Computers & Graphics (2018)

Contents lists available at ScienceDirect

Computers & Graphics

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

3D Synthesis of Man-made Objects based on Fine-grained Parts

Diego Gonzalez[a], Oliver van Kaick[a,*]

[a]*School of Computer Science, Carleton University, Ottawa, ON, Canada*

ARTICLE INFO

Article history:
Accepted 15 May 2018
Available online To appear

Keywords: Shape synthesis, Shape segmentation, Shape analysis

ABSTRACT

We present a novel approach for 3D shape synthesis from a collection of existing models. The main idea of our approach is to synthesize shapes by recombining fine-grained parts extracted from the existing models based purely on the objects' geometry. Thus, unlike most previous works, a key advantage of our method is that it does not require a semantic segmentation, nor part correspondences between the shapes of the input set. Our method uses a template shape to guide the synthesis. After extracting a set of fine-grained segments from the input dataset, we compute the similarity among the segments in the collection and segments of the template using shape descriptors. Next, we use the similarity estimates to select, from the set of fine-grained segments, compatible replacements for each part of the template. By sampling different segments for each part of the template, and by using different templates, our method can synthesize many distinct shapes that have a variety of local fine details. Additionally, we maintain the plausibility of the objects by preserving the general structure of the template. We show with several experiments performed on different datasets that our algorithm can be used for synthesizing a wide variety of man-made objects.

1. Introduction

There is an ongoing need for digital content in the form of 3D models in fields such as entertainment and product design. However, the manual creation of 3D models is often a difficult and time-consuming task. Hence, facilitating the creation of 3D models is a fundamental problem in computer graphics.

A popular approach for the automatic creation of novel 3D objects is to reuse parts from existing shapes. Several works on shape synthesis propose to extract and recombine parts from a dataset of existing objects [1–6]. Most of these techniques make two important assumptions about the input models and their parts. First, the input objects should be segmented into coarse *semantic* parts, that is, parts with a meaningful semantic meaning, and possibly a specific functionality. Second, a *correspondence* or consistent *labeling* should exist among the different parts of the input shapes.

Fig. 1. Scene created with shapes synthesized by our method. Note how the three shapes exhibit a variety of fine details and a coherent structure.

One shortcoming of these synthesis methods is that the generated shapes may display a limited range of variations in their fine geometrical details, since semantic parts are often large, coarse and exchanged as a whole [7, 8]. Furthermore, although much progress has been made to obtain semantic segmentations

*Corresponding author:
e-mail: oliver.vankaick@carleton.ca (Oliver van Kaick)

of 3D shapes [9, 10], computing highly accurate and consistent segmentations and part correspondences for large collections of shapes is still a challenging problem [11, 12].

In our work, we propose to synthesize shapes by exchanging *fine-grained* parts among a set of input shapes. As shown in Figure 1, the use of fine-grained parts allows us to generate shapes with large variation in fine geometric details, which is not achievable with methods that exchange semantic parts as a whole. Figure 2 shows a comparison between a typical semantic segmentation of a table, and the fine-grained segments that we use for synthesis. The recombination of fine-grained segments is guided by the geometric similarity of the segments, enabling us to exchange segments that have the same overall geometry but which can possess variations in their fine details. The recombination guided by geometry allows us to even exchange segments between models from different families.

Moreover, we do not require a semantic segmentation nor a part correspondence as input, since the fine-grained parts can be obtained by analyzing the local geometry of the shapes, as done by traditional geometry-based segmentation methods [9]. Our main requirement on the segmentation is that the shape segments should possess roughly the same size. However, one problem arising from the elimination of part semantics is that we require an alternative mechanism to guide and constrain the synthesis of a new shape. Thus, we propose to use for guidance an input template, which can be a simple configuration of geometric proxies or an example shape. The template constrains the topology of the synthesized shape, which helps to ensure a certain level of plausibility in the generated shapes, and provides a domain for formulating the synthesis of a shape as a graph assignment problem, as we discuss in Sections 3 and 5.

In summary, we present a pipeline for synthesizing 3D shapes using fine-grained segments extracted from an input set of shapes. Our method does not require a segmentation of the input models into semantic parts, nor a part correspondence, but mainly a guiding template and a segmentation of a collection where the segments are consistent in size, which is easier to obtain with automatic or semi-automatic methods. Specifically, for the results shown in this paper, we employ a semi-automatic segmentation method. Moreover, we demonstrate the effectiveness of our approach by presenting and analyzing a variety of results synthesized with our method. In addition, we explore variations of our pipeline that enable us to control different aspects of the shape generation.

Fig. 2. Comparison of two segmentations for the table model shown in the inset. Left: typical semantic segmentation. Right: segmentation into fine-grained segments used in our work.

2. Related work

In this section, we discuss the previous work most related to our method, i.e., shape synthesis and segmentation approaches.

Shape synthesis. Earlier approaches for synthesis employed statistical shape models to generate shapes, according to the variability learned from a collection. For example, Blanz and Vetter [13] developed a deformable model of 3D faces, while Allen et al. [14] introduced a statistical model of human bodies. These methods are quite general as they are applicable to any type of shape (biological, man-made), but require a compatible mesh with correspondences across the entire collection.

The seminal work by Funkhouser et al. [1] proposed a system that allows the user to browse a library of 3D models and compose new shapes by assembling together parts of the existing 3D objects. This *part reuse framework* is widely applicable to objects that can be decomposed into parts in a meaningful manner, such as man-made objects. Thus, much of the subsequent work on shape synthesis has been based on this idea.

One line of work has proposed interfaces that facilitate the extraction and reuse of shape parts. For example, Kraevoy et al. [15] use compatible segmentations of objects to enable part exchange, while Sharf et al. [16] and Takayama et al. [17] introduce interfaces that facilitate the selection of part or regions to be exchanged between shapes. Chaudhuri et al. [18] incorporate semantics into this framework with a learning approach, to suggest suitable part replacements while building a shape from existing parts. Recently, Jaiswal et al. [19] and Sung et al. [20] also developed suggestion mechanisms based on learning approaches which however do not require labeled parts.

Another line of work proposes methods that *automatically* synthesize shapes while requiring little to no user input. A few works are based on the idea of *blending* existing shapes together as a whole, such as the methods of Jain et al. [3] and Alhashim et al. [7]. Blending is advantageous in that it can generate a variety of objects from two input shapes, however, these methods require a semantic segmentation of the input models.

Moreover, much of the recent work on automatic shape synthesis focuses on learning statistics of part co-occurrence based on a semantic segmentation of the objects in a collection, and using this information to automatically generate objects. Kalogerakis et al. [4] and Huang et al. [2] introduce approaches that select parts to compose a shape based on the co-occurrence of semantic labels. Zheng et al. [21] exchange specific arrangements of parts to synthesize objects that satisfy certain functionalities, while Huang et al. [22] extend this approach to non-symmetric arrangements. Su et al. [8] exchange more complex substructures among shapes. Moreover, Xu et al. [6] use an evolution-based approach to synthesize shapes.

A few methods also focus on the specific problem of generating valid configurations of shape parts, i.e., defining the relative positioning and orientation of the parts that compose the shapes. Fish et al. [23] and Yumer and Kara [24] learn models from a collection of shapes that capture the probability of part configurations. Averkiou et al. [25] represent the shapes of a dataset as box-like templates which can be used for exploration of the set but also for synthesizing new objects via part deformation.

The main requirement of all of these blending and automatic synthesis methods is a semantic segmentation of the objects in the analyzed collections. Often, a semantic labeling of the segments is also required. Although there has been significant work in recent years for obtaining such segmentations automatically [10], obtaining and accurate segmentation of a large collection is still a challenging problem with much room for improvement to the segmentation accuracy [12]. In contrast, the objective of our method is to synthesize shapes by exchanging parts obtained with a less demanding type of segmentation, which can be computed mainly from geometric constraints without involving semantics. Moreover, the use of fine-grained segments in our work enables to exchange fine details between objects and increase the variability in the synthesized shapes, which is not possible when using semantic segments, as these are often larger and more coarse in relation to the entire shape.

Finally, a few methods also aim to extract parts that can be suitable for exchange, in a sense, treating the segmentation and part exchange problems together. Bokeloh et al. [26], and later Kalojanov et al. [27], discover a shape grammar that can be used to generate the input shape and, subsequently, novel objects. Liu et al. [5] combine the discovery of shape grammars with the analysis of substructures for shape synthesis. Although the parts extracted by these methods are not semantic in nature, the discovered parts tend to be large as the goal of these methods is to discover the smallest set of elementary parts. Thus, the exchange of finer details is also difficult to achieve.

Shape segmentation. There has been a significant amount of work in shape segmentation, where the goal is to decompose an input object into meaningful parts. Much of earlier work employed geometric criteria combined with clustering or region growing heuristics to partition a single input object into parts, as surveyed by Shamir [9] and more recently by Theologou et al. [10]. More contemporary approaches make use of learning to incorporate prior knowledge about the desired segmentation, typically given in the form of example segmented shapes. Notable works in this area include the learning segmentation method of Kalogerakis et al. [28] and recent works that perform the learning with deep networks [11], also targeting a hierarchical segmentation [12]. Although most of the literature has focused on the extraction of semantic segments, the existing approaches can also be leveraged to decompose shapes into segments with application-specific requirements [9], as we discuss in Section 4 for the segmentation required by our method.

3. Overview

Our approach is summarized in the overview shown in Figure 3. The input to our synthesis pipeline is a set of triangular meshes. We start by obtaining a point cloud for each mesh by sampling points from the triangles of the mesh. Similarly to recent shape analysis methods [29–31], we opt to work with point cloud representations to alleviate problems in our analysis that can be caused by the presence of non-uniformly sampled geometry and *non-manifold* shape structures. Next, we segment each point cloud into fine-grained segments with a semi-automatic method, generating a pool of parts that can be used for synthesis. Then, we compute a set of descriptors to represent each segment, and we use these descriptors to define a similarity metric for the comparison of segments.

To synthesize a shape, we take as input a geometric template representing the general structure and topology of the target shape. In practice, the template is a shape segmented in the same manner as the collection. We define a graph based on the part structure of the template, and pose the synthesis as a probabilistic sampling of segments. Our goal is to assign segments from the pool of parts to the nodes of the graph, in order to maximize a *shape energy*. This energy takes into account the similarity between the template and sampled segments, in the form of a unary term, as well as the consistency between neighboring segments, captured by a pairwise term.

Finally, after sampling the segments that constitute the shape, we perform a series of geometric operations to align the segments and ensure that we obtain a plausible shape as output. At the end of this process, we obtain a shape that respects the topology of the template and possesses a consistent structure, while containing local geometric variations. Since our method can consider a large set of fine-grained segments, we can generate different variations for the same template by sampling multiple shapes according to the shape energy.

Note that the set of descriptors used for sampling the segments capture the overall geometric properties of the segments, but they are relatively insensitive to differences in the fine details of the parts. Hence, the segments sampled by our method preserve the general form of the template while leading to variations in the geometry of the synthesized shape.

4. Extraction of fine-grained segments

In this section, we explain how we pre-process and segment the input collections of shapes used for synthesis, while we describe our synthesis approach in detail in Section 5.

Pre-processing. The input to our method is a set of triangular meshes, where we orient all the meshes consistently. After the alignment, the +Y axis corresponds to the upward direction of the objects, and the +X axis to the frontal direction. We perform the alignment manually, although it would be possible to incorporate automatic alignment methods [32] into our pipeline. In addition, we normalize our shapes in scale, so that the axis-aligned bounding box of each shape is a cube centered at the origin where all point dimensions are in the range [-1, 1].

Point sampling. We transform each input mesh into a point cloud, using the sampling algorithm described by Osada et al. [33], which samples points from the triangles of the mesh without biasing the sampling by the shape of the triangles. For all the experiments that we discuss in this work, we sampled $N = 15,000$ points from each mesh. During the sampling, we also compute the normal of each face, and associate this normal with each point sampled from the given face. We use the normals associated to each point to compute some of our descriptors, as we describe in Section 5.

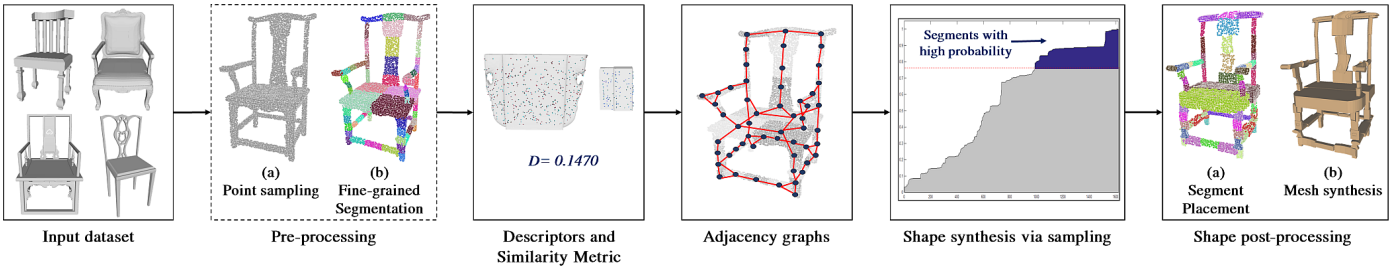

Fig. 3. Overview of our 3D shape synthesis approach.

4.1. Fine-grained segmentation

From each point cloud computed from the input meshes, we perform a segmentation that generates a set Ω of fine-grained segments that we use in our synthesis process. Our fine-grained segmentation has two main goals: (i) to provide segments that capture fine geometrical details of the shapes, such as ornaments and salient features, which can then be transferred to the synthesized shapes; and (ii) to provide segments that are consistent in size across all the shapes of the input set, so that they can be easily reused for the generation of novel shapes.

Although the segmentation of 3D shapes is a fundamental component for applications in diverse areas such as geometric modeling and shape analysis, obtaining a meaningful segmentation of a 3D shape is still a challenging task [10, 11], especially when a consistent segmentation of different 3D objects of the same class is required. Moreover, most works about shape segmentation presented in the literature are tailored for computing semantic segments for 3D meshes, while our method employs fine-grained segments extracted from a point cloud.

Thus, to obtain a segmentation that satisfies our requirements, we currently segment the input shapes with a semi-automatic method, based on an interactive program for segmentation of point clouds. Figure 4 shows the interface of our program and an example of a fine-grained segmentation obtained with the program. Our program provides a set of high-level tools that the user can employ to easily extract fine-grained segments. The user can select a set of points from a point cloud using a simple selection box drawn directly on the screen, and assign all the points that project onto the box to a segment.

In addition, our segmentation interface provides an option for merging two or more segments, as well as an option for splitting an existing segment into smaller segments of a given size. Finally, our system can take as input point clouds that have already been pre-segmented into coarse semantic parts with an external tool. In our study, we pre-segmented a portion of the meshes with the point cloud segmentation method of van Kaick et al. [34], which partitions models into approximately convex segments. By combining all of these tools together, the user can segment the shapes into segments of consistent size.

In this paper, we focus on how to use the set Ω of fine-grained parts to synthesize a variety of man-made shapes. Thus, we leave the study of computing consistent fine-grained segmentations in a fully automatic manner for future work. Nonetheless, we note that a segmentation into fine-grained segments poses less demands on the development of the segmentation method,

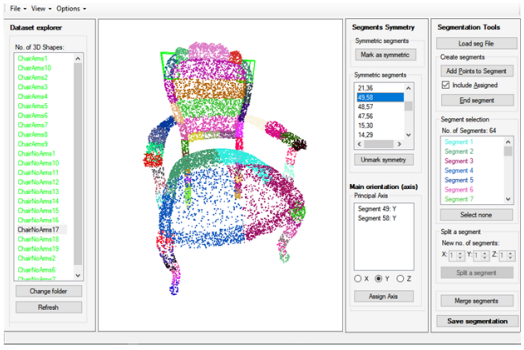

Fig. 4. Our interactive tool for point cloud segmentation and an example of a fine-grained segmentation obtained with the tool.

as the generated parts do not need to satisfy semantic requirements, but only match an expected type of geometric boundary and size. Thus, the development of such a segmentation method is expected to be an easier task in comparison to a fully semantic segmentation. We discuss alternatives for the automatic extraction of our fine-grained segments in Section 7.

Furthermore, our program also allows the user to mark pairs of segments having reflectional symmetry [35]. These annotations let us determine the symmetry relations of our input shapes, and generate novel shapes where these relations are preserved. Similarly, the user can employ our segmentation system to create annotations about the principal axis of orientation of each fine-grained part. We employ the annotations about the principal axis of each part to compute a rotation between each segment of the template and the corresponding segment selected to generate a new 3D shape in a consistent manner, as we will describe in the following section.

5. Synthesis based on fine-grained parts

In this section, we explain the details of how we employ the set Ω of fine-grained segments to generate man-made shapes.

5.1. Shape descriptors and similarity metric

We use a set of shape descriptors to estimate the similarity between two fine-grained segments. The similarity is used to guide the shape synthesis, as we describe in Section 5.3. Our descriptors encode several characteristics of the overall geometry of each segment, while at the same time being insensitive to the fine details of the parts. We classify our descriptors into two types: point-level descriptors and segment-level descriptors.

Point-level descriptors. We capture the distribution of the values of each point-level descriptor using histograms. We employ four point-level descriptors: (i) The Point Feature Histogram (PFH) [36], which captures the overall variation in the orientation of the segment's surface by using the normals that we associated to each sampled point; (ii) The mass distribution of the segment, given by the distribution of the points in 3D space [37]; (iii) The volume of the segment obtained using the Shape Diameter Function (SDF) [38], where we adopt the method of van Kaick et al. [34] to compute the SDF for a point cloud; (iv) The *surface variation*, which captures the curvature of points in the segment in a robust manner [39].

Segment-level descriptors. We make use of three segment-level descriptors: (i) The overall geometry (G), which allows us to estimate how linear, spherical, or planar is the shape of each segment [30, 40]; (ii) The relative position (P) of the segment with respect to its containing shape [37]; and (iii) The minimum and maximum coordinate along each dimension for all the points in a segment, which captures the axis-aligned bounding box of the segment relative to its source shape (B).

5.1.1. Similarity measure between segments

We compute the similarity (more precisely, a distance) between any two segments using the set of descriptors. Given two segments i and j, we define their distance D as:

$$\begin{aligned} D(i,j) &= \sqrt{D_p(i,j) + D_s(i,j)}, \quad \text{where:} \\ D_p(i,j) &= \sum_{d=1}^{4} \mathrm{EMD}^2(h_i^d, h_j^d), \quad \text{and} \\ D_s(i,j) &= \|\mathrm{G}_i - \mathrm{G}_j\|^2 + \|\mathrm{P}_i - \mathrm{P}_j\|^2 + \|\mathrm{B}_i - \mathrm{B}_j\|^2, \end{aligned} \tag{1}$$

where $D_p(i,j)$ denotes the distance measure for the point-level descriptors, h_i^d denotes the histogram for the segment i and point-level descriptor d, and EMD is the Earth-Mover's Distance, which allows to measure the similarity between two probability distributions given in the form of histograms [40]. Similarly, $D_s(i,j)$ is the distance for the segment-level descriptors, denoted according to the acronyms defined above.

Our similarity metric allows us to find segments that are similar in their overall geometry, while preventing fine detail from influencing the similarity too much, which enables us to synthesize shapes with variations in their details.

5.2. Adjacency graphs of the shapes

For each shape S in our input set, we create a graph G_S that allows us to capture the neighborhood information of each segment in S. We also use this information during the synthesis. The graph has a node for each segment of the shape, and an edge between each pair of segments that are neighbors in the shape. To determine if a pair of segments i and j of S are neighbors, we adapt the method employed by Jaiswal et al. [19]. Specifically, we say that the segments i and j are neighbors if we can find a pair of points $p \in i$ and $q \in j$ such that: $\|p - q\| \leq 0.035\sqrt[3]{V}$, where V is the volume of the bounding box of S. Figure 3 presents an example of the adjacency graph of one of the shapes of our input set of chairs.

5.3. Shape synthesis via sampling

Template selection. To obtain synthesized objects with a plausible structure, we select one shape from our input set as a template. This template $\mathcal{T}$ guides the synthesis process, determining the global structure and topology of the generated shape. Note that, as part of the procedure explained in Section 5.2, we have also computed the adjacency graph for the template $\mathcal{T}$, which we will denote as $G_{\mathcal{T}}$.

For synthesizing a new 3D shape, we select a specific subset of segments from the set Ω to compose the shape, according to the similarity metric and the selected template.

5.3.1. Shape energy

To synthesize a 3D shape with variations in its local geometry while maintaining a plausible structure, we sample segments from our pool of parts Ω that can replace each segment i of the template $\mathcal{T}$ while forming an adequate shape. We formulate this goal as the maximization of a shape energy of the form:

$$E(m) = \sum_{i \in T} E(i, m), \quad \text{with:} \tag{2}$$

$$E(i,m) = P_i(m(i)) + P_i^{PW}(m), \tag{3}$$

where T is the set of segments that form the template $\mathcal{T}$, m is a mapping that encodes the assignment of segments in Ω to the segments of T, and the two terms of Eq. (3), P_i and P_i^{PW} are defined below.

The energy defined in Eq. (2) estimates whether the segments selected to compose a new shape contribute to create a plausible object. The estimation involves the two terms of Eq. (3). The first term, P_i, is a unary energy that measures the overall similarity between segments i of the template and their replacements $m(i)$. The second term of Eq. (3) is a pairwise energy that measures the similarity between the context of each segment in the synthesized shape compared to the context of the original segment.

5.3.2. Sampling of segments for shape synthesis

To synthesize a plausible shape, we construct the mapping m so that the energy $E(m)$ given by Eq. (3) is maximized. Since directly optimizing Eq. (3) is only possible for certain convex energies [41], we employ a heuristic sampling approach instead. To perform the sampling of the segments that will compose a shape, we transform the distances between the segment $i \in T$ and all the segments in the set Ω into a discretized Probability Density Function (PDF) by means of the following expression:

$$P_i(r) = \frac{\exp(-D(i,r))}{\sum_{j \in \Omega} \exp(-D(i,j))}, \tag{4}$$

where $P_i(r)$ is the probability of segment $r \in \Omega$ being a suitable replacement for the segment $i \in T$, according to the similarity given by Eq. (1). In our heuristic algorithm, we randomly sample a segment with a probability higher than a threshold τ, to obtain a segment with high similarity in comparison to the segment $i \in T$. Specifically, for the experiments that we performed for this paper, we sampled segments with a probability higher than $\tau = 0.75$. In addition, we also consider the following filtering conditions to compute the final mapping m:

- We compute the ratios between the length of each pair of sides of the bounding box of each segment, and we find the difference between the maximum of these ratios for every pair of segments in the set Ω. Then, we discard as possible replacement for the segment $i \in T$ any segment for which such difference is larger than a threshold $\eta = 1.8$, which was determined experimentally.

- Additionally, we filter out segments based on an adjacency score defined for a segment $i \in T$ as:

$$H_i = \sum_{j \in N_i} D(i, j), \tag{5}$$

where D is the distance defined in Eq. 1, and N_i is the set of segments that are neighbors of $i \in T$. We normalize the score H_i by the mean and standard deviation of the entire shape $\mathcal{T}$, yielding a normalized score $\hat{H}_i$. Similarly, we compute the score $\hat{H}_r$ for the segment r sampled from the PDF, and we calculate the difference $\hat{h}$ between the score of $i \in T$ and the score of r, i.e.: $\hat{h} = |\hat{H}_i - \hat{H}_r|$. We then filter out segments where $\hat{h}$ is larger than a threshold. In practice, we use a threshold of 2, which allows us to discard segments with very dissimilar contexts.

In general, we also filter out segments coming from the template shape, to obtain synthesized shapes with more variety in their fine details in comparison to the original template. Nonetheless, our method can also preserve selected segments of the template during synthesis, as we show in Section 6.4.

Furthermore, to obtain a shape that preserves the main symmetry relations of the template, we use the list of symmetric pairs created during the segmentation, which determines if a segment $i \in T$ has a symmetric segment $j \in T$. For such symmetric pairs, we compare the probability computed by Eq. (3) for each corresponding replacement segment $r = m(i)$ and $s = m(j)$, and we keep in our synthesized shape the segment having the highest probability. During our shape post-processing step, we reflect this segment through the corresponding symmetry plane of the shape [35].

Next, we employ the probability associated with the segment r sampled from the unary PDF as the value of the first term of the energy function of Eq. (3). To compute the second term of Eq. (3), we compute a pairwise PDF given by:

$$P_i^{PW}(m) = \exp\left(- \sum_{j \in N_i} |D(i, j) - D(m(i), m(j)| \right), \tag{6}$$

where, as before, D is the distance defined in Eq. 1, and N_i is the set of segments that are adjacent to i according to the adjacency graph of the template. After normalizing Eq. (6) by the total number of segments of Ω, we compute the cumulative sum for the values of this pairwise distribution, and we use this cumulative sum to extract the probability that represents the value of the second term of Eq. (3).

By sampling different segments from the unary PDF computed for each segment i, we can generate several new shapes from the same template, as we will see in Section 6.2. Finally, after sampling each segment that will compose a new shape, we compute the shape energy according to Eq. (2).

5.4. Shape post-processing

Given the segments selected to synthesize a new shape, we finalize the creation of a 3D object with a two-stage procedure that we describe next.

5.4.1. Segment placement for point cloud synthesis

To generate a point cloud from the mapping m, we compute, for each replacement segment, a transformation composed of a rotation, a non-uniform scaling, and a translation, that allow us to find the optimal placement of the segment in the synthesized shape, with respect to the corresponding size, scale and orientation of the original segment of the template.

First, given a replacement segment r, we compute a rotation so that the orientation of r matches the orientation of the segment i of T. To obtain this rotation, we extract the oriented bounding box (OBB) of the segments r and i using the method of Fish et al. [23]. To obtain the principal axis of orientation of each fine-grained segment, we initially used the principal components given by PCA, as proposed in previous works [37]. However, we found that this approach did not provide consistent results in many cases. Thus, we resort to the user-given annotations on the principal axis of each segment, which are specified during the segmentation process.

Afterwards, we compute a non-uniform scaling to make the proportions of the bounding box of the segment r match as best as possible the size of the bounding box of i. In addition, we obtain a translation from the position of the replacement segment r towards the position of i by calculating the difference between the centroid of the segment i and the centroid of r.

It is important to note that the method that defines the mapping m selects the same replacement r for symmetric segments of the template. Therefore, in practice, we only compute the transformations described above for one of the symmetric segments of the template T, and we reflect the transformed segment through the main symmetry plane of $\mathcal{T}$, generating, thus, a new shape that respects the symmetry properties of the template.

Moreover, due to the differences in the geometrical features between the parts of the template and the parts of the synthesized shape, after applying the alignment described above, the synthesized shape can have disconnections and misalignments between its adjacent segments. Thus, similarly to previous works [4, 21], we refine the placement of the parts of the new shape by aligning a set of contact points on the segments.

More precisely, we obtain a set of four matching contacts for each segment boundary that will be aligned. Suppose first that we would like to align two adjacent segments r and s. To acquire the set of contact points, we use the OBBs for the segments r and s, and we find the face on the OBB of r that is closest to the points of the segment s. We denote such face by f_r. Likewise, we find the face on the OBB of s that is closest to the segment r, denoted f_s. Then, we compute the area of the faces f_r and f_s. We assign as the first set of contacts the corners of the face with the smallest area. Let us suppose, w.l.o.g., that f_r is the face with the smallest area. Then, we assign as the four contact points for the segment r the corners of the face f_r. To find the contact points for the other segment (s in this case), we project the contact points of the segment r towards the plane

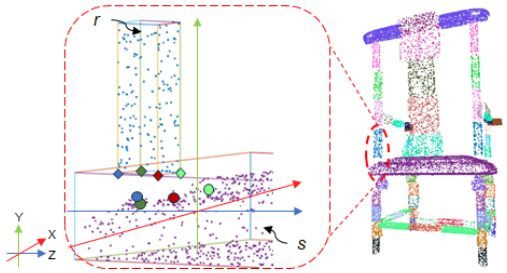

Fig. 5. Synthesized point cloud before the alignment between adjacent components, showing two neighboring segments and their contact points.

given by the face f_s and we assign as contacts for the segment s the four points of the segment that are closest to each point that was projected on the face f_s. Figure 5 shows an example of the synthesized point cloud of a chair before the alignment, where we zoom in two adjacent segments, denoted as r and s, and we show the contact points of each segment, illustrating each pair of matching contact points with corresponding matching colors.

Using the set of four pairs of contacts, we compute the alignment between adjacent segments in a similar manner to previous works [4, 5, 21]. The local alignment between neighbor segments is composed of a translation and a non-uniform scaling that optimize the alignment between the contact points in a least-squares sense [21]. Specifically, we find an optimal translation computing the difference between the centroid of the contact points of r, and the centroid of the contact points of its neighbor segment s. To compute the scaling for the segment r, we calculate the average of the scalings required to align each contact point of r to its matching contact points on s.

Furthermore, to perform the alignment for all pairs of segments in the synthesized shape, we traverse the graph of the template $G_{\mathcal{T}}$ by means of a breadth-first search algorithm [42], aligning neighboring segments during the traversal. This procedure allows us to align the segments of the shape in an incremental manner, avoiding the need to realign segments that were already aligned in earlier stages of the process.

Specifically, we select one the segments of T that has the largest number of neighbors in the template to start the traversal of the graph $G_{\mathcal{T}}$. Let r be the segment on the synthesized shape corresponding to this initial segment of T. We align first each neighbor of r based on their respective contact points. Next, we continue the traversal with one of the neighbors of r, say s, and we compute the alignment between s and each one of its respective neighbors in the new shape, before moving to the next neighbor of r. Finally, we store the full set of transformations computed for each segment. We use these transformations for mesh synthesis, as we describe next.

5.4.2. Mesh synthesis

To create a shape with a well-defined surface, we directly transfer parts of the input 3D meshes to compose the new mesh. However, since the fine-grained segments are extracted from the sampled point clouds, there are many cases where the boundaries of the segments do not match the edges of the triangles on the meshes. Hence, to alleviate this problem, we first compute a subdivision of the input meshes.

Mesh subdivision. We compute the subdivision of each input mesh using the Loop scheme, which iteratively splits all the faces having edges longer than a threshold into four triangles. We employ a threshold $\xi = 5\%$ of the length of the longest edge of the input mesh $\mathcal{M}$. The result is a subdivided mesh $\hat{\mathcal{M}}$ where all the faces have a length smaller or equal than the threshold ξ. Note that, differently from the original Loop scheme, we do not interpolate newly-created vertices to smooth the meshes, since our goal is mainly to increase the resolution of the meshes.

Extraction of mesh segments. Next, we extract from the subdivided meshes a set of mesh segments that approximately correspond to each fine-grained segment in our set Ω. However, the mesh segments obtained directly from the subdivided meshes can contain faces with edges that lie slightly outside or slightly inside the boundaries of the bounding box of the segment, yielding mesh segments with jagged boundaries. To correct this problem, we find the vertices closest to the boundaries of each mesh segment that do not match exactly with the boundary of the bounding box of the corresponding fine-grained segment. Then, we project these vertices towards the plane given by the corresponding closest face of the bounding box of the segment.

Placement of mesh segments. In this step, we first transform the vertices of each mesh segment employing the transformation that was stored during the process of point cloud synthesis. Next, we optimize the alignment between the adjacent mesh segments on the synthesized shape by means of a local alignment analogous to the process that we employ during the synthesis of point clouds. Hence, we find a set of vertices on each adjacent mesh segment that we use as contact points to compute the alignment. To find these contacts, we employ the same procedure that we described for the case of the adjacent segments of a point cloud. Additionally, we also compute the alignment of the adjacent mesh segments of the synthesized mesh by traversing the nodes of the graph $G_{\mathcal{T}}$ in a breadth-first manner, as we did for the generation of a new point cloud.

6. Experimental results

In this section, we present the results of diverse experiments performed to validate our synthesis method.

Input datasets. In our work, we focus on man-made objects, which often exhibit rich local geometrical details and multiple symmetric parts, which can be captured well by fine-grained segments. Specifically, we applied our method on three datasets: a family of 30 chairs, a family of 15 tables, and a hybrid set that combines all the table models with a subset of 15 chair models. We collected our input shapes from three different sources: the dataset of furniture models of diverse styles of Hu et al. [31], and two public online repositories, *ShapeNet* [43] and *3D Warehouse* [44].

6.1. Shapes synthesized from different templates

Figure 6 shows a set of shapes created with our method using the input set of tables, while Figure 7 shows several shapes synthesized using the set of chairs. Each shape displayed in Figures 6 and 7 was created using a different input template. Note

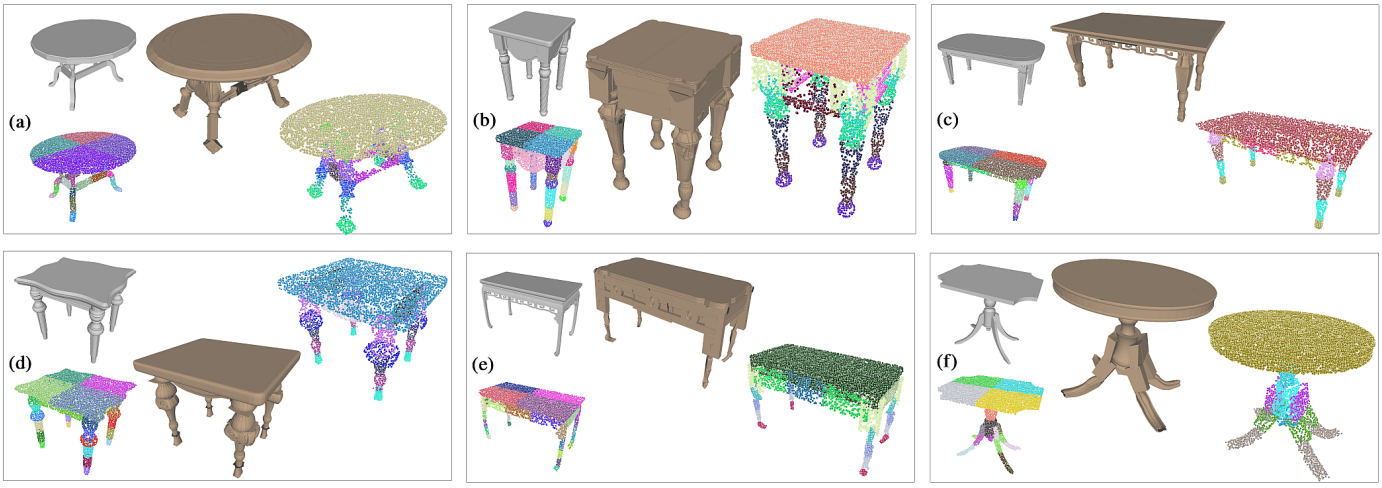

Fig. 6. Shapes synthesized using different templates from our input set of tables. For each group of results, we show the template shape (top-left, in gray), the fine-grained segmentation of the template (bottom-left), the synthesized mesh (center), and the synthesized point cloud (right).

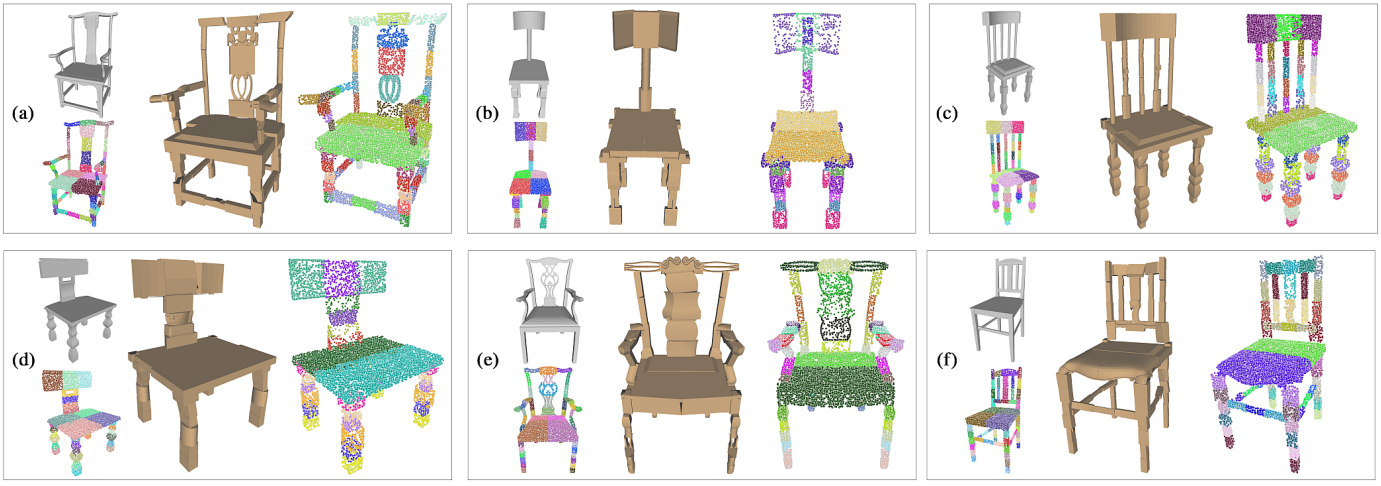

Fig. 7. Shapes synthesized using different templates from our input set of chairs. For each group of results, we show the template shape (top-left, in gray), the fine-grained segmentation of the template (bottom-left), the synthesized mesh (center), and the synthesized point cloud (right).

that all the synthesized shapes that we present in this and the following two sections do not contain any of the original segments of the input template.

Following the symmetry annotations that are made for each shape during the segmentation, our method employs the same segment to replace each part that was identified as symmetric in the template. Thus, all the synthesized point clouds that we present in this work show the same color for all the symmetric parts. As we can see in Figures 6 and 7, using the same segment for all the symmetric parts allows us to better preserve the structure of the template in the synthesized models.

Moreover, the sampling process used to synthesize a shape finds segments that have an overall geometry that is quite similar to the geometry of the original segments of the template. Hence, our method can generate shapes that keep a plausible structure while using templates that have diverse forms and topologies. For example, in Figure 6(a), (b), and (f) and Figure 7(a), (b), and (c), we can see that the synthesized shapes maintain the general form of the corresponding templates. We analyze in more detail the process of sampling the segments for the synthesis process in Section 6.5.

6.2. Shapes generated from the same template

Figure 8 presents three shapes synthesized from a single template selected from the set of chairs, and the corresponding value of the shape energy computed using Eq. 2 for each shape. We observe that our approach generates three shapes that are distinct from each other and from the template, possessing many local variations in their geometry. This can be seen especially in the backs of each synthesized chair. Additionally, all the shapes maintain the general form of the template, although the synthesized shape in Figure 8(a), which has the highest shape energy, presents a form that resembles more closely the structure of the template, in comparison to the shapes shown in (b) and (c). As we explained in Section 5, the value of the shape energy defined by Eq. 2 captures the general

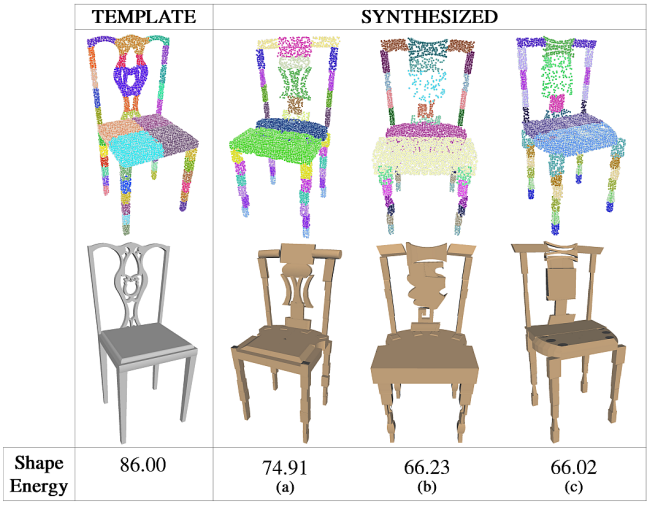

Fig. 8. Models generated from a single template chair. Top row: segmented point cloud for the template and synthesized point clouds; middle row: input and synthesized meshes; bottom row: shape energy for each shape.

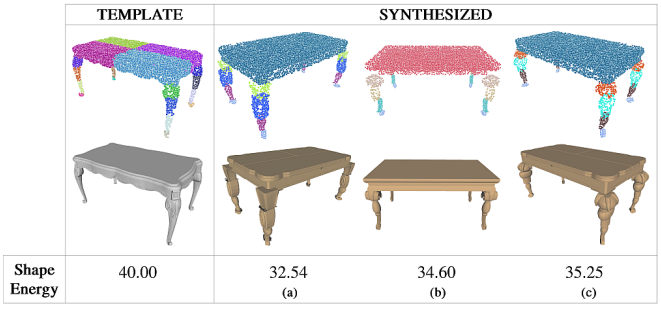

Fig. 9. Shapes synthesized from a single template table. Top row: segmented point cloud for the template and synthesized point clouds; middle row: input and synthesized meshes; bottom row: shape energies.

similarity of the individual segments, as well as the similarity between the contexts of each segment of the synthesized shapes in comparison to the corresponding segments of the template.

Furthermore, all the table models that we present in Figure 9 present several fine detail variations in their legs and, at the same time, they all preserve the general structure of the template. Correspondingly, the values of the shape energy for the three synthesized shapes are similar, which shows that our shape energy captures in an appropriate manner the similarities between the segments of the template and the fine-grained parts sampled for the synthesized shapes. Nonetheless, the form of the segments of the shape of Figure 9(a) presents some differences with respect to the original segments of the template and this is captured in a correct manner by our shape energy, since the lowest energy value shown in Figure 9 corresponds to the shape shown in (a), which is most dissimilar to the template.

6.3. Shapes synthesized from a hybrid set

Since our synthesis approach is based on the geometric properties of the segments used for synthesis and not on their semantic labeling, our method can be employed in a straightforward manner on a hybrid set containing shapes of different classes. In this way, we can introduce additional variations into our synthesized shapes. In Figure 10, we show various shapes generated with our method using the hybrid dataset. In each synthesized model, we highlight a segment that was extracted from a family that is different from the family of the template shape.

Our approach can combine the fine-grained segments of the two families of the hybrid set generating models that are plausible variations of the original template shape. In particular, the chairs in Figure 10(a) and (e) contain in their backs several segments that have been extracted from shapes of the family of tables. Meanwhile, for the table shown in Figure 10(c), only its top segments were taken from another input table, while all the other segments, including the fine details on the sides of the table, come from segments extracted from chair models.

In addition, the synthesized tables that we show in Figure 10(b) and (f) employ several segments from the chairs family. For instance, we can observe that the seat of a chair has been used in a feasible manner to replace the top of the input table template in both (b) and (f). Moreover, we see in Figure 10(d) a chair model created from a template with a complex structure, which also contains various details in the legs and the back.

6.4. Shapes synthesized preserving parts of the template

Using our method, we can also select specific segments of the template shape that must be preserved in the synthesized shapes, yielding additional variations in our results. Figure 11 shows several shapes that we have synthesized maintaining different parts of a given template. For each synthesized shape, we highlight the segments of the template that were preserved.

As we can observe, our method can place and align in an appropriate manner the segments of the original template together with the segments that were sampled from other shapes, producing a cohesive and plausible shape. We can select several adjacent segments from the template to be preserved in the novel shapes, as we show in the example of Figure 11(a). Additionally, we can also preserve in the synthesized shape smaller portions of the template, as illustrated in Figure 11(b).

6.5. Analysis of the method

Shape energy. By means of our shape energy function, which incorporates the similarities of each individual segment, and the similarities between the segments that are adjacent in the 3D shapes, our method can find suitable parts to replace the segments of the input templates. We show in Figure 12 the sources of some of the segments that were used by our method to generate the shape of Figure 7(b). We see in this example that the segments that belong to the legs of other shapes are the segments most commonly selected by our approach for the legs of the generated shape, and the same occurs with other parts.

In addition, Figure 13 shows an example of the individual energy values computed with Eq. 3 for two segments of three shapes generated from a single input template. In particular, the segment denoted as $C1$ has the highest energy, and we observe that this segment is the most similar with respect to the corresponding segments $T1$ of the template, since the segment $C1$ has a curved form that is similar to the form of segment $T1$, while segments $A1$ and $B1$ do not exhibit such curved form.

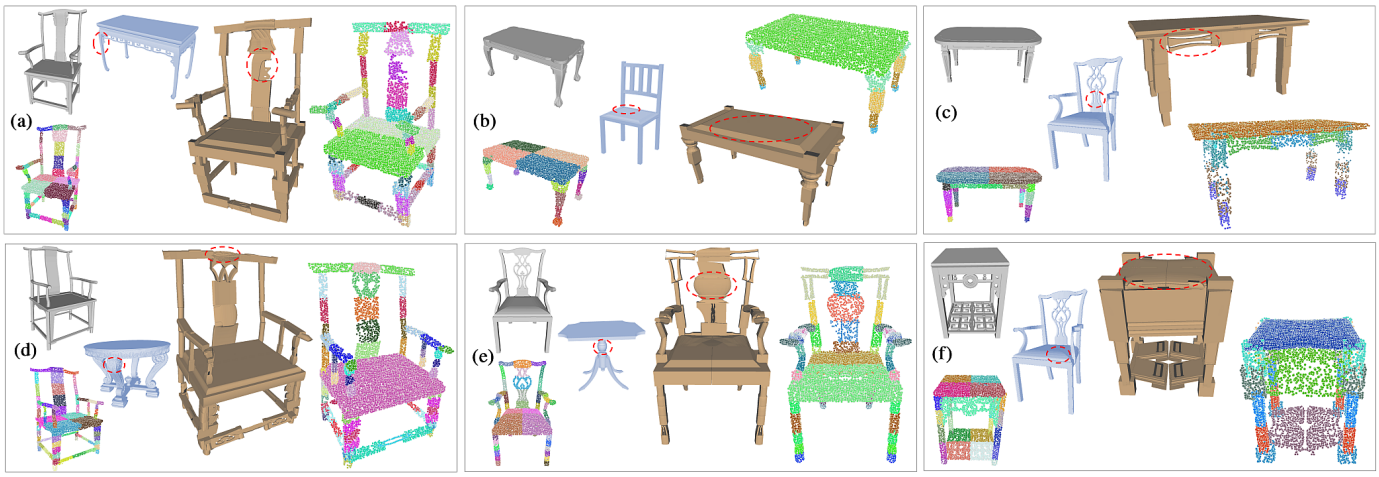

Fig. 10. Shapes synthesized with the hybrid set. For each group of results, we show the original template shape (gray) and its fine-grained segmentation. We highlight with a red circle a segment that was extracted from a shape (blue) that belongs to a family that is different from the template's family.

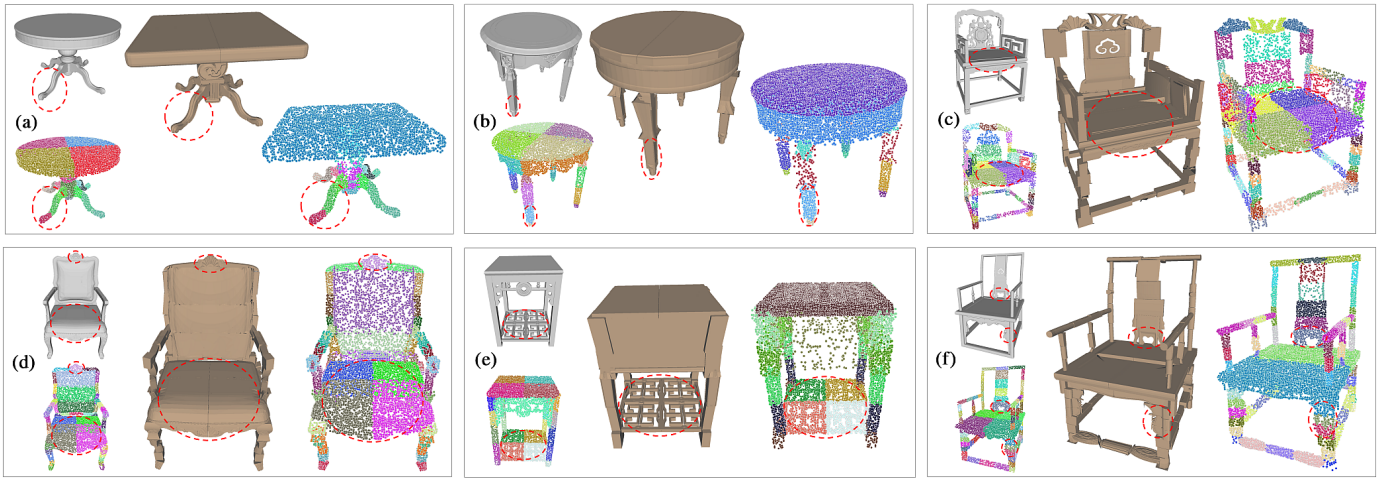

Fig. 11. Shapes synthesized using different input datasets, and preserving specific parts of the template shape. The preserved parts are circled in red.

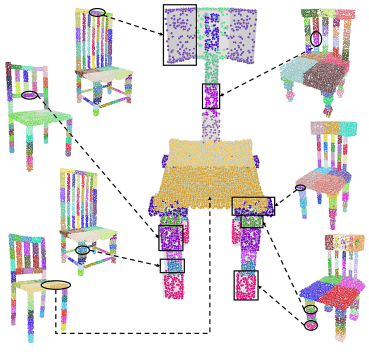

Fig. 12. Example of some of the fine-grained segments selected by our method for synthesizing the chair shown in the center. The mesh synthesized from this point cloud is shown in Figure 7(b).

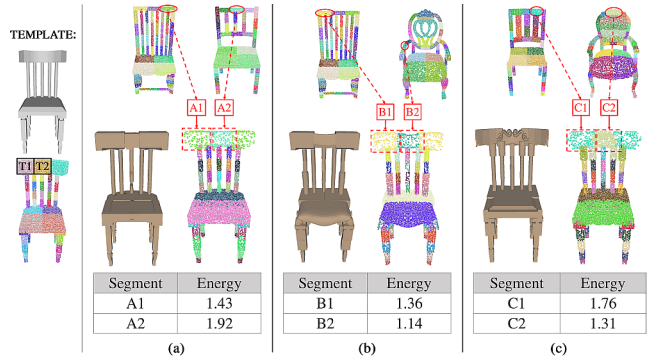

Fig. 13. Energy values computed for pairs of segments sampled to synthesize three chair models (brown) from a single template (gray).

Likewise, the lower energy values of the segments $B2$ and $C2$ capture the differences in geometry and context between these two segments and the segment $T2$ of the template.

Timing statistics. Segmenting an input set with our interactive program requires considerable time, especially for shapes with complex geometries. Segmenting a shape with our program requires on average 1.8 hours for the set of chairs, and 1.3 hours

for the tables dataset. On the other hand, for shape synthesis, on a PC with a 3.4Ghz Intel Core i7 6700 and 16Gb of RAM, the generation of a new point cloud with our unoptimized Matlab implementation requires an average time of 14.98s for the set of tables that contains 415 segments; 13.92s for the set of chairs (1,627 segments); and 16.21s for our hybrid set (1,229 segments). Generating a new mesh only requires a fraction of a second in all cases. However, file I/O takes considerable time for the subdivided meshes, and thus the total time for synthesizing a mesh and writing its file is on average 10.20s.

7. Conclusions, limitations, and future work

In this paper, we presented a pipeline for 3D synthesis of man-made shapes based on recombining fine-grained segments extracted from an input set of shapes. The results of diverse experiments performed with different datasets, including a hybrid set that contains shapes of different classes, show that our method can generate 3D models that have a wide range of fine details. In addition, we showed that our method can be employed to preserve certain portions of the original input template, allowing us to constrain the synthesis.

We showed that we can generate a plausible shape preserving the structure and topology of a template by maximizing an energy function that captures in an appropriate manner the overall similarities between the parts of the template and the parts selected to synthesize the shape. Although we showed with a qualitative analysis that the use of our selected descriptors and the maximization of the shape energy lead to the synthesis of meaningful shapes, as future work, we would like to perform a more thorough analysis of our shape energy and possibly a user study to evaluate our results in a quantitative manner. Specifically, we could ask users to evaluate the plausibility and quality of the shapes generated by our method. Moreover, to the best of our knowledge, our method is the first to synthesize shapes from more granular parts. Thus, a comparison to existing shape synthesis approaches is difficult, as our objective and the type of segmentation that we use are different in nature from those of previous methods. Nevertheless, a user study would also allow us to compare our shapes with the ones produced by previous methods, e.g. [4, 6], in terms of quality and also variability.

Additionally, our method does not require a semantic segmentation, nor part correspondences between the shapes of the input dataset, unlike most previous approaches. However, we currently employ an interactive tool to obtain our fine-grained segments with consistency in size across all the shapes of our dataset. Moreover, our pre-processing stage requires considerable manual effort, as implied by the large average times required to segment a shape with our interactive program. Several alternatives can be explored to obtain our fine-grained segments automatically. First, we could combine the tool for regular partitioning of segments that is part of our interactive segmentation program with automatic approaches for shape segmentation that are based only on the geometric information of the shapes [9], which would yield a segmentation with meaningful boundaries and parts of consistent size. Also, we could improve the consistency of the segments by training a model to perform our fine-grained segmentation, by means of traditional learning methods [28] or recent deep-learning techniques [11]. Such a method would need to learn how to partition parts into consistent sizes and following natural boundaries, but would not need to account for all possible semantics (part labels) of the shapes.

On the other hand, we observed in some of our results that the alignment process does not prevent intersections or other types of visual defects from happening between adjacent segments, as shown in Figure 14(a). In addition, when there are significant differences in the geometry of the boundaries of adjacent segments, the alignment process can connect only a small portion of the boundaries of the segments, as shown in the inset of Figure 14(b). To solve this problem, instead of computing an alignment, we could deform the boundaries of the segments so that they match to each other. This could be achieved, for instance, with a deformation approach based on transformation propagation [45], where we would allow the parts to deform more near the boundaries.

In some cases, conflicts could occur in the alignment process while traversing the adjacency graph of the template. Specifically, a segment that should be aligned with some of its neighbors, could have been aligned previously with respect to another neighbor. To avoid disconnecting pairs of segments that were aligned in previous stages of the process, we "freeze" the segments that have already been aligned, even though this could leave gaps between some adjacent segments in the generated shapes. However, we note that traversing the adjacency graph of the template shape using a breadth-first approach, and starting the traversal from one of the segments that has the largest number of neighbors, allows us to avoid this issue in most cases.

Another possibility for improving the segment alignment and connectivity consists in obtaining a mesh from the synthesized point cloud with a surface reconstruction approach, instead of extracting the mesh segments from the original input shapes. In particular, we could employ methods like Poisson surface reconstruction [46], where we could constrain the reconstruction to preserve the internal parts of the segments while smoothing the boundary regions, avoiding the loss of the fine details that we seek to maintain in our shapes. This would prevent some segmentation artifacts in our synthesized meshes, which can appear in some of our results as small black patches, as shown in one of the segments of the legs of the shape in Figure 14(b).

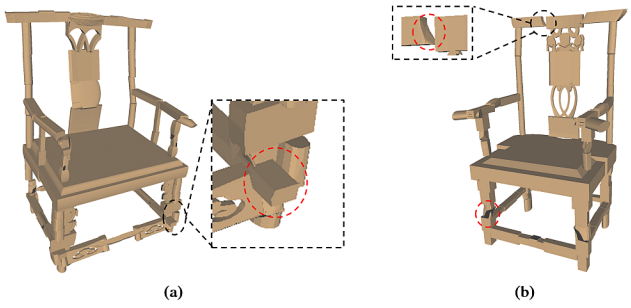

Fig. 14. (a) Example of a case where the alignment process produced intersections between segments; (b) Synthesized shape where we show limitations of the process of alignment between adjacent segments, and segmentation artifacts.

Finally, another interesting direction for future work could involve the use of different options to constrain the synthesis process. For instance, by using not just one input template, but by combining parts coming from different templates, we could introduce additional variability, such as topological changes, to our synthesized shapes.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported by NSERC (2015-05407).

References

[1] Funkhouser, T, Kazhdan, M, Shilane, P, Min, P, Kiefer, W, Tal, A, et al. Modeling by example. ACM Trans on Graphics 2004;23(3):652–663.

[2] Huang, H, Kalogerakis, E, Marlin, B. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. Computer Graphics Forum 2015;34(5):25–38.

[3] Jain, A, Thormählen, T, Ritschel, T, Seidel, HP. Exploring shape variations by 3D-model decomposition and part-based recombination. Computer Graphics Forum 2012;31(2pt3):631–640.

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

[5] Liu, H, Vimont, U, Wand, M, Cani, MP, Hahmann, S, Rohmer, D, et al. Replaceable substructures for efficient part-based modeling. Computer Graphics Forum 2015;34(2):503–513.

[6] Xu, K, Zhang, H, Cohen-Or, D, Chen, B. Fit and diverse: set evolution for inspiring 3D shape galleries. ACM Trans on Graphics 2012;31(4):57:1–57:10.

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

[8] Su, X, Chen, X, Fu, Q, Fu, H. Cross-class 3D object synthesis guided by reference examples. Computers & Graphics 2015;54:145–153.

[9] Shamir, A. A survey on mesh segmentation techniques. Computer Graphics Forum 2008;27(6):1539–1556.

[10] Theologou, P, Pratikakis, I, Theoharis, T. A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation. Comp Vision and Image Understanding 2015;135:49–82.

[11] Shu, Z, Qi, C, Xin, S, Hu, C, Wang, L, Zhang, Y, et al. Unsupervised 3D shape segmentation and co-segmentation via deep learning. Computer Aided Geometric Design 2016;43:39–52.

[12] Yi, L, Guibas, L, Hertzmann, A, Kim, VG, Su, H, Yumer, E. Learning hierarchical shape segmentation and labeling from online repositories. ACM Trans on Graphics 2017;36(4):70:1–70:12.

[13] Blanz, V, Vetter, T. A morphable model for the synthesis of 3D faces. In: Proc. SIGGRAPH. ACM Press; 1999, p. 187–194.

[14] Allen, B, Curless, B, Popović, Z. The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans on Graphics 2003;22(3):587–594.

[15] Kraevoy, V, Julius, D, Sheffer, A. Model composition from interchangeable components. In: Proc. Pacific Graphics. IEEE; 2007, p. 129–138.

[16] Sharf, A, Blumenkrants, M, Shamir, A, Cohen-Or, D. Snappaste: an interactive technique for easy mesh composition. The Visual Computer 2006;22(9):835–844.

[17] Takayama, K, Schmidt, R, Singh, K, Igarashi, T, Boubekeur, T, Sorkine, O. Geobrush: Interactive mesh geometry cloning. Computer Graphics Forum 2011;30(2):613–622.

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

[19] Jaiswal, P, Huang, J, Rai, R. Assembly-based conceptual 3D modeling with unlabeled components using probabilistic factor graph. Computer-Aided Design 2016;74:45–54.

[20] Sung, M, Su, H, Kim, VG, Chaudhuri, S, Guibas, L. ComplementMe: weakly-supervised component suggestions for 3d modeling. ACM Trans on Graphics 2017;36(6):226:1–226:12.

[21] Zheng, Y, Cohen-Or, D, Mitra, NJ. Smart variations: Functional substructures for part compatibility. Computer Graphics Forum 2013;32(2):195–204.

[22] Huang, SS, Fu, H, Wei, LY, Hu, SM. Support substructures: support-induced part-level structural representation. IEEE Trans Visualization & Computer Graphics 2016;22(8):2024–2036.

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

[24] Yumer, ME, Chaudhuri, S, Hodgins, JK, Kara, LB. Semantic shape editing using deformation handles. ACM Trans on Graphics 2015;34(4):86:1–86:12.

[25] Averkiou, M, Kim, VG, Zheng, Y, Mitra, NJ. ShapeSynth: parameterizing model collections for coupled shape exploration and synthesis. Computer Graphics Forum 2014;33(2):125–134.

[26] Bokeloh, M, Wand, M, Seidel, HP. A connection between partial symmetry and inverse procedural modeling. ACM Trans on Graphics 2010;29(4):104:1–104:10.

[27] Kalojanov, J, Bokeloh, M, Wand, M, Guibas, L, Seidel, HP, Slusallek, P. Microtiles: Extracting building blocks from correspondences. Computer Graphics Forum 2012;31(5):1597–1606.

[28] Kalogerakis, E, Hertzmann, A, Singh, K. Learning 3D mesh segmentation and labeling. ACM Trans on Graphics 2010;29(4):102:1–102:12.

[29] Kim, VG, Li, W, Mitra, NJ, Chaudhuri, S, DiVerdi, S, Funkhouser, T. Learning part-based templates from large collections of 3D shapes. ACM Trans on Graphics 2013;32(4):70:1–70:12.

[30] Hu, R, van Kaick, O, Wu, B, Huang, H, Shamir, A, Zhang, H. Learning how objects function via co-analysis of interactions. ACM Trans on Graphics 2016;35(4):47:1–47:13.

[31] Hu, R, Li, W, Kaick, OV, Huang, H, Averkiou, M, Cohen-Or, D, et al. Co-Locating Style-Defining Elements on 3D Shapes. ACM Trans on Graphics 2017;36(3):33:1–33:15.

[32] Liu, Z, Zhang, J, Liu, L. Upright orientation of 3D shapes with convolutional networks. Graphical Models 2016;85:22–29.

[33] Osada, R, Funkhouser, T, Chazelle, B, Dobkin, D. Shape distributions. ACM Trans on Graphics 2002;21(4):807–832.

[34] van Kaick, O, Fish, N, Kleiman, Y, Asafi, S, Cohen-Or, D. Shape segmentation by approximate convexity analysis. ACM Trans on Graphics 2014;34(1):4:1–4:11.

[35] Mitra, NJ, Guibas, LJ, Pauly, M. Partial and approximate symmetry detection for 3D geometry. ACM Trans on Graphics 2006;25(3):560–568.

[36] Zhao, X, Wang, H, Komura, T. Indexing 3D scenes using the interaction bisector surface. ACM Trans on Graphics 2014;33(3):22:1–22:14.

[37] Lun, Z, Kalogerakis, E, Wang, R, Sheffer, A. Functionality preserving shape style transfer. ACM Trans on Graphics 2016;35(6):209:1–209:14.

[38] Shapira, L, Shamir, A, Cohen-Or, D. Consistent mesh partitioning and skeletonisation using the shape diameter function. The Visual Computer 2008;24(4):249.

[39] Pauly, M, Gross, M, Kobbelt, LP. Efficient simplification of point-sampled surfaces. In: Proc. Visualization. IEEE Computer Society; 2002, p. 163–170.

[40] Sidi, O, van Kaick, O, Kleiman, Y, Zhang, H, Cohen-Or, D. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. ACM Trans on Graphics 2011;30(6):126:1–126:10.

[41] Boykov, Y, Veksler, O, Zabih, R. Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Analysis & Machine Intelligence 2001;23(11):1222–1239.

[42] Cormen, TH. Introduction to algorithms. MIT press; 2009.

[43] Chang, AX, Funkhouser, T, Guibas, L, Hanrahan, P, Huang, Q, Li, Z, et al. ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR]; Stanford University — Princeton University — Toyota Technological Institute at Chicago; 2015.

[44] Trimble 3D Warehouse. `https://3dwarehouse.sketchup.com/index.html`; 2017. Accessed: 2017-10-02.

[45] Sorkine, O, Cohen-Or, D, Lipman, Y, Alexa, M, Rssl, C, Seidel, HP. Laplacian surface editing. In: Symposium on Geometry Processing. Eurographics; 2004,.

[46] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Transactions on Graphics (TOG) 2013;32(3):29:1–29:13.