Structure-aware Generative Network for 3D-Shape Modeling

ZHIJIE WU, Shenzhen University
XIANG WANG, Shenzhen University
DI LIN, Shenzhen University
DANI LISCHINSKI, The Hebrew University of Jerusalem
DANIEL COHEN-OR, Shenzhen University and Tel Aviv University
HUI HUANG*, Shenzhen University

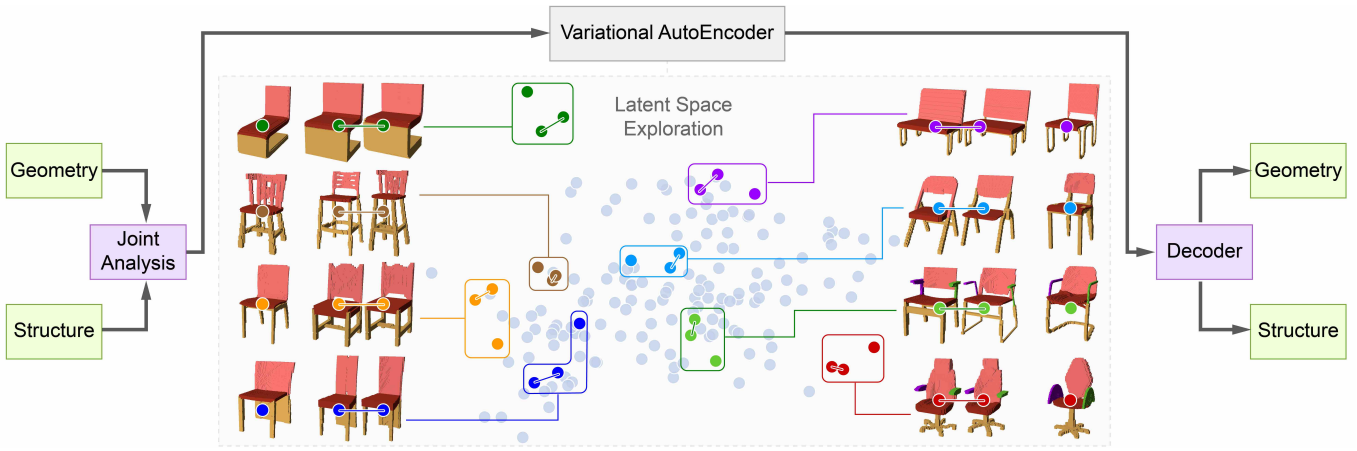

Fig. 1. Our generative model jointly analyzes the structure and geometry of shapes, encoding them into a single latent code. The highlighted triplets above demonstrate that, in this joint latent space, pairs of nearby points represent models that are close to each other in both geometry and structure, while stepping away from the pair introduces differences in either geometry, or structure, or both.

We present SAGNet, a structure-aware generative model for 3D shapes. Given a set of segmented objects of a certain class, the geometry of their parts and the pairwise relationships between them (the structure) are jointly learned and embedded in a latent space by an autoencoder. The encoder intertwines the geometry and structure features into a single latent code, while the decoder disentangles the features and reconstructs the geometry and structure of the 3D model. Our autoencoder consists of two branches, one for the structure and one for the geometry. The key idea is that during the analysis, the two branches exchange information between them, thereby learning the dependencies between structure and geometry and encoding two augmented features, which are then fused into a single latent code. This explicit intertwining of information enables separately controlling the geometry and the structure of the generated models. We evaluate the performance of our method and conduct an ablation study. We explicitly show that encoding of shapes accounts for both similarities in structure and geometry. A variety of quality results generated by SAGNet are presented.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Shape analysis**;

*Corresponding author: hhzhiyan@gmail.com

Authors' addresses: Zhijie Wu, Shenzhen University; Xiang Wang, Shenzhen University; Di Lin, Shenzhen University; Dani Lischinski, The Hebrew University of Jerusalem; Daniel Cohen-Or, Shenzhen University and Tel Aviv University; Hui Huang, Shenzhen University.

2018. XXXX-XXXX/2018/8-ART $15.00
https://doi.org/0000001.0000001_2

Additional Key Words and Phrases: Shape synthesis, shape analysis, generative model, variational autoencoder

ACM Reference Format:
Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2018. Structure-aware Generative Network for 3D-Shape Modeling. 1, 1 (August 2018), 11 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

Modeling of 3D shapes is a central problem in computer graphics. In recent years more attention has been given to structure-aware modeling techniques, where the relations among parts are carefully considered. Analyzing the structure provides a high-level understanding of the shape, and it goes beyond the low-level analysis of the local geometry [Mitra et al. 2014]. The structure of a shape can be inferred from a single instance, but analyzing a family of shapes that share some similar structural characteristics can yield a much more powerful representation [Fish et al. 2014]. However, such an analysis is challenging, since structure and geometry are often interdependent, exhibiting complex relations and dependencies that are not easy to model directly.

Our work is motivated by recent advances in the competence of neural networks in analyzing data. We present a generative network that analyzes and encodes latent relationships between structure and geometry in a class of (man-made) shapes. For a given set of shapes, structure is already implicitly represented in their geometric

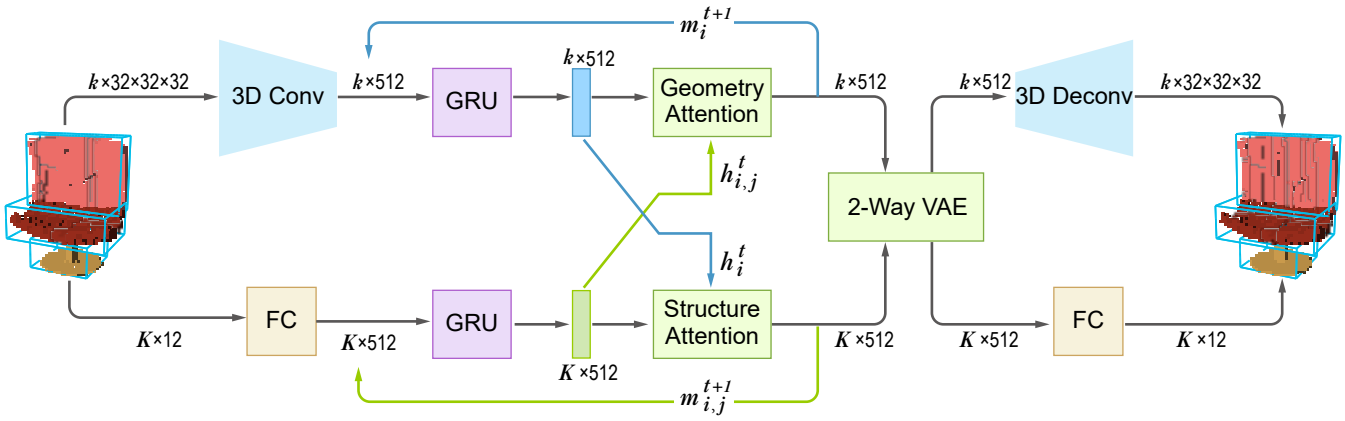

Fig. 2. Overview of SAGNet. Given 3D shapes as training data, the network has traditional 3D convolutional and fully-connected layers to extract visual features for shape parts. The network is equipped with GRU-based encoder and attention component, which jointly analyzes the geometry and structural information of shapes. All the information are provided for the 2-way VAE, which offers the generative power to the network. Our network eventually decodes the geometry and structural information to generate 3D shapes.

models, however geometric and structural information is entangled together in a manner that does not provide a way to control each of them separately. Thus, instead of performing unsupervised training with geometric models of entire objects, we use a weakly supervised training strategy, where we provide the geometry as a collection of separate parts and the structure as pairs of their bounding boxes. The two-branch autoencoder trained using this data learns to intertwine these two types of information into a single latent code, capturing geometry, structure, and their interdependencies. Thanks to this weak supervision, it becomes possible to separately control geometry and structure when synthesizing new models.

The high-level concept of our structure-aware generative network (SAGNet) is illustrated in Fig. 1. The network is trained using a set of shapes segmented into parts. The geometry of each part is provided as a voxel map and the structure is provided as a list of the bounding boxes, one for each part. Note that nothing is assumed about the relationships between the parts, thus the structure of the shape is not specified explicitly. The network learns to perform a joint analysis of the geometry and the implicit structure, feeding the results into an autoencoder that embeds the shape into a joint structure-geometry latent space. The resulting latent space representation can then be decoded back into geometry and structure.

As demonstrated in Fig. 1, nearby points in the joint latent space correspond to shapes that are similar to each other in both geometry and structure, while taking a step away introduces differences in geometry, structure, or both. Such latent space supports separate control of geometry and structure, and operations such as shape interpolation and completion.

The key novel component of our network is the exchange of data between the structure and geometry branches that takes place during the joint analysis stage. Previous attempts in developing generative neural networks for 3D shapes, included adversarial networks based on voxels or point representations of the geometry [Choy et al. 2016; Girdhar et al. 2016; Wu et al. 2015; Yan et al. 2016], or structural-based approaches [Li et al. 2017; Zou et al. 2017]. These methods do not leverage the power of a joint analysis of geometry and structure. We show that our generative network generates plausible structure-aware shapes that adhere to the characteristics of the learned class. We also demonstrate that our approach supports inferring the geometry from structure and vise-versa, enabling applications such as shape completion, and constrained modeling.

2 RELATED WORK

In recent years there have been efforts to leverage the success of deep neural networks to develop generative models of 3D shapes. Wu et al. [2015] develop a neural framework based on deep belief network to synthesize novel samples. Later, Wu et al. [2016] model the distribution of voxels of 3D objects using an adversarial approach. Their model can take a random noise as input and generate a voxel grid as output. The latent representation that they learn supports simple arithmetic and interpolation operations on the latent codes. Girdhar et al. [2016] embed voxel maps of shapes and their corresponding images in a shared latent space, making it possible to predicts a voxel map from a single 2D image. More advanced 3D generative voxel-based models are presented [Gwak et al. 2017; Yan et al. 2016]. Achlioptas et al. [2017] introduce a generative adversarial network (GAN) in the latent space. Nash et al. [2017] present a generative model based on a variational autoencoder (VAE) [Kingma and Welling 2014].

In our approach, we also employ a VAE to generate new shapes and use regular 3D voxel maps to represent their geometry. However, we represent each part using its own voxel map. Differently from the previous works above, we use recurrent neural networks (RNNs) to analyze the data. RNNs are widely used in generative models to analyze and generate novel sequences. For example, van den Oord et al. [2016] regard natural images as sequences and generates images row by row, pixel by pixel. Rezende et al. [2016] develop a novel model based on DRAW [Gregor et al. 2015], which can reconstruct 3D structures from 2D images. Zou et al. [2017] use an RNN decoder to generate shape primitives step by step from given depth images.

Our architecture consists of several RNNs and an RNN-based 2-way VAE that learns a generative representation for 3D shapes. We consider geometry and structure jointly, where the challenge is to learn a joint distribution of signals from different domains [Liu et al. 2017; Liu and Tuzel 2016]. Other examples of methods that learn a joint distribution include Choy et al. [2016], who use 3D-R2N2 to build a joint distribution to reconstruct 3D voxel maps from images. Also, Li et al. [2015] get a joint embedding of images and 3D shapes via CNN purification.

Relationships among entities, or spatial layouts of objects, are known to be useful for understanding visual information. Some previous works explore the physical relationship [Jia et al. 2013; Zheng et al. 2015], while others [Socher et al. 2012, 2011] use a recursive structure, and a recursive autoencoder, to capture the relationship by iteratively collapsing edges of a graph to yield a hierarchy. Li et al. [2017] adapt such recursive structures, and present a generative neural network model for the 3D structures of shapes, which can capture the structural information of different shapes within a class. Unlike us, they do not jointly consider the geometry domain and its corresponding structure domain, and they do not learn the dependencies between the geometries of different parts in an object.

3 STRUCTURE-AWARE GENERATIVE NETWORK

The main idea of our method is to analyze and generate shapes by jointly considering their structure and geometry, learning them and their inter-relations. Each shape is represented with k parts, where each part consists of a bounding box that contains a voxel map representing the part geometry. We represent the shape structure as the set of all $K = k \times (k-1)/2$ pairwise spatial relationships between the k parts. Thus, a shape is represented by two series (i) k voxel maps, and (ii) K pairs of axis-aligned bounding boxes, where each pair is represented by 2×6 coordinates.

Fig. 2 shows an overview of our architecture. Generally speaking, it is a two-branch autoencoder. The network takes two streams of input: one is a series of k voxel maps (the upper branch in the figure), and the other is a series of K pairwise spatial relationships, represented by pairs of bounding boxes (the lower branch). The geometry stream is analyzed by convolutional layers, and fed into an RNN component that analyzes the series of the resulting k features. The structural stream is analyzed by fully-connected layers and fed into an RNN that analyzes the series of K pairwise relation features. Each of these RNN units is implemented using a GRU [Cho et al. 2014] to achieve greater flexibility in a more principled training framework. The outputs of these two GRUs are then fed into modules that exchange information between the geometry and structure streams. The exchange of information is weighted by the influence of the respective data. This is commonly called "attention": In the Geometry Attention module, the geometric features are given attention by their $k-1$ related structural features. Likewise, in the Structure Attention module, each structure feature is given attention by its two geometry features. Each attention module yields a deep feature, one representing the geometry and one the structure. More details are provided in Section 4.

These two deep features are next fed into a 2-way VAE, which accepts two inputs (one from each branch), rather than one. The purpose of the 2-way VAE is to combine and and fuse the two features representing the geometry and the structure into a single vector, thereby embedding them in a joint latent space. On the output end, the 2-way VAE produces two features, associated with the two streams, which are fed into two corresponding branches with two decoders that generate the output streams.

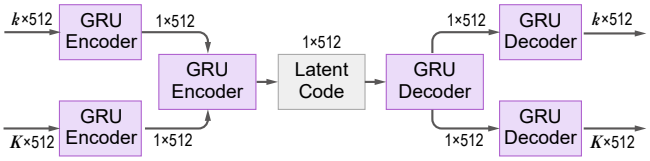

Fig. 3. Architecture of 2-Way VAE. Our VAE has internal GRU encoders, which takes input as the geometry and structural information analyzed by the attention component. To generate a 3D shape, the encoders output a latent code that is processed by the internal GRU-decoders of the VAE.

The architecture of the 2-way VAE is depicted in Fig. 3. First, the two input feature streams are fed into corresponding GRU units whose goal is to collapse each feature stream into a single feature vector of size 1×512. The two resulting features are next fed into another GRU encoder, which fuses them into a single latent code. This latent code represents the coordinates of the shape in the joint embedding space. The fused features represent the shapes, encapsulating their geometries and structures. In other words, the fused features are structure-aware since they encode the geometry and structure information, as well as the relation between them.

As shown in Fig. 3, the decoding end of the 2-way VAE exactly mirrors the encoding end, and consists of GRU decoders that split the features into two streams, which are then decoded into geometric and structural series.

Note that information from the geometry and structure streams is exchanged in two stages of the pipeline. First, during their analysis, using the Geometry and Structure attention modules, and then in the 2-Way VAE, where the two streams collapsed and fused.

4 TECHNICAL DETAILS

In this section, we elaborate on some technical details, and describe the training of our model.

4.1 Two-Branch Autoencoder

Fig. 2 shows the two-branch autoencoder, whose upper branch is intended for processing the geometry, while the lower branch processes the structure. The geometry branch consists of five 3D convolutional layers on the encoder side, accepting a series of k $32 \times 32 \times 32$ voxel maps as input. The 3D convolutional layers downsample the voxel maps by a ratio of 16 and are followed by a fully-connected layer to compute k 512D features. In parallel, the structure branch has a fully-connected layer to process K pairs of bounding boxes, producing K 512D features.

The features output by the encoder are fed into two different GRUs. These GRUs account for the relationships between parts in terms of geometry and structure, exchange information between them using the *Geometry and Structure Attention* components, and eventually output the k 512D features and K 512D features to the

and *2-Way VAE*. Finally, the decoder echoes the encoder with five 3D deconvolutional layers that transform latent features back into voxel maps. It also has a fully-connected layer to regress latent features to k bounding boxes for all parts.

4.2 Geometry and Structure Attention Component

The geometry and structure attention components are used to exchange the information between the upper and lower branches. The attention components are implemented with two fully-connected layers [Xu et al. 2017]. We formulate the information exchanged between the the upper and lower branches as:

$$m_i^{t+1} = \sum_{j \neq i} f([h_i^t, h_{i,j}^t]) h_{i,j}^t, \tag{1}$$

$$m_{i,j}^{t+1} = f([h_{i,j}^t, h_i^t]) h_i^t + f([h_{i,j}^t, h_j^t]) h_j^t, \tag{2}$$

where m_i^{t+1} and $m_{i,j}^{t+1}$ are the feedback messages for updating the hidden state of the upper- and lower-branch GRUs, respectively. $h_i^t \in \mathbb{R}^{512}$ is a geometry feature for the i-th part, which is produced by the upper-branch GRU. $h_{i,j}^t$ is the structure feature between the i-th and j-th parts, which is produced by the lower-branch GRU (see Fig. 2). t indicates the t-th iteration of GRUs. f represents fully-connected sigmoid-activated layers. To simplify notations, we denote all full-connected layers as f.

The feedback message m_i^{t+1} is computed as follows. For the i-th object part, we concatenate h_i^t with a structure feature $h_{i,j}^t$, feeding them to a fully-connected layer with sigmoid activation function. The outcome feature of the fully-connected layer weights the structure feature $h_{i,j}^t$, attending to the relevant components of $h_{i,j}^t$ that can be employed by the upper-branch GRU to update the geometric feature h_i^{t+1}. The message $m_{i,j}^{t+1}$ is computed by two terms. Using the structural feature $h_{i,j}^t$, the first term models the attention of the geometry feature h_i^t, while the second term attends to the geometry feature h_j^t. Eq. (2) summarizes the geometry information of the i-th and j-th parts for updating their structural feature $h_{i,j}^{t+1}$.

Using the messages defined in Eq. (1) and (2), we exchange the features representing the geometry and structure information between the two branches of SAGNet, as shown in Fig. 2. It is crucial to generate accurate 3D object models, which rely on the joint information of geometry and structure. We omit the superscript t in the last iteration of GRUs, denoting the geometry feature as h_i for the i-th part and the structural feature $h_{i,j}$ for the i-th and j-th parts. The resulting geometry and structural features are fed into the 2-way VAE, which is elaborated below.

4.3 2-Way VAE

The 2-way VAE also has an encoder-decoder architecture, which focuses on learning the dependencies between the geometry and structural features of a shape. As shown in Fig. 3, the 2-way VAE has an internal encoder, which consists of three GRUs. One GRU takes a sequence of geometry features $H_g = \{h_i | i = 1, ..., k\}$ (i.e., the k 512D features) as input. A second GRU processes the sequence of structural features $H_s = \{h_{i,j} | i = 1, ..., k, j >= i\}$ (i.e., the K 512D features). Each of these two GRUs encodes its input feature sequence into a single 512D feature as:

$$h_g = G_g^e(H_g),\ h_s = G_s^e(H_s), \tag{3}$$

where G_g^e and G_s^e represent GRUs. Thus, $h_g, h_s \in \mathbb{R}^{512}$ encodes the global geometry and structural information of all parts, respectively.

Since different shapes in the analyzed family may consist of different subsets of the k parts, in order to reconstruct such shapes it is necessary to provide a part mask $c \in \mathbb{R}^k$. Each element of c is a binary variable that indicates the presence (1) or absence (0) of a part. Using the part mask c together with the geometry and structural features h_g and h_s, we produce a new joint feature $h_v \in \mathbb{R}^{512}$ as:

$$h_v = G_v^e(f([h_g, c]), f([h_s, c])), \tag{4}$$

where G_v^e is a third GRU in the encoder of the 2-way VAE. The new feature h_v passes through an extra fully-connected layer to yield two 512D vectors, which are the mean and standard deviation of a Gaussian distribution. We then generate a random variable $n \in \mathbb{R}^{512}$ to produce a latent vector $z \in \mathbb{R}^{512}$ as:

$$z = \mu + \sigma n, \tag{5}$$

where μ and σ represent the mean and standard deviation.

The 2-way VAE also has an internal decoder that processes the latent vector z. Again, the decoder has three GRUs. Following the decoding procedure of [Bowman et al. 2016; Roberts et al. 2017], z is input to a GRU that outputs two 512D features. One 512D feature is fed to a decoder GRU to generate k 512D geometry features, and another 512D feature is used by another decoder GRU to produce K 512D structural features. Finally, the geometry and structural features are further processed by the decoder of our two-branch autoencoder to produce voxel maps and corresponding bounding boxes for all parts, as already explained earlier.

4.4 Network Training

The training of SAGNet includes two phases. In the first phase, we use a reconstruction loss to guide the training of the whole two-branch autoencoder. The first phase warms up the network training, avoiding the posterior collapse problem of the VAE [Bowman et al. 2016; Shen et al. 2018]. In the second phase, we keep the reconstruction loss for the two-branch autoencoder, while adding a KL loss and feature regularization for the training of our 2-way VAE. We detail the two phases below.

In the first phase, we define the training objective function as:

$$L_f = -E_{q_\phi(z|v,b,c)}[log(p_\varphi(v, b|z, c))], \tag{6}$$

where v and b denote the voxel maps and bounding boxes. c is the part mask that indicates the presence/absence of parts. z is the latent feature produced in the 2-way VAE. The distribution $q_\phi(z|v, b, c)$ is output by the encoder part of 2-way VAE, and the distribution $p_\varphi(v, b|z, c)$ is output by our two-branch autoencoder. The objective function L_f penalizes the reconstruction loss of the voxel maps and bounding boxes using the latent vector z.

In the second phase, we define the training objective as:

$$L_s = L_f + \lambda L_{KL} + \eta R, \tag{7}$$

where

$$L_{KL} = KL(q_\phi(z|x, y, c)||p_\phi(z|c)), \tag{8}$$

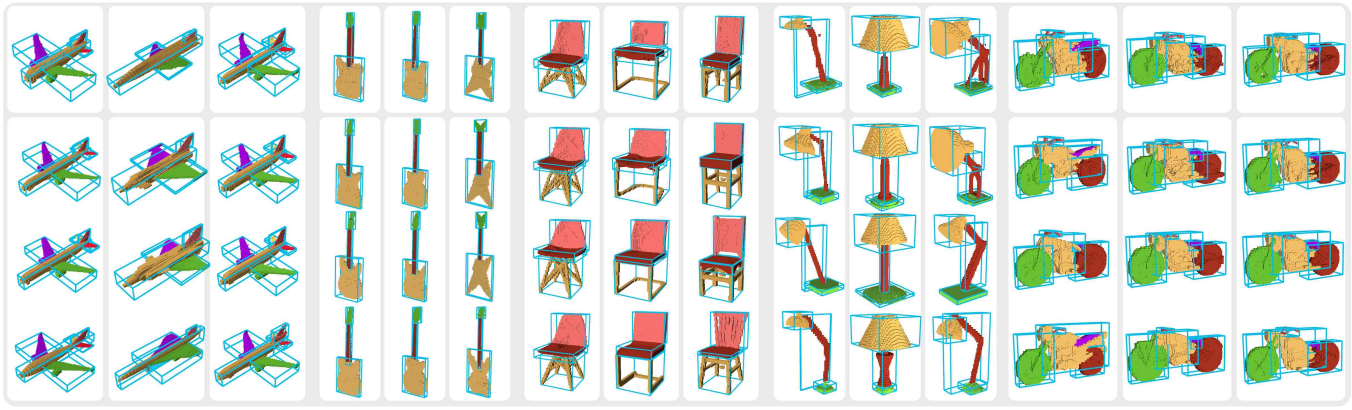

Fig. 4. For each generated sample in the top row, we retrieve the 3-nearest neighbors in the training data. It may be seen that the generated shapes are original.

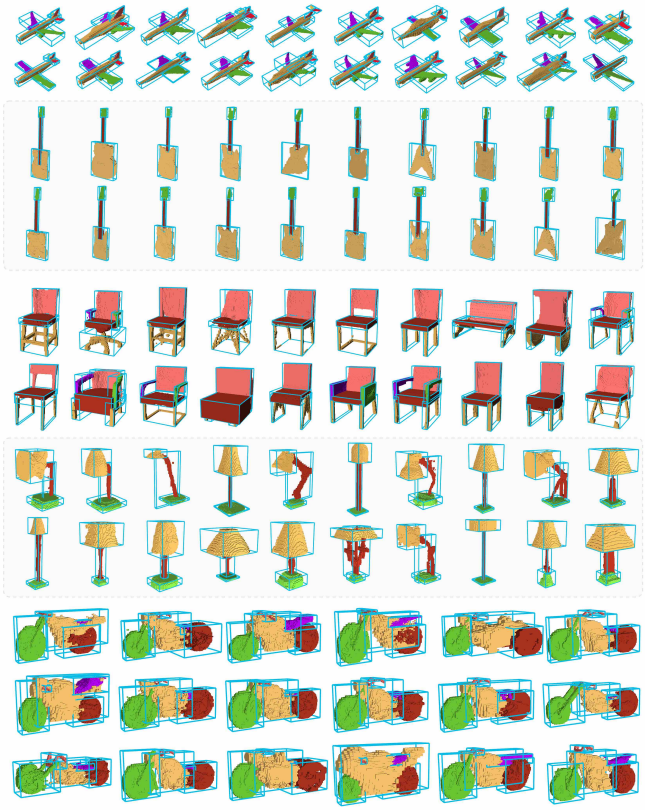

Fig. 5. A gallery of shapes generated from random latent codes. Note that each part is associated with its bounding box.

$$R = \sum_{i=1}^{k} ||h'_i - h_i||_2^2 + \sum_{i=1}^{k} \sum_{j=i+1}^{k} ||h'_{i,j} - h_{i,j}||_2^2, \tag{9}$$

where $p_\phi(z|c)$ is a standard Gaussian distribution as prior, h'_i denotes the i-th part's geometry feature, and $h'_{i,j}$ denotes the structural feature of the i-th and j-th parts. Both h'_i and $h'_{i,j}$ are produced by the 2-way VAE. During the second phase, we gradually increase the factors λ and η, which provide SAGNet its generation power.

We use the TensorFlow platform [Abadi et al. 2016] to construct SAGNet. All the parameters are randomly initialized and optimized by the standard SGD solver. The network is trained with a learning rate of 0.001 for 70000 mini-batches, using a mini-batch size of 10.

class	airplane	chair	guitar	lamp	motorcycle
object #	1907	3700	779	1482	202
part #	6	7	3	4	3

Table 1. Shape classes and numbers of objects and parts in our dataset.

5 RESULTS AND EVALUATION

We evaluate SAGNet using the data collected by Yi et al. [2016] and Kae et al. [2013]. There are five classes of objects, including airplane, chair, guitar, lamp and motorcycle. We correct the problematic meshes provided in [Kae et al. 2013; Yi et al. 2016]. We divide each object into parts, using the meshes to newly generate voxel maps and bounding boxes for parts. We have a total of 8183 objects for training and testing our network. See Table 1 for more details on the dataset. We have also created a set of simple 3D shapes with two parts (see Fig. 11), where the dependency relation between the two parts is clear and easy to evaluate quantitatively. We show using this all the data that our method indeed learns the latent relationship, successfully encodes it, and generates new shapes where this relationship is maintained.

5.1 Shape Generation

We train a SAGNet for each class of objects. Using the trained network, we can generate object shapes that are represented by voxel maps and bounding boxes for each of their parts. To generate an object shape, we first sample a 512D latent code from a standard Gaussian distribution, using Eq. (5) that generates the latent code with the internal encoder of 2-way VAE. Next, we input the sampled latent code to the decoder of 2-way VAE, computing the voxel maps and bounding boxes of all parts. See Fig. 5 for a gallery of generated

shapes. For each part of a generated shape, there is an associated voxel maps and a bounding box.

The generative network creates novel parts for the shapes, which are different from the given training data. To show the generative power of SAGNet, we compare the generated shapes with the training data. For a generated shape, we retrieve its 3-nearest neighbors in the training data, where the distance between two shapes is the sum of Chamfer distances between corresponding parts [Achlioptas et al. 2017]. As may be seen in Fig. 4, the generated shapes are original, and exhibit various differences from their nearest neighbors.

The distance between two shapes are defined based on their bounding boxes and voxel maps. Given the bounding boxes, we compute and sum the Euclidean distances between the corresponding parts of two shapes. In the voxel maps of a shape, we employ the 3D coordinates of the associated bounding boxes to compute the point cloud for representing each occupied voxel. For each voxel of a shape, we compute the Euclidean distance to the nearest voxel of another shape. All the distances between voxels are summed to form a Chamfer distance. The Euclidean distance of bounding boxes and Chamfer distance of point clouds are summed again as the overall distance between two shape.

We conjecture that SAGNet combines the existing patterns of the training data, creating new patterns for the generated shapes. To verify this, we select two different shapes in the same class from the training set. Their voxel maps and bounding boxes of parts are fed into SAGNet, which outputs two latent vectors corresponding to the given input shapes. Then we perform linear interpolation using the two latent vectors. By controlling the interpolation rate, we compute the latent code of novel shapes, which can be regarded as the combination of the training data. The interpolated latent code is input to the decoder of 2-way VAE for generating voxel maps and bounding boxes of the interpolated shape. We show the training data (see the left- and right-most columns of Fig. 6) and the generated results (see the middle columns of Fig. 6) based on the computed latent vectors. Obviously, the generated shapes have different patterns with the training data.

We have shown that vectors sampled from the latent space correspond to object shapes. Here, we further investigate the properties of the latent space. For better visualization, we retrain SAGNet by modifying the dimension of the latent space to two. This allows us to show and compare the latent vectors in a 2D Euclidean space. We find that similar shapes appear to be close in the 2D latent space, forming apparent clusters, as shown in Fig. 1. This demonstrates that SAGNet builds a proper relationship between the shapes and latent vectors, which is important for the shape generation task.

5.2 Learning the Geometry-Structure Relationships

SAGNet has an GRU-based encoder and attention component, used to jointly learn the relationship between the geometry and structure of the shape parts. We perform an ablation study to show the importance of the inclusion of the GRUs and Attention modules. We compare SAGNet with two other baseline networks that use simpler schemes to analyze the geometry and structure information. The first *no-attention* baseline is defined by removing the attention component to disable the information exchange between the geometry

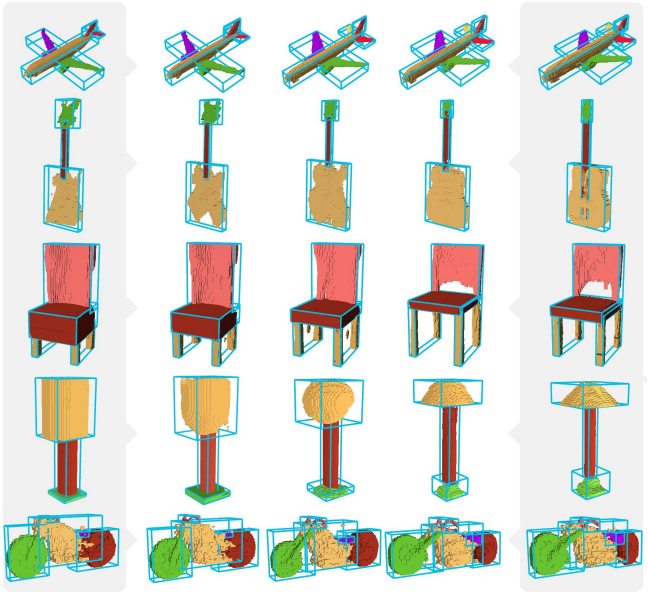

Fig. 6. The left- and right-most samples are randomly selected and paired from the training data. Their latent codes are then linearly interpolated to create three intermediate latent codes that are used to generate the 3D shapes shown in the three middle columns.

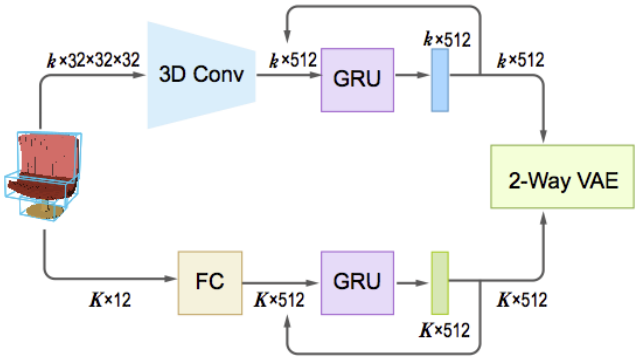

Fig. 7. The architecture of the no-attention baseline model.

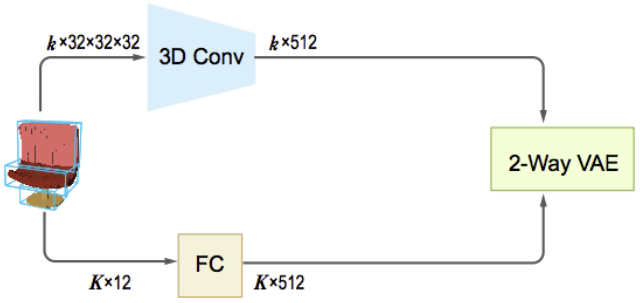

Fig. 8. The architecture of the linear baseline model.

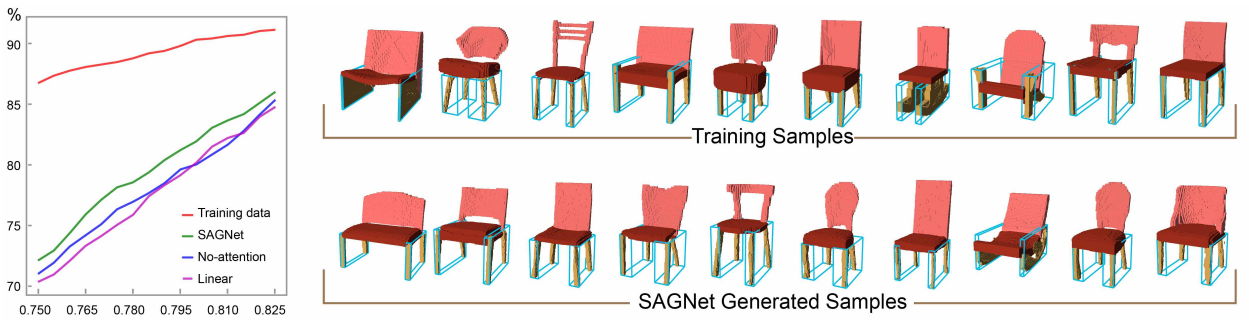

Fig. 9. We measure the symmetry scores for legs of chairs generated by SAGNet and the baseline models. Along the horizontal axis, we set different thresholds for the scores. Along the vertical axis, we provide the percentage of shapes, which have smaller scores than the given thresholds.

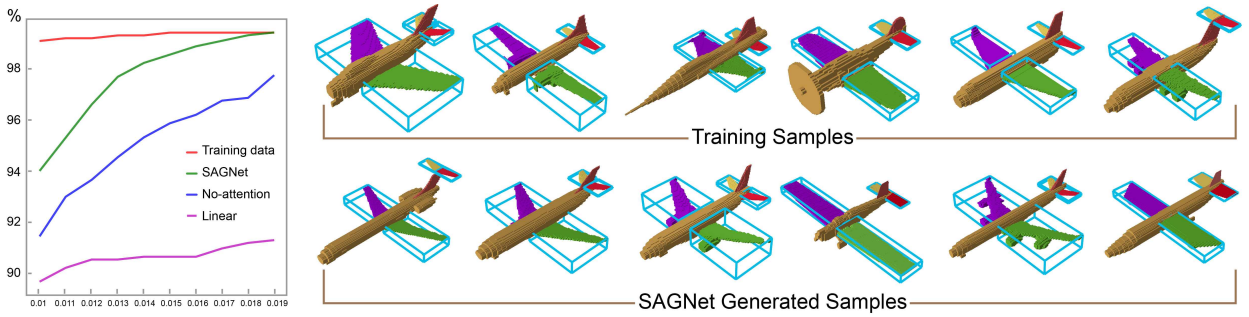

Fig. 10. We measure the centroid-to-plane distances for each airplane generated by SAGNet and the baseline models. The centroid-to-plane distances are computed using the fore- and back-wings of airplanes. Along the horizontal axis, we provide different thresholds for the distances. Along the vertical axis, we provide the percentage of shapes, which have smaller distances than the given thresholds.

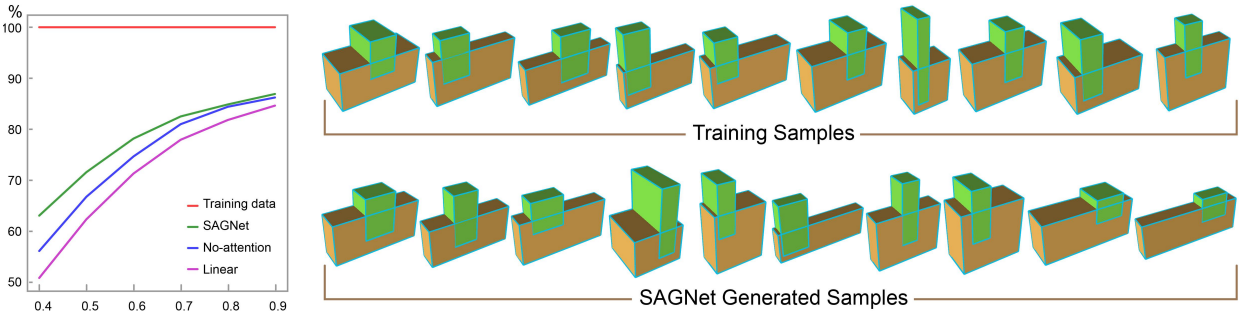

Fig. 11. A class of two-part shapes, where a convex part fits exactly into the cavity of the non-convex one. It is hard to infer the geometry of the non-convex part from its bounding box only. To infer it, the relationship with the convex part must be learned. Along the horizontal axis, we provide different thresholds for the fitting accuracy. Along the vertical axis, we provide the percentage of shapes, which have better fitting accuracy than the given thresholds.

and structure branches of SAGNet, as shown in Fig. 7. A second, *linear* baseline is defined by removing the GRUs from both of these branches, as shown in Fig. 8. The removal of the GRUs, effectively disables the modeling of the relationship between parts.

Symmetry Analysis. We conduct a part symmetry analysis to evaluate the quality of generated results produced by SAGNet and the baseline models. Given a generated shape, there are pairs of parts (e.g., the legs of chairs and the wings of airplanes) that are supposed to be symmetric (see Fig. 9). We compute the symmetry score for these parts. Specifically, here we focus on the legs of chairs. We perform a mirror reflection of one leg to another. To measure how similar the legs are after the reflection, we use the distance defined in Section 5.1 as the score. The model that yields lower scores performs better. In Fig. 9, we randomly select 1000 training shapes, computing their scores. We provide the percentage of shapes having lower scores than the given thresholds. We find

that the training shapes generally have lower scores, showing strong symmetric property.

For each model, we randomly collect 1000 generated chairs and compute their symmetry scores. In Fig. 9, we report the scores of SAGNet and the baseline models. SAGNet results in lower scores than those of the baseline models, which demonstrates that it learns the symmetric property of shapes better.

Coplanarity Analysis. Similarly to symmetry, many objects exhibit coplanarity of parts (see Fig. 10). Here we use the wings (i.e., two fore-wings and two back-wings) of airplanes, measuring their coplanarity to show the quality of the generated shapes. Using the voxel map and the corresponding bounding box, we compute a centroid for each wing of a generated airplane. Then we use the centroids of two fore-wings and the left back-wing to determine a plane in 3D space. We then compute the Euclidean distance from the centroid of the right back-wing to that plane. Smaller distances imply a better coplanarity property. We select 1000 airplanes from the training data, and compute the centroid-to-plane distance for each airplane. In Fig. 10, we report the percentage of training shapes having smaller distances than given thresholds. The training shapes shows high coplanarity.

We also report the generative models' performances in terms of coplanarity. We randomly generate 1000 airplanes using each model, and compute the centroid-to-plane distance for each airplane. In Fig. 10, we show that SAGNet produces shapes having better coplanarity property than other models do.

Cavity Analysis. In order to explicitly show the ability to process and learn the relationship between geometry and structure, we build a simple synthetic dataset to train our neural network. Each shape in the dataset, consists of just two parts, as shown in Fig. 11. One part, in orange, has a cavity into which the second part, in green, exactly fits. The idea of such apparently simple shape class is that the geometry of the orange, non-convex part is hard to infer just from its bounding box. Only by learning the geometry-structure relationships between the two parts, it is possible to infer the geometry of the non-convex part, and in particular, the location of the cavity. The same must hold in order for the generated convex part to exactly fit the cavity.

We use 20000 synthetic example shapes to train our framework, some of which are shown in the top row of Fig. 11. Next, we randomly generate 3000 test samples, feed them to the trained network and measure how well the convex parts fit into the cavity of the non-convex ones. To quantitatively measure the fitting accuracy, we calculate, $R_o(R_e)$, the portion of occupancy(empty) voxels of the non-convex part that locate in the bounding box of the convex part. With smaller R_o, there are less occupancy voxels wrongly placed in the convex part. A larger R_e means that the convex part fits the cavity of the non-convex part better. Then the score $R = 1 - (R_e - R_o)$ is used to measure how well a generated sample satisfies the dependency. The smaller R indicates better fitting status between the two parts. Fig. 11 shows the performance of each model. Like in Fig. 9, the value in the horizontal axis is a threshold and the vertical axis indicates the percentage of objects for which R is lower than the given thresholds. As we can see, the performance of SAGNet is significantly better than other two baselines.

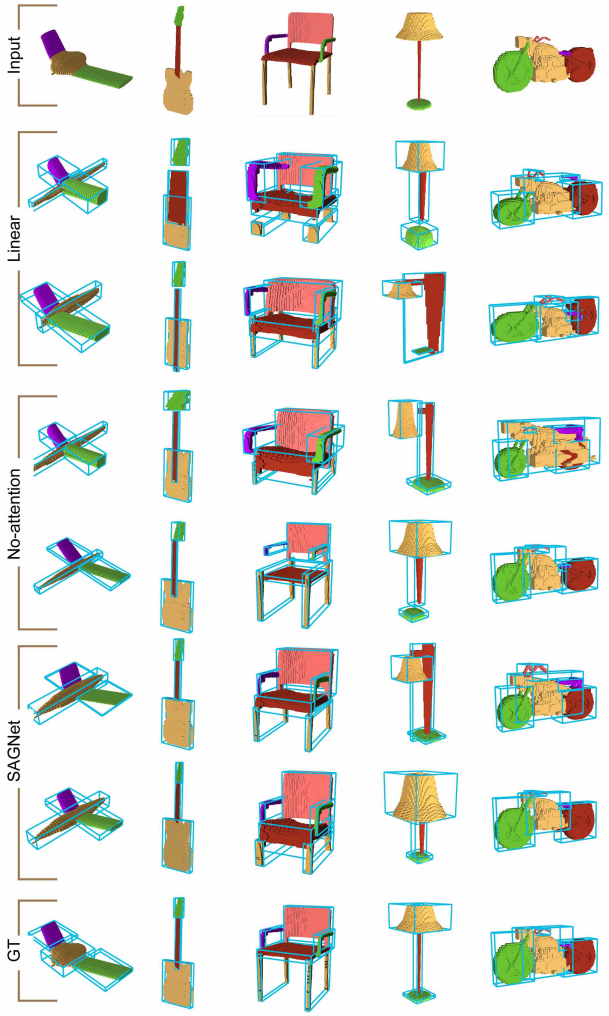

Fig. 12. Visual comparison of geometry-to-structure mapping results.

5.3 Geometry-Structure Mapping

Jointly analyzing the geometry and structural information learns their interdependency. Given geometry information only, a generative model that captures the dependency better is supposed to provide a reasonable inference of the structural information, and vice-versa. Thus we conduct a bidirectional mapping between the geometry and structural information to examine whether SAGNet learns the dependency well; see, e.g., Figs. 12 and 13.

We take all the training data from each class to test the models. Given the voxel maps of all parts only, we randomly initialize the corresponding bounding boxes. Then the voxel maps and bounding boxes are input to the two-branch autoencoder, generating a new set of voxel maps and bounding boxes, which are fed back into the autoencoder. This is repeated for 300 iterations. We compute the Euclidean distance between the ground-truth and generated bounding boxes for each object. The distances for different objects are averaged. In Table 2, we list the average distances produced by SAGNet

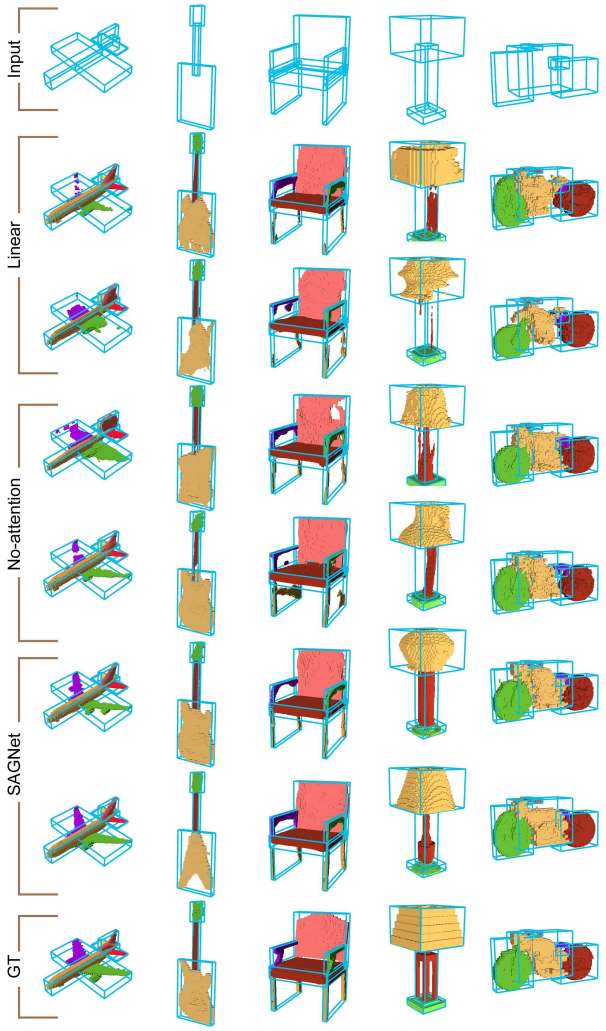

Fig. 13. Visual comparison of structure-to-geometry mapping results.

class	structure-geometry/geometry-structure		
	SAGNet	no-attention	linear
airplane	**0.0051**/**0.0265**	**0.0051**/0.0269	0.0053/0.0282
chair	**0.0258**/0.0581	0.0271/**0.0579**	0.0264/0.061
guitar	**0.0038**/0.0238	0.0039/0.0242	0.0039/**0.0237**
lamp	**0.0140**/**0.0824**	0.0141/0.0853	0.0161/0.0832
motor	**0.0090**/**0.0350**	0.0092/0.0365	0.0094/0.0365

Table 2. The comparisons between SAGNet and baseline models on the tasks of structure-to-geometry and geometry-to-structure mapping.

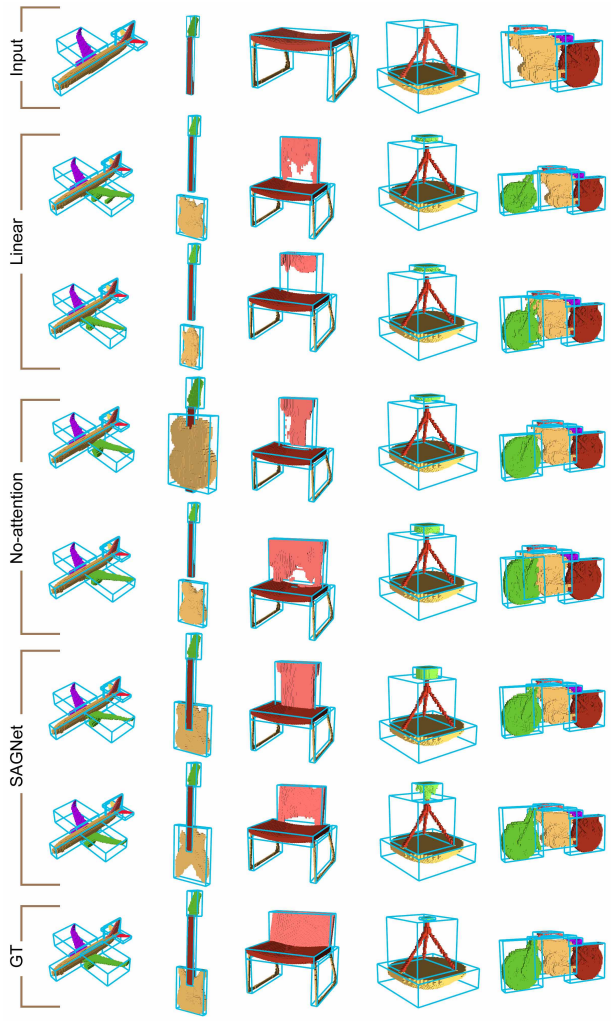

Fig. 14. Visual comparison of shape completion results.

and baseline models (see the entry *geometry-structure*). We follow a similar procedure to map bounding boxes to voxel maps. The voxel maps are randomly initialized and input to the two-branch autoencoder for 300 iterations. We compute the Chamfer distance using the ground-truth and generated voxel maps, averaging the distances for different object. See the entry *structure-geometry* of Table 2. It may be seen that SAGNet outperforms the baseline models. The mapping between the geometry and structure demonstrates that SAGNet effectively exchanges information to capture the dependencies between geometry and structure.

5.4 Shape Completion

We further conduct a shape completion task to evaluate the models. Given an object with missing parts, the models should complete these parts by inferring their voxel maps and corresponding bounding boxes. It requires the models to learn the underlying relationship between parts effectively. Given an object, the missing parts are randomly initialized with voxel maps and bounding boxes that are input to the two-branch autoencoder. We follow a similar completion process in [Nash and Williams 2017; Rezende et al. 2014]. The voxel maps and bounding boxes of the missing parts are output

class	shape completion		
	SAGNet	no-attention	linear
airplane	0.0240	**0.0239**	0.0269
chair	**0.0600**	0.0618	0.0625
guitar	**0.0066**	0.0067	0.0067
lamp	**0.0649**	0.0674	0.0650
motor	**0.0056**	0.0058	0.0059

Table 3. The comparisons between SAGNet and baseline models on the task of shape completion.

by the autoencoder and fed back into it again for 300 iterations to produce the final completion result.

For each class, we take the training data as ground-truth. For each object, we remove some of their parts. The objects with missing parts are then input into SAGNet and baseline models, respectively. To measure the quality of completion, we compute the distance (see Section 5.1) between the ground-truth parts and the corresponding generated parts. The distances for different objects are averaged. Again, lower score is better. We report the scores of SAGNet and two baseline models in Table 3. In general, SAGNet achieves lower scores than the baseline models. This is because SAGNet is better equipped to exploit the relationship between parts. In Fig. 14, we compare our shape completion results with the baseline models. SAGNet completes the missing parts with better visual details.

6 CONCLUSIONS AND FUTURE WORK

We have presented a network that allows generating 3D shapes with separate control over their geometry and structure. We use weak supervision, in the form of a semantically segmented training set, in order to learn the implicit dependencies between the geometry of parts and their spatial arrangement. More specifically, we have explicitly demonstrated that the geometry generated in one bounding box, representing one part, is aware of the geometry generated in another bounding box. Since the learned pairwise relationships among the different parts reflect the structure of the shape, we refer to our generative model as structure-aware.

It should be noted that our two-branch autoencoder has similarities with conditional autoencoders in the sense that it encodes information coming from two sources. However, here the two branches learn to extract and intertwine geometry and structure features. This opens up more possibilities for future research. One is to learn other properties in parallel using two separate branches, and intertwine them by a two-way autoencoder. For example, one branch could learn the style of an object and encode it in a feature, while the other branch learns the geometry, and fuse these two features together. Another direction is the development of a k-way autoencoder (with $k > 2$), where k properties are learned in parallel using k interconnected branches. The challenge is then to create or collect proper datasets to weakly supervise the learning.

The current training data assumes the objects are segmented into semantic parts. The generative model that we presented does not fully exploit the potential of such data. One can learn more about the geometry of the parts themselves, possibly by employing part-level generators that could potentially generate finer details.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2017. Learning Representations and Generative Models for 3D Point Clouds. *arXiv preprint arXiv:1707.02392* (2017).

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. (2016), 10–21.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. (2014), 103–111.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Proc. Euro. Conf. on Computer Vision*. 628–644.

Noa Fish, Melinos Averkiou, Oliver van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J. Mitra. 2014. Meta-representation of Shape Families. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 33, 4 (2014), 34:1–34:11.

Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. 2016. Learning a predictable and generative vector representation for objects. In *Proc. Euro. Conf. on Computer Vision*. 484–499.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. (2015), 1462–1471.

JunYoung Gwak, Christopher B Choy, Animesh Garg, Manmohan Chandraker, and Silvio Savarese. 2017. Weakly supervised generative adversarial networks for 3D reconstruction. *arXiv preprint arXiv:1705.10904* (2017).

Zhaoyin Jia, Andrew Gallagher, Ashutosh Saxena, and Tsuhan Chen. 2013. 3D-based reasoning with blocks, support, and stability. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 1–8.

Andrew Kae, Kihyuk Sohn, Honglak Lee, and Erik Learned-Miller. 2013. Augmenting CRFs with Boltzmann machine shape priors for image labeling. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019–2026.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding variational bayes. In *Proc. Int. Conf. on Learning Representations*.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 36, 4 (2017), 52:1–52:14.

Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J Guibas. 2015. Joint embeddings of shapes and images via CNN image purification. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 34, 6 (2015), 234:1–234:12.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In *Proc. Int. Conf. on Neural Information Processing Systems*. 700–708.

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Proc. Int. Conf. on Neural Information Processing Systems*. 469–477.

Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. 2014. Structure-aware Shape Processing. In *ACM SIGGRAPH Courses*. 13:1–13:21.

Charlie Nash and Chris KI Williams. 2017. The shape variational autoencoder: A deep generative model of part-segmented 3D objects. In *Computer Graphics Forum*, Vol. 36. 1–12.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. (2016), 1747–1756.

Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. 2016. Unsupervised learning of 3D structure from images. In *Proc. Int. Conf. on Neural Information Processing Systems*. 4996–5004.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. (2014), 1278–1286.

Adam Roberts, Jesse Engel, and Douglas Eck. 2017. Hierarchical variational autoencoders for music. (2017).

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving Variational Encoder-Decoders in Dialogue Generation. (2018).

Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. 2012. Convolutional-recursive deep learning for 3D object classification. In *Proc. Int. Conf. on Neural Information Processing Systems*. 656–664.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proc. Int. Conf. on Machine Learning*. 129–136.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial

modeling. In *Proc. Int. Conf. on Neural Information Processing Systems*. 82–90.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D shapenets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 1912–1920.

Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, Vol. 2. 3097–3106.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. 2016. Perspective transformer nets: Learning single-view 3D object reconstruction without 3d supervision. In *Proc. Int. Conf. on Neural Information Processing Systems*. 1696–1704.

Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. 2016. A scalable active framework for region annotation in 3D shape collections. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 35, 6 (2016), 210:1–210:12.

Bo Zheng, Yibiao Zhao, Joey Yu, Katsushi Ikeuchi, and Song-Chun Zhu. 2015. Scene understanding by reasoning stability and safety. *Int. J. Computer Vision* 112, 2 (2015), 221–238.

Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 2017. 3D-PRNN: Generating shape primitives with recurrent neural networks. In *Proc. Int. Conf. on Computer Vision*. 900–909.