EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Replaceable Substructures for Efficient Part-Based Modeling

Han Liu[1] Ulysse Vimont[2] Michael Wand[3] Marie-Paule Cani[2] Stefanie Hahmann[2] Damien Rohmer[2] Niloy J. Mitra[4,1]

[1]KAUST [2]INRIA Grenoble [3]Universiteit Utrecht [4]University College London

Abstract

A popular mode of shape synthesis involves mixing and matching parts from different objects to form a coherent whole. The key challenge is to efficiently synthesize shape variations that are plausible, both locally and globally. A major obstacle is to assemble the objects with local consistency, i.e., all the connections between parts are valid with no dangling open connections. The combinatorial complexity of this problem limits existing methods in geometric and/or topological variations of the synthesized models. In this work, we introduce ***replaceable substructures*** *as arrangements of parts that can be interchanged while ensuring boundary consistency. The consistency information is extracted from part labels and connections in the original source models. We present a polynomial time algorithm that discovers such substructures by working on a dual of the original shape graph that encodes inter-part connectivity. We demonstrate the algorithm on a range of test examples producing plausible shape variations, both from a geometric and from a topological viewpoint.*

1. Introduction

Geometric content creation remains one of the central goals in shape modeling. However, creating shapes from scratch requires both raw creativity and (3D) modeling skills, and hence remains accessible to only a handful of experts. With the growing volumes of easily available shape repositories (e.g., Trimble warehouse, Turbosquid, etc.), content creation by combining existing models has emerged as an alternative, particularly for proposing new shapes to novice users.

One of the most popular choice for such data-driven shape synthesis is part-based modeling (recently surveyed in [MWZ*13]). The idea is to first split example geometry into smaller parts, which are subsequently copied and reassembled to obtain shape variations [FKS*04]. Plausibility is ensured by subjecting part decomposition and assembly to suitable constraints.

There are two main classes of constraints: *Local constraints* enforce geometric plausibility at a local level, i.e., all small, local neighborhoods of the synthesized shapes must look plausible, regardless of the overall shape. Data driven methods usually formulate this as geometric similarity of all local neighborhoods to portions of the example data. In the context of part-based modeling, this constrains the connections between parts, as exemplified in the input.

Local consistency is necessary but not sufficient, i.e., out of locally plausibly connected pieces, one can build composite shapes with unsuitable global structures. This is addressed by *global constraints*, which describe non-local correlations in part usage in order to further narrow down the space of synthesized shapes to a plausible subset. Both geometric approaches (e.g., [WXL*11]) as well as machine-learning methods (e.g., [FRS*12]) are employed to this end.

In this paper, we study the problem of guaranteeing local consistency in shapes. We also integrate simple global constraints to examine their impact on our approach. Despite its importance, analysis of even the local consistency problem in part-based modeling has received little attention. We start by building a theoretical framework for part-based shape modeling, describing models as *shape graphs* and reducing consistency conditions as pairwise constraints corresponding to the formal construct of a *tiling grammar*. As a

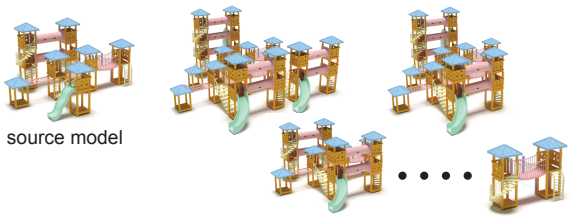


Figure 1: *Starting from part-based source models, we propose a polynomial time algorithm to synthesize plausible new models using* replaceable subgraphs.

negative result, we show that such a modeling problem is undecidable and even restricted variants, e.g., shape synthesis with limited number of pieces, are NP-hard.

A negative complexity result for a general model does yet exclude still being able to solve the in problem in a variety of practically relevant special cases. Our paper gives a positive result for such a subclass. We propose a suitable restriction of the search space that trades-off some variability for efficiency: As the main algorithmic contribution, we describe a novel algorithm that efficiently enumerates all *replaceable substructures* that lead to shape modification operations. Rather than synthesizing new shapes from scratch (permitting arbitrary part combinations), the algorithm enumerates all possible replacements of subgraphs of an *existing* shape graph (thereby restricting the search space), which, surprisingly, can be achieved in low-order polynomial time under mild restrictions. Specifically, we obtain a quadratic time algorithm in two steps: (i) We assume a local ordering of outgoing edges in each node, which is trivial for manifold models. (ii) We factor out an exponential overhead of redundant solutions with equivalent effect. We improve practical performance further by working with the dual of the original shape graphs to encodes inter-part connectivity.

We extend our method to also take into account *global constraints*. With our current implementation, we examine a small set of demonstrator examples such as topological constraints to ensure routes between entry-exit points, geometric constraints to ensure stability and functional validity of models. Here, the large class of candidate variations considered at once by our approach permits a more efficient discovery of solutions than a naïve part-by-part assembly strategy.

We validate our conceptual approach by building a prototype system for part-based shape modeling (see supplementary demo and video). General types of parts and their permitted connectivity are user-annotated through a simple surface coloring interface, local and global graph variations are computed, and then the part-synthesized graphs are embedded as 3D models. The user can select among multiple feasible subgraph replacements, with shape variations being proposed in real-time, and iterate the process (see Figure 1). We apply our system to a range of different models and explore examples of global constraints, which are motivated by the scenes considered.

2. Related Work

In the recent years, significant progress has been made towards analysis and synthesis of shapes using collections of 3D models, particularly man-made objects.

Part-based modeling. Motivated by the availability of 3D model collections, the interactive modeling-by-example framework [FKS*04] proposed to combine parts from various models to synthesize new models, instead of modeling from scratch. However, the users were expected to assist in the tasks of establishing semantic part correspondence, placement of potential candidate parts, and geometric adaptation of the parts. Various refinements have subsequently been proposed. For example, [KJS07] uses contact-graphs to determine interchangeable part candidates, [SI07, LF08] use 2D sketches to assist in part retrieval, [SBSCO06] uses geometric snapping to create plausible part connections, while [CK10] employ geometric matching to retrieve components for ill-defined shapes.

In another approach, [MM08] poses the part-based model synthesis problem as an instance of MRF labeling. However, they work under the assumption at node of graphs already exists and the task is to determine suitable labels, under neighborhood constraints. The approach was later extended to handle global constraints [TYK*12, YBY*13]. In contrast, our method synthesizes the domain of the labeling function of the MRF itself, which is a harder problem. Previous methods fix the domain of the MRF by an a priori discretization of space into fixed cells. This constraints geometric variability to subsets of regular spatial subdivisions. Our graph-modification approach does not have this restriction, and this advantage is our main motivation.

[CKGK11, KCKK12] improve parts selection by a probabilistic model that incorporates geometric style and semantics to select relevant components and to automate plausible synthesis. These methods mainly focus on replacing one part at a time and keep the part-level model topology fixed. In contrast, we focus on replacing substructures with which we obtain variations in both geometry and topology.

Texture synthesis. In the context of images, texture synthesis has emerged as a powerful approach to create content from examples [KSE*03]. The approach has been extended to model deformations of organically shaped meshes with details, where details on the surface duplicate rather than stretch when the mesh is deformed [AZL12]. Details of the mesh can be generated from example-based textures [BIT04], automatically extracted from an input mesh [ZTS09], smartly annotated [GTB14], or encoded as parametric structures [LS10]. Although texture synthesis can handle generic surface details and can even adapt to temporal continuity [BBT09], their extensions to handle structured 3D shapes structured shapes are not obvious.

Inverse-procedural modeling. Shapes can be synthesized by (i) inferring building rules directly from an input 3D model, and (ii) using the grammar to synthesize new variations. One option is to detect and exploit partial symmetry to automatically find rules for procedural modeling that guarantee local consistency. Bokeloh et al. [BWS10] detect r-symmetric geometry as *docking sites* and replace the differing parts with each other. A generalization, slippable dockers [BWKS11], provide an option for pattern-aware free-form shape editing. Kalojanov et al. [KBW*12] consider all boundaries of partial r-symmetry simultaneously to span the space of all locally similar shapes. More recently, for con-

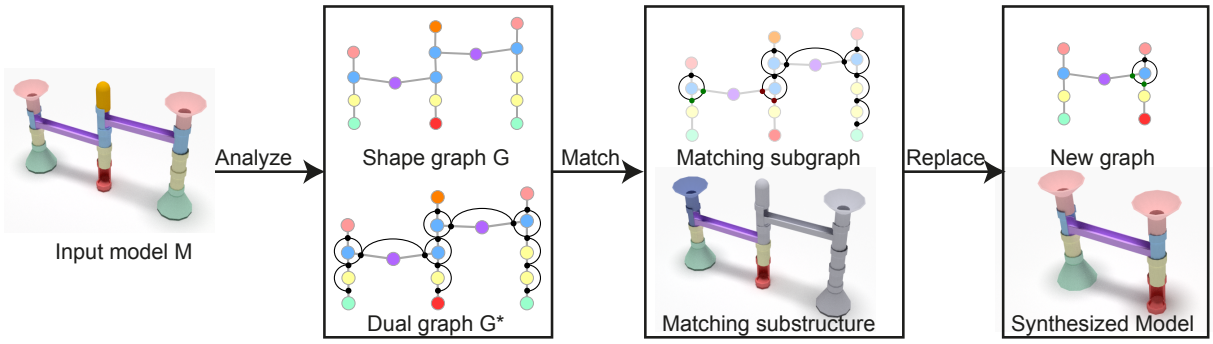


Figure 2: *Pipeline: The input is one or more segmented 3D models, with parts of related functionality labeled consistently as shown in the shape(dual) graph. We propose a polynomial time algorithm to enumerate all the matching subgraphs that could be replaced at alternative locations in the model to create plausible shape variations (see Figure 4 and Section 4.4 for details).*

tinuous geometry synthesis, [DK14] propose geometry seam carving to to duplicate self-similar details on a shape.

In this work, we look beyond regular patterns. At an abstract level, our algorithm is a generalization of the grammar extraction algorithm of Bokeloh et al. [BWS10]. There are three key improvements: First, our method is not restricted to rigid pieces. We take a topological point of view, permitting corresponding parts to vary strongly in geometry. Second, our dynamic subgraph replacement permits, when iterated, more general topological variations than iterated applications of rules from precomputed context-free shape grammars. Third, we introduce global constraints that cannot be captured by the earlier static framework. Our paper explains why such an approach is intractable and proposes a practical alternative by constraining the problem suitably.

Global constraints. Generating complex shapes plausibly often requires additional global constraints. Wang et al. [WXL*11] construct a symmetry hierarchy that supports structure-aware editing by changing the symmetry parameters while holding the part-level topology fixed. Xu et al. [XZCOC12] propose mixing and matching parts using a genetic algorithm to *evolve* novel and interesting shape variations. Zheng et al. [ZCOM13] search for pre-authored part substructures to create functionally plausible novel variations. Our paper studies several examples of how such global constraints can be integrated in each step of our iterative graph modification algorithm.

In another thread, Jain et al. [JTRS12] propose symmetry-structure-guided morphing between shape parts to create shape in-betweens, while more recently skeleton graphs have been used to create interesting and plausible in-between shapes with topological variations [ALX*14]. In contrast to blending methods, we focus on part assembly with formal guarantees on local consistency.

3. Part-Based Modeling

In this section, we analyze the structure of part-based modeling and analyze the complexity implications arising from local consistency constraints.

3.1. Parts and Tiling Grammars

First, we describe part-based modeling by an abstract framework that clearly identifies the combinatorial challenges.

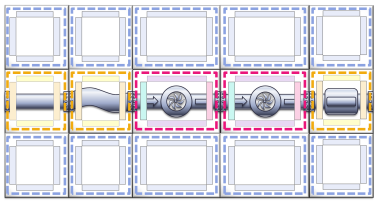
Data-driven part-based modeling typically starts by segmenting the input geometry into disjoint parts. These are later instantiated (copied, potentially multiple times), transformed (rigidly or with some deformation), and reassembled to create shape variations. The different approaches [MWZ*13] mainly differ in how the segmentation is performed, and the allowed types of transformations and constraints.

Shape graphs. We abstract an input model M by a purely topological description, a *shape graph* $G := (V, E)$ that encodes parts as nodes connected by edges to capture the coarse-scale model topology.

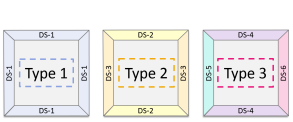
Parts. We assume a segmentation of the input geometry into a finite number of disjoint, connected regions that we call *parts*. Parts are the nodes V in our shape graph. Importantly, each part $v \in V$ has an associated *type* $\tau(v)$. The part designation can be a geometric property (for example, rigid copies [BWS10]) or a purely semantic property based on

Table 1: *Notation used in this paper.*

Symbol	Description
M	model shape
$G := (V, E)$	shape graph with part and part-relation sets
$\tau(\cdot)$	node type of a part in V
k	maximum node degree
$G^* := (V^*, E^*)$	dual graph of G with node and edge sets
dE^*	dummy edges in G^*



(a) Annotated example data.



(b) Learned part types.

(c) Learned docking rules.

Figure 3: *A simple example for learning docking rules. (a) The original input data is annotated with parts and docking sites. (b) This yields parts of different types and (c) rules for connecting them, visualized as a compatibility matrix.*

functionality. We assume that such segmentation and annotation are given. Our prototype system uses a simple interactive tool to "color" presegmented mesh parts to indicate matching part types, without constraints on geometric variability (see supplementary video).

Docking sites. Parts are connected to each other through *docking sites* linked by graph edges E. As an important consistency condition, we demand that *parts of the same type must have the same configuration of docking sites*. Each docking site on a part type specifies a list type-compatible docking sites on other parts to connect to.

Tiling grammars. Our rule system can be interpreted as formal language with a specific, context-sensitive use of non-terminals. The parts are the terminals, the docking sites are non-terminals, and the rules require non-terminals to match for valid assembly. We call this formalism *tiling grammars* (restricted 2D variants have previously been described as "tiling systems" and "tile rewriting grammars" [RP03]). Tiling grammars are a natural formalism for encoding local consistency constraints in part-based modeling.

In our interactive annotation setup, we allow users to define the shape grammar by defining suitable connector parts. Simply coloring the boundary region as parts of the same types makes them connectible.

3.2. Complexity of Locally Consistent Synthesis

The key problem is to build a whole consistently connected graph that describes a synthesized shape while abiding by the observed grammar rules.

Complexity. How difficult is this? Unlike context-free shape grammars [SG71, MWH*06], tiling grammars specify only local constraints on how pieces can be glued together, thus permitting context-sensitive rules. While traditional, context free grammars permit efficient algorithms for sampling valid

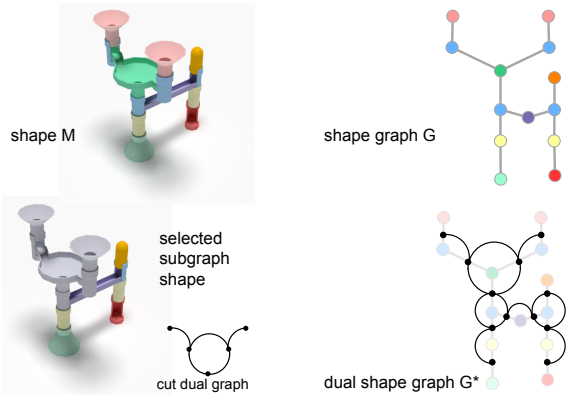


Figure 4: *From a shape M with annotated part types (indicated by color) we construct a shape graph G where each part becomes a node and each part-connection an edge. The dual graph G^* encodes how a set of parts can be removed to extract a subgraph. In this example (bottom-left), the cut is formed by removing two nodes from the dual graph G^*.*

scene variations [BWS10], the additional expressiveness of a tiling grammar comes at a price: Related problems have been studied extensively in theoretical computer science [RP03], and a number of negative results are known.

Negative result. Rule systems with parts and local connection rules can have arbitrarily complex behavior. As an well-known example, Penrose tilings create globally aperiodic patterns in the 2D plane with as few as two simple polygons as tile types and carefully chose connection rules. In general, tiling grammars have been shown to be Turing-capable, even in very restrictive settings (e.g., Wang tiling). Being able to encode any Turing computation, tiling grammars fall under Rice's theorem [Hop79], which implies that any non-trivial property of the grammar becomes undecidable, including just finding a single, closed, consistent graph. The problem becomes decidable if we permit only a finite number of tiles to be used, but even this restricted variant is NP-hard [DD07]. Our 3D tiling grammar is able to express the 2D tilings as special cases, and the same restrictions apply.

4. Discovering Replaceable Substructures

Now, we present an efficient algorithm for finding locally consistent assemblies of parts under suitable restrictions. We first introduce the conceptual idea and discuss grammar consistent graph operations. Afterwards, we make it asymptotically efficient by avoiding redundancy and imposing local ordering constraints. Finally, we make the algorithm more concise and practically faster by operating on the dual G^* of graph G (see Figure 4).

4.1. Conceptual Overview

Rather than synthesizing graphs from scratch, we restrict our consideration to *modifications* of existing graphs. Here, the

input example provides a starting point to not only learn the rules, but also learn how to assemble the shape graphs.

Graph operations. We find pairs of two different subgraphs that can be replaced with each other. In other words, *replaceable subgraphs* are characterized as pairs of corresponding cuts that still respect all of the learned rules when the interior of one is exchanged with the interior of another. We call the resultant operations *(grammar-consistent) graph operations*.

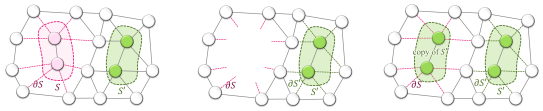
Fast search for graph operations. A naïve search for replaceable subgraphs still has exponential runtime costs. Specifically, symmetry in this set of cuts that leads to an artificial, exponential blow-up of possible variations. By factoring these symmetries out, along with local ordering assumption, we design an algorithm to enumerate all the possible graph modification operations in polynomial time. Specifically, for graphs of n nodes, m edges, and maximum node degree $k \in \mathcal{O}(1)$, there will be no more than $\mathcal{O}(k \cdot n^2) = \mathcal{O}(n^2)$ *different* operations. The cut pairs leading to different operations can be directly enumerated and each can be computed in linear, $\mathcal{O}(n+m)$ time, yielding a cubic worst-case bound for bounded k. A simple hashing argument reduces this further to $\mathcal{O}(k \cdot n \cdot (n+m)) = \mathcal{O}(n^2)$, i.e., quadratic time.

Dual graph algorithm. The search algorithm can be more conveniently formulated using dual shape graphs. The asymptotic complexity then changes only slightly to $\mathcal{O}(k \cdot m^2)$ trials of $\mathcal{O}(k \cdot m)$ costs. For $k \in \mathcal{O}(1)$, we also obtain a worst-case cubic bound. However, the practical performance is better due to early termination of the step that extracts operations in worst-case linear time.

Robustness to real-world imperfections. Finally, we extend our algorithm to make it *robust* against a small number of defects in the connection through *dummy edges* in the dual graph. The modification helps to recover from mistakes in user annotation or imperfect data (for example, an imperfect mesh segmentation). Such robust graph operations still maintain plausibility, as shown in our tests. The explanation for this observation is that a long cut with very sparse violations is still unlikely to be observed by chance.

Global constraints. We further validate the generated graph variations against violations of global constraints, which cannot be directly encoded in a tiling grammar. As we have a larger collection of large scale modifications, a filter that tests of each option is still effective unless the constraints are very restrictive. See Section 5.3 for details.

Geometric embedding. Finally, we convert the modified graphs back to geometry by instantiating corresponding geometrical parts for each graph node. We formulate an optimization problem for placing these elements rigidly with minimal stretch, and use linear blend-skinning to create the final composite. We take the global constraints into account in the geometric embedding step.



(a) subgraph match (b) remove S (c) insert S'

Figure 5: *Graph operations modify existing shape graphs by replacing matching subgraphs iteratively.*

4.2. Grammar Consistent Graph Operations

We perform a graph operation in three steps (see Figure 5). First, we extract a subgraph S from a shape graph G by cutting the necessary edges as to fully disconnect S from the remainder graph R. We call the set of edges cut ∂S. Subgraph S will be replaced and can be discarded. Second, we find a replacement S' that is also a subgraph of the original graph G. Similar to S, S' is also separated from the remainder graph R' by cutting all edges $\partial S'$ that would connect S' and R'. Finally, we copy S' into the hole left by S, reconnecting it with R along the boundary. Importantly, S' is not removed from the graph but just copied. As it is copied from the unmodified original shape graph G, S and S' are also allowed to overlap. Figure 2 shows an example replacement.

Grammar consistency. S may only be replaced by S' if the rules of the shape grammar are satisfied. In particular, the number of edges along the cuts (boundaries of S and S') must match, i.e., $|\partial S| = |\partial S'|$, and all node reconnections along the boundary must have been observed in the example graph.

Remark. The relation between S and S' determines the nature of the operation. If $S' \subset S$, we have found a *recursive* rule to arbitrarily extend G with regular copies of $S \setminus S'$. By switching S and S', we can also shrink the graph until the regularity vanishes (subgraphs S and S' can always be exchanged because graph edges are undirected). If $S' \not\subset S$ and $S \not\subset S'$, the rule is not recursive and cannot be trivially repeated. When performing more than one graph operation, we always recompute all matching cuts dynamically. Hence, more complex rules can arise. This is an important structural difference to [BWS10] who precompute a context free grammar by enumerating non-intersecting cuts. Our method permits arbitrary sequences of cuts, reconsidering all options every time, thereby generating a richer set of graph variants.

4.3. Fast Search For Replaceable Subgraphs

While the idea of using matching cuts is straightforward, it still has a serious complexity problem. There are two main challenges: First, the number of matching pairs of cuts can be exponential and thus cannot be enumerated naïvely. Second, general subgraph matching itself is NP-hard.

4.3.1. Polynomial Number of Graph Operations

Even with very simple graphs, we might obtain an exponentially large number of matching cut pairs. Consider for

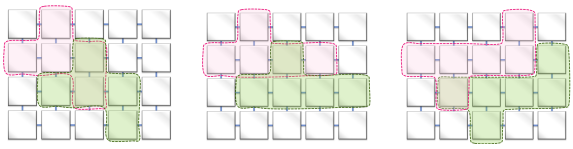


Figure 6: *Even simple graphs (here: regular grid of identical parts) can have an exponential number of matching cuts. However, due to symmetry, all but an $\mathcal{O}(k \cdot n^2)$-sized subset have the same effect.*

example a 2D grid of squares where all four sides can be mutually connected (see Figure 6). In each subfigure, a pair of matching cuts and substructures are represented by different colors (red and green); each pair have the same shape of boundaries in the grid example, so swapping them does not result in any change. Rather than enumerating all the matching cuts, we create equivalence classes of cuts that have the same effect on the graph and enumerate only one representative from each class.

Equivalent operations. The counter-example also gives a hint on how to avoid redundancy. Given an operation that replaces S by S', we can convert it into an equivalent operation by including additional pairs of nodes along the boundary of S and S' (to be replaced with each other) that *both have the same type*. Leaving out such a matching node pair or not does not affect the result. We designate a *normalized,* representative graph operation out of the whole equivalence class by just leaving out all of such ineffective replacements.

Partial graph symmetry. The normalization above can be understood by looking at partial symmetries of the graph: A *partial graph symmetry* is node-wise matching between two subgraphs, matching (i) only nodes of the same type, and (ii) preserving the graph topology within matched regions.

We can characterize all graph operations as the complement of all maximal partial symmetries (with respect to set inclusion): We enumerate all partial graph matchings (always with the maximal sets of nodes matched), and obtain the shape operations as pairs of cuts along the boundaries towards the unmatchable regions (including boundaries). These cut-out regions are always replaceable because the connections between the nodes along the cut have, by definition, been observed in the exemplar geometry. This also holds for iterated application of shape operations, as this makes grammar consistency an invariant of the process.

4.3.2. Efficient Subgraph Matching

The above method still requires solving the subgraph matching problem, which is NP-hard in general. We make a small restriction and introduce a *local ordering constraint*. First, we consider manifold inputs, where the ordering constraint is implicitly given by the manifold structure and graph matching is always a polynomial problem.

Manifold training data. We only allow matches that preserve *ordering* of the docking sites. The ordering is trivially induced by manifold data: Recall that parts on a surface (a triangle mesh) a obtained by segmentation into disjoint, connected regions. On a manifold surface, each part boundary can only be neighboring at most two parts. Hence, docking sites are simple curve segments along the part boundaries, and the boundary curve of a cut is a simple curve, consisting of one connected component without intersection (see Figure 5). In this case, the ordering is prescribed by the ordering along the boundary curve, and similar to planar graphs, maximal subgraph matching can be efficiently performed.

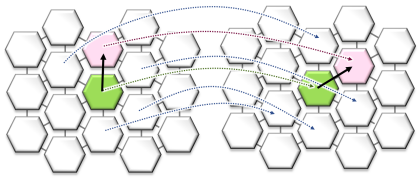
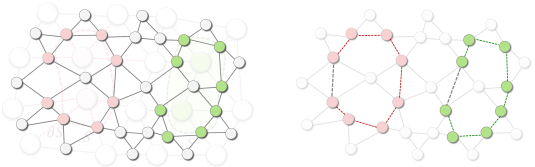


Figure 7: *A partial graph symmetry is fixed by one node and one edge assignment.*

Complexity manifold graph matching. We start by picking a pair of matching nodes of the same type. There are at most $\mathcal{O}(n^2)$ such pairs, n denoting the number of nodes. For each such pick, we fix a permissible correspondence between two docking sites, i.e., the "rotation" and "translation" of the match. There are at most $k \ll n$ such options with k being the maximum degree of nodes in G, see Figure 7. This choice uniquely determines the maximal subset that can be matched: We just perform region growing into adjacent nodes as connected by their docking sites. If the node types are the same, the pair can be included in the match $(S, f(S))$ and this cannot interfere with other choices. If it does not match or if we encounter a node that has already been matched previously, we can stop region growing. Overall, we obtain at most $\mathcal{O}(n^2 \cdot k)$ matching pairs of cuts, each of which can be computed in $\mathcal{O}(n+m)$ time with m edges in the graph. This yields a third order worst-case bound assuming $k = \mathcal{O}(1)$.

Improved bound and practical efficiency. The worst-case estimate is still too pessimistic. With a small modification, we obtain a theoretical *quadratic* bound by using the fact that at most $\mathcal{O}(k) = \mathcal{O}(1)$ overlapping, distinct matches because overlapping maximal partial symmetries must differ in their "rotation" of the docking sites against each other. Otherwise, they would be identical. Hence, we use a simple hash table to avoid rematching subgraphs multiple times that have different starting pairs of nodes. Then the linear matching costs cannot be accumulated more than $\mathcal{O}(n \cdot k)$-times. This reduces the worst-case compute time to $\mathcal{O}(n \cdot k \cdot (m+n)) = \mathcal{O}(n^2)$. In practice, large scale matches are rather unlikely, such that we might even often expect linear scaling behavior, permitting interactive applications.

Non-manifold data. In practice, we want to utilize our method on general triangle soup, as real-world shape repos-



(a) dual graph (b) dual graph cutting

Figure 8: *Manifold-like data is created by ordering docking sites (see Figure 5). In a dual graph, matching cuts is performed by node propagation.*

itories rarely offer clean manifold meshes. In this case, we make a small adaptation. The key assumption that leads to polynomial matching costs is the fixed ordering of the outgoing edges. Hence, we impose a pseudo-manifold ordering for general meshes, as explained next.

For each part in the graph, we consider all directly connected parts. We compute the centroids of all of these and determine a least-squares fitting plane using a local PCA analysis. The outgoing edges (vectors between centroids) are then ordered by projecting them into the plane and ordering the projected edges by increasing angle. A manifold-like structure can then be created like the dual graph in Figure 8 where dual nodes are obtained by such local ordering and dual edges simulates the traversal of half edges. The initial orientation of the plane is uniquely determined during matching such that the incoming edge, the plane normal, and the next edge form a right-handed coordinate system.

4.4. Simultaneous Dual-Graph Cutting

In practice, we compute the matching cuts directly using the dual graph G^* of the shape graph. The dual-domain algorithm is faster in practice as it avoids explicitly detecting symmetry through flood-filling but only walks along boundaries. Further, we can easily extend it to make it robust to small grammatical defects.

Dual graph. We consider the dual G^* of the shape graph G, i.e., the edges in G form the nodes in G^* and vice versa (Figure 4). We match pairs of nodes in G^* (i.e., edges of G) if they are compatible with respect to the shape grammar. We simply walk along a path in the dual graph that contains only matching edges (Figure 8). Further, we avoid mismatching dual nodes. Dual nodes are trivially matched if they connect the same types of nodes in the original graph G. We enumerate all the cuts by starting an exhaustive depth-first search at each pair of dual nodes that can be non-trivially matched. Enumerating these cuts is equivalent to finding the boundaries of partial symmetry in the primal graph algorithm and therefore the same worst-case complexity bounds hold.

Robust matching. Exact matching can fail in practice due to imperfect input, e.g., inconsistent annotation, imprecise segmentation, small shape variations. In particular, we have observed in our experiments that the ordering of non-manifold

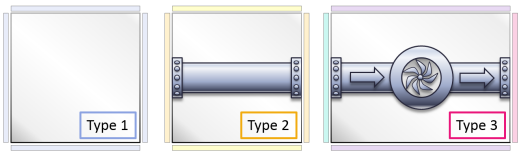


Figure 9: *Annotations of docking sites: By sharing labels, symmetry can be expressed. In type 1, all docking sites are equivalent. In type 2, left-right docking sites differ semantically from top-bottom ones. Type 3 finally also distinguishes between left and right.*

graphs using PCA-based tangent-plane projection is not always reliable. We therefore permit limited violations of the ordering by introducing *dummy edges* that cover sparse outliers. We add a potential dummy edge to the dual graph between all nodes with a graph distance of two (nodes with distance one directly connected), irrespective of grammatical constraints, thereby short-cutting over local defects. We minimize the number of violations by exhaustively searching graphs with up to $q \in \{0,1,2,3\dots\}$ violations. This search is exponential in q. In our experiments, we used $q \leq 2$.

5. Implementation Details

5.1. Interactive Labeling

Our framework expects meshes with annotated part types as input. For our experiments, we build a simple system where the user manually annotates parts. We use presegmented mesh models and the user can select segments and "paint" them in a specific color to designate a part type.

Tiling grammars are learned from color alone. We do not explicitly distinguish multiple corresponding docking sites on the same type but additional intermediate pieces with varying colors must be created manually in order to constrain with respect to distinguishable docking sites, as described in the formal model. This step could possibly be supported by novel surface texel extraction methods [HGM14].

5.2. Assembling Models

We now present a method to embed concrete geometry from synthesized shape graph variations.

Partial merging of two models. We fuse geometry of subgraphs from two input graphs $G = (V,E)$ and $G' = (V',E')$. Operation within a single graph work analogously, by setting $G = G'$. Let $S \subset G, S' \subset G'$ denote the corresponding replaceable subgraphs. We denote their boundaries by $\partial S \subset E$ and $\partial S' \subset E'$, respectively.

We compute a non-linear mapping $\mathcal{T}$ that fits the geometry of S into that of S'. To this end, we equip each edge in G and G' with an orthonormal 3D frame. For boundary edges of ∂S that have a corresponding edge $\partial S'$, the corresponding frames yield rigid transformations. Transformation $\mathcal{T}$ is obtained by interpolating the boundary transformations.

Boundary frames. We first fix a frame and use simple heuristic. For an edge $e = (v, w)$, we use the center of the intersection of the two bounding boxes of the parts $v, w \in V$ as origin, then orient the x axis towards the center of part w, and orient the z axis upwards in world coordinates. If desired, the user can manually refine the result.

Interpolation. Subsequently, we obtain the final mapping $\mathcal{T}$ as a per-vertex linear interpolation of those transformations arising from matching corresponding frames, resulting in a globally non-rigid transformation. Interpolation weights within S are obtained as inverse distance between the center of the part and the origin of the corresponding frame. In addition, the user can make manual annotations to choose whether parts should be rigidly transformed according to these coefficients, or the coefficients to be per-vertex interpolated within the part (non-rigid mapping); see Figure 10.

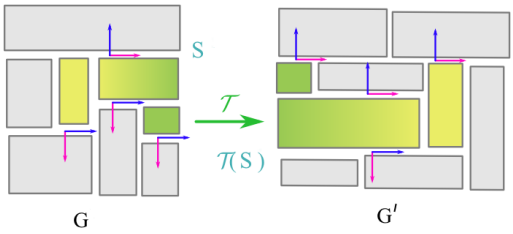


Figure 10: *Non uniform interpolation weighting across parts and inside parts create non-rigid transformations.*

5.3. Global Constraints

We verify global constraints after the embedding step, thereby permitting the algorithm to avoid constraint violations. We support the following simple constraints.

Support constraint ensures that the structure is well grounded (see Figure 14). We detect the horizontal support relation between neighboring nodes by fitting a plane to their contact area and check whether the normal is parallel to the upright vector. Then, the heights of nodes can be calculated starting from ground touching nodes, as well as the edges. We compare the height difference between two consecutive edges in matching cuts, and filter the failure cases, i.e., a lifted ground touching node, or a tilted substructure.

Path constraint ensures that connected paths exist from every entry to at least one exit, and are traversable under gravity (see Figure 14). We verify it in two steps: First, we compute a path by traversing the graph and then reject examples that do not have any path from an entry to exits. Second, we check whether the path is geometrically monotonically decreasing in height. At this point, the embedding can still be modified by non-uniform resizing of the parts thereby changing the heights. We perform greedy resizing along the path and reject the whole graph operation if no valid solution is found (see Figure 11).

Other constraints can also be supported. For example, we check orientation by comparing the angle between two consecutive edges in two matching cuts, to avoid distortion in geometry assembly, e.g., two parallel nodes will not be replaced by two perpendicular nodes. This could be modeled by more complex local rules, but a global validation is easier and efficient in pruning invalid results.

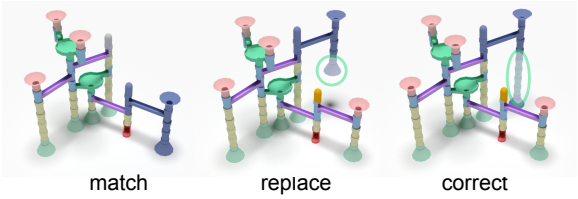


Figure 11: *Synthesized models, if found to be geometrically invalid, can be corrected by adapting the model parts subject to the geometric constraints. In this example, the exit part is anisotropically rescaled to allow it to be properly grounded.*

6. Results and Discussion

We tested the proposed algorithms on different families of models, including toy tractors, bikes, balltracks, racetracks, castles, and playgrounds. The scale of the graphs ranged from 5-70 nodes, and expanded to hundreds of nodes after progressive subgraph replacement. Both the search and geometric embeddings were at interactive rates.

Preprocessing. An input model is first annotated(colored) using our interactive tool, and abstracted by bounding boxes so that users can further group functionally related parts. The system then generates corresponding shape graphs based on geometric contacts among the mesh parts, and thereafter builds dual graphs used for discovering replaceable subgraphs. We support two main modes of replacements:

(i) In-model synthesis. For models with partial repetitive structures like playgrounds, racetracks, and ball tracks, replaceable subgraphs can be explored in just a single model.

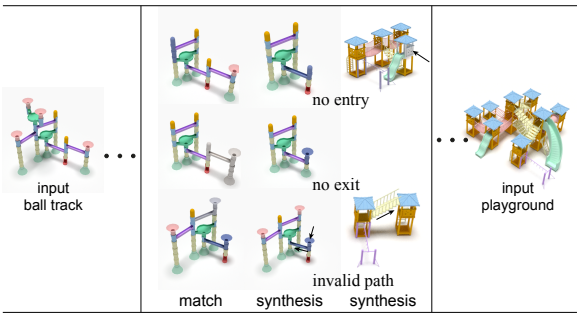


Figure 12: *Various synthesis results marked as invalid due to violation of global constraints (e.g., no entry, no exit, or no connections between entry-exit).*

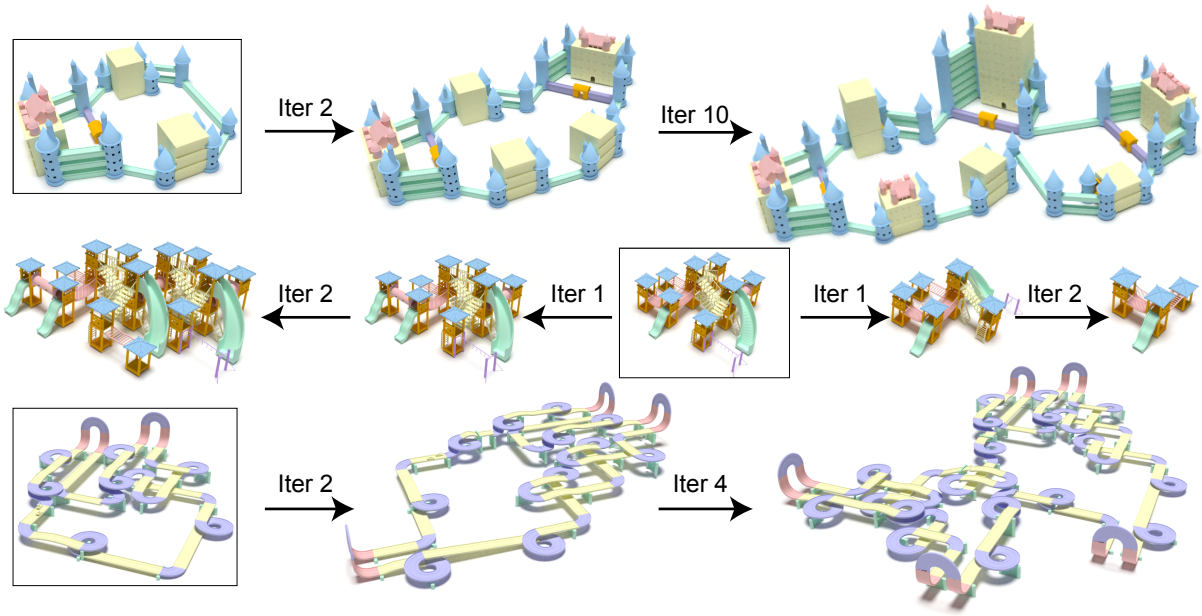


Figure 13: *Starting from single castle, playground, racetrack models, matching subgraphs are progressively found and replaced.*

Hence, we progressively replace substructures from a source model. Figure 13 presents three kinds of models: castle, playground, and race track. As the replacement of two matching subgraphs is bilateral, therefore, we can either choose to use a large subgraph to replace a smaller one or viceversa. This leads to interesting variations as shown by the inflated and the deflated playground model.

(ii) In-group model synthesis. Matching subgraphs can also be discovered and swapped between different models. In Figure 14, starting from two source models, the synthesized models are inserted as new inputs for further variation.

We obtain a variety of plausible shapes with varied topologies as well as geometries, as illustrated by the synthesized bike models in Figure 15, such as the two-seats tricycle resulted from replacing a subgraph from the tandem to the tricycle while preserving the geometry of the tricycle. The geometry can also be replaced along with the subgraph. We keep the non trivial results, i.e., those with topological and geometric changes from source models, while filtering out the trivial ones discussed in Section 4.4. Note that some of the cuts cannot be formed by only using the adjacent edge propagation from the shape graph. Here, dummy edges are used (maximum length 2) to expand the solution space, at the cost of slightly increasing the computational time.

Validity. The topological validity of synthesized models are preserved as the match-and-replace process respects the observed connection rules. We implemented functional validity according to the global geometric constraints as described in section 5.3. For example, from any labeled entry node in a balltrack model, there always exist at least one path leading to an exit node (see supplementary video), and such functionality is similarly proved for playground models.

Performance. On a 2.9GHz laptop with 4GB RAM, it took up to 3s to enumerate all the matching subgraphs across our tests. The geometry embedding is then performed at interactive rates.

Limitations. Although the graph can be extracted automatically, manual edits are sometimes necessary to correct obvious errors. For example, if the meshes of two perpendicular bridges attached to the same platform intersect, a wrong edge connecting these two nodes will be created and the graph will become complicated involving a set of meaningless edges. Therefore, input models are required to be clean at least along boundaries of mesh parts.

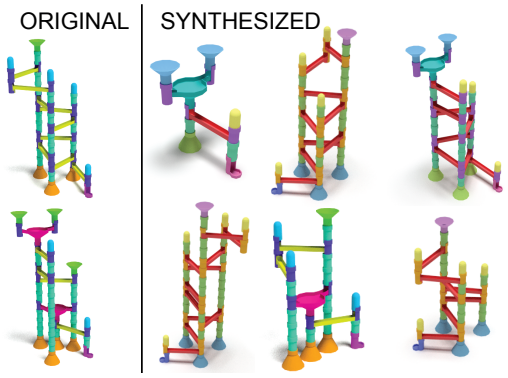


Figure 14: *Starting ball track models, replaceable subgraphs result in plausible synthesis results (see video).*

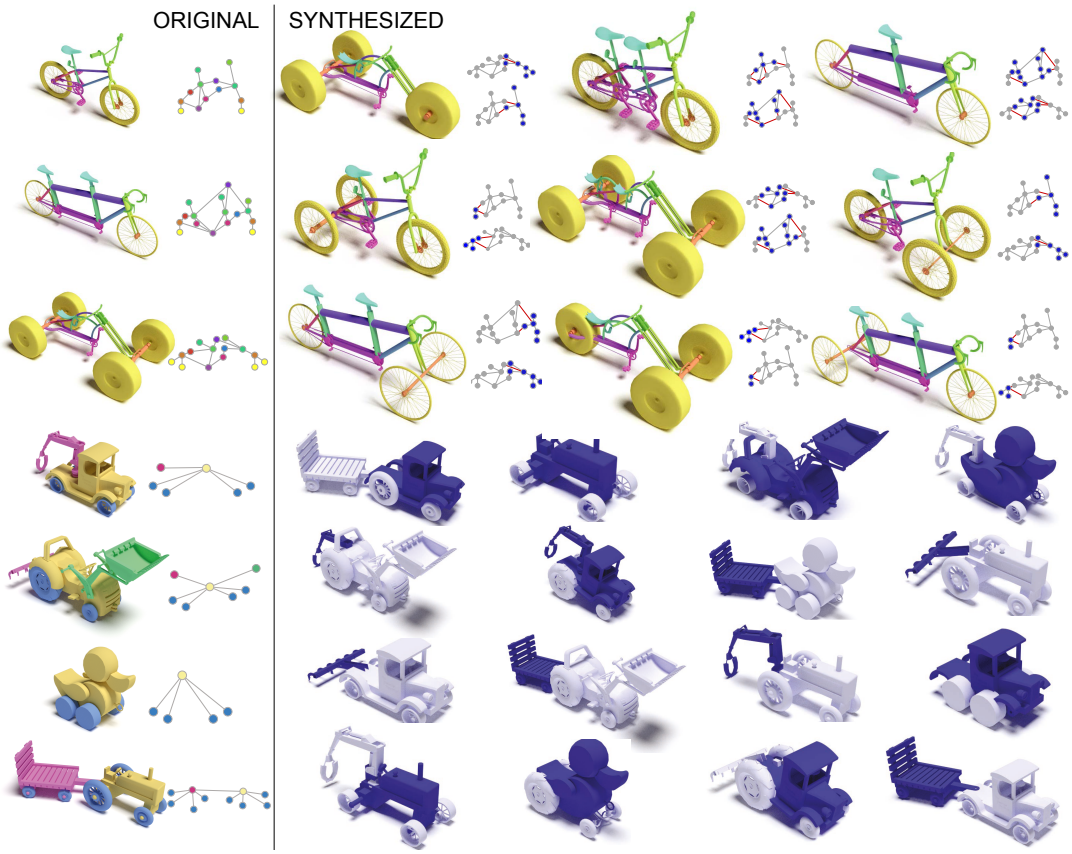


Figure 15: *Synthesized bikes created from only 3 bike models, with 13, 15, and 16 parts, respectively. We show two graphs and associated highlight matching cuts/subgraphs on the right of each synthesized bike, the subgraph in the upper graph get replaced by the matching subgraph in the lower one. The synthesized results of toy tractors were created from 4 input models.*

Further, interpolating the boundary transformations does not always yield good results. For more plausible geometry, the user can specify the desired behavior of deformation for the different node types of the shape graph (rigid or non-rigid). However, distortions happen when attempting to squeeze a very large substructure to a small space, such as the too close seats in the synthesized tandem bicycle.

7. Conclusion and Future Work

We introduced *replaceable substructures* for part-based synthesis of novel shapes. Beside theoretical characterization of the problem, a key contribution is a polynomial time algorithm to compute all possible replacements of subgraphs. Our approach circumvents the undecidable problem of unconstrained instantiation of tiling grammars by restricting operations to replacements of contiguous subgraphs at a time. Further, ordering constraints and factoring out symmetry avoids the exponential costs of subgraph isomorphism. For graphs with n nodes, m edges, and maximum degree k, we obtain at most $\mathcal{O}(n^2 \cdot k) = \mathcal{O}(n^2)$ complexity computed altogether in worst-case $\mathcal{O}(k \cdot n \cdot (n+m)) = \mathcal{O}(n^2)$ time.

The algorithm operates on dual shape graphs to simplify the discovery of consistent replaceable subgraphs that can then be geometrically realized in an deformation inducing embedding. Additionally, the proposed dummy edges enables model variations for imperfect inputs. We used the algorithm to create non-trivial and semantically plausible models with significant topological and geometric variations at a scale not previously demonstrated (involving hundreds of nodes, instead of tens of nodes, in order of a few seconds).

In the future, we would like to automatically order the replacement suggestions proposed by the algorithm. One approach would be to use the necessary deformation energy to predict plausibility. As further direction of research, it would be interesting to combine the segmentation and the synthesis more tightly. For example, we can hypothesize various segmentations and optimize for simple decompositions into few parts that still results in many model variations. This can, in turn, lead to an information theoretic measure for quantifying part-level redundancy within shape families.

Acknowledgements

We are grateful to the anonymous reviewers for their comments, suggestions, and additional references. We would like to thank Alexander Berner, Martin Bokeloh, Oliver Burghard, Javor Kalojanov and Youyi Zheng for discussions. The project was supported in part by the Marie Curie Career Integration Grant 303541, the ERC Starting Grant SmartGeometry (StG-2013- 335373), the ERC Advanced Grant Expressive (ADG-2011 291184), and gifts from Adobe.

References

[ALX*14] ALHASHIM I., LI H., XU K., CAO J., MA R., ZHANG H.: Topology-varying 3d shape creation via structural blending. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2014) 33* (2014). 3

[AZL12] ALHASHIM I., ZHANG H., LIU L.: Detail-replicating shape stretching. *The Visual Computer 28* (2012), 1153–1166. 2

[BBT09] BÉNARD P., BOUSSEAU A., THOLLOT J.: Dynamic solid textures for real-time coherent stylization. In *Proc. I3D* (2009), pp. 121–127. 2

[BIT04] BHAT P., INGRAM S., TURK G.: Geometric texture synthesis by example. In *Proc. SGP* (2004), pp. 41–44. 2

[BWKS11] BOKELOH M., WAND M., KOLTUN V., SEIDEL H.-P.: Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics 30*, 6 (2011). 2

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29* (July 2010), 104:1–104:10. 2, 3, 4, 5

[CK10] CHAUDHURI S., KOLTUN V.: Data-driven suggestions for creativity support in 3d modeling. *ACM Trans. Graph. 29*, 6 (2010), 183. 2

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

[DD07] DEMAINE E. D., DEMAINE M. L.: Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics* (June 2007), 195–208. 4

[DK14] DEKKERS E., KOBBELT L.: Geometry seam carving. *Computer-Aided Design 46* (2014), 120 – 128. 3

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph. 23*, 3 (2004). 1, 2

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3d object arrangements. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* (2012). 1

[GTB14] GUY E., THIERY J.-M., BOUBEKEUR T.: Similarity-based selection for 3d surfacess. *Computer Graphics Forum 31*, 2 (2014). 2

[HGM14] HUANG Q., GUIBAS L. J., MITRA N. J.: Near-regular structure discovery using linear programming. *ACM Trans. Graph. 33*, 3 (2014), 23:1–23:17. 7

[Hop79] HOPCROFT JOHN; ULLMAN J.: *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979. 4

[JTRS12] JAIN A., THORMAHLEN T., RITSCHEL T., SEIDEL H.-P.: Exploring shape variations by 3d-model decomposition and part-based recombination. *Computer Graphics Forum 31*, 2pt3 (2012), 631–640. 3

[KBW*12] KALOJANOV J., BOKELOH M., WAND M., GUIBAS L., SEIDEL H.-P., SLUSALLEK P.: Microtiles: Extracting building blocks from correspondences. In *Computer Graphics Forum (Proc. SGP)* (2012). 2

[KCKK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D., KOLTUN V.: A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics 31*, 4 (2012). 2

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

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (2003), 277–286. 2

[LF08] LEE J., FUNKHOUSER T.: Sketch-based search and composition of 3D models. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2008). 2

[LS10] LANDRENEAU E., SCHAEFER S.: Scale and scale-like structures. *CGF (Proc. SGP) 29*, 5 (2010). 2

[MM08] MERRELL P., MANOCHA D.: Continuous model synthesis. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 158. 2

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (2006), 614–623. 4

[MWZ*13] MITRA N. J., WAND M., ZHANG H., COHEN-OR D., BOKELOH M.: Structure-aware shape processing. In *EUROGRAPHICS State-of-the-art Report* (2013). 1, 3

[RP03] REGHIZZI S. C., PRADELLA M.: Tile rewriting grammars. In *7th International Conference on Developments in Language Theory* (2003). 4

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snappaste: An interactive technique for easy mesh composition. *Vis. Comput. 22*, 9 (2006), 835–844. 2

[SG71] STINY G., GIPS J.: Shape grammars and the generative specification of painting and sculpture. In *IFIP Congress 71* (Ljubljana, Yugoslavia, 1971). 4

[SI07] SHIN H., IGARASHI T.: Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. In *Proc. GI* (2007), pp. 63–70. 2

[TYK*12] TALTON J., YANG L., KUMAR R., LIM M., GOODMAN N., MĚCH R.: Learning design patterns with bayesian grammar induction. In *Proc. UIST* (2012), ACM, pp. 63–74. 2

[WXL*11] WANG Y., XU K., LI J., ZHANG H., SHAMIR A., LIU L., CHENG Z., XIONG Y.: Symmetry hierarchy of man-made objects. In *Proc. Eurographics* (2011). 1, 3

[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), 57:1–10. 3

[YBY*13] YEH Y.-T., BREEDEN K., YANG L., FISHER M., HANRAHAN P.: Synthesis of tiled patterns using factor graphs. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 3. 2

[ZCOM13] ZHENG Y., COHEN-OR D., MITRA N. J.: Smart variations: Functional substructures for part compatibility. *CGF (Eurographics) 32*, 2pt2 (2013), 195–204. 3

[ZTS09] ZATZARINNI R., TAL A., SHAMIR A.: Relief analysis and extraction. *Proc. SIGGRAPH Asia 28*, 5 (2009). 2