DOI: 10.1111/cgf.12039
EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Smart Variations: Functional Substructures for Part Compatibility

Youyi Zheng KAUST
Daniel Cohen-Or TAU
Niloy J. Mitra University College London

Abstract

*As collections of 3D models continue to grow, reusing model parts allows generation of novel model variations. Naïvely swapping parts across models, however, leads to implausible results, especially when mixing parts across different model families. Hence, the user has to manually ensure that the final model remains functionally valid. We claim that certain symmetric functional arrangements (*SFARR-*s), which are special arrangements among symmetrically related substructures, bear close relation to object functions. Hence, we propose a purely geometric approach based on such substructures to match, replace, and position triplets of parts to create non-trivial, yet functionally plausible, model variations. We demonstrate that starting even from a small set of models such a simple geometric approach can produce a diverse set of non-trivial and plausible model variations.*

1. Introduction

Geometric design remains challenging. For most users, 3D modeling from scratch is tedious, cumbersome, difficult, and often not a realistic option. Alternatively, new models can be created by reusing existing parts. The key challenge is to create novel, non-trivial, and interesting variations, while still maintaining functional plausibility.

Model collections have grown in popularity with the availability of online 3D model repositories (e.g., Google Warehouse, Turbosquid, etc.), especially for man-made objects. Various automatic and semi-automatic approaches have been proposed to organize and consistently segment models of such collections, leading to interesting reuse possibilities (e.g., [ARSF09, WAvK*12]). Naïve methods to interchange model parts can easily fail: random swapping of parts quickly destroys model plausibility, or even text label-based part replacement can lead to inconsistency, e.g., a 'mug handle' is different from the 'handle' of a racket. Hence, researchers have proposed a mix-and-match by swapping parts based on geometric similarity only [FKS*04], or considering both geometric similarity and contact information for part replacements [KJS07]. The task of content creation, however, remains difficult since the user has to carefully control the final shape to ensure its plausibility.

In another approach, probabilistic models are learned from training datasets to encode part-based composition [CKGK11, KCKK12]. Subsequently, model synthesis amounts to sampling from such probabilistic models. These

Figure 1: *Starting from only three models (top), our algorithm detects and exploits symmetric functional arrangements (*SFARR-*s) to generate more than 20 variations. Many of the variations have strong geometric differences, but still preserve the object sub-structures and functionalities.*

Figure 2: *Algorithm overview. Given two (pre-segmented) shapes (a), we extract and encode mutual part relations (e.g., symmetry, contact, and support) in the form of relation graphs (b). Next, certain compatible symmetric functional arrangements (*SFARR*) are detected in each graph (shown by boxes in both graphs) and subsequently reshuffled to produce model variations. Note that even two models can produce dozens of variations (c), yet the synthesized models remain plausible.*

methods, however, require a moderate to large number of part-labeled models for training. Furthermore, while the methods are good at producing probable and expected variations for classes of learned models (e.g., 4-legged animals, battleships, etc.), they cannot reuse parts across different model families (e.g., a play horse and a bus stop).

Starting from only a *small* number of (segmented) models, our goal is to create a large number of *non-trivial* model variations that are also *functionally* plausible. This is challenging since functionality is rarely explicitly encoded in the raw geometric descriptions. We make two observations that simplify the problem: (i) structure (e.g., contact, symmetry, arrangements, etc.) rather that the actual geometry of the parts is critical for model reuse; more importantly, (ii) certain sub-structures often relate to actual functionality of the models. Hence, even objects from different categories that share common sub-structures (e.g., legs of tables and chairs) can be reused to produce novel models with functional validity (see Figure 1). Further, beyond symmetry relations, certain mutual arrangements of parts, as captured by such sub-structures, often are vital for object functions.

We first abstract each input 3D model, assumed to be pre-segmented, into a graph where nodes denote parts and edges capture relations among part-pairs. Note that unlike other methods, we do not require part labels or part-level correspondence. Instead, we hypothesize that certain prescribed subgraphs, which we call *symmetric functional arrangements* (SFARR-s), are often closely linked to the core functionality of objects and hence are critical to the model plausibility. Subsequently, our algorithm identifies such compatible SFARR-s across model pairs, swaps the matched SFARR-s, and finally resizes and positions the replaced SFARR-s using structural cues to create valid model variations.

We demonstrate the performance of our algorithm in creating large numbers of in-class and across-class variations. Our tests indicate that such a direct approach focusing exclusively on structural similarity at the level of SFARR-s is surprisingly sufficient to produce novel and non-trivial variations that look functionally plausible. Our user study reveals that most users found most of the automatically generated variations to be interesting and plausible, while only a small number of across-class variations were found to be too extreme and unrealistic.

2. Related Work

Data-driven 3D creation. In a seminal effort, modeling by example [FKS*04] proposed a mix-and-match approach to combine model parts for creating new models. Parts were replaced based on geometric similarity with the example parts (or proxy boxes) with the user having the option to guide selection, placement, and scaling of the parts. Later, the Shuffler system [KJS07] used contact relations to consistently co-segment models to facilitate such part-level replacements. The task of content creation, however, remained difficult since the user needed to have a good idea of the shape of the final model.

Recently, as model collections grew, researchers have focused on data-driven content creation. In a recent system [CK10], artists create rough 3D content that is then enhanced using customized examples. The system uses component matching and shape retrieval to facilitate the process. In followup efforts, probabilistic models were learned and used for suggestion generation for assembly-based modeling [CKGK11] or generating shape variations from existing ones [KCKK12]. While the results are impressive, the methods assume the availability of large (labeled) training sets consisting of 10-100 models and cannot combine parts across model families.

Bokeloh et al. [BWS10] analyze models to extract potential docking sites leading to interesting inverse procedu-

ral modeling possibilities. Xu et al. [XZZ*] propose photo-inspired creation of 3D models by deforming parts using image silhouette constraints. Jain et al. [JTRS12] present an interesting solution to create new *interpolated* shapes by combining parts retrieved from objects based on hierarchy analysis. In another effort [XZCOC12], shape structures among the same class are exploited to generate shape variations based on fit-and-diverse rules. These methods assume access to labeled parts, or explicit part correspondence, and are targeted to create variations in the same category of objects. Instead, we directly exploit sFARR-s to identify interchangeable parts both intra- and inter-classes, while largely preserving functional plausibility.

Exploring shape collections. Growing volumes of 3D data sets have opened new data exploration possibilities. Early leads in this direction were provided by the ShapeAnnotator framework [ARSF09] where an intuitive interface helps the user in the creation of a part-based shape ontology. 'more recently, researchers have extracted low degree of freedom deformation to discover shape common structures [OLGM11] or diffused correspondence to organize model collections [KLM*12]. We also rely on common substructures. Our analysis, however, is purely geometric based on sFARR-s yielding interesting and novel variations, most of which are plausible.

Shape analysis. Given the close relation between form and function, it is not surprising that relations and symmetries are dominant in man-made objects. Various methods have been proposed to extract such high level abstractions and hierarchies (see survey [MPWC12]). Such high level relations have been used for smart shape editing [GSMCO09, ZFCO*11], co-segmentation [WAvK*12], and upright orientation identification [FCODS08]. In our work, we also exploit shape structure analysis for exchanging parts, but focus on more general shape characteristics using sFARR-s.

3. Symmetry Functional Arrangement (sFARR)

Form follows function, which inspired many in industrial design and modern architecture, suggests that form-finding is primarily driven by the intended function of objects [Sul96, Gre58]. As a result, certain arrangements of parts are consistently observed across objects with similar functions, though the objects can be very different in form (e.g., most functional chairs have legs supporting the seat from below). Hence, we hypothesize that preserving such special arrangements among parts (i.e., part substructures) in the course of model creation maintains the object functionality, and hence its plausibility. In this attempt, we identify such functional substructures for the purpose of part reshuffling.

In our approach, we leverage mutual (geometric) relations among different arrangements of shape parts to identify component-level compatible functional substructures that can be interchanged across different objects towards object creation. Such functional substructures should ideally be: (i) commonly occurring in objects with similar functions, and (ii) characterized by geometric relations among the object parts. In a first attempt, we *define* one particular form of such substructures that we term *symmetry functional arrangements* (sFARR).

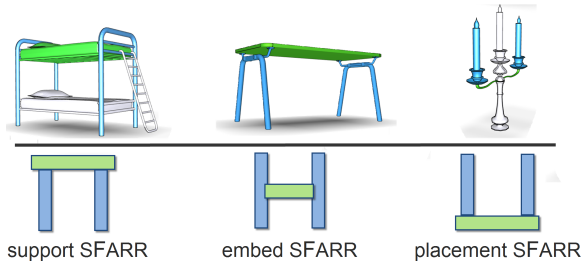

Figure 3: *Examples of three types of predefined* sFARR-*s : support, embed, and placement.*

We define sFARR as a triplet of shape parts, where two are coupled via symmetry, while the third part connects them (see Figure 3). For example, the two supporting legs and the table top, as shown in Figure 3-top, form such a triplet, or sFARR. Since symmetry is ubiquitous in man-made objects, such simple forms of substructures are dominant in many man-made objects (see Figures 1, 5, and 12).

While such sFARR-s are common in man-made environment, they can be non-trivial to detect and replace (see Figure 3). Naïvely interchanging them, especially across objects of different families, can potentially generate an exponential number of invalid objects. Hence, we first carefully examine and classify sFARR-s according to their functions based on the mutual relation of the two symmetric parts with the third one. Specifically, we define three types: *support* sFARR, *embed* sFARR, and *placement* sFARR , respectively (see Figure 3) assuming we know the upvector. Let us denote the three elements of a sFARR triplet as S, O, and S, where S-s are the symmetric parts, and O connects to both S-s. By *support*, we mean O touches both S-s at their top faces; by *embed* we mean O is attached to vertical side faces of the two Ss, and by *placement* we mean two S-s are supported by the top face of O. We found these sFARR-s to be surprising expressive and focus on them, although alternative substructures can potentially be defined and added to the framework.

These sFARR types still do not fully characterize potential object functionalities. For example, in Figure 4, although the two supporting stands of the piano (left) and the two legs of the table (middle) both support sFARR-s, they are not-interchangeable as they do not perform the same physical functionality (one combination stands, while the other is unstable and topples).

To identify such potential functionality of sFARR-s, we propose two simple additional attributes: (i) We classify a sFARR to be stable if it does not topple. We test stability by checking if the projection of its center of mass on to the

Figure 4: *Additional attributes for further verifying functionality of* sFARR*-s to ensure interchangeability. We check for stability by verifying if the projected center falls within the convex hull of the ground-touching points of the* sFARR *parts (left, middle); and if the parts lie on a common axis (right).*

ground falls inside the convex hull of its ground-touching vertices; (ii) We classify a sFARR as coaxial if its three parts are coaxial and the symmetric parts are cylindrical (e.g., the component *O* resembles a rotational axis with two *S*s, such as tyres of a car). We validate if the symmetric components are cylindrical by primitive fitting (e.g., see [ZFCO*11]). Later, we ensure that coaxial sFARR-s are replaced only by other coaxial sFARR-s.

By defining sFARR types and their attributes, we enable replacement of part triplets while maintaining object functionality. They also simplify their comparison during replacement since we only have to examine the graph substructures with matching sFARR-s both in type and attributes. We focus on the two attributes (i.e., stable and coaxial), which according to our observation, are critical and prevents most of the common functionally implausible arrangements. Note that multiple sFARR-s may share common elements (Figure 9) to form clusters of sFARR-s. In Section 6, we describe how to handle such clusters.

4. Algorithm Overview

Our algorithm runs in two main stages, an off-line analysis stage and an online reshuffle stage. In the analysis stage, we extract the mutual relations among object parts and encode them as a spatial relation graph. We then identify all sFARR-s in each object, classify their types, and associate them with attributes. Next, groups of sFARR-s are identified and their attributes examined. We also globally align the shapes to a common orientation to facilitate subsequent part reshuffling.

In the reshuffle stage, the constructed spatial graphs along with the sFARR-s are utilized to generate shape variations with potentially large geometric variations, while preserving object functionality. Note that we allow part reshuffling even across different shape families (where part names are ineffective), as long as the sFARR-s are compatible.

At runtime, we identify compatible sFARR-s and establish correspondence across them. Next, we order the compatible sFARR-s for potential replacements. Before replacing

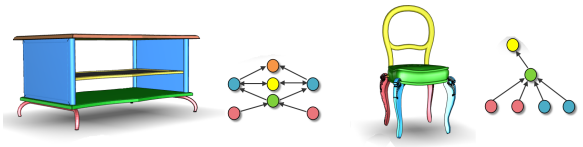

Figure 5: *The spatial relation graph. Each component corresponds to a node. Graph edges are attributed as supportive and embedding. Symmetric nodes are in the same color.*

a single sFARR, we also check its compatibility and graph structure (i.e., relations) to ensure model validity. Having decided on a replacement, in a constrained optimization setting we solve for the best transformation to properly fit the retrieved parts to their new positions, using both topological and geometrical constraints (see Figure 2). Note, here we benefit from part-proxy level optimization as proposed in [ZFCO*11].

5. Pre-processing

In the off-line stage, we start with models from different shape families, each assumed to be segmented into meaningful parts. Note that we do not require the parts to be labeled or having correspondence across different models. Recent advances in semi-automatic annotation/co-segmentation algorithms provide access to such pre-segmented data (see [ARSF09,WAvK*12] and references therein). For our experiments we used a dataset comprising of tables, chairs, cars, beds, sofas, lamps etc, where symmetries are abundant. For example, in Figure 5, the chair is decomposed into six parts, each representing a semantic part such as chair leg, chair back, chair seat, etc. Further, we allow the user to interactively group some of the components into semantic meaningful parts (see also [XZCOC12]).

We focus on man-made objects and assume them to have up-right orientations [FCODS08]. Since we do not have correspondence information among different objects parts and to facilitate subsequent reshuffling (e.g., prevent front-facing from being replaced with back-facing), we first consistently align each object to a common facing orientation. Specifically, we use the global reflection plane of each object to align all models with the computed reflectional plane. To address the flip problem (i.e., the problem of back-facing or front-facing), we employ a simple graph-based formulation (see Section 5.2). The user can override the automatic suggestions, if necessary.

5.1. The Spatial Relation Graph

We first abstract the geometric structure (and sFARR-s) as a relation graph to facilitate subsequent analysis and part reshuffling. For each object, we create a spatial relation graph, where a node is a part and directed edges represent relations among pairs of parts. Further, we analyze the relation types and classify them using the sFARR types. Figure

5 shows an example: the legs of the chair and the seat are connected with a directed support edge, where the direction points to the component that is being supported. Note that placement is opposite of support and hence represented by the same type of graph edges. We do not explicitly encode relations like parallelism, coplanarity, or concentricity, but consider them implicitly during part placement optimization. For example, if a part being replaced is coaxial with an existing part, we constrain the existing parts and the new part to be coaxial during optimization [ZFCO*11].

In our implementation, we mark two components a and b to be connected *iff* they are touching each other at some points or share common vertices. A component a is marked to support another component b *iff* they are in contact and a is under b. We detect symmetry relations between part pairs using the method of [MGP06]. Figure 5 shows relation graphs constructed for a chair and a counter model. We also mark a graph node as ground-touching node if its corresponding component touches the ground plane (i.e., lowest in height).

5.2. Graph-based Orientation Rectification

The connectivity of the relation graph abstracts the topology of part connections and symmetry relations (see Figure 5). To find the correct orientation of the part to be placed in, we need consistent orientation. Although we have the global symmetry plane to rectify the upright orientation, ambiguities can still arise due to symmetry flipping, i.e., front facing vs. back facing. We now address this problem as a simple labeling problem.

Each graph has two choices of orientation (by flip) as the choices for spatial graph G_i. Hence there will be a total $2n$ possible labels if we have n graphs to rectify. Let us denote the labels to be $\{l_1, l_2, ..., l_n\}$, where n is the number of objects in the category and l_i denoting flip/no-flip. If there is only a single object, the choice is arbitrary. However, when there are multiple objects to be consistently aligned, we define a labeling cost as the matching distance between two graphs:

$$P(G_i \rightharpoonup l_i, G_j \rightharpoonup l_j) = dist(G_i, G_j). \quad (1)$$

The best alignment is found as the minimum value of pairwise matching cost using a MRF formulation. To define distance between two relation graphs, i.e., $dist(G_i, G_j)$, we solve a graph assignment problem using the Hungarian algorithm based on the Euclidean distances between the centroids of nodes of the spatial relation graphs. Figure 6 shows an overview of the process. Note that we perform this alignment only across objects from the same family. For objects from two different families, the user manually aligns a pair (one coming from each family), thus assigning a consistent alignment among all model pairs.

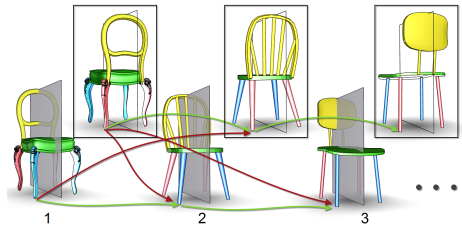

Figure 6: *Model orientation rectification with MRF labeling. Each model has two orientation possibilities. The cost function (pairwise term) measures the matching distance between two objects (in terms of graph distances).*

6. sFARR-based Parts Reshuffle

A natural way to mix objects is to identify compatible subgraphs among two relation graphs, e.g., subgraphs with the same structure and being functionally compatible. In this work we only focus on sFARR-based substructures, which consistently seems to result in compelling variations. In this section, we describe how we identify, group, replace, and position triplets of parts using sFARR-s.

6.1. Identify Compatible Symmetry Triplets

To synthesize from a set of shapes, we first identify sFARR-s in all the relation graphs and find compatible sFARR-s within these graphs. Given two graphs, G_i and G_j, for each sFARR $s_k^i \in G_i$, we first identify all its compatible sFARR-s in G_j. Two sFARR-s are defined to be compatible *iff* the following conditions hold (Figures 1 and 2): (i) they are of the same type, i.e., support, embed, or placement; and (ii) their share the same attributes, i.e., stable, or coaxial.

6.2. Group sFARR-s

When there are multiple sFARR-s sharing the same components, we examine them with additional compatibility criteria to prevent undesirable replacements. For example, in Figure 9, the four table legs both support the table top and being supported by the bottom slot, i.e., there are both support sFARR-s and placements sFARR-s. In this case, the legs which are placement sFARR-s also act as support, hence we probably would not wish to replace them with another placement sFARR that cannot act as support, for example like the candles shown in Figure 3. Thus, we add new functional attributes to individual nodes in sFARR once they are shared by multiple sFARR-s, i.e., when groups of sFARR-s are found.

Let us denote the three types of sFARR-s as S (support), E (embed), and P (placement), respectively. The operator $\otimes$ denotes that two triplets have common elements, i.e., $X \otimes Y \Rightarrow Z$ means two sFARR-s with types of X and Y share common node(s), and their common node(s) will be associated with attribute Z (the nodes in each sFARR are originally

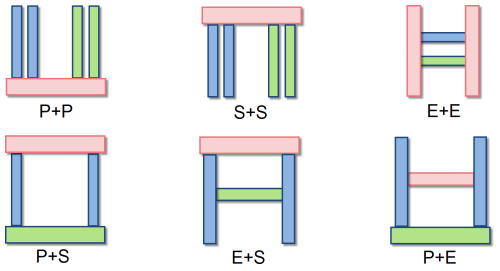

Figure 7: *Different combinations of* SFARR*-s form clusters. We define new functional attributes for elements in a* SFARR *once it is shared by another.*

assigned with the corresponding attributes). Now we define the following rules ($\varnothing$ means no attribute to be added):

$$P \otimes P \Rightarrow \varnothing \qquad S \otimes S \Rightarrow \varnothing \qquad E \otimes E \Rightarrow \varnothing$$
$$P \otimes S \Rightarrow P+S \qquad E \otimes S \Rightarrow E+S \qquad P \otimes E \Rightarrow \varnothing$$

Figure 7 shows six basic combinations. Note that if both the symmetric components in a SFARR touch the ground, we also add the S attribute to the two elements. Let the set of attributes for nodes n_i and n_j be Δ_i and Δ_j, respectively. Once we add such functional attributes to the SFARR elements, we define a SFARR node n_i as replaceable by n_j *iff* both of the following holds:

- $\Delta_i \subseteq \Delta_j$ or $(\Delta_i \setminus \Delta_j) \cap \{S,P\} = \varnothing$
- n_i and n_j have the same number of contact slots (see Section 6.4).

Note that we define a preference across the three types: $S = P \geq E$ because the embed attribute has lower priority than the other two attributes in relative occurrence in typical man-made objects (see also Section 6.4 for contact compatibility).

When SFARR-s form shared elements, multiple SFARR-s of the same type can further form new SFARR-s. Figure 8 shows such an example, where the original SFARR-s formed by the four chair legs are not compatible to the foot rest of the counter, but grouping them forms new compatible support SFARR-s. In our algorithm, we identify all such possible groups of SFARR-s and add them in part reshuffling, which greatly increases the variations in the output.

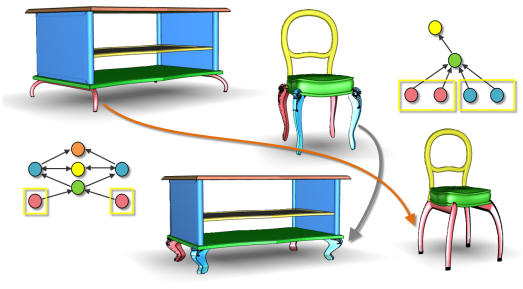

Figure 8: *Groups of* SFARR*-s can form new* SFARR*-s, and offers new possibilities for compatibility. The two pairs of symmetry legs of the chair shares a common seat, grouping the symmetries can lead to one single* SFARR *which is compatible to the foot rest of the counter.*

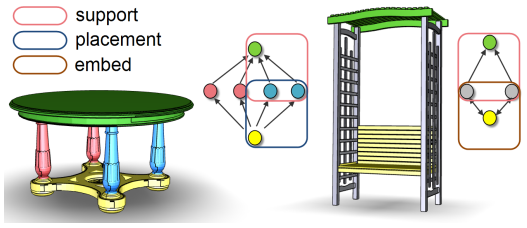

Figure 9: *Multiple* SFARR*-s can share common elements leading to new functionalities of the shared elements.*

6.3. Replace SFARR

Next, we search for SFARR replacements. Given a set of graphs Ω and a graph $G_i \in \Omega$, we first find all the SFARR-s in G_i and their compatible SFARR-s in all the rest of $G_j \in \Omega$ forming a set say Ξ^i. We then search for all possible SFARR clusters in G_i and sort them by their sizes. Assume the list of sorted SFARR clusters is $\{L_1, L_2, ..., L_k\}$. Within each SFARR cluster L_j, we use a greedy algorithm to sort the SFARR-s by their compatibility to the SFARR-s in Ξ^i. For any two compatible SFARR-s $t_k \in G_i$ and $t_l \in G_j$, we measure their geometry compatibility $\Upsilon(t_k, t_l)$ based on how the parts got relatively scaled, how the relative arrangement of the part triplets changed in terms of subtended angles and mutual distances. Specifically,

$$\Upsilon(t_k, t_l) = \lambda \cdot \theta \cdot \vartheta, \tag{2}$$

where $\lambda = \Pi_{i=0}^{2} \Pi_{t=x,y,z} g(s_t^k, s_t^l)$ is the scale compatibility between the corresponding nodes in t_k and t_l with $g(x,y) = 1 + |x-y|/(x+y)$ is a monotonic function and $g(s_t^k, s_t^l)$ denotes the scale ratio of their corresponding i-th nodes in $t = x,y,z$ dimensions (measured as bounding box scales); $\theta = g(\theta^k, \theta^l)$ is the angle ratio of the triangles that are spanned by the bounding box centers of the three nodes in the triplets; and $\vartheta = g(\vartheta^k, \vartheta^l)$ is the perimeter ratio of the two triangles. The SFARR-s replacement is then performed greedily, starting from the SFARR which has the minimum value of $\Upsilon(.,.)$.

The cluster sizes along with the values of $\Upsilon(.,.)$ define replacement order for all triplets in G_i. When a SFARR is replaced, we tag all the nodes to prevent them from being replaced again in the current round of reshuffling. For example, in Figure 12 dozens of variations are generated starting from only six models.

6.4. Position SFARR

Since we mainly considered topology and SFARR compatibility rather that geometric coherence, we now warp the placed-in components to make them cohesive with the current model. Since the SFARR parts have different scales, we

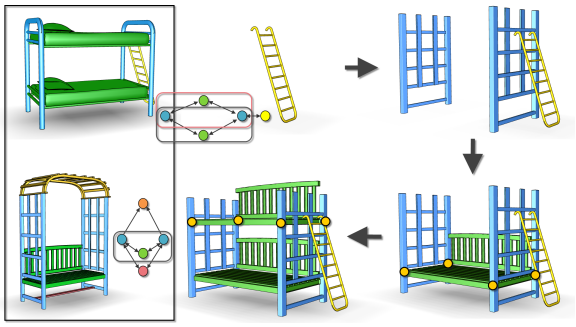

Figure 10: *Optimizing* SFARR*-s placement. The optimization proceeds node by node. Each time a new node is identified and optimized with respect to the existing content.*

handle the replacements of the three nodes individually. We start by fixing all other parts that stay unchanged (i.e., not being replaced) or already have been substituted, denote this set of nodes as Λ. Then, to replace a SFARR t_i, we first identify a node in t_i, which is connected to the nodes in Λ. To place it, we proceed similar to [KCKK12]. Specifically, if node a is going to be replaced by node b, we find all nodes in Λ that share a relation with a, then formulate these relations as constraints in an optimization. In practice, both contact and supporting relations are represented in terms of contacting slots between two components (see Figure 10). We also add constraints that require the placed-in node to have similar scale with the node being replaced. If the constraining contact slots are less than three, we directly solve for best translation, scaling, and possible rotation by looking at the centers of contacts. If there are more than three contact constraints, we solve linear equations for the best translation and scaling as in [KCKK12].

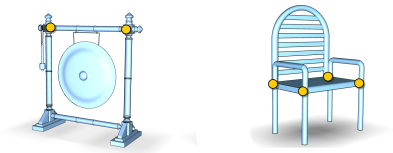

Figure 11: *The numbers of contact slots further classifies the compatibility between components. The chair seat that has four contact slots is not compatible with the cylindrical bar shown on the left, which has only two contact slots.*

Importantly, the contact slots do not only serve as constraints for the part placements, but also provide hints about part compatibility. For example, the chair in Figure 11 has four slots of contacts while the top bar of the gong model has only two. Hence, the two parts are incompatible since they have different number of contact slots. We define contact slots as follows: for cylindrical parts, we define one contact slot at its contacting end; for cuboid parts, we define two slots at the contact edge (see Figures 3 and 10).

Once the first node is properly positioned, we add it to Λ and repeat the process until all three nodes are added. Symmetry is handled as soon as its opponent node is replaced by mapping the corresponding transformation. Figure 10 shows the process. Note that in some cases the best transformation may cause the components to break their structural properties such as deforming a circular chape to an elliptical one. This can be solved using state-of-the-art structure-preserving shape editing techniques [KSSCO08, GSMCO09]. Note that when replacing a SFARR with another SFARR, there are four possible combination of outcomes, i.e., $aba \otimes cdc = \{aba, ada, cbc, cdc\}$, leading to potentially further variations.

7. Assessing Reshuffle Quality

The reshuffle framework uses geometric compatibility Υ to decide which compatible SFARR to first replace. This offers a quality control for the reshuffle, i.e., we can use Υ to define a threshold for compatibility measurements. Figure 13 shows Υ values to compare the upper SFARR-s of the double-bed with SFARR-s in other models. While smaller Υ can indeed produce variations geometrically similar to the input model, the medium/high ranges of Υ lead to more non-trivial and interesting models, which are particularly useful for creative design. As default, for our tests, we set $\Upsilon = 10$. While without a physical simulation we cannot guarantee functional validity, we found a very large percentage of the synthesized models are useful (see also user study).

8. Results and Discussion

We tested our framework with various families of shapes including tables, chairs, cars, beds, sofas, lamps, lights, candles, shelves, and street lights, etc. The main goal of our algorithm is to demonstrate the effectiveness of the SFARR-s in terms of structure and functionality preservation. Hence, we focus more on cross-family shape reshuffling. Figures 12 and 14 show the main results. After the off-line stage, the online reshuffle is fast. The complexity depends on the number of permissible permutations while preserving SFARR compatibility. Generally it takes less than 2s to synthesize a model including the component loading time on an Intel i7 2.7GHZ laptop with 4 GB memory.

In Figure 12, we mix among six objects including tables, bus-stops, and a play horse to generate dozens of new models, a subset of which are shown. Many of the synthesized shapes have rather different geometry than the original ones, but still most of them retain the functionality, i.e., a bus-stop remains a bus-stop, unless in some particular cases the replaced node has extra functionalities, which we did not explore. In such cases, surprising functionalities may appear in the new shapes. For example, the table supports shown in the green boxes come from the legs of the play horse. Besides supporting the table, the new models can also be *rocked*. On the other hand, such unexamined individual component

Figure 12: *Results demonstrate the support* SFARR. *Starting from six different shapes from different families, our framework produces dozens of non-trivial variations, yet preserving functional plausibility. In green, we highlight some tables that were ranked as interesting; in yellow, we highlight some tables that were ranked as functionally unsatisfactory; in red, we highlight tables that are physically unstable.*

functionality can lead to failures, as shown in yellow and red boxes of Figure 12. In the object marked by the yellow box, the table top is replaced by the top of a bus stop, which does not allow the object to be used as a table. We believe this can be solved by adding more physical constraining attributes, such as horizontal top, placeable, rollable, etc., to the framework by analyzing the shape geometry [UIM12]. We leave this for future work. Figure 14 shows further examples synthesized by our framework.

User interaction. Although our system automatically groups the component parts, manual reassignment is sometimes required to semantically group parts. We support simple manual regrouping, e.g., to merge redundant components across the bus stop in Figure 1. For models with simple structure like chairs, sofa, and lamps, no interaction was required. For more complex models, interactions typically took total of 1-2 minutes for 3-5 complex models. Table 1 lists time required for individual models used in the paper. The corrected groups ensure meaningful component grouping for model parts and thus lead to plausible reshuffling. Since our method works at the level of detected symmetry triplets, the presence of non-meaningful components in SFARR-s may lead to non-meaningful reshuffle results (e.g., parts of the symmetry bars in the bus-stop being replaced with the entire bed bars in Figure 1).

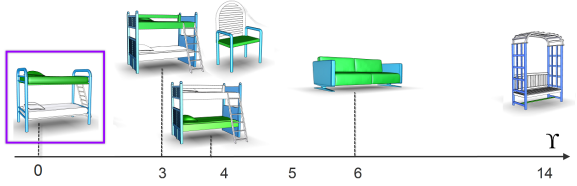

Figure 13: *Geometric compatibility of* SFARR-*s according to* ϒ*: Larger values allow replacement of* SFARR-*s with substantially different geometry, while smaller values restrict the replacements among geometrically similar ones.*

bus stops (Fig. 1, 12) < 40s	beds (Fig. 1) < 20s	tables (Fig. 3, 14) < 10s
sofa (Fig 14) (no interaction)	cars(Fig. 3, 14) < 30s	cabinet (Fig. 14) (no interaction)

Table 1: *Average time for interactive regrouping required for typical models.*

User study. We conducted a user study to assess the results of our framework (see project page for an application demo). Specifically, we showed a small team of 15 people 35 models (evenly distributed among different sets and randomly picked among our results pool), within which 5 were original models. The users were asked to name the model among provided original names (i.e., model families) from which we generated our results. If the users did not agree with the provided names based on object functionality, they had the option of providing a new name, or mark the model as null, i.e., non-functional.

The feedback was largely positive. All the 15 subjects (all computer science students, some with computer graphics background) consistently found most of the models to be functional and agreed with the original names, e.g., a model derived from a table was marked as table. The average hit

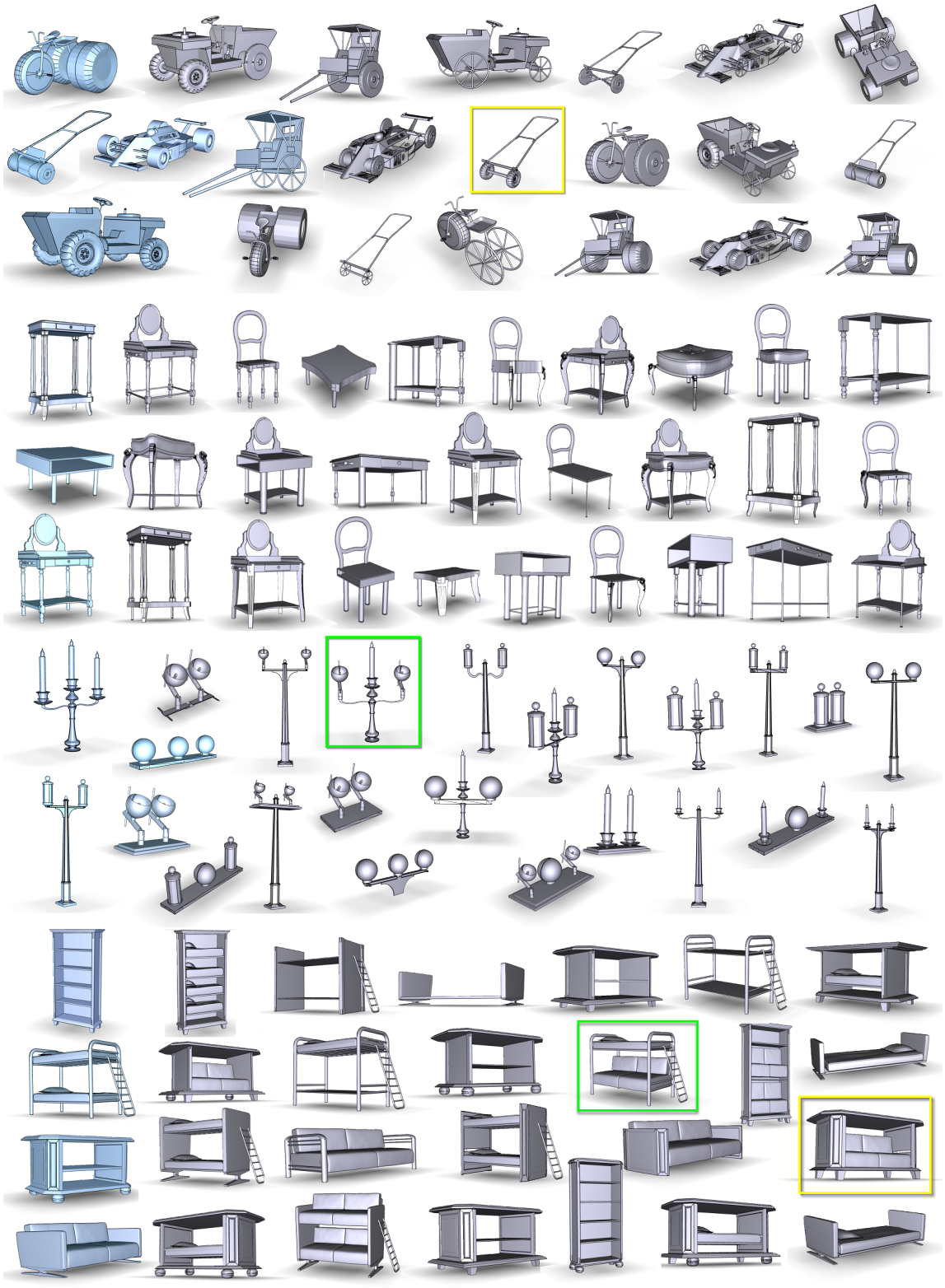

Figure 14: *Reshuffling results among various object families consistently indicate that the proposed* SFARR*-s can synthesize non-trivial shape variations across classes, while maintaining plausible functionalities of the objects.*

rate was 85% (average 5 misses) when removing one highest (95%) and one lowest (77%). The failure cases are similar, as shown in yellow boxes in Figures 12 and 14, where original functionalities of individual parts were not replaced (play horse, or bus stop), or additional functionalities introduced that lead to ambiguities in judgement, e.g., sofa seats placed on a counter or bed placed within a shelf (see Figure 14). Nevertheless, the user study demonstrated that the variations mostly preserved the original designed functionalities, while being rather different geometrically.

Limitations. The main limitations of the algorithm is the reliance on good quality pre-segmented method. Another important limitation is that we hand picked interesting SFARR configurations in Section 3. In the future, we expect to automatically extract interesting SFARR-s directly by analyzing data collections. However, at this stage it remains a hypothesis that such consistent SFARR-s exist in real model collections and can be automatically extracted.

9. Conclusions and Future Work

We presented a simple reshuffling framework for non-trivial 3D model creation across different shape families. We exploit a particular form of compatible shape substructure, which we call *symmetry functional arrangements* (SFARR-s) to match, replace, and position triplets of parts to create plausible yet functional model variations. We demonstrated that such simple forms of SFARR-s can produce a rich set of novel shape variations, which are otherwise difficult to achieve with existing methods. We also validated the functional plausibility of the synthesized models via a user study.

We take a first step to relate shape functionalities to geometric substructures. In the future, we will like to further search and explore other functional substructures, especially in conjunction with basic physical constraints. This can lead to new model creation possibilities, especially towards fabrication-aware form-finding.

Acknowledgement. We thank the anonymous reviewers for their useful suggestions and Melinos Averkiou for carefully proofreading the paper. The work was partially supported by the Marie Curie Career Integration Grant 303541.

References

[ARSF09] ATTENE M., ROBBIANO F., SPAGNUOLO M., FALCIDIENO B.: Characterization of 3d shape parts for semantic annotation. *Comput. Aided Des. 41*, 10 (2009), 756–763. 1, 3, 4

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. In *ACM (Siggraph)* (2010), pp. 104:1–104:10. 2

[CK10] CHAUDHURI S., KOLTUN V.: Data-driven suggestions for creativity support in 3d modeling. *ACM Trans. Graph.* (2010), 183–183. 2

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3D modeling. *ACM Siggraph 30*, 4 (2011). 1, 2

[FCODS08] FU H., COHEN-OR D., DROR G., SHEFFER A.: Upright orientation of man-made objects. *ACM Trans. Graph. 27*, 3 (2008). 3, 4

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Siggraph* (2004). 1, 2

[Gre58] GREENOUGH H.: *Form and Function: Remarks on Art, Design and Architecture*. University of California Press, 1958. 3

[GSMCO09] GAL R., SORKINE O., MITRA N. J., COHEN-OR D.: iwires: An analyze-and-edit approach to shape manipulation. *ACM Siggraph 28*, 3 (2009), #33, 1–10. 3, 7

[JTRS12] JAIN A., THORMÄHLEN T., RITSCHEL T., SEIDEL H.-P.: Exploring shape variations by 3d-model decomposition and part-based recombination. *CGF EG 31*, 2 (2012). 3

[KCKK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D., KOLTUN V.: A Probabilistic Model of Component-Based Shape Synthesis. *ACM Transactions on Graphics 31*, 4 (2012). 1, 2, 7

[KJS07] KRAEVOY V., JULIUS D., SHEFFER A.: Shuffler: Model composition from interchangeable components. *In Proc. Pacific Graphics* (2007), 129–138. 1, 2

[KLM*12] KIM V. G., LI W., MITRA N. J., DIVERDI S., FUNKHOUSER T.: Exploring collections of 3d models using fuzzy correspondences. *ACM Transactions on Graphics 31*, 4 (2012), 54:1–54:11. 3

[KSSCO08] KRAEVOY V., SHEFFER A., SHAMIR A., COHEN-OR D.: Non-homogeneous resizing of complex models. *ACM Trans. Graph. 27*, 5 (2008), 111:1–111:9. 7

[MGP06] MITRA N. J., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH) 25*, 3 (2006), 560–568. 5

[MPWC12] MITRA N. J., PAULY M., WAND M., CEYLAN D.: Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report* (2012). 3

[OLGM11] OVSJANIKOV M., LI W., GUIBAS L., MITRA N. J.: Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics 30*, 4 (2011), 33:1–33:10. 3

[Sul96] SULLIVAN L.: The tall office building artistically considered. *Lippincott's Magazine 57* (1896). 3

[UIM12] UMETANI N., IGARASHI T., MITRA N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics (Siggraph) 31*, 4 (2012), 86. 8

[WAvK*12] WANG Y., ASAFI S., VAN KAICK O., ZHANG H., COHEN-OR D., CHEN B.: Active co-analysis of a set of shapes. *ACM Transactions on Graphics (Siggraph) 31*, 6 (2012). 1, 3, 4

[XZCOC12] XU K., ZHANG H., COHEN-OR D., CHEN B.: Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2012) 31*, 4 (2012), 57:1–57:10. 3, 4

[XZZ*] XU K., ZHENG H., ZHANG H., COHEN-OR D., LIU L., XIONG Y.: Photo-inspired model-driven 3d object modeling. *ACM Siggraph*. 3

[ZFCO*11] ZHENG Y., FU H., COHEN-OR D., AU O. K.-C., TAI C.-L.: Component-wise controllers for structure-preserving shape manipulation. *Comput. Graph. Forum 30*, 2 (2011), 563–572. 3, 4, 5