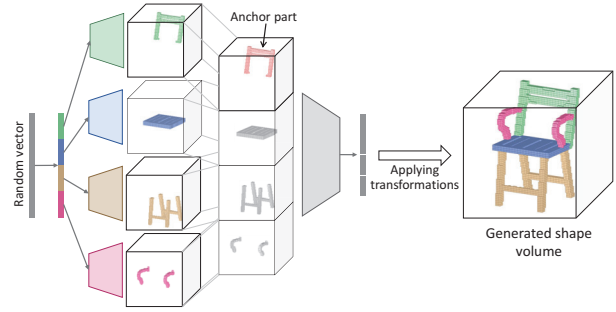

Figure 4: Part-aware 3D shape generation. Given a random vector, our network splits it into several sections and pass them to respective part decoders, yielding a set of semantic parts. The parts are then assembled together based on the predicted per-part transformations.

encoder, except that a $Sigmoid$ nonlinearity is used in the last layer. Following the decoder, a discriminator is devised by reusing the encoder architecture (with an extra 1D MLP output layer). The discriminator is trained to tell whether a given part volume is real, i.e., the voxelization of a real shape part, or fake, i.e., a volume generated by the generator.

Therefore, the loss function for a part generator consists of three terms: a part volume reconstruction loss L_{recon}, a Kullback-Leibler divergence loss L_{KL} and a adversarial loss L_{adv}. In addition, we introduce a reflective symmetry loss L_{ref} to penalize generating asymmetric parts. This loss would help regularize the part generation, since most parts are reflective symmetric. For those asymmetric parts, the weight of L_{ref} is set to 0. In summary, the loss is:

$$L_{\text{partgen}}(p) = L_{\text{recon}} + \alpha_1 L_{\text{KL}} + \alpha_2 L_{\text{adv}} + \delta_{\text{ref}}(p) L_{\text{ref}}, \quad (1)$$

where $L_{\text{recon}} = D(x_{\text{in}}, x_{\text{out}})$ measures the mean-square-error (MSE) loss between the input volume x_{in} and output volume x_{out}; $L_{\text{ref}}(x) = D(x, ref(x))$ is the MSE loss between a volume x and its reflection $ref(x)$ about the reflection plane of the input shape; $\delta_{\text{ref}}(p)$ is a Kronecker delta function indicating whether part p shares reflective symmetry with the full shape. In training, we detect reflective symmetry using the method in (Mitra, Guibas, and Pauly 2006) for a training shape and its semantic parts, to evaluate δ_{ref} for each part.

For the adversarial training, we follow WGAN-GP (Gulrajani et al. 2017) which improves Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017) with a gradient penalty to train our generative model,

$$L_{\text{adv}} = \underset{\widetilde{x} \sim P_g}{E[D(\widetilde{x})]} - \underset{x \sim P_r}{E[D(x)]} + \lambda \underset{\hat{x}}{E[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]},$$
$$(2)$$

where D is the discriminator, P_g and P_r are the distributions of generated part volumes and real part volumes, respectively. The last term is the gradient penalty and $\hat{x}$ is sampled uniformly along straight lines between pairs of points from the data distribution P_r and the generator distribution P_g. The discriminator attempts to minimize L_{adv} while the generator maximizes the first term in Equation (2).

3.3 Part assembler

Since the part volumes are generated independently, their scales may not match with each other and their positions may disconnect adjacent parts. Taking K part volumes generated from the part-wise generative network, part assembler regresses a transformation, including a scaling and a translation, for each part. It relates different semantic parts, with proper resizing and repositioning, to assemble them into a valid and complete shape volume. Essentially, it learns the

spatial relations between semantic parts (or part arrangements) in terms of relative size and position, as well as mutual connection between different parts.

The part assembler takes K part volumes as input, which amount to a $64 \times 64 \times 64 \times K$ input tensor. The input tensor is passed through five volumetric fully convolutional layers of kernel sizes $4 \times 4 \times 4$ with a stride of 2. Similar to part encoders, batch normalization and ReLU layers are used between convolutional layers. In the last layer, a sigmoid layer is added to regress the scaling and translation parameters. To ease the training, we normalize all scaling and translation parameters into $[0, 1]$, based on the allowed range of scaling ($[0.5, 1.5]$) and translation ($[-20, 20]$, with the unit being voxel size). The actual values of scaling and translation parameters are recovered when being applied.

Anchored transformation. Given K part volumes, the transformations assembling them together is not unique. Taking the chair model in Figure 3 as an example, the chair seat can be stretched to match the back, while the back can also be shrunk to conform to the seat; both result in a valid shape structure. This also adds some diversity to the generation. To make the transformation estimation determined and the assembly network easier to train, we introduce an extra input to the part assembler to indicate a *anchor part*. When estimating part transformations, the anchor part is kept fixed (with an identity transformation) while all the other parts are transformed to match the anchor. To do this, one option is to input an indicator vector (a one-hot vector with the corresponding part being 1). However, the dimension of this indicator vector is too small, making its information easily overwhelmed by the large tensor of part volumes. Therefore, we opt to infuse anchor information by setting the occupied voxels in the anchor part volume to -1, to strongly contrast against the 1's in the volumes of the free parts.

During test, novel shapes can be generated from a random vector of all parts (Figure 4), where the anchor part can be randomly selected or user-specified.

3.4 Training details

We train and test our model on the ShapeNet part dataset (Yi et al. 2016), which is a subset of the ShapeNet dataset (Chang et al. 2015) and provides consistent alignment and semantic labeling for all shapes. In the main paper, we test on four representative categories exhibiting rich part structure variation, including *chairs* (3746), *airplanes* (2690), *lamps* (1546), *motorbikes* (202). The results on more categories is provided in the supplemental material. Each object category has a fixed number of semantic parts: a chair contains a *back*, a *seat*, a *leg* and an *armrest*; an airplane consists of a *body*, a *wing*, a *tail* and an *engine*; a lamp has a *base*, a *shade*, and a *tube*; a motorbike is composed of a *light*, a *gas tank*, a *seat*, a *body*, a *wheel* and a *handle*. Note that a shape may not contain all semantic parts belonging to the corresponding category. The dataset is divided into two parts, according to the official training/test split. To enhance the training set, we employ the structure-aware deformation technique in (Zheng et al. 2011) to deform each shape, generating about 10 variations of the shape. Finally, each shape and its semantic parts are voxelized to form our training set.

To train the part generators, we augment the dataset of part volumes via randomly scaling and translating the parts, with the ranges in $[0.5, 1.5]$ for scaling and $[-20, 20]$ (in voxels) for translation. As Wasserstein GANs usually have large gradients, which might result in unstable training, we first pre-train the VAEs and then fine-tune them via joint training with the discriminators. To train the part assembler, we generate a large set of training pairs of messed-up part arrangement and ground-truth assembly, with randomly selected anchor part. The messed-up arrangements are generated by randomly scaling and translating the semantic parts of a shape. The inverse of the messing-up transformations are used as ground-truth assembling transformations. Besides that, we also introduce some random noise to the training part volumes, to accommodate the imperfect volume generation during testing.

For all modules, we use ADAM ($\beta = 0.5$) for network optimization with an initial learning rate of 0.001. Batch size is set to 32. The parameters in the loss in Equation (1) are set as $\alpha_1 = 2$ and $\alpha_2 = 1 \times 10^{-3}$ for all experiments. The λ in Equation (2) is set to 10 as in (Gulrajani et al. 2017). The average training time is 12 hours for each part generator and 7 hours for part assembler.

Note that the part generators and part assembler are not trained jointly, in an end-to-end fashion, since there is no ground-truth assembly for the parts generated by VAE (random generation). It is, however, possible to make the whole pipeline end-to-end trainable if a discriminator network could be devised to judge whether the *final assembled shape* is reasonable or not. We leave this for future work.

4 Results and Evaluations

Part generation. Symmetry preservation is an important aspect of structure reservation. Through imposing reflective symmetry regularization for those parts which are reflectively symmetric, our model is able to produce structurally more plausible shapes. To evaluate symmetry preservation in

Table 1: Comparing average symmetry measure over 1000 generated shapes between our method (with and w/o symmetry loss) and 3D-GAN (Wu et al. 2016), G2L (Wang et al. 2018a), on two shape categories. For each category, we report the measure for both full shape and semantic parts (**red** indicates the best and **green** the second best). Note how our method is able to discriminate between reflectively symmetric and asymmetric legs of chairs in preserving symmetries.

| | | PAGENet | | 3D-GAN | G2L | GT |
		w/ sym. loss	w/o sym. loss			
Chair	back	**0.88**	0.80	0.71	**0.93**	0.93
	seat	**0.90**	0.82	0.76	**0.94**	0.95
	leg (sym.)	**0.68**	0.56	0.40	**0.74**	0.86
	leg (asym.)	0.29	0.28	–	–	0.32
	armrest	**0.64**	**0.59**	0.16	**0.64**	0.81
	full	**0.85**	0.77	0.70	**0.91**	0.92
Airplane	body	**0.84**	0.78	0.67	**0.89**	0.92
	wing	**0.81**	0.73	0.63	**0.84**	0.86
	tail	**0.75**	0.69	0.46	**0.77**	0.85
	engine	**0.71**	**0.61**	0.14	**0.71**	0.83
	full	**0.80**	0.72	0.64	**0.84**	0.89

shape generation, we define a *symmetry measure* for generated shapes. Given a generated shape volume, the reflective plane is the vertical bisector plane of the volume, since all training data were globally aligned and centered before voxelization. The symmetry measure can be obtained simply by reflecting the left half of the shape volume and computing the IoU against the right half.

Table 1 shows the average symmetry measures on 1000 randomly generated shapes. The results are reported both for full shape and individual semantic parts. We also compare to a baseline model trained without symmetry loss, as well as 3D-GAN (Wu et al. 2016) and G2L (Wang et al. 2018a). In G2L, an extra part refinement network is trained via minimizing the average reconstruction loss against three nearest neighbors retrieved from the training set. While achieving a higher symmetry score, such refinement also limited the diversity of the generated shapes, which is indicated by the lower inception score than ours in Table 3.

An interesting feature of our part generators is that it learns when to impose symmetry constraint on the generated parts, through judging from the input random vectors. This is due to the discriminative treatment of reflectively symmetric and asymmetric parts during the generator training (Equation (1)). Take the leg part generator for example. If a random vector of a four-leg chair is input, the leg generator will preserve the reflective symmetry in the leg part. If the input random vector implies a swivel chair, the symmetry preservation will be automatically disabled since the leg part of a swivel chair is mostly *not* reflectively symmetric in reality. This is reflected in the average symmetry measures of symmetric and asymmetric legs in Table 1.

Part assembly. To evaluate the ability of our part assembler, for each test shape, we perturb each of its semantic parts with random scaling and translation, and use our network to regress assembly transformation. Assembly quality is measured by the IoU between the assembled shape volume and

Table 2: Comparison of part assembly quality (**red** indicates the best and **green** the second best).

|	Chair				Plane				Motorbike						Lamp		
Template-based	0.74				0.72				0.69						0.70		
Anchor part	seat	back	leg	armrest	body	wing	tail	engine	light	gas tank	seat	handle	wheel	body	base	shade	tube
One-hot vector	0.77	0.78	0.79	0.72	0.76	0.75	0.73	0.74	0.72	0.76	0.74	0.73	0.71	0.76	0.79	0.77	0.70
Ours	**0.83**	**0.81**	**0.82**	**0.79**	**0.80**	**0.82**	**0.76**	**0.82**	**0.80**	**0.79**	**0.82**	**0.79**	**0.78**	**0.82**	**0.81**	**0.83**	**0.77**
Ours (training data)	**0.89**	**0.91**	**0.90**	**0.88**	**0.89**	**0.87**	**0.90**	**0.88**	**0.87**	**0.92**	**0.86**	**0.86**	**0.85**	**0.87**	**0.91**	**0.89**	**0.81**
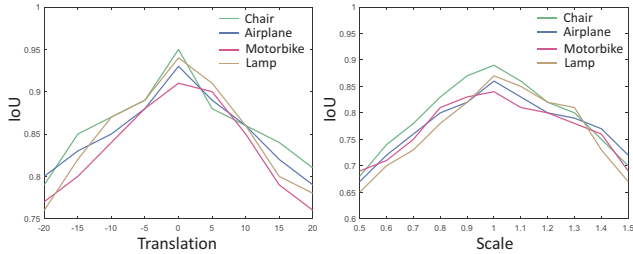
Figure 5: Plots of assembly quality measure (average IoU w.r.t. ground-truth) over varying amount of translation (left; with scaling being fixed to 1.2) and scaling (right; with translation being fixed to 10).

the ground-truth. In testing, we choose each semantic part as anchor and report the average IoU as the assembly quality. In Table 2, we compare the assembly performance over *three* methods. The *first* is our method. The *second* is our method in which the anchor part is indicated by a one-hot vector. The *third* one is a template-based part assembly where we retrieve a template shape from the training set based on part-wise CNN features. We then transform the shape parts according to the corresponding parts in the template, since part correspondence is available for all shapes in the training set and the generated shapes. For contrasting, we also show the performance on the training shapes (the last row).

The results show that our part assembly generalizes well to unseen shapes. The numbers reported in Table 2 are under messy part arrangement with random scale from $[0.5, 1.5]$ and random translation from $[-20, 20]$. For each part, its local axes are scaled and translated independently. Figure 5 plots the assembly quality measure over varying amount of translation and scaling. Our method obtains reasonably good assembly results within the range of $[-20, 20]$ for translation and $[0.5, 1.5]$ for scaling. Note, however, the goal of our part assembler is *not* to reconstruct an input shape. In fact, there is not a unique solution to structurally plausible part assembly. Therefore, this experiment only approximately evaluates the assembly ability of our model.

Random shape generation. Figure 6 shows a few examples of random generation for all four shape categories. For each shape, both the generate part volumes (overlaid) and the final assembling result are shown. A nice feature of our method is that the generated shapes all possess semantic segmentation by construction, which can be used in training data enhancement for shape segmentation. More generation results can be found in the supplemental material.

Table 3: Comparing diversity (inception score) of random shape generation with three state-of-the-art methods.

	Chair	Airplane	Motorbike	Lamp
3DGAN (Wu et al. 2016)	5.84	5.61	5.01	5.19
GRASS (Li et al. 2017)	5.79	5.68	4.93	5.26
G2L (Wang et al. 2018a)	6.17	5.89	5.52	5.45
PAGENet	**6.55**	**6.05**	**5.80**	**5.59**

In Table 3, we compare the diversity of random generation by our method and three alternatives including 3D-GAN (Wu et al. 2016), GRASS (Li et al. 2017) and G2L (Wang et al. 2018a). Similar to (Wang et al. 2018a), we use the inception score (Salimans et al. 2016) to measure the diversity of shape sets. In particular, we first cluster the training shapes and then train a classifier targeting the clusters. The inception score for a given set of shapes is then measured based on the confidence and variance of the classification over the set. From the results, our method achieves consistently more diverse generation than alternatives, thanks to the part-wise shape variation modeling. Please refer to the supplemental material for a *qualitative* study on diversity.

Comparison with GRASS (Li et al. 2017). Figure 7 shows a visual comparison to GRASS. Although GRASS can recover part relations (adjacency and symmetry) in the generated shapes, it does not explicitly learn how adjacent parts are connected. Therefore, parts generated by GRASS can sometimes mistakenly detach. In contrast, PAGENet leads to better part connection thanks to the learned part assembly prior. To quantitatively evaluate part assembly quality, we propose two measures, one objective and one subjective. The objective measure simply examines the voxel connectivity of a generated shape volume. In the subjective evaluation, we recruited 30 human participants (including both CS and non-CS graduate students) to visually inspect and rate the correctness (binary choice) of part connections of a generated shape. All generated shapes are presented to the human viewers with the same rendering style without part coloring. For both measures, we compute the average *correctness rate* of part assembly for each shape category.

Table 4: Comparing with GRASS (Li et al. 2017) on objective and subjective correctness rate of part assembly (in %).

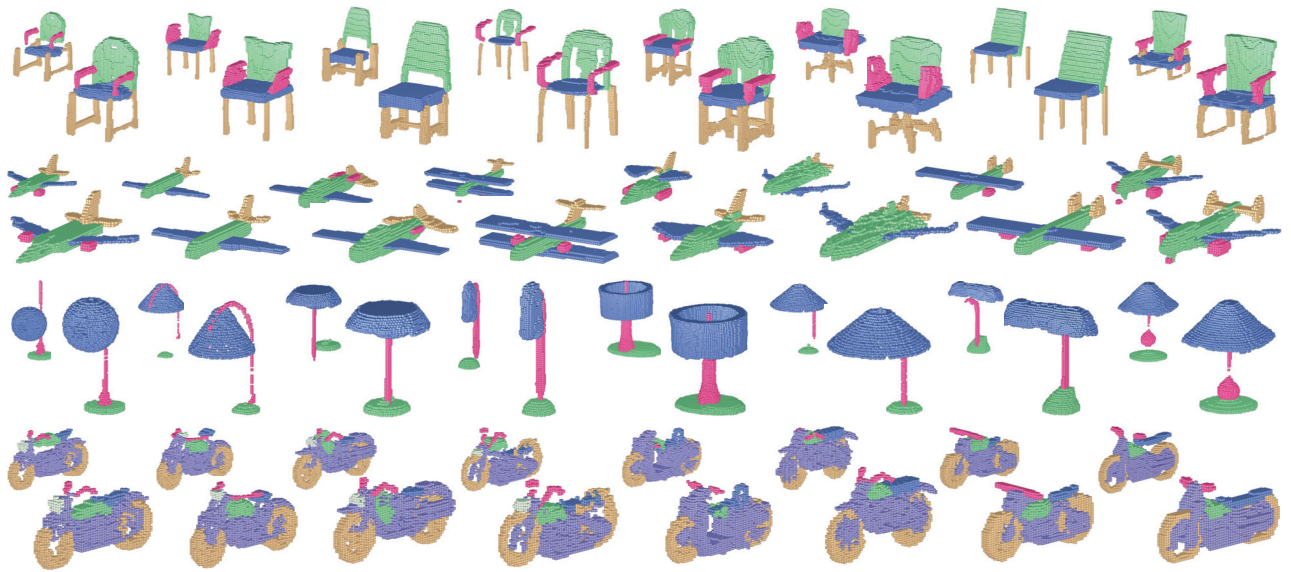|	Chair		Airplane		Motorbike		Lamp	
	Obj.	Subj.	Obj.	Subj.	Obj.	Subj.	Obj.	Subj.
GRASS	81.8	79.6	83.7	80.2	93.6	90.1	91.4	87.4
PAGENet	90.2	88.2	92.6	91.8	97.0	95.2	96.8	95.0

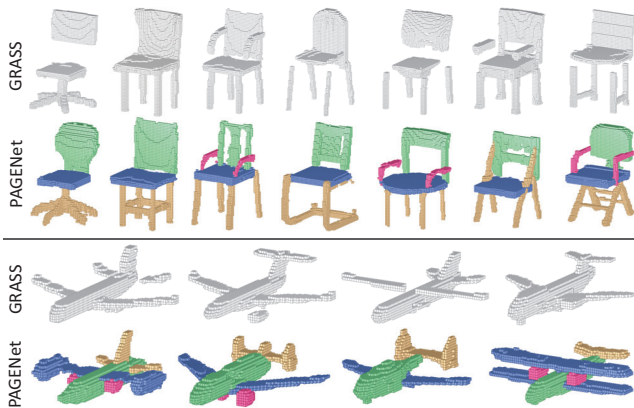Figure 6: Results of random shape generation on four categories. All generated shapes have semantic segmentation.

Figure 7: A visual comparison of part connection with GRASS (Li et al. 2017).
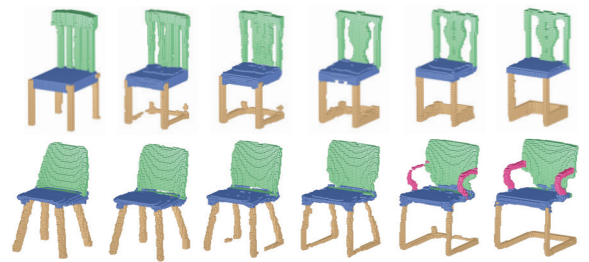

Figure 8: Shape interpolation in latent space.

Figure 9: 3D shape crossover at the granularity of semantic part, enabled by our structured latent space.

Table 4 compares the success rate of the two methods over 500 randomly generated shapes for each category.

Latent space shape manipulation. Through breaking down 3D shape generation into part-wise generation and part assembly inference, our method is able to model significant variation of 3D shape structures. This can be demonstrated by shape interpolation between shapes with significantly different structures. Figure 8 shows two such examples. Our method can generate high quality in-between shapes with detailed shape structures, although the source and target shapes have considerably different structures. More interpolation results can be found in the supplemental material. Due to the part-aware representation in our model, the latent space for full shapes is by construction structured. Therefore, our model naturally supports arithmetic modeling of
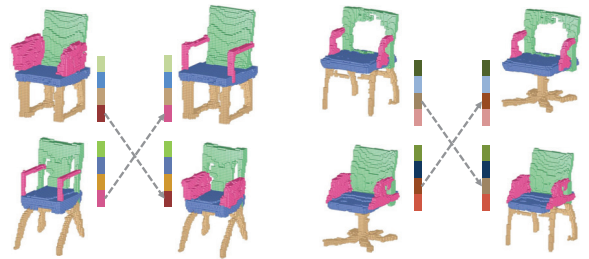
3D shapes, at the granularity of semantic part. Results can be found in the supplemental material.

5 Applications

Structure-aware part-based shape editing. This is an important way of creating 3D shape variations. Our model supports two modes of part-based editing: *latent space control* and *direct manipulation*. With the structured latent representation learned by our model, novel shapes can be created via varying the entries of the latent vector of a generated
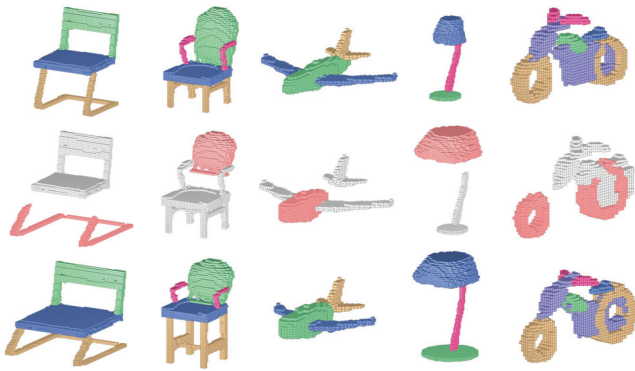
Figure 10: Top: 3D shapes generated by PAGENet. Middle: User editing on anchor parts (pink). Bottom: Editing results after re-assembling.
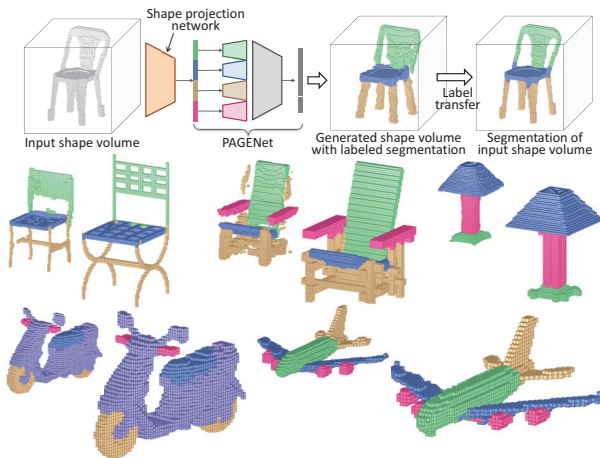

Figure 11: Top: The network of using PAGENet for 3D shape segmentation. Bottom: A few segmentation results. For each example, the recovered shape used for semantic label transfer is shown to the top-left.

shape. This way, the latent code can be viewed as "shape DNA" of 3D shapes on which chromosome operations such as mutation and crossover can be conducted. New shapes are recovered (decoded) from the resultant chromosomes (code) in analogous to genetic expression, facilitating bio-inspired shape generation (Xu et al. 2012). Figure 9 shows two examples of crossover operation. Given a shape generated by our method, the *direct manipulation* mode allows the user to select one part as the anchor and perform direct editing (scaling and translation in our implementation) over it. After user editing, we leverage the part assembler to compute for each part an assembling transformation, forming a structurally valid shape. Figure 10 demonstrates a few examples of interactive part-based shape editing.

Shape segmentation. The part-wise generation of our model can also be used to segment a 3D shape. The network is shown in Figure 11 (top). Given a 3D shape in volumetric representation, we first project it into the latent space of

Table 5: A quantitative comparison of shape segmentation.

	Chair	Airplane	Motorbike	Lamp
PointNet (Su et al. 2017)	89.6	83.4	65.2	80.8
SyncSpecCNN (Yi et al. 2017)	90.2	81.6	**66.7**	84.7
O-CNN (Wang et al. 2017)	91.1	**85.5**	56.9	83.3
PAGENet	**92.0**	83.5	63.2	**85.2**

PAGENet based on a trained shape projection network. It encodes the shape volume with five volumetric convolutional layers and project the input shape volume to a latent code. Then, our pre-trained PAGENet is used to reconstruct a 3D shape (with semantic segmentation). The projection network is trained by minimizing the reconstruction loss against the input shape volume, while keeping the PAGENet part fixed. During testing, passing a 3D shape volume into the network results in a reconstructed 3D volume with semantic segmentation. Since the recovered 3D volume is geometrically close to the input volume (due to the self-reconstruction training), we can accurately transfer its voxel labels onto the input volume, thus obtaining a semantic segmentation for the input.

Figure 11 shows a few examples of such segmentation on the testing set. Essentially, we learn a deep model of *segmentation transfer*. The model integrates a pre-learned *part-aware shape manifold* with a *shape-to-manifold projector*: To segment a shape, the projector retrieves the nearest neighbor from the manifold and generate semantic segmentation for the retrieved shape, whose segmentation can be easily transferred to the input due to shape resemblance. Table 5 shows some quantitative results of segmentation with comparison with a few state-of-the-art methods. It shows that our method achieves comparable performance with the state-of-the-arts. Extended results of shape segmentation can be found in the supplemental material.

6   Conclusion

We have proposed a simple and effective generative model for quality and diverse 3D shape generation. Our model knows what it generates (semantic parts) and how the generated entities correlate with each other (by assembling transformation). This makes the generation part-aware and structure-revealing. There are two main *limitations*. *First*, our model relies on hard-coded split of semantic part generators which is hard to adapt to a different label set. *Second*, our method currently works with major semantic parts. Extending it to deal with fine-grained parts would find difficulty in part assembly, which could be alleviated with the help of a hierarchical part organization as in (Zhu et al. 2018; Yu et al. 2018).

Acknowledgments

References

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L. 2018. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Balashova, E.; Singh, V.; Wang, J.; Teixeira, B.; Chen, T.; and Funkhouser, T. 2018. Structure-aware shape synthesis. In *3D Vision (3DV)*.

Chang, A. X.; Funkhouser, T.; J. Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. (arXiv:1512.03012 [cs.GR]).

Chen, Z., and Zhang, H. 2019. Learning implicit fields for generative shape modeling. In *CVPR*.

Dubrovina, A.; Xia, F.; Achlioptas, P.; Shalah, M.; and Guibas, L. 2019. Composite shape modeling via latent space factorization. *arXiv preprint arXiv:1803.10932*.

Fan, H.; Su, H.; and Guibas, L. 2016. A point set generation network for 3d object reconstruction from a single image. *arXiv preprint arXiv:1612.00603*.

Girdhar, R.; Fouhey, D. F.; Rodriguez, M.; and Gupta, A. 2016. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, 484–499. Springer.

Groueix, T.; Fisher, M.; Kim, V. G.; Russell, B. C.; and Aubry, M. 2018. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 216–224.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *NIPS*, 5769–5779.

Huang, H.; Kalogerakis, E.; and Marlin, B. 2015. Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. *Computer Graphics Forum* 34(5).

Li, J.; Xu, K.; Chaudhuri, S.; Yumer, E.; Zhang, H.; and Guibas, L. 2017. Grass: Generative recursive autoencoders for shape structures. *arXiv preprint arXiv:1705.02090*.

Mitra, N.; Wand, M.; Zhang, H. R.; Cohen-Or, D.; Kim, V.; and Huang, Q.-X. 2013. Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, 1. ACM.

Mitra, N. J.; Guibas, L. J.; and Pauly, M. 2006. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. on Graph.* 25(3):560–568.

Mo, K.; Guerrero, P.; Yi, L.; Su, H.; Wonka, P.; Mitra, N.; and Guibas, L. J. 2019. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Trans. on Graph. (SIGGRAPH Asia)*.

Nash, C., and Williams, C. K. 2017. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Computer Graphics Forum (SGP 2017)* 36(5):1–12.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*.

Riegler, G.; Ulusoy, A. O.; and Geiger, A. 2017. Octnet: Learning deep 3d representations at high resolutions. In *Proc. CVPR*, volume 3.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Proc. NIPS*, 2234–2242.

Schor, N.; Katzier, O.; Zhang, H.; and Cohen-Or, D. 2019. Learning to generate the "unseen" via part synthesis and composition. In *IEEE International Conference on Computer Vision*.

Soltani, A. A.; Huang, H.; Wu, J.; Kulkarni, T. D.; and Tenenbaum, J. B. 2017. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1511–1519.

Su, H.; Qi, C. R.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, to appear.

Wang, P.-S.; Liu, Y.; Guo, Y.-X.; Sun, C.-Y.; and Tong, X. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (SIGGRAPH)* 36(4).

Wang, H.; Schor, N.; Hu, R.; Huang, H.; Cohen-Or, D.; and Huang, H. 2018a. Global-to-local generative model for 3d shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)* 37(6):214:1–214:10.

Wang, N.; Zhang, Y.; Li, Z.; Fu, Y.; Liu, W.; and Jiang, Y.-G. 2018b. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 52–67.

Wu, J.; Zhang, C.; Xue, T.; Freeman, B.; and Tenenbaum, J. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, 82–90.

Wu, Z.; Wang, X.; Lin, D.; Lischinski, D.; Cohen-Or, D.; and Huang, H. 2018. Structure-aware generative network for 3d-shape modeling. *arXiv preprint arXiv:1808.03981*.

Xu, K.; Zhang, H.; Cohen-Or, D.; and Chen, B. 2012. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics (TOG)* 31(4):57.

Xu, K.; Kim, V. G.; Huang, Q.; Mitra, N.; and Kalogerakis, E. 2016. Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses*, 4. ACM.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*.

Yi, L.; Su, H.; Guo, X.; and Guibas, L. J. 2017. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, 6584–6592.

Yu, F.; Liu, K.; Zhang, Y.; Zhu, C.; and Xu, K. 2018. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *CVPR*.

Zheng, Y.; Fu, H.; Cohen-Or, D.; Au, O. K.-C.; and Tai, C.-L. 2011. Component-wise controllers for structure-preserving shape manipulation. In *Computer Graphics Forum*, volume 30, 563–572. Wiley Online Library.

Zhu, C.; Xu, K.; Chaudhuri, S.; Yi, R.; and Zhang, H. 2018. Scores: Shape composition with recursive substructure priors. *ACM Transactions on Graphics (SIGGRAPH Asia 2018)* 37(6):to appear.

Zou, C.; Yumer, E.; Yang, J.; Ceylan, D.; and Hoiem, D. 2017. 3d-prnn: Generating shape primitives with recurrent neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*.