Graphical Models 129 (2023) 101185

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Component-aware generative autoencoder for structure hybrid and shape completion

Fan Zhang, Qiang Fu *, Yang Liu, Xueming Li

School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications, Beijing, 100876, China

ARTICLE INFO

Keywords:
Generative autoencoder
Structure hybrid
Shape completion
3D modeling

ABSTRACT

Assembling components of man-made objects to create new structures or complete 3D shapes is a popular approach in 3D modeling techniques. Recently, leveraging deep neural networks for assembly-based 3D modeling has been widely studied. However, exploring new component combinations even across different categories is still challenging for most of the deep-learning-based 3D modeling methods. In this paper, we propose a novel generative autoencoder that tackles the component combinations for 3D modeling of man-made objects. We use the segmented input objects to create component volumes that have redundant components and random configurations. By using the input objects and the associated component volumes to train the autoencoder, we can obtain an object volume consisting of components with proper quality and structure as the network output. Such a generative autoencoder can be applied to either multiple object categories for structure hybrid or a single object category for shape completion. We conduct a series of evaluations and experimental results to demonstrate the usability and practicability of our method.

1. Introduction

With the development of deep learning methods, various deep networks have been proposed to tackle 3D modeling problems in the field of computer graphics. For example, modeling tasks such as shape estimation [1], multi-view reconstruction [2], and virtual data synthesis [3] have been well addressed by applying deep networks. As one of the most common-seen modeling targets, man-made objects across different object categories can have various shapes and structures. For most of the traditional non-deep-learning-based 3D modeling methods, analyzing the structure of man-made objects in terms of components is an essential step before synthesizing new shapes. By representing man-made objects with components and the way how they are combined, components in the same object category can be re-assembled following the known structures [4,5]. Similarly, deep networks that encode component-based structures of man-made objects have been well-studied for 3D modeling with shape variations (e.g., [6]).

On the other hand, exploring new ways of component combination [7] and even across different object categories [8,9] could enrich the synthesized man-made objects. Besides the academic research, we also observe that products like furniture could have hybrid structures for multi-function purposes, which has been popular in industrial product design (e.g., Fig. 1). However, such a goal is challenging for the current deep-learning-based 3D modeling methods, either these represent shapes by volume or these use component relations encoding certain structures. That is mainly because, when constructing a 3D model with a new component combination, it needs to determine which components should exist and where they should be. Such questions also involve design issues including supporting relation, affordance, and aesthetics. Therefore, arbitrarily creating new component combinations would easily fail.

In this paper, we propose a novel generative model that tackles the component combinations for the 3D modeling of man-made objects. Our method can be applied to either multiple object categories for structure hybrid, or a single object category for shape completion (e.g., Fig. 2). Our key idea is to use a GAN-based architecture and adopt the autoencoder as the generator. Since the task of combining two models into a new structure can hardly find ground truth for supervised learning, using a discriminator that distinguishes between synthetic models and database models can improve the quality of the generated 3D model. We train the network by creating each input model and the hybrid of the inputs so that the input hybrid can also profit from the trained discriminator. For example, the discriminator could ensure the integrity of a hybrid structure given two 3D models. In our implementation, we only use the two input models instead of a large-scale dataset to create the training data. This enabled our network to extract the individual structure of each input model and explore the potential component combination of the inputs simultaneously.

* Corresponding author.
E-mail addresses: zhangfan.kanv@gmail.com (F. Zhang), fu.john.qiang@gmail.com (Q. Fu), yang.liu@bupt.edu.cn (Y. Liu), lixm@bupt.edu.cn (X. Li).

https://doi.org/10.1016/j.gmod.2023.101185
Received 8 September 2022; Received in revised form 1 May 2023; Accepted 14 June 2023
Available online 18 July 2023

Fig. 1. Examples of IKEA's furniture with hybrid structures.

During the training stage, we use the volume-based components with randomized positions, directions, and scales as the training data. This could avoid the network from learning a fixed relation between the input components and the output structure. Additionally, we use redundant component inputs to train our network's ability to select proper components from the inputs, which is important for generating hybrid shapes.

In this manner, for the application of structure hybrid, our generative model is trained with the segmented reference objects across multiple categories. We use all components of the references to create component volumes as the inputs of the trained model. The output 3D shape is a plausible component combination determined by the generative autoencoder. For the application of shape completion, we use a similar approach as structure hybrid, but apply it to a single object category. Given an incomplete 3D shape, we explore a 3D shape from the dataset similar to the input as the auxiliary. Then we use the components of both the input and the retrieved dataset object to create a new 3D shape through our generative autoencoder. Benefiting from the discriminator, the components of the output shape would be complete.

To sum up, our work makes two major contributions: (1) a novel generative autoencoder that converts volume-based components to 3D models with plausible structure, (2) a framework that leverages the generative model to tackle the multi-category structure hybrid and single-category shape completion. We show a series of experimental results on both structure hybrid and shape completion, to demonstrate the usability and practicability of our method.

2. Related work

A large amount of current work is based on the assembly of components for 3D modeling. This approach is inspired by generic component assemblies of fixed design, such as Lego assembly models and IKEA furniture. The reusability and extensibility of assembled components are its advantages. In this section, we will discuss work related to 3D modeling based on the assembly of components. However, most of the current work is limited to single-category or geometrically similar component assemblies, and there are still a lot of possibilities to develop work across categories.

Assembly-based modeling. The 3D modeling approach, inspired by the real-world assembly process described above, has been earlier proposed in the Modeling by Example system [10], which allowed the user to select interesting components from a large database of 3D objects and combine them to form new objects. To combine the components of 3D objects across categories in a rational way, Lee et al. [11] provided an interactive system based on 2D sketches to leverage the user inputs to explore proper components from the database to create new 3D shapes. Some ideas for automated suggestions were then proposed. Based on a basic shape given by the user, Chaudhuri

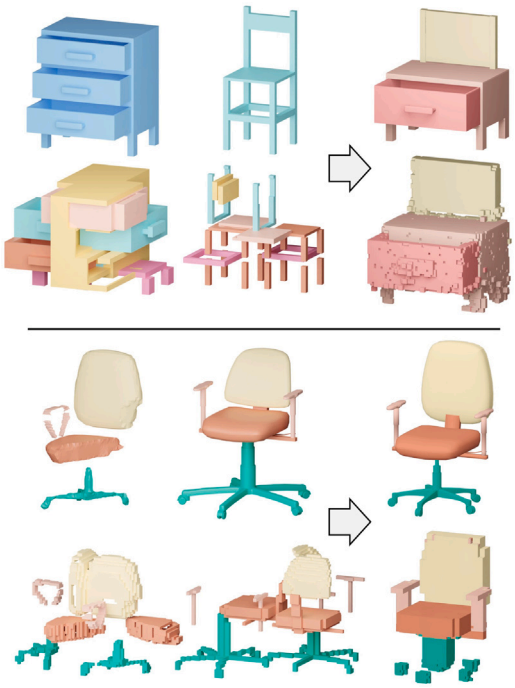

Fig. 2. Top: Given two shapes from different categories, our method uses their component volumes as the network inputs to generate new shapes with hybrid structures. **Bottom:** Given an incomplete shape, our method can retrieve a similar shape from a dataset as the reference to complete the input shape.

et al. [12] used a data-driven approach to suggest to the users to extend this basic shape in ways that stimulate creative exploration. Ovsjanikov et al. [13] leveraged low-dimensional manifolds in the descriptor space to learn the template of the same class of 3D objects and help the user to select the most meaningful deformation. Fish et al. [14] used meta-representations to learn the assembly statistics positions of individual components for the same class of 3D objects, which can automatically assign components to reasonable positions with high probability. Su et al. [9] proposed to automatically combine user-supplied cross-class models into new models based on a reference shape. Fu et al. [15] developed a system that changes the structure of the user-edited 3D shape adaptively, by switching the structure between different ways of assembly. Mo et al. [16] proposed an order-invariant encoding of n-ary graphs and utilized a hierarchical graph network that directly encodes shapes to generate realistic structured shape geometries. Yin et al. [17] proposed a data-driven framework for synthesizing part connection by removing the mismatched portions and filling the part's gap. Wan et al. [18] represented shape as contact-based reasoning, and generated complete shape from retrieved parts by a graph-neural-network-based model. [19] proposed to model 3D shape variations with a part-aware deep generative network, which is composed of an array of per-part VAE-GANs, generating semantic parts composing a complete shape.

Components exploration. Chaudhuri et al. [20] learned probabilistic graphical models in a given shape library to discover semantic and geometric relationships between components, suggesting to the user components that are semantically and stylistically compatible with the new 3D model. Kreavoy et al. [21] proposed methods specifically for artificial 3D objects to implement compatible partitioning on a group of 3D objects of the same or similar class entered by the user, and support the user to interchange components to create new models. Fu et al. [8] developed a system that leverages human pose input to indirectly infer the potential components for shape modeling from a cross-category man-made object dataset. Sung et al. [22] suggested complementary components and their placement for an incomplete 3D part assembly by neural network architecture. Hertz et al. [23] proposed neural

implicit fields to generate or edit 3D shapes. It decomposes shapes into distinct embeddings which correspond to sub-components. Then the contextual embeddings are combined with the distinct embeddings to reduce them to 3D shapes. As a prior study of component replacement, Kim et al. [24] proposed a method that allows users to browse a 3D object datasets based on similarities and differences between the shapes of user-specified regions of interest (ROIs) by computing similarity relationships between points on 3D objects. Van Kaick et al. [25] used segmented and labeled components as prior knowledge to explore the component correspondence of a set of 3D objects that are similar geometrically. Some works proposed several different encoding methods and segmentation methods based on the symmetry level of artificial objects (e.g., [6,26,27]), while some others proposed a collaborative component segmentation method for the same class of 3D object collections (e.g., [28,29]).

Shape evolution. Automatically combining or exchanging objects' components could lead to shape evolution. Jain et al. [30] proposed to create new shapes by recombining components from different shapes based on constraints derived from shape analysis. Kalogerakis et al. [31] analyzed a generative probabilistic model of the shape structure of a small database of 3D objects of the same class to automatically synthesize reasonable new shapes from the class domain. This application can extend the original dataset. Xu et al. [32] developed a *fit and diverse* system that utilizes evolution, where the 3D model groups are evolved into several generations of new shapes, and the new shapes are presented to the user after the reasonableness assessment to provide modeling suggestions to the user. Zheng et al. [33] took into account that the particular ordering between symmetry-related substructures is closely related to the object function, so they base their pure geometric approach to matching and replacing components on symmetry substructures. Fu et al. [34] proposed to make shape variations of man-made objects by using the artistic lines extracted from natural objects. Bokelohd et al. [35] considered local similarity when generating new models, where each local neighborhood of the new model must match some local neighborhood of the exemplar. Similarly, the work of Alhashim et al. [36] attempted to merge two shapes with different topologies into a new shape based on topological geometry. The work of Xu et al. [37] considered the influence of style. This method first separates the style and content of 3D objects, then synthesizes new structures using component segmentation with content changes, and applies style transfer to the new structures to synthesize new objects. Zhu et al. [4] used a recursive autoencoder network to synthesize a new shape structure using the set of components from two or more source 3D shapes and the rough initial positions of the components as input. Jin et al. [38] proposed a VAE-based embedding space of 3D shapes and 2D contours, thus supporting contour-based shape modeling and shape evolution. Guan et al. [39] proposed an evolution-based method that uses a set of cross-category shapes to generate novel part recombination while aware of functionalities. This work has a very similar goal to ours. The main difference is that we do not need extra effort to analyze the structures and functions of the input objects, such as the structure graph and the part connect relations. Instead, our generative model can learn such information when training the decoder and discriminator with the component and object volume pairs.

3. Overview

For man-made objects, especially ones across different categories, some components might share similar shapes but have different structural relations. Therefore, training a generative autoencoder by using all components from a large-scale dataset with cross-category objects, might fail to extract meaningful structure priors required by the applications like structure hybrid and shape completion. In our work, we directly train our generative autoencoder with the components of the input objects, thus ensuring the hybrid structure or the completed shape satisfies the constraints of the inputs.

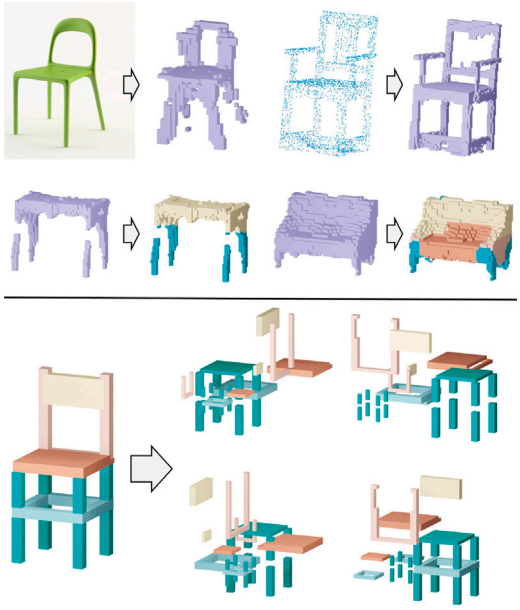

Fig. 3. Top: The volume inputs of our method can be obtained from images by [40] or from scanning data by resampling. We employ a segmentation network to extract semantic- and structure-related components from the object volume. **Bottom:** By randomly resetting the component amount and the positions/directions/scales of the components, we can obtain various component volumes (Right) from the segmented object (Left).

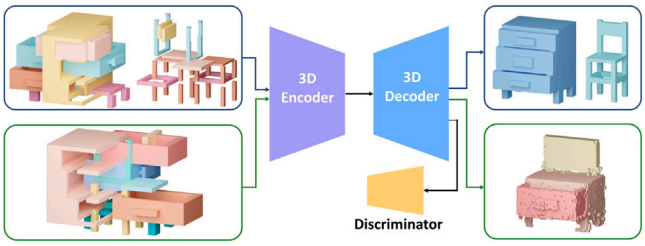

Fig. 4. The pipeline of our method for structure hybrid. We use the randomly generated component volumes and their associated input 3D shapes to train the generative model, which includes an encoder, a decoder, and a discriminator. We apply the trained generative model to a component volume mixed with the components of the two input shapes, thus obtaining a novel object volume with a hybrid structure.

The training stage of the architecture of our generative autoencoder includes an encoder, a decoder, and a discriminator (Section 4). For both the two applications, we first conduct a pre-processing stage to segment the volumes of the input models into components, and create the component volumes that have redundant components and random configurations. Then, we use the component volumes of the inputs as well as a mixed component volume as the network inputs, and train the generative autoencoder by using the input object volumes as the ground truths. The trained autoencoder can choose the proper numbers of components and determine their positions/directions/scales to construct a meaningful 3D object.

Pre-processing. In the pre-processing stage, the input models are manually edited to have real scales and aligned with the coincident centers and front directions. Such an operation is also performed on all datasets adopted in our method. Then, the inputs are converted to volumes and segmented into volume-based components (e.g., Fig. 3-Top). Note that since our method accesses volume inputs, users can also use point clouds, depth maps, or images [40] to convert the input models to volume data. The volume data is normalized into a $64 \times 64 \times 64$ voxel

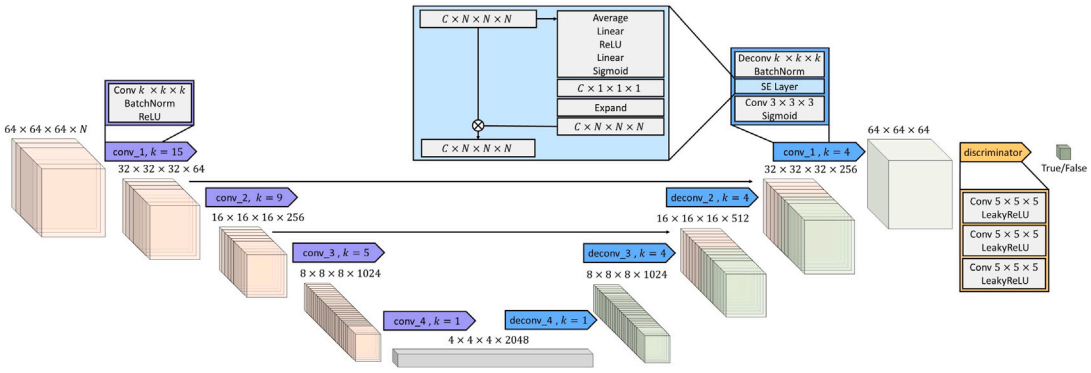

Fig. 5. The detailed network architecture. Note that the purple, blue, and yellow layers are associated with the 3D encoder, 3D decoder, and discriminator, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

space. We employ a segmentation network that has 1 encoder and K decoders, where K is the empirical number of components. For example, tables $K = 4$, chairs $K = 5$, beds $K = 5$, and cabinets $K = 7$. We use the data provided by [41], and manually segment the 3D meshes of these data into meaningful parts, to train the segmentation network for each object category. Totally, we have 139 objects across 7 categories to train the segmentation network. After that, we increase the amount of the segmented components by repeating components, aiming at providing redundant components for the generative model to choose. Note that all components of the volumes are independent and have individual numbers as their labels. And then, we configure these components with random positions/directions/scales (limited in reasonable ranges) thus creating various component volumes (e.g., Fig. 3-Bottom). More specifically, the steps of translation and rotation are 1 voxel and 90°, respectively. The rotation is only about the vertical axis to ensure the components have correct upright directions. The scaling factor is set between 0.8 to 1.2 to avoid components from excessive changes.

Post-processing. Since the network outputs in our work are also represented by volumes, we need post-processing to refine the outputs and convert them to mesh models. For the network-output volumes, considering most of the man-made objects have symmetrical structures, we can leverage symmetry to improve the quality of the generated object volumes. Namely, for an object volume, we can use the mirror images of its front, back, left, or right half sides, to fix the other side that has issues. Afterward, we also intend to convert the volume-represented components of the network output to 3D meshes. Since the output object volume has component labels that are corresponding to the components of the inputs (which will be discussed in Section 4.1), for input models with 3D meshes, we can directly use the component correspondences to select component meshes of the inputs and fit them to the output one by one. For inputs only with volumes, the missing component meshes should be explored from other 3D meshes with similar shapes as the inputs. These meshes could be directly provided by users or retrieved from a dataset by using shape similarity metrics (which will be discussed in Section 5). The fitting operation adopts the same method of [42]. Specifically, we adjust the center position, direction, and scale of the component mesh in order to fit the component volume of the output.

4. Generative autoencoder

4.1. Network architecture

The pipeline of our generative model is shown in Fig. 4. We adopt a GAN-based architecture, where the generator is set as an AutoEncoder, and the discriminator is a three-layer 3D convolutional module. The encoder and decoder of the AutoEncoder focus on two tasks, i.e., component-based structure encoding and redundancy-eliminating assembly, respectively. We illustrate the details of our network architecture in Fig. 5. During the learning phase, the network is trained to create both the original structures of the inputs and their hybrid structure simultaneously. By leveraging the redundancy and randomness of the component volumes and the known correspondence between each component volume and a 3D shape, the network can learn the implicit assembly structure of a certain object through supervised training. During the generation phase, we directly use the generated hybrid structure as the generated model. Note that when given multi-category 3D model inputs, the trained network could create a hybrid structure, while single-category 3D model inputs (one of which is incomplete with missing parts) could lead the network to complete one input shape by using the components of the other one.

The encoder consists of a set of 3D convolutions to learn the assembly structure, which is called structure encoding. This is achieved by the known correspondence between man-made objects and the component volumes. Each layer in the encoder has a 3D convolution module with down-sampling. The downsampling module is used to ensure that the network learns the macroscopic structural features rather than the decorative and non-structural factors belonging to the object surface. The structure encoding can encode the structural information of all components in the component volume for usage in the next step of assembly and hybrid. Note that each component has a unique $64 \times 64 \times 64$ volume in which the voxels that belong to the component have a specific value based on their labels. The volumes of all components, assuming their quantity is N, are fed to the encoder simultaneously.

To design the decoder, we use the Squeeze-and-Excitation (SE) block [43] to calculate the weight of each feature group encoded from the component volume, and use the weight to evaluate the component redundancy. In other words, the SE block can not only restore the shape of the component, but also determine the redundancy based on the component feature. In this manner, the decoder can achieve the purpose of eliminating redundancy and reassembling the component in the meanwhile. We also use a discriminator with a three-layer convolutional structure to evaluate the quality of the network-output objects. The network output is a $64 \times 64 \times 64$ volume in which each voxel has a value range from 0 to 1. Since the voxel values of the input components have been encoded with specific values as their labels, the decoder can also ensure that the output voxels belonging to the same component could have similar values.

The discriminator is also trained with the component volumes of the input shapes. Thus, it can evaluate whether the component combination is plausible, especially for the mixed component volume which contains the components from more than one object. Since the training data of the discriminator also only includes the component volumes of the input shapes, the consistency of the generator and discriminator can be guaranteed.

4.2. Network training

The training process of the generative autoencoder is summarized with a pseudo-code in Algorithm 1. Specifically, since we use the autoencoder as the generator of our network, training the generator G involves training both the encoder and decoder of the autoencoder. In this manner, the trained generator can create the object volumes of both the two inputs and their hybrid, while the trained discriminator ensures the output object volumes have plausible structures. Without losing generality, we use two input 3D shapes to represent the cross-category shapes for structure hybrid, as well as the incomplete shape and the reference for shape completion. We perform n training loops totally, and in each iteration, component volumes C_a, C_b (associated with input shapes S_a and S_b, respectively), and the volume S_h which is mixed by the components of S_a and S_b are randomly generated by the above-mentioned approach. Besides, we also perform m loops in each iteration to better train the discriminator. The loss functions L_G and L_D, which are different in the two applications, would then be introduced in the next section.

Algorithm 1: Network Training Algorithm

Input: Two input 3D shapes S_a and S_b
Output: Network weights ω_G and ω_D
for $i = 0$ *to* n **do**
  Use S_a, and S_b to generate component volumes C_a, C_b, and C_h;
  for $j = 0$ *to* m **do**
    $S'_a = \mathbf{G}(\omega_G, C_a)$, $S'_b = \mathbf{G}(\omega_G, C_b)$, $S'_h = \mathbf{G}(\omega_G, C_h)$;
    $D_a = \mathbf{D}(\omega_D, S'_a)$, $D_b = \mathbf{D}(\omega_D, S'_b)$, $D_h = \mathbf{D}(\omega_D, S'_h)$;
    $l_d = L_D(D_a, D_b, D_h)$;
    Update ω_D according to l_d;
  $S'_a = \mathbf{G}(\omega_G, C_a)$, $S'_b = \mathbf{G}(\omega_G, C_b)$, $S'_h = \mathbf{G}(\omega_G, C_h)$;
  $l_g = L_G(S'_a, S'_b, S'_h)$;
  Update ω_G according to l_g;
return ω_G and ω_D

4.3. Task expanding

For the network output, we empirically set a threshold of 0.01 to filter out the voxels whose values are smaller than the threshold, thus obtaining the object volume. Such values can also be used to establish the correspondence between the voxel of the object volume to a certain component of the component volume. The network-output object volume thus have component labels associated with the input components. Hence, the network output can further participate structure hybrid with another object. Hence, our method is also applied to the inputs of more than two objects, i.e., we can iteratively tackle the inputs in pairs. In fact, our current network can be directly trained and used with more than two inputs. For example, when using the network to learn the structure of three input objects, the loss function needs to be changed to accommodate such an expansion. And the component volume should also include these three objects as the network input. We will discuss this in Section 6.1.

5. Applications

In this section, we give more details about how to leverage our component-aware generative autoencoder to tackle the applications of structure hybrid and shape completion.

Structure hybrid. The purpose of structure hybrid is to enable the trained generative model to explore plausible component combinations. After we convert various user inputs (e.g., images, point clouds, 3D meshes) of cross-category objects into voxels, the first task is to find the correct segmentation network for the given inputs. Even though the segmentation networks in our method are associated with certain object categories and independently trained, choosing the proper segmentation network for the input voxel can still accomplish automatically. Here we adopt an assumption that encoders of the segmentation networks can map both the input voxel and the training data of the segmentation networks to a hidden space. Based on this assumption, the distance between the input voxel and the group of the in-category training data of one segmentation network on the hidden space, can be used to determine how such a segmentation network suits the input voxel.

The second task is to train the generative model by using the user inputs. Aiming at reducing the difficulty of training, we expect the generative model to tackle component volume with a fixed component number. Given two 3D shapes S_a and S_b that have been respectively segmented into p and q components, the component amounts of the component volumes for all inputs are set to $p + q$. Namely, after randomly duplicating certain components to create component Volumes C_a and C_b, S_a and S_b would have q and p redundant components, respectively. For the component volume C_h mixed by two inputs, it consists of all components of S_a and S_b in random order, but with no redundant components.

After preparing all component volumes, we train the generative model and discriminator using the Algorithm 1 mentioned in Section 4, where the two loss functions L_G and L_D are as follows:

$$\begin{aligned} L_G =& \|S_a, S'_a\|_2^2 + \|S_b, S'_b\|_2^2 + \\ & \omega\left(\frac{1}{\|S_h, S'_a\|_2^2} + \frac{1}{\|S_h, S'_b\|_2^2}\right) + \\ & \log(\mathbf{D}(S'_a)) + \log(\mathbf{D}(S'_b)) + \log(\mathbf{D}(S'_h)), \\ L_D =& \log \mathbf{D}(S_a) + \log \mathbf{D}(S_b) + \log(1 - \mathbf{D}(S'_a)) + \\ & \log(1 - \mathbf{D}(S'_b)) + \log(1 - \mathbf{D}(S'_h)), \end{aligned} \tag{1}$$

where S'_a, S'_b, and S'_h are the network-outputs for the component volumes C_a, C_b, and C_h, respectively. The second term is for anti-over-fitting, which ensures the network output is not over-fitting either S_a or S_b. We set $\omega = 1$ in our implementation. $\mathbf{D}$ denotes the discriminator-output whose values are in the range of 0 to 1. In loss L_G, the Mean-Square Error loss ensures that the generator recovers the appearance of the object, and discriminator loss ensures that the generated object is structurally sensible.

Shape completion. The purpose of shape completion is to enable the trained generative model to fill or replace the incomplete components of the input by using components that have similar functions in the structure of reference. Since the input shape and the reference belong to the same category, the trained generative model can learn how to assemble the shape in a targeted manner and ignores the incomplete components.

Similar to the application of structure hybrid, the inputs are an incomplete shape S_a and a complete one S_b in the same category of S_a as the reference. The reference shape could be user-specified as an input or retrieved from a dataset. We collect a lightweight dataset with 91 models from 4 categories to validate the shape completion application. Like the network inputs, the dataset models also have real scales and have been well-aligned. To define the metrics for retrieval, we convert the volume as a vector. Hence the distance between two volumes can be calculated as the L1 norm of the difference between two vectors. Since the shapes in the same category have similar structures, we use the average volumes of a shape category to determine the incomplete components, which have obviously large distances based on the defined metrics. Then, we explore the reference shape that has a minimum distance to the rest components of the input. Afterward, the component volumes C_a, C_b, and C_h can be created. Note that C_a also includes the incomplete components so that the trained network could have the ability to determine whether fill or replace a certain incomplete component.

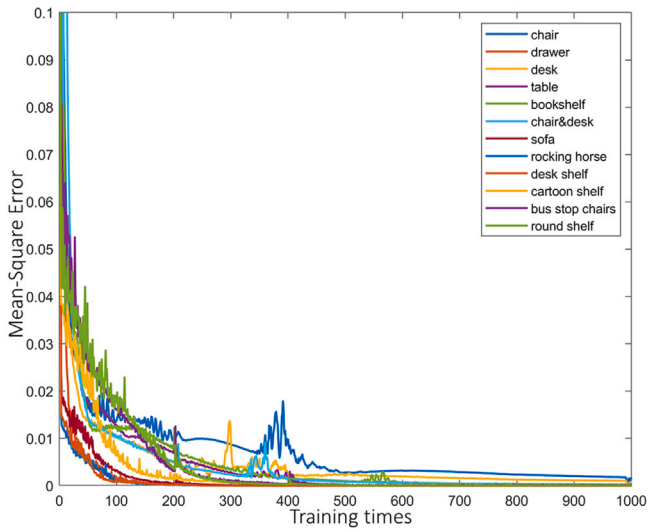

Fig. 6. The MSE curves during the network training for 12 shapes across different categories.

We use the following two loss functions to train the network for shape completion:

$$L_G = \|S_a, S_a \cap S'_a\|_2^2 + \|S_b, S'_b\|_2^2 + \log(\mathbf{D}(S'_b)) + \log(\mathbf{D}(S'_h)), \quad (2)$$
$$L_D = \log \mathbf{D}(S_b) + \log(1 - \mathbf{D}(S'_b)) + \log(1 - \mathbf{D}(S'_h)).$$

Such two loss functions are similar to the ones in Eq. (1). The differences are, in the loss L_G for the generative model, we the term $\|S_a, S_a \cap S'_a\|_2^2$ to ensure the generated shape S'_a should contain S_a but more than it. We also remove the discriminator-output $\mathbf{D}(S'_a)$ in both two loss functions, aiming at avoiding the generative model learning the incomplete shape of S_a.

6. Results and discussions

In this section, we first evaluate our method by analyzing the network performance, showing structure hybrid and shape completion results, and conducting both robustness tests and ablation experiments. Then, compared our method with the state-of-the-art methods on structure hybrid [39] and shape completion [44]. These experiments were run on a PC with Intel Core i9-10900K CPU, 64 GB RAM, and an RTX 3080 GPU.

6.1. Evaluations

Network performance. We evaluate our generative autoencoder in terms of astringency and reliability. As illustrated in Fig. 6, for 12 man-made objects across 6 different categories, we use the autoencoder to convert the component volumes to object volumes, and compare the network outputs with the corresponding ground truths by using mean-square error (MSE). The average MSE of the generated object volumes is 3.5×10^{-4}, and the standard deviation is 5.2×10^{-4}. This demonstrates that, given redundant components with random configurations, the autoencoder is able to choose the proper numbers of components and determine their positions/directions/scales to construct a meaningful 3D object.

Structure hybrid. In Fig. 7, we illustrate four groups of structure hybrid results. In each group, we show two reference objects, two network-output object volumes, and the associated 3D models. We can see that our generative autoencoder is able to learn the component relations and structure priors from the reference objects, then apply them to the input redundant components with random configurations, to generate 3D models with plausible structures.

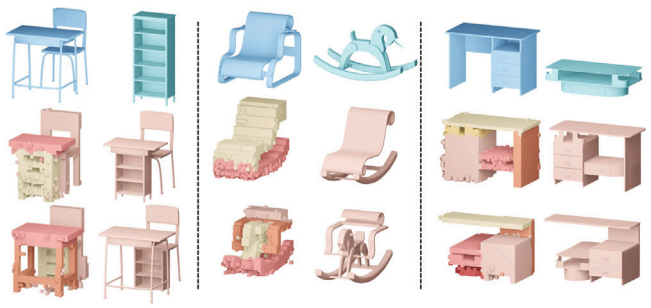

Fig. 7. Three groups of structure hybrid results. In each group, we show two input shapes and two different network-output object volumes as well as the refined 3D object.

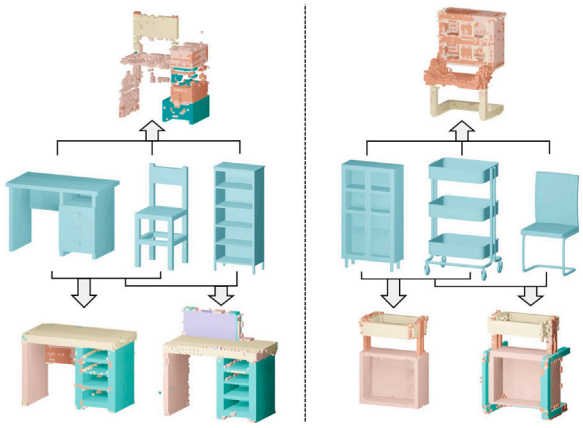

Fig. 8. Two groups of structure hybrid results with three object inputs. In each group, we show the direct hybrid results (Top) and stepwise hybrid results (Bottom).

As discussed earlier in Section 4.3, we can extend our methods to more than inputs in two ways. Namely, stepwise structure hybrid through iteratively tackling the inputs in pairs, and direct structure hybrid that trains and uses our network with more than two inputs. In Fig. 8, we also test our method with three object inputs. The bottom cases show the stepwise hybrid results while the top cases show the direct hybrid results. We can see that the latter is more convenient to generate the hybrid structure, however, the former allows the users to control the hybrid results by adjusting the order of the input object.

Shape completion. In Fig. 9, we illustrate six cases of shape completion. In each row, we show the incomplete shape which is manually created by breaking a complete shape. Hence, the complete shape can be treated as the ground truth of shape completion. We also show the reference shape, the network-output object volume, and the completed 3D shape. We can see that our network automatically picks a proper number of parts from the reference shapes and determine their positions/directions/scales, to complete the input incomplete shapes and preserves the original structures of the inputs as well. Aiming at quantitatively comparing the completed shape and the reference shape, we first convert both the reference shape and the ground truth of the incomplete shape to volumes. Then we calculate the Intersection of Union (IOU) between the volumes of the completed/reference shape and the ground truth, respectively, to represent their similarity to the ground truth. On average, the reference shapes get an IOU of 0.17, while the completed shapes created by our method get an IOU of 0.41. This demonstrates our results are better than directly using the reference shape as the shape completion of the input.

Robustness tests. To verify that our network can generate the correct input model shape for component volume inputs with different

Fig. 9. Six cases of shape completion. Note the component colors of the completed 3D shapes illustrate the sources of the components, i.e., from input incomplete shapes or references. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

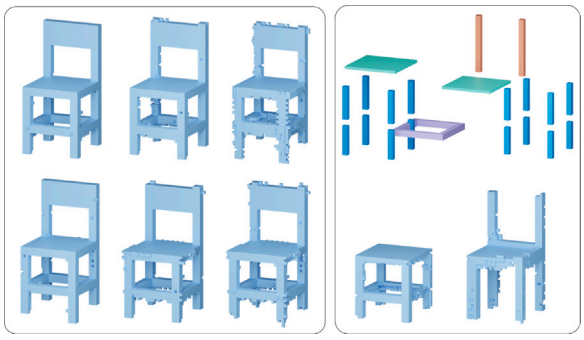

Fig. 10. Left: Two rows of network-output object volumes given components with different redundancy (from left to right), and component positions/directions (between two rows). **Right:** Two cases of the network-output object volumes (Bottom) given component volume with missing parts (Top).

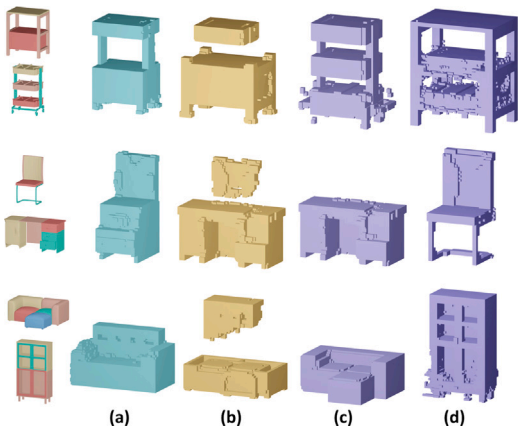

Fig. 11. Three cases of structure hybrid by using our network (a), our network without the discriminator (b), and our network without the anti-over-fitting term (c&d).

redundancy and configurations, we tested inputs with 100%, 200%, and 300% redundancy (from left to right) in Fig. 10-Left, while the inputs in the top and bottom rows have different component positions/directions. From the output results, we can see that our network can produce object volumes with similar qualities given components with different positions/directions. But the quality of the output object volume decreases slightly when increasing the redundancy. In Fig. 10-Right, we test our network in the scenarios when the input component volume loses certain parts. We find that the output object volume does not have the corresponding parts either. Therefore, it can be proved that our network can select an appropriate number of parts from the inputs and find their correct positions and directions.

Ablation experiments. To validate the proposed network architecture, we performed a series of ablation experiments. Specifically, we abandon the discriminator and anti-over-fitting term (i.e., the second term in Eq. (1)), respectively, and then compare these results with the original network outputs. As illustrated in Fig. 11, given the same object inputs, each row shows the structure hybrid results by using our network (a), our network without the discriminator (b), and our network without the anti-over-fitting term (c & d). We can see that, the discriminator is mainly to ensure the structural rationality (i.e., the connection relationship of components) of the generated shapes. On the other hand, the anti-over-fitting term helps the network be able to create new structures which are the hybrids of the inputs. Without the anti-over-fitting term, the autoencoder would create shapes similar to one of the inputs (either (c) or (b) in Fig. 10-Right) instead of their hybrid.

6.2. Comparisons

Qualitative comparison. Different from the structure hybrid methods [8,33], our method does not need the inputs to have structural similarity like the former one or rely on extra input to indicate which parts should participate in the hybrid like the latter one. We observe that the method of [39] could achieve a goal of structure hybrid very similar to ours. Hence, we compare the unconstrained first-generation shape generation results of [39] with ours. Since [39] does not release their data, we choose the 3D models with very similar appearance and structure to [39] as our inputs. In Fig. 12, we show two groups of structure hybrid results created by [39] and our method. We present two hybrid structures created by our method as a result of the randomness. Due to the structural similarities between the inputs, the two methods obtain very similar hybrid structures. However, the method [39] needs more steps of evolution to make the variations on the hybrid structures, while ours leverage the randomness of the mixed component volume to achieve the same goal. In Fig. 13, given similar two groups of inputs with significant structural differences, both [39] and our method can generate plausible hybrid structures. We can see that our network-output object volumes are already detailed to distinguish their structures. This demonstrates that our network is able to reveal potential component connections to form plausible structures, while the method [39] needs to compare the functional similarity of parts between the inputs to achieve the same goal. To validate the ability of shape completion for our method, we compare our method with AutoSDF [44] in which an autoregressive prior is proposed to solve multimodal 3D tasks including shape completion. In Fig. 14, the four

Fig. 12. Two cases of comparisons with [39] given shapes with similar structures. We show two hybrid structures created by our network due to the randomness of the mixed component volume.

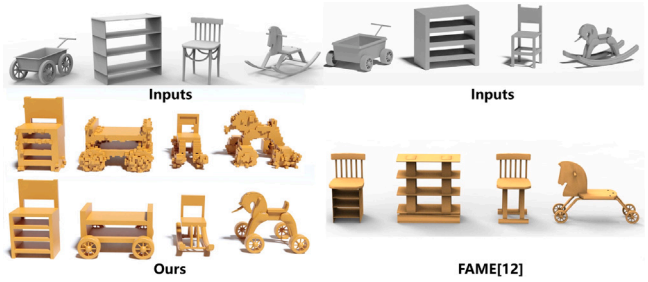

Fig. 13. Comparisons with [39] given shapes with significant structural differences.

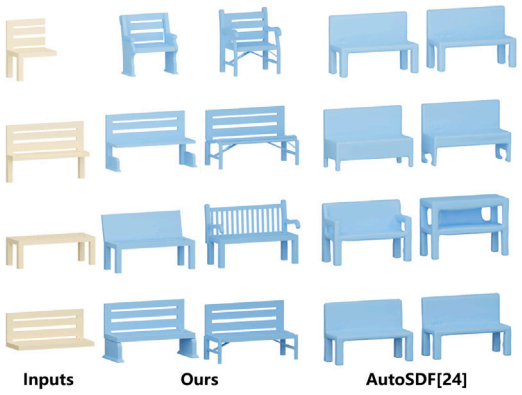

Fig. 14. Comparisons of the shape completion results between our method and AutoSDF [44], given incomplete shapes (Left). Note that each column of our results has the same reference object.

rows of different incomplete shapes belong to the same model, our method preserves the style of the inputs while using parts from the references to complete the missing parts of the input models. On the contrary, AutoSDF relies on shape priors learned from databases. Therefore, when the input incomplete shapes are ambiguous (such as that in the first row), our method chooses a completion result that is closer to the input shape and better preserves the style of the input model.

User study. We also conducted a user study, in which we invited 12 post-graduate students who majored in digital media technology, to grade both the appearance and affordance of the structure hybrid results. Specifically, each participant was instructed on the idea of structure hybrid before being asked to rate each 3D model's appearance and affordance on a scale of 0 (worst) to 5 (best), respectively. We select 5 pairs of 3D models with similar structures from Figs. 12 and 13, generated by [39] and our method. The user study results are summarized in Fig. 15. The average scores of appearance/affordance are 2.55/2.62 for the results of [39] and 2.75/2.80 for ours. This quantitative comparison results demonstrate our network performs as well as the method of [39].

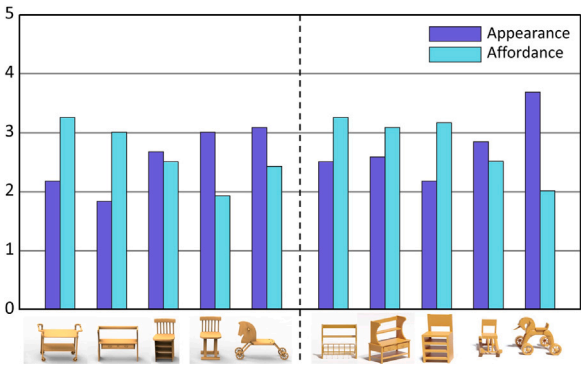

Fig. 15. Quantitative user evaluation results of the hybrid structures in terms of appearance and affordance. The results of [39] are on the left, while ours are on the right.

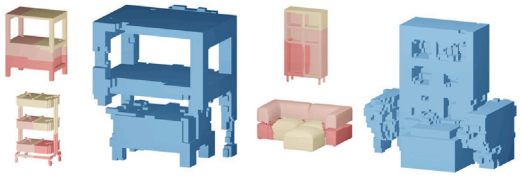

Fig. 16. Failure cases are caused by the incorrect segmentation of the inputs.

7. Conclusion

In this paper, we introduce a novel method that leverages deep networks to explore proper combinations given certain volume-based components, thus creating 3D shapes with plausible structures. The proposed networks can be trained with 3D shapes beyond a single category. Benefiting from a discriminator cooperating with the generative autoencoder, both single-or multi-category components can be mixed with randomly increasing quantities. The proper combinations of volume-based components can be explored thus creating a hybrid structure between objects from different categories, or completing the shape given volume models with missing parts. We demonstrate the usability and practicability of our method with various experiments of structure hybrid and shape completion.

Limitations. Our method, however, still has several limitations. First, our current method needs to train the autoencoder for each case of structure hybrid or shape completion. Namely, when the inputs change, the generative autoencoder needs to be trained again. It takes about 5 min to complete the training. But for the same inputs, the trained autoencoder could be reused to create different hybrid structures. Second, for the structure hybrid application, the generated model is essentially constructed by the random combination of the components of the reference objects. Even though our generative autoencoder can choose the proper numbers of the components and determine their configurations, the variations of the structure hybrid are still limited by the inputs. Therefore, a heuristic mechanism that explores more components from a dataset is helpful to improve the structural variations of the generated models. Third, for the shape completion application, our generative autoencoder mainly focuses on volume outputs. That means the quality and variation of the generated 3D model rely on the dataset models. Considering more and more large-scale man-made object datasets have been published, it is not very hard for the dataset collection. Finally, our network relies on the correct object segmentation for the inputs. As shown in Fig. 16, when using low-quality segmented objects to feed the network, the network outputs are more likely to have unreasonable structures and shapes.

In the future, we are interested in introducing a heuristic mechanism for structural variation in our networks. This could be achieved by

configuring the discriminator with the ability to analyze the physical or ergonomic relations of the components, rather than only based on the shape of the combined components, as we currently do. We think with the development of artificial intelligence, especially deep learning methods, the shape and structure of man-made objects could be better understood by the computer, and more interesting products might be designed with AI assistance or entirely designed by AI.

CRediT authorship contribution statement

Fan Zhang: Methodology, Software, Validation, Formal analysis, Data curation. **Qiang Fu:** Conceptualization, Writing – original draft, Writing – review & editing, Visualization. **Yang Liu:** Investigation, Resources. **Xueming Li:** Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by a grant from the NSFC, China (No. 61902032).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.gmod.2023.101185.

References

[1] D. Tome, C. Russell, L. Agapito, Lifting from the deep: Convolutional 3d pose estimation from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2500–2509.

[2] Y. Furukawa, J. Ponce, Accurate, dense, and robust multiview stereopsis, IEEE Trans. Pattern Anal. Mach. Intell. 32 (8) (2009) 1362–1376.

[3] J. Wu, C. Zhang, T. Xue, W.T. Freeman, J.B. Tenenbaum, Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS '16, Red Hook, NY, USA, 2016, pp. 82–90.

[4] C. Zhu, K. Xu, S. Chaudhuri, R. Yi, H. Zhang, SCORES: Shape composition with recursive substructure priors, ACM Trans. Graph. 37 (6) (2018) 211:1–211:14.

[5] N. Schor, O. Katzir, H. Zhang, D. Cohen-Or, CompoNet: Learning to generate the unseen by part synthesis and composition, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8759–8768.

[6] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, L. Guibas, Grass: Generative recursive autoencoders for shape structures, ACM Trans. Graph. 36 (4) (2017) 52:1–52:14.

[7] K. Wang, P. Guerrero, V. Kim, S. Chaudhuri, M. Sung, D. Ritchie, The shape part slot machine: Contact-based reasoning for generating 3D shapes from parts, 2021, pp. 1–19, arXiv preprint arXiv:2112.00584.

[8] Q. Fu, X. Chen, X. Su, H. Fu, Pose-inspired shape synthesis and functional hybrid, IEEE Trans. Vis. Comput. Graphics 23 (12) (2017) 2574–2585.

[9] X. Su, X. Chen, Q. Fu, H. Fu, Cross-class 3D object synthesis guided by reference examples, Comput. Graph. 54 (2016) 145–153.

[10] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by example, ACM Trans. Graph. 23 (2004) 652–663.

[11] J. Lee, T.A. Funkhouser, Sketch-based search and composition of 3D models, in: SBIM, 2008, pp. 97–104.

[12] S. Chaudhuri, V. Koltun, Data-driven suggestions for creativity support in 3D modeling, ACM Trans. Graph. 29 (6) (2010) 183:1–183:10.

[13] M. Ovsjanikov, W. Li, L. Guibas, N.J. Mitra, Exploration of continuous variability in collections of 3d shapes, ACM Trans. Graph. 30 (4) (2011) 33:1–33:10.

[14] N. Fish, M. Averkiou, O. Van Kaick, O. Sorkine-Hornung, D. Cohen-Or, N.J. Mitra, Meta-representation of shape families, ACM Trans. Graph. 33 (4) (2014) 34:1–34:11.

[15] Q. Fu, X. Chen, X. Su, J. Li, H. Fu, Structure-adaptive shape editing for man-made objects, in: Computer Graphics Forum, Vol. 35, No. 2, Wiley Online Library, 2016, pp. 27–36.

[16] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, L.J. Guibas, Structurenet: Hierarchical graph networks for 3d shape generation, 2019, arXiv preprint arXiv:1908.00575.

[17] K. Yin, Z. Chen, S. Chaudhuri, M. Fisher, V.G. Kim, H. Zhang, Coalesce: Component assembly by learning to synthesize connections, in: 2020 International Conference on 3D Vision, 3DV, IEEE, 2020, pp. 61–70.

[18] K. Wang, P. Guerrero, V.G. Kim, S. Chaudhuri, M. Sung, D. Ritchie, The shape part slot machine: Contact-based reasoning for generating 3D shapes from parts, in: European Conference on Computer Vision, Springer, 2022, pp. 610–626.

[19] J. Li, C. Niu, K. Xu, Learning part generation and assembly for structure-aware shape synthesis, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 11362–11369.

[20] S. Chaudhuri, E. Kalogerakis, L. Guibas, V. Koltun, Probabilistic reasoning for assembly-based 3D modeling, ACM Trans. Graph. 30 (4) (2011) 35:1–35:10.

[21] V. Kreavoy, D. Julius, A. Sheffer, Model composition from interchangeable components, in: 15th Pacific Conference on Computer Graphics and Applications, PG'07, IEEE, 2007, pp. 129–138.

[22] M. Sung, H. Su, V.G. Kim, S. Chaudhuri, L. Guibas, ComplementMe: Weakly-supervised component suggestions for 3D modeling, ACM Trans. Graph. 36 (6) (2017) 1–12.

[23] A. Hertz, O. Perel, R. Giryes, O. Sorkine-Hornung, D. Cohen-Or, SPAGHETTI: Editing implicit shapes through part aware generation, 2022, arXiv preprint arXiv:2201.13168.

[24] V.G. Kim, W. Li, N.J. Mitra, S. DiVerdi, T. Funkhouser, Exploring collections of 3d models using fuzzy correspondences, ACM Trans. Graph. 31 (4) (2012) 54:1–54:11.

[25] O. Van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, G. Hamarneh, Prior knowledge for part correspondence, in: Computer Graphics Forum, Vol. 30, No. 2, Wiley Online Library, 2011, pp. 553–562.

[26] Y. Wang, K. Xu, J. Li, H. Zhang, A. Shamir, L. Liu, Z. Cheng, Y. Xiong, Symmetry hierarchy of man-made objects, in: Computer Graphics Forum, Vol. 30, No. 2, Wiley Online Library, 2011, pp. 287–296.

[27] O. Van Kaick, K. Xu, H. Zhang, Y. Wang, S. Sun, A. Shamir, D. Cohen-Or, Co-hierarchical analysis of shape structures, ACM Trans. Graph. 32 (4) (2013) 69:1–69:10.

[28] Y. Zheng, D. Cohen-Or, M. Averkiou, N.J. Mitra, Recurring part arrangements in shape collections, in: Computer Graphics Forum, Vol. 33, No. 2, Wiley Online Library, 2014, pp. 115–124.

[29] V.G. Kim, W. Li, N.J. Mitra, S. Chaudhuri, S. DiVerdi, T. Funkhouser, Learning part-based templates from large collections of 3D shapes, ACM Trans. Graph. 32 (4) (2013) 70:1–70:12.

[30] A. Jain, T. Thormählen, T. Ritschel, H.-P. Seidel, Exploring shape variations by 3d-model decomposition and part-based recombination, in: Computer Graphics Forum, Vol. 31, No. 2pt3, Wiley Online Library, 2012, pp. 631–640.

[31] E. Kalogerakis, S. Chaudhuri, D. Koller, V. Koltun, A probabilistic model for component-based shape synthesis, ACM Trans. Graph. (TOG) 31 (4) (2012) 55:1–55:11.

[32] K. Xu, H. Zhang, D. Cohen-Or, B. Chen, Fit and diverse: Set evolution for inspiring 3d shape galleries, ACM Trans. Graph. 31 (4) (2012) 57:1–57:10.

[33] Y. Zheng, D. Cohen-Or, N.J. Mitra, Smart variations: Functional substructures for part compatibility, in: Computer Graphics Forum, Vol. 32, No. 2pt2, Wiley Online Library, 2013, pp. 195–204.

[34] Q. Fu, X. Chen, X. Su, H. Fu, Natural lines inspired 3D shape re-design, Graph. Models 85 (2016) 1–10.

[35] M. Bokeloh, M. Wand, H.-P. Seidel, A connection between partial symmetry and inverse procedural modeling, ACM Trans. Graph. 29 (4) (2010) 104:1–104:10.

[36] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, H. Zhang, Topology-varying 3D shape creation via structural blending, ACM Trans. Graph. 33 (4) (2014) 158:1–158:10.

[37] K. Xu, H. Li, H. Zhang, D. Cohen-Or, Y. Xiong, Z.-Q. Cheng, Style-content separation by anisotropic part scales, ACM Trans. Graph. 29 (6) (2010) 184:1–184:10.

[38] A. Jin, Q. Fu, Z. Deng, Contour-based 3D modeling through joint embedding of shapes and contours, in: Symposium on Interactive 3D Graphics and Games, in: I3D '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 9:1–9:10.

[39] Y. Guan, H. Liu, K. Liu, K. Yin, R. Hu, O. van Kaick, Y. Zhang, E. Yumer, N. Carr, R. Mech, H. Zhang, FAME: 3D shape generation via functionality-aware model evolution, IEEE Trans. Vis. Comput. Graphics 28 (4) (2022) 1758–1772.

[40] H. Xie, H. Yao, X. Sun, S. Zhou, S. Zhang, Pix2Vox: Context-aware 3D reconstruction from single and multi-view images, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV, IEEE Computer Society, 2019, pp. 2690–2698.

[41] A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., Shapenet: An information-rich 3d model repository, 2015, arXiv preprint arXiv:1512.03012.

[42] C.-H. Shen, H. Fu, K. Chen, S.-M. Hu, Structure recovery by part assembly, ACM Trans. Graph. 31 (6) (2012) 180:1–180:11.

[43] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.

[44] P. Mittal, Y.-C. Cheng, M. Singh, S. Tulsiani, AutoSDF: Shape priors for 3D completion, reconstruction and generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 306–315.