EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis

Melinos Averkiou[1] Vladimir G. Kim[2] Youyi Zheng[3] Niloy J. Mitra[1]

[1]University College London [2]Stanford University [3]Yale University

Abstract

Recent advances in modeling tools enable non-expert users to synthesize novel shapes by assembling parts extracted from model databases. A major challenge for these tools is to provide users with relevant parts, which is especially difficult for large repositories with significant geometric variations. In this paper we analyze unorganized collections of 3D models to facilitate explorative shape synthesis by providing high-level feedback of possible synthesizable shapes. By jointly analyzing arrangements and shapes of parts across models, we hierarchically embed the models into low-dimensional spaces. The user can then use the parameterization to explore the existing models by clicking in different areas or by selecting groups to zoom on specific shape clusters. More importantly, any point in the embedded space can be lifted to an arrangement of parts to provide an abstracted view of possible shape variations. The abstraction can further be realized by appropriately deforming parts from neighboring models to produce synthesized geometry. Our experiments show that users can rapidly generate plausible and diverse shapes using our system, which also performs favorably with respect to previous modeling tools.

1. Introduction

Modelers often lack a clear mental image when creating a new shape and would ideally prefer to first browse through different possible geometric realizations of the target object. For example, one may want to flip through various existing designs before sketching a new chair. The ever growing 3D model repositories (e.g., Trimble 3D Warehouse) provide rich samplings of such design possibilities, but are typically unorganized and do not come with any parameterization of the underlying shape spaces. The challenge then is how to characterize, organize, and effectively navigate such shape spaces implicitly specified by model collections. An even more important challenge is how to provide previews of possible shapes that are *missing* from the input collections. In this paper we propose how to effectively organize unorganized model collections, and systematically detect and populate the *missing shape variations* (see Figure 1).

There are a few common paradigms for exploring model repositories: the user can provide a query 3D shape [FMK*03] or scribble a target shape in 2D [ERB*12], and then the system retrieves the matching shape(s). Such approaches rely on the user having a clear conceptual model of the target shape and being able to effectively communicate the same. Further, the repositories should contain sufficiently similar shapes that can be retrieved, an assumption that is often violated. More recently, qualitative exploration tools have been proposed [HSS*13, KFLCO13] using dynamic embedding. Although such systems provide intuitive local exploration of existing models, they do not support synthesis of the missing shapes. Talton et al. [TGY*09] propose an intuitive interface that tightly couples exploration and synthesis for parameterized model families. However, their method assumes a compact parameterizable de-

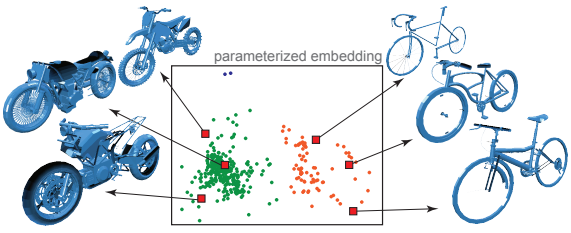

Figure 1: *We analyze unorganized model collections using template-based abstractions to create a low-dimensional parameterized embedding of the underlying shape space. The user then explores the parameterized space to create novel models by probing the empty regions (e.g., in red rectangles). In each case, a model was synthesized by significantly deforming and combining parts from the input models.*

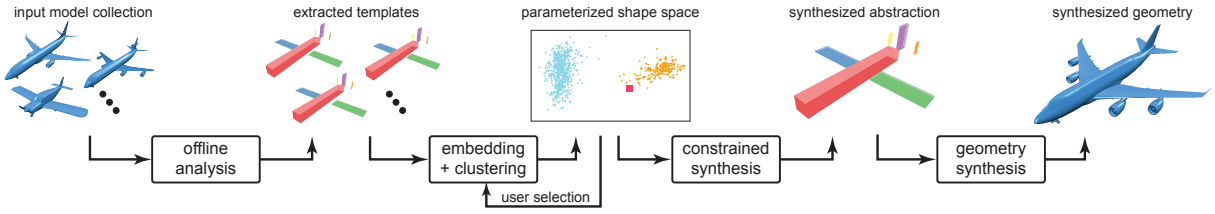

Figure 2: *We analyze unorganized model collections to obtain a template-based parameterized shape space. Exploration of the abstracted shape space reveals possible shape variations, which can then be realized by appropriately mixing the input models. The combined exploration and synthesis allows users to quickly get an overview of the modeling space and subsequently create novel shape variations via the parameterized shape space.*

sign space [WP95, ACP03], which quickly becomes infeasible for raw model collections with diverse shape variations.

In the context of synthesis, modeling from scratch requires a lot of expertise and even professional training. A simpler interface that is suitable for novice users is to directly combine parts from existing models to synthesize new models [FKS*04]. This type of direct manipulation-based interface still faces the challenge of finding and extracting appropriate parts from model repositories which is cumbersome with standard retrieval techniques. Hence, smarter alternatives have been proposed, searching for parts based on geometric [KJS07] and relational [ZCOM13] similarity, or by sampling suitable probabilistic graphical models [CKGK11, KCKK12]. Such methods, however, do not provide the user with a high-level preview of the space of possible models that can be synthesized. Further, the user needs a clear vision of the final shape in order to effectively probe such systems to retrieve suitable models or model parts. This is often difficult, especially in the presence of significant model variations (see Figure 4).

We propose *ShapeSynth*, a novel approach that directly integrates exploration with synthesis. The key to our method is extracting a template-based parameterizable space that effectively factors out and encodes (part-) deformation across the collection. Our method has an offline analysis step and an online hierarchical encoding stage to extract local embeddings of the underlying shape spaces. The embeddings facilitate both fast exploration of the existing shapes and previewing of possible shapes that are missing from the input collections. The offline analysis reveals the structure of the shapes in the collection using a set of deformable templates. The automatically-learned templates are fitted to each shape producing co-aligned and cosegmented models with known part deformations. The fast online embedding reveals the different shape groupings inside the collection, exposes the main modes of variability for each group, and lends itself to fast and intuitive synthesis of novel shapes. We performed a user study to test our system on several large model collections and compared with alternative systems (see also supplementary video and demo). Figure 1 shows some of the models created by the different users of our system. In summary, our contributions are:

- An efficient embedding technique for part-aware shape descriptors that enables a hierarchical organization of large model collections and provides a parameterized basis for exploring the underlying shape space;
- a novel exploration interface based on the embedding that reveals groupings of shapes and summarizes the main modes of variation inside each cluster; and
- a novel tool for synthesizing new shapes using 3D model repositories, seamlessly tied to the exploration interface.

2. Related Work

In this work we demonstrate the synergy between two problems that were traditionally studied independently: *exploration* of geometric collections and *synthesis* of novel shapes.

Exploration of shape collections. A large body of research has been devoted towards shape retrieval from collections of 3D models. Typically, the user either provides an example shape [FMK*03, FKS*04], a 2D sketch [ERB*12, LF08, SXY*11], or 3D scene context [FH10, XCF*13], while the corresponding system finds an appropriate model from the repository. Topological information [STP12] can also be used for non-rigid shape retrieval. Ovsjanikov et al. [OLGM11] observed that shape retrieval is only suitable when the user has prior knowledge of a database, which is itself a challenging task for unorganized datasets. Hence, they proposed an alternative tool for exploring novel collections. Several exploration interfaces have been proposed since then, including navigating in the space of 3D models by deforming a proxy shape [OLGM11], selecting regions of interest to define an ordering of the models [KLM*12], and embedding shapes into a 2D space based on their geometric differences [HSS*13, KFLCO13, ROA*13]. Although these exploration techniques provide context for a potential modeler (e.g., a better understanding of variations in part arrangements and part geometry), none of these methods allow synthesizing novel shapes during the exploration session, which is the key contribution of our work. In an in-

teresting earlier attempt, Talton et al. [TGY*09] proposed the use of parametric design spaces to support exploratory modeling based on human preferences. Although our work is inspired by their interface, their system is only suitable for parametric shape spaces (e.g., human bodies, procedural plants) and would not work with raw collections of 3D models as its input.

Part-based model synthesis. The seminal modeling-by-example system [FKS*04] demonstrated that even inexperienced users can synthesize interesting shapes by mixing parts from model repositories. Although several follow-up methods have been proposed for stitching compatible parts into a final 3D model (e.g., [SBSCO06, CKGK11]), two key challenges remain: choosing appropriate parts and effectively presenting them to the user.

In the Shuffler system, appropriate parts were chosen from a set of consistent segments [KJS07]. Subsequently, alternative methods measured part compatibility with respect to the rest of the shape, including a probabilistic model learned from training examples [CKGK11, KCKK12], a fuzzy part correspondence based on similarity between their bounding boxes [XHCOB12], part-based contextual information [XXM*13], and functional arrangements which encode structural relations such as symmetry and support [ZCOM13]. Most existing methods present compatible parts as sorted lists [FKS*04,CKGK11], which can be limiting if the number of parts is large. Jain et al. [JTRS12] allow choosing pairs of models and produce intermediate shapes with similar part arrangements. However, choosing which shapes to blend requires prior knowledge about the collection and is only appropriate for small datasets. Further, these methods assume compatible parts, which is easily violated in the case of large shape variations. Recently, Chaudhuri et al. [CKGF13] proposed a browsing interface that uses 1D sliders associated with adjectives, allowing to search for parts that have more or less of a certain property. Although this results in a simple and effective interface, assigning natural language tags to geometry requires significant manual effort and also does not provide sufficient geometric context.

3. System Overview

Our system takes as input a collection of semantically-related man-made shapes, e.g., a set of 3D chair models, which were downloaded from the web. The user starts by loading such a shape collection, which has been pre-analyzed in an off-line stage (see Section 4).

The system interface, see Figure 3, is split into three panels: the *icon view* that presents the different representatives for quickly selecting among different style groupings; the *exploration view* that presents a set of embedded points, one for each shape among the current selection of models; and the *model view* that presents the current synthesized model,

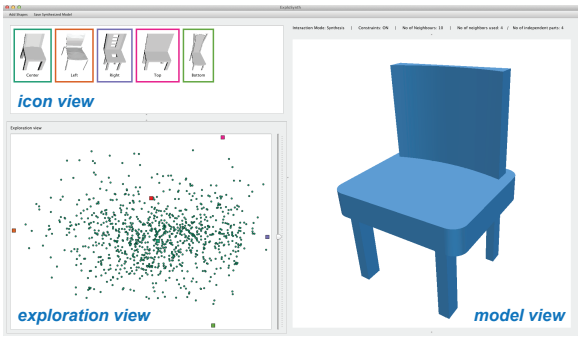

Figure 3: *Our system comprises of an **exploration view** to show the embedding of the input models; an **icon view** to show representative models for the current group(s); and a **model view** for displaying synthesized models created using our coupled exploration and synthesis algorithm.*

either abstracted as a set of box proxies, or as a part-based geometric realization.

Input models get embedded such that models with similar deformations end up as neighbors, while dissimilar models are embedded to distant points (see Section 4). The embedded points are automatically grouped to provide the user with a high-level overview of representative shapes in the *icon view* panel, with different colors indicating different groups. As the user selects one of the groups, the corresponding models are re-embedded and the process continues. Essentially, the user traverses the hierarchically organized models that are grouped based on their similarity, with shape variations being automatically factored out. Furthermore, the user can explore variations within any single group by studying the main variation modes across models in the selected group.

More importantly, one can use our system to preview plausible shapes that are missing in the original collections. As the user hovers over any empty region in the *exploration view*, at any level of the hierarchy, the system shows abstracted views of synthesized shapes in the *model view*. This allows users to progressively conceive and refine the missing shapes that they want to create. Note that the users can work at any level of the hierarchy and freely combine models across clusters, if they desire. By providing the user an idea of the shapes she wants to combine into new shapes, we simplify the content creation process. Essentially we provide a quick glimpse of possible models by exposing the space spanned by the input collections. Constraints such as symmetry, contact, and size are directly preserved by our system. Finally, the user can also view plausible geometric realizations of the abstracted model. In the background, the system produces such geometric realizations by combining deformed parts from appropriate neighboring models (see supplementary video and demo).

Figure 4: *Combining a random selection of chair models (top), even when they are consistently segmented, is challenging. The models have different proportions of parts, that make a part selection based on visual inspection of the superimposed models (bottom-left) confusing and can easily result in meaningless part ensembles (bottom-right). Instead, we expose a constrained and intuitive part-based shape space for easy exploration and synthesis.*

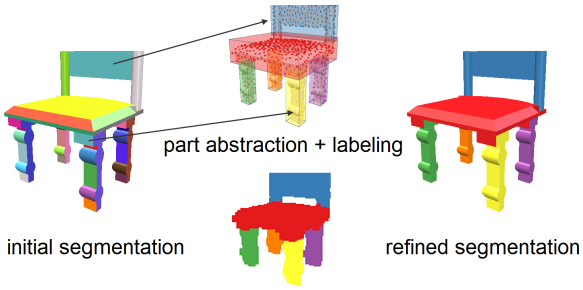

Figure 5: *In the case of models with multiple components (left), we use the extracted part distributions obtained from the shape collection [KLM*13] to obtain an initial point labeling (middle-bottom) and part abstraction (middle-top). We refine the segments using a labeling optimization (right).*

4. Algorithm

Starting from an unorganized model collection, our goal is to allow the user to meaningfully navigate the input models, and more importantly, provide a preview of the possible shapes that can be synthesized by appropriately combining parts from the different input models. In order to enable such exploration of missing shapes, we have to overcome a few key challenges: (i) the input models typically have large shape variations that in turn obscure any inherent consistency across the collection; (ii) the models do not come with any segmentation or consistent parameterization; and (iii) the collections typically include thousands of models, making realtime analysis non-trivial. For example, a visual investigation of the models in Figure 4 does not immediately reveal the space of possible models that can be realized by part-level combination of the input models.

We overcome these challenges by computing intrinsic embeddings of the models, which in turn facilitate previewing possible part-based synthesized models. The embeddings help to identify *which* models can be combined; *what* respective parts can be combined; and finally *how* the parts can be deformed to produce a consistent model. Thus, the user can directly explore the space of plausible models, with the system factoring out the variations in configurations that obscure novel synthesis possibilities. Intuitively, the embeddings offer parameterizations to systematically factor out part deformations, making subsequent synthesis simple and intuitive. We now describe the key steps of our method (see Figure 2).

Initial analysis. First, in an offline phase [KLM*13], we analyze the input collection of models $\{M_1, \dots M_N\}$. Starting with an initial template model, the method jointly optimizes for part segmentation, point-to-point surface correspondence, and a compact deformation model to best explain the input. The deformation model assumes that each shape can be approximated with a set of box-like parts that differ in position and scale.

Abstracted encoding. Based on the extracted distribution of the template parameters, we refine the segmentations of the individual models M_i. We assume that each model comes with multiple disconnected components, which is true for most models in Trimble Warehouse [Tri13] that we used in our experiments. If this assumption does not hold, we simply assign each triangle to its nearest box. Let $\{p_1, p_2, \dots\}$ be the set of components for model M_i. Our task is to associate each component with a template box. This is essentially a labeling problem, where each p_i can be assigned to a set of t candidate boxes $\{l_1, \dots, l_t\}$. We formulate the labeling as a MRF minimization:

$$\{l_i\}^\star := \arg\min_{\{l_i\}} \sum_i E(p_i \to l_j) + \sum_{i,j} E(p_i \to l_k, p_j \to l_l) \quad (1)$$

The unary term $E(p_i \to l_j) := vol(B_{p_i \cup l_j}) - vol(B_{l_j})$, where $vol(B_{p_i \cup l_j})$ denotes the bounding volume of component p_i and template box l_j, and $vol(B_{l_j})$ denotes the bounding volume of template box l_j, measures the increase of bounding volume of the template box when component p_i is assigned to it. The pairwise term $E(p_i \to l_k, p_j \to l_l)$ measures the penalty when two neighboring components (based on shortest distance between them) are assigned different labels (set to 1e-5 in our tests). In the end, for each model we get a set of abstracted boxes, each enclosing a part of the input model (see Figure 5-right).

Let there be t different parts across all the extracted templates. We represent each model M_i as a configuration vector $X_i \in \mathbb{R}^{6t}$, where each (axis-aligned) template box is represented by its centroid $\mathbf{c}$ and its dimensions $\mathbf{s}$, i.e., its length/breadth/height. Parameters corresponding to missing

Data: Input model collection $\mathbf{M} := \{M_1, \ldots M_N\}$.
Result: Embedding coordinates of the selected models.
1. Analyze the input collection $\mathbf{M}$ in an offline stage [KLM*13];
2. For each model $M_i \in \mathbf{M}$, refine initial segmentation to obtain configuration vector $X_i \in \mathbb{R}^{6t}$;
3. Set selection $\mathbb{M} \leftarrow \mathbf{M}$;
4. **while** *new selection* $\mathbb{M}$ *available* **do**
   i. Randomly pick n models $\tilde{M}_j$ for $j = 1, \ldots n$ from $\mathbb{M}$ as landmarks;
   ii. Construct distance matrix $\mathbf{D}_{n \times n}$ with elements $d_{i,j} := d(\tilde{M}_i, \tilde{M}_j)$ for $i, j = 1, \ldots n$;
   iii. Compute MDS embedding of the landmark models $\tilde{M}_j$ using $\mathbf{D}$ to obtain $\tilde{Y}_j \in \mathbb{R}^2$;
   iv. For all $M_i \in \mathbb{M}$ and $M_i \notin \{\tilde{M}_j\}$, find its k nearest neighbor models among the landmark models and use distance-based interpolation to obtain embedded coordinates $\{Y_i\}$.
   v. Compute (linear) basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ for inverse mapping of embedded coordinates to configuration vectors.
   vi. Apply mean-shift clustering on the points $\{Y_i\}$;
   vii. Update selection $\mathbb{M}$ and repeat hierarchically by returning to step #4 ;
end

Algorithm 1: Iteratively embed a selection of models $\mathbb{M}$, which is then used for exploration and synthesis.

parts are set to zero. We also detect the potential relations among the individual parts, i.e., symmetry and contact relations, and propagate the information to their associated templates, which are later used in the constrained synthesis phase. Note that the relations should be unified across templates within a given family. To address this in a consensus stage, we collect the relations among all the templates and use a greedy selection strategy to filter out falsely detected relations. Improperly identified or conflicting relations can be manually corrected, although advanced automated methods can potentially be used [MWZ*13].

Efficient embedding. Both exploration and synthesis require a notion of *neighborhood* among the models. We use the coarse abstraction obtained above to define such a dissimilarity distance between model pairs M_i and M_j as follows: $d(M_i, M_j) := \|X_i - X_j\|$. Note that since we also have the parameter distributions, one can instead use Mahalanobis distance. Thus, similar models have near-zero dissimilarity score, while dissimilar model pairs get high scores.

We use the similarity values to embed the models into a low-dimensional parameterized space. One option is to construct a $N \times N$ matrix with all the pairwise similarity values (e.g., $\exp(-d(M_i, M_j)^2/2\sigma^2)$) between the input models and compute its spectral embedding [KLM*12]. Such a direct computation, however, can be prohibitively expensive (i.e., $O(N^3)$) for large model collections. Instead, we propose a sampling-based approach to efficiently build an embedding of the models (see Algorithm 1). The key observation is that the embedding is largely dictated by the members from different (unknown) clusters (c.f., [dST04]). Hence, working with a random sampling of models as representatives yields an approximate embedding. Note that we use the approximate embedding only when the number of models is large, otherwise we perform the embedding with all the selected models.

We start by picking at random n landmark models, say $\{\tilde{M}_j\}$, from the current set of models $\mathbb{M}$. Using the n landmark models, we compute their pairwise distance matrix $\mathbf{D}_{n \times n}$ and embed the models to $\mathbb{R}^2$ using multi-dimensional scaling (MDS) based on singular value decomposition (SVD), to get $\{\tilde{Y}_j\}$. For any other model $M_i \in \mathbb{M}$, we compute its k nearest neighbors among the landmark models. We then interpolate the embedded coordinates of the landmark models to compute the embedding of M_i, i.e., $Y_i \leftarrow \sum_{j=1:k} w_j \tilde{Y}_j / \sum_{j=1:k} w_j$, where $w_j := \exp(-d(M_i, \tilde{M}_j)^2/2\sigma^2)$, with σ set to the diameter of set $\{\tilde{X}_i\}$ and $\tilde{M}_j$ denoting the j-th closest of the landmark models .

The above embedding method has a complexity of $O(n^2 + Nk \log n)$. For example, for a chair dataset with 2036 models, the sparse embedding takes about 0.6 sec, which is about 12 times faster than full embedding (see Figure 6).

Abstracting missing shapes. At this stage, we have mapped the initial coordinates $\{X_i\} \rightarrow \{Y_i\}$. We solve for the dominant linear variation modes $\mathbf{e}_1$ and $\mathbf{e}_2$ such that $Y_i \approx [(X_i \cdot \mathbf{e}_1), (X_i \cdot \mathbf{e}_2)]$ for all $i \in [1, N]$ using a least squares formulation. Now, given any point (α, β), we can lift up the

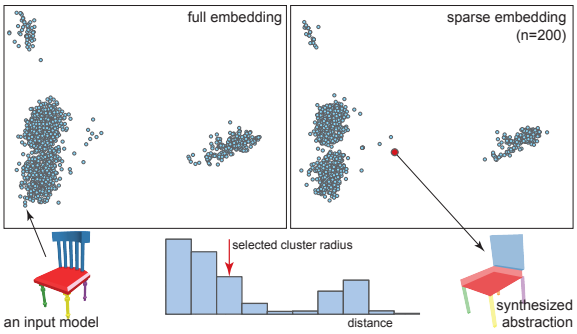

Figure 6: *We embed the input models using their corresponding fitted template-based abstractions. We perform an efficient landmark-based embedding and analyze the points to obtain a parameterized template abstracting the underlying shape space. As the user navigates the embedded space, the extracted variation modes are used to lift the points (shown in red) to synthesize template abstractions. The distribution of the pairwise distances between the embedded points is used to estimate a suitable clustering radius.*

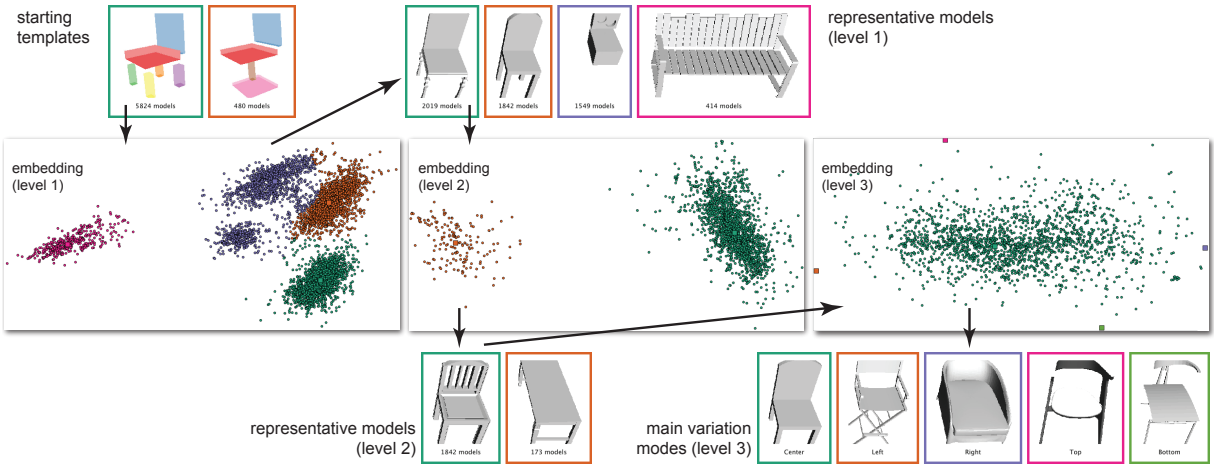

Figure 7: *A typical hierarchical exploration session using our interface. After the initial analysis, the system displays the top level templates (top-left). As the user selects the green mode, its member models are embedded (level 1) and 4 dominant clusters are detected. The user selects the next representative and its member models are re-embedded. When a single cluster is discovered, its representative and dominant variation modes are shown (bottom-right).*

point (from the empty region) to the configuration vector as $(\alpha,\beta) \rightarrow \alpha\mathbf{e}_1 + \beta\mathbf{e}_2$, thus providing an abstracted model X as a preview for the empty region.

Grouping models. We cluster the embedded points using mean-shift clustering [CM02] in order to organize the data into a hierarchy. We automatically select the clustering radius based on the histogram of the pairwise distances between the embedded points. Intuitively, in the case of points that can be grouped into multiple clusters, we can estimate a good clustering radius based on the first valley (if any) of the histogram (see Figure 6). Each of the extracted clusters is then re-embedded to extract corresponding basis vectors (i.e., $\mathbf{e}_1$ and $\mathbf{e}_2$), eventually forming a hierarchical organization (see Figure 7). As the user selects one of the clusters, she zooms into that particular cluster in order to better study the fine scale variations.

Constrained synthesis. A direct derivation of box configuration $X = \alpha\mathbf{e}_1 + \beta\mathbf{e}_2$ from the embedding space can easily result in models that deviate from a semantically valid one, e.g., symmetry being broken, part-to-part contacts being lost, etc. We project the box configuration parameters to the valid shape space using a constrained optimization. We observe that many relations of interest (c.f., [MWZ*13]) simply amount to linear constraints involving parameters of the configuration vector, e.g., contact, reflective symmetry about known plane, equal dimensions of certain parts, etc. Our goal is to obtain a new configuration $\tilde{X}$ such that potential symmetry and contact relations among parts are restored, while $\tilde{X}$ being as close to X as possible (see Figure 8). This amounts to solving the following minimization:

$$\min_{\tilde{X}} \|\tilde{X} - X\|^2 \quad \text{such that} \quad f_j(\tilde{X}) = 0 \; \forall j = 1,\ldots c \qquad (2)$$

where, $f_i(\tilde{X})$ is a set of c semantic constraints derived from the relations among the parts. We support three main types of relations: symmetry, contact, and equal length.

Symmetry. Let two boxes, say $(\mathbf{c}_i,\mathbf{s}_i)$ and $(\mathbf{c}_j,\mathbf{s}_j)$, have reflective symmetry with respect to a given plane. The corresponding constraints take the form:

$$((\mathbf{c}_i + \mathbf{c}_j)/2 - \mathbf{o}) \cdot \mathbf{n} = 0; \;\; (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{n} = \mathbf{0}; \;\; \mathbf{s}_i - \mathbf{s}_j = \mathbf{0}$$

where $\mathbf{n}$ and $\mathbf{o}$ are the normal to the reflection plane and a point on the plane, respectively.

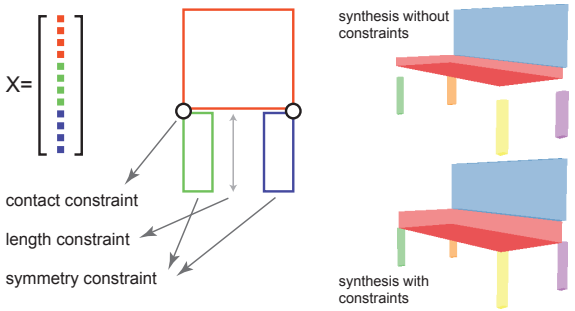

Figure 8: *Illustrative example of the different constraints handled in our framework. (Left) In this 2D example, the configuration vector $X \in \mathbb{R}^{12}$ represents the abstracted model with 3 parts. For example, the two contact constraints involve the orange-green and orange-blue boxes and hence the corresponding $f_i(X)$ involves the corresponding coordinates of X. (Right) Our system restores these constraints during the real-time exploration using a QP formulation.*

Contact. When two boxes are in contact, they share a common contact point. Between two boxes in contact, we assign the closest points as contact points. Alternatively, the user can directly specify the contact points. Thus, between two boxes, the contact constraint takes the form: $\mathbf{c}_i + \mathbf{s}_i/2 = \mathbf{c}_j + \mathbf{s}_j/2$ (up to sign changes due to which corners get selected).

Equal length. In certain cases, we want a pair of boxes to have similar dimensions (for example, the legs of a chair should have equal height even if they are not symmetric). Since the abstracted boxes are axis-aligned, such a constraint takes the form: $\mathbf{s}_i^y = \mathbf{s}_j^y$ when equality along y-direction is desired.

The optimization in Equation 2 amounts to solving a quadratic program with linear constraints. As the user explores the configuration space extracted from the input collection, we perform the optimization to directly show the constrained solution (see Figure 8 and supplementary video). Note that the input templates, i.e. $\{X_i\}$, do not necessarily satisfy the constraints. However, we do not project them to the constrained space since the model parts are later deformed to a constrained model, as described next.

Synthesizing models. As the user moves the mouse over a point (α, β), our system shows the corresponding abstracted box model, $\alpha\mathbf{e}_1 + \beta\mathbf{e}_2$, which is then constrained to produce abstracted model $\tilde{X}$. Each such feature vector $\tilde{X} \in \mathbb{R}^{6t}$ represents concatenated parameters for t boxes $\tilde{X} = [x_1, \ldots x_t]$, where x_i is a 6-dimensional vector that encodes position and dimensions of the i-th box. When the user is satisfied with the coarse arrangement of parts, she can click and lock the system to the current box model. This immediately prompts our system to *fill* the boxes with geometric parts that have the most similar positions and dimensions (i.e., $\arg\min_{x_{\text{part}}} \|x_{\text{part}} - x_i\|$), which essentially picks parts that are to be least deformed. The user can continue exploration and visualize alternative part arrangements drawn over the selection, or refine the choice of selected parts by clicking on a corresponding box x_i in the *model view*. The click prompts the system to cycle through the candidate parts, where the parts are taken from the k nearest models M_j from the selected model X (see Figure 9). One can use geometric similarity to further sort the selected box x_i.

5. Evaluation

In this section, we evaluate the proposed shape synthesis tool. First, we describe the data and experimental setup, and evaluate performance of our algorithm on diverse datasets obtained from 3D Warehouse. While our coupled analysis, exploration, and synthesis framework is the first of its kind, we compare parts of our system to state-of-art shape synthesis methods ([CKGK11] and [JTRS12]) and evaluate our constrained synthesis algorithm.

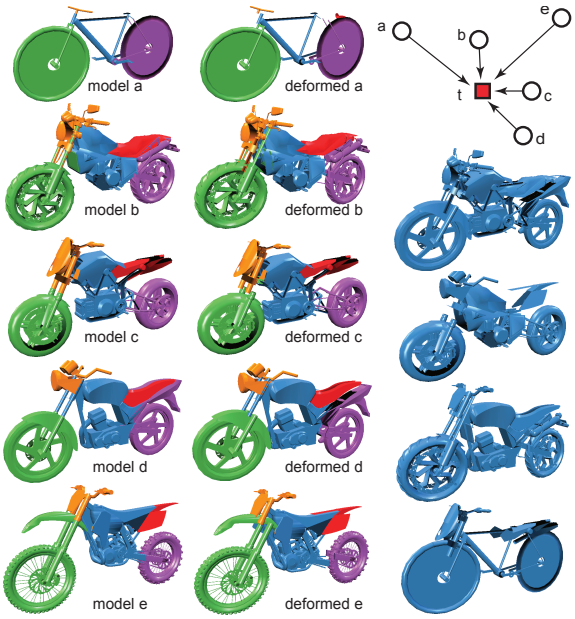

Figure 9: *Our system allows to preview possible geometric realizations in an empty region around the embedded points (top-right). Each of the retrieved models (models a-e) is deformed to match the query configuration (indicated as a red box). Parts from the deformed models (middle) are then combined to create different plausible shapes (right).*

Dataset. We tested our framework using Trimble Warehouse models [Tri13] obtained from [KLM*13]. We selected a subset of models such that the template fitting energy is below 30, and hence our final analysis included 2062 chairs, 636 planes, and 114 bikes. As an additional dataset, to compare with the authors' implementation of synthesis method [CKGK11], we use their manually-segmented dataset of 100 airplanes from Digimation ModelBank.

Results. Several participants used our system to explore existing model collections and create new shape variations. Overall the feedback was positive and they created a variety of different models (see supplementary). Most people appreciated the ability to quickly obtain a high-level overview of the modeling space, refine the selection to a region via hierarchical navigation, and finally obtain immediate preview of geometric realizations. Some users complained that the final models were at times disconnected making them look unrealistic. This is expected as we only enforce contact constraints at the abstracted level of the boxes, and not on the original geometry.

User experiments. We evaluated our system on 3 large and diverse datasets from 3D Warehouse. We asked 8 volunteers from a computer science department to use our system to perform an open-ended task on each dataset:

T1: *Create the most diverse set of 5 shapes.*

Next, we validated the results by comparing to a random selection of groups of 5 models from the existing datasets. Similarly to previous work [CKGK11, CKGF13], we recruited a different group of volunteers to compare random pairs of results created by users of our method and models belonging to the random selection. For each pair of results, we described the task **T1** and asked (i) if the resulting shapes look plausible, and (ii) if the resulting shapes look diverse. For each question, the evaluator had an option of selecting one of the results, both, or none that satisfy the plausibility or diversity condition.

Figure 10: *Sets of models created in our user experiment (please refer to supplementary for full results). Our system enables rapid synthesis of diverse models.*

In Figure 10, we present user-created models in each category (all results are provided in the supplemental material). Table 1 also shows results of the validation. Note that our method is comparable to real models in terms of plausibility of shapes, and allows easy creation of diverse models in comparison to random selection among the input models. As the user clicks on the parameterized space, new shape variations are immediately shown (see supplementary demo).

Table 1: *User study on Task **T1** comparing our method with a random selection from a dataset. Voting indicates number of times users voted for our method vs random selection (where votes for both are summed with individual votes), and timings are in minutes.*

	Voting						Time
	Plausibility			Diversity			(min)
Dataset	**Our**	**Rnd**	**None**	**O**	**R**	**N**	**O**
bikes	77	64	2	69	58	11	5
chair	48	104	4	100	19	5	4
planes	53	81	2	68	48	6	3

Comparison to Chaudhuri et al. [2011]. We compare against this system by focusing on high-level exploratory design tasks. We used the authors' implementation of the method on their dataset of 100 consistently pre-segmented airplanes. Our initial analysis was modified to create deformable templates from segmented models and fit the templates to all models without changing the segmentation. We recruited a different group of 10 volunteers with CS background and asked them to perform task **T1**, and:

T2: *Create an airplane that is best suited to win a dogfight (one-on-one aerial combat) in a computer game. This is the same task as in [CKGF13].*

Each user performed both tasks using both systems in a consistent order (**T1, T2**), while we randomly permuted the order in which systems were used. Figure 11 summarizes the results produced by the same user (selected at random) in both systems. We again validated the results using another group of volunteers and summarize the statistics in Table 2. Although users of our system often produced implausible models when aiming at diversity in task **T1**, often the resulting sets of models were deemed comparable in diversity. Furthermore, we found that once the users became familiar with the space of shapes, they could very rapidly synthesize airplanes prescribed in task **T2** that are comparable to results produced by Chaudhuri et al. [CKGK11]. We find the variability result particularly surprising since our parameterization is based on coarse box-abstractions rather than geometric details. Even then, our system exposes interesting high-level part placement variations.

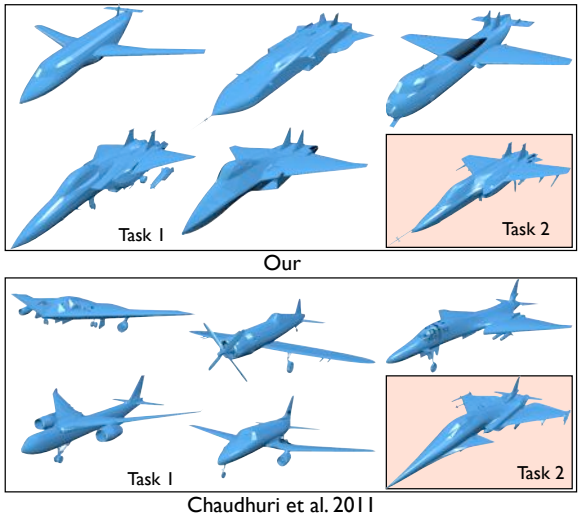

Figure 11: *Example models created by User 1 (picked at random) for comparison to Chaudhuri et al.[2011] system. Note that both sets of models created for task **T1** contain diverse and plausible shapes, as was requested in the task.*

Table 2: *Comparison to Chaudhuri et al. [CKGK11], tasks **T1** and **T2** were accomplished with two different interfaces. Voting columns indicate number of time users voted for results produced with our method vs their approach (where individual votes are summed with votes for both methods).*

	Voting						Time	
	Plausibility			Accomplished			(min)	
Task	**Our**	**Ch.**	**None**	**O**	**C**	**N**	**O**	**C**
T1	67	174	17	81	131	38	6	14
T2	132	150	13	158	180	12	2	4

Comparison to Jain et al. [2012]. Our method can also be used to interpolate between pairs of shapes, as in [JTRS12]. In Figure 12 we demonstrate two interpolations produced by our method, and our simplified implementation of their method. A visible advantage of our method is that it produces more plausible shape variations for intermediate shapes where the deformation is high (e.g., look at the chair legs). The quality comes from: (i) appropriate part scaling to facilitate model assembly; and (ii) neighboring models contributing to the intermediate models allowing richer variations. Finally, unlike Jain et al. [JTRS12], our analysis automatically segments the input models and establishes part-level correspondence. In the future, it will be interesting to use their contact and relation graphs to maintain the model structure of the synthesized variations. It should also be noted that as our approach is data-driven, performance improves as the dataset grows. The example in Figure 12 is illustrated on a very small dataset and hence this is an extreme case where our method can produce visible distortions. In practise, we did not observe this effect when working with the full datasets.

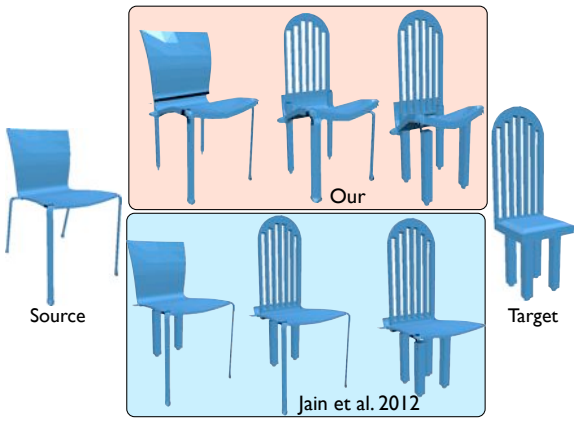

Figure 12: *We evaluate our method in a shape interpolation scenario such as in [JTRS12]. Note that our method is more robust to strong deformations because it uses other shapes from the dataset to produce intermediate results.*

Constrained synthesis. We evaluate the quality of our constrained synthesis via a leave-one-out experiment. First, starting with model collection $\mathbf{M}$, we compute its embedding $\mathbf{e}_1, \mathbf{e}_2$ and then select a particular model M_i embedded as (α^i, β^i). We then remove the model M_i and reanalyze $\mathbf{M} \backslash M_i$ to obtain embedding $\mathbf{f}_1, \mathbf{f}_2$ and re-synthesize the model as $\alpha^i \mathbf{f}_1 + \beta^i \mathbf{f}_2$. We then compare the effect of leaving out M_i on the embedding . We found the variation negligible in most cases. This is not surprising since our landmark-based embedding, and hence the extracted dominant modes are mainly dictated by the n randomly selected models. If the landmarks remain unchanged, the embedding does not change. Even with different landmarks the changes are small as shown in Figure 6. Furthermore, we use the new embedding to reconstruct the box structure for the model M_i given its coordinates (α^i, β^i), and we found that the reconstruction error is within 1% of the original box dimensions.

Restoring contacts. In a postprocessing step, it is possible to restore contacts using a simple post-processing step as shown in Figure 13. This step however is orthogonal to the focus of this paper, and we leave it to future work to fully evaluate its effect.

Figure 13: *The synthesized models can be further refined using docker-based part deformation. In these examples, the parts of the chair and bike are brought back into contact based on nearest part dockers.*

6. Conclusions

We presented an analysis approach that extracts a template-based hierarchical parameterization for input model collections. The extracted parameterization factors out part deformations to enable intuitive exploration of the models and provides a high-level overview of the underlying shape space. Subsequently, the analysis results are used for interactive creation of novel shape variations, both at an abstract box-level and also with geometric details. We evaluated our system using 4 datasets ranging from 100-2000 models with 18 users synthesizing 500+ novel shapes, validated by 30+ (different) people, with 164 pairwise comparisons.

In this work we focused on abstracting the input models by axis-aligned box-templates. Given that, our system is not suitable for datasets where the coarse structure of the shape does not correlate with its functionality or desired properties. In the future, we would like to study the use of oriented box-templates. Such an abstraction can be challenging to parameterize, but can potentially provide better un-

derstanding of the underlying shape space, improved clustering and more meaningful exploration and synthesis of novel shapes. Furthermore, we also plan to investigate supplementing the shape space with part-based geometric descriptors. The challenge is to appropriately combine the high-level box descriptors with low-level geometric descriptors. Lastly, the extracted parameterized space provides an abstracted representation of the underlying shape space: we plan to investigate other problems like pose estimation, scan completion, etc., as projections to this parameterized space.

Acknowledgements. We are grateful to Evangelos Kalogerakis and the reviewers for their comments; Siddhartha Chaudhuri for helping with the comparison to [CKGK11]. We thank Bongjin Koo and Yanir Kleiman for their comments, and James Hennessey for the video voiceover. This project was supported in part by a Marie Curie CIG and ERC Starting Grant SmartGeometry.

References

[ACP03] ALLEN B., CURLESS B., POPOVIĆ Z.: The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph. 22*, 3 (2003), 587–594. 2

[CKGF13] CHAUDHURI S., KALOGERAKIS E., GIGUERE S., FUNKHOUSER T.: Attribit: Content Creation with Semantic Attributes. In *Proceedings of UIST* (2013), pp. 193–202. 3, 8

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3D modeling. *ACM Trans. Graph. 30*, 4 (2011), 35:1–35:10. 2, 3, 7, 8, 9, 10

[CM02] COMANICIU D., MEER P.: Mean shift: a robust approach toward feature space analysis. *IEEE PAMI 24*, 5 (2002), 603–619. 6

[dST04] DE SILVA V., TENENBAUM J. B.: *Sparse multidimensional scaling using landmark points*. Tech. rep., Stanford University, 2004. 5

[ERB*12] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-based shape retrieval. *ACM Trans. Graph. 31*, 4 (2012), 31:1–31:10. 1, 2

[FH10] FISHER M., HANRAHAN P.: Context-based search for 3D models. *ACM Trans. Graph. 29*, 6 (2010), 182:1–182:10. 2

[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), 652–663. 2, 3

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3D models. *ACM Trans. Graph. 22*, 1 (2003), 83–105. 1, 2

[HSS*13] HUANG S.-S., SHAMIR A., SHEN C.-H., ZHANG H., SHEFFER A., HU S.-M., COHEN-OR D.: Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. Graph. 32*, 4 (2013), 71:1–71:10. 1, 2

[JTRS12] JAIN A., THORMÄHLEN 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, 7, 9

[KCKK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D., KOLTUN V.: A Probabilistic Model for Component-Based Shape Synthesis. *ACM Trans. Graph. 31*, 4 (2012), 55:1–55:11. 2, 3

[KFLCO13] KLEIMAN Y., FISH N., LANIR J., COHEN-OR D.: Dynamic Maps for Exploring and Browsing Shapes. *Computer Graphics Forum 32*, 5 (2013), 187–196. 1, 2

[KJS07] KREAVOY V., JULIUS D., SHEFFER A.: Model Composition from Interchangeable Components. In *Proceedings of the Pacific Conference on Computer Graphics and Applications* (2007), pp. 129–138. 2, 3

[KLM*12] KIM V. G., LI W., MITRA N. J., DIVERDI S., FUNKHOUSER T.: Exploring Collections of 3D Models using Fuzzy Correspondences. *ACM Trans. Graph. 31*, 4 (2012), 54:1–54:11. 2, 5

[KLM*13] KIM V. G., LI W., MITRA N. J., CHAUDHURI S., DIVERDI S., FUNKHOUSER T.: Learning part-based templates from large collections of 3D shapes. *ACM Trans. Graph. 32*, 4 (2013), 70:1–70:12. 4, 5, 7

[LF08] LEE J., FUNKHOUSER T.: Sketch-Based Search and Composition of 3D Models. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2008), pp. 97–104. 2

[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), pp. 175–197. 5, 6

[OLGM11] OVSJANIKOV M., LI W., GUIBAS L., MITRA N. J.: Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Trans. Graph. 30*, 4 (2011), 33:1–33:10. 2

[ROA*13] RUSTAMOV R., OVSJANIKOV M., AZENCOT O., BEN-CHEN M., CHAZAL F., GUIBAS L.: Map-Based Exploration of Intrinsic Shape Differences and Variability . *ACM Trans. Graph. 32*, 4 (2013), 72:1–72:12. 2

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: SnapPaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9-11 (2006), 835–844. 3

[STP12] SFIKAS K., THEOHARIS T., PRATIKAKIS I.: Non-rigid 3D object retrieval using topological information guided by conformal factors. *The Visual Computer 28*, 9 (2012), 943–955. 2

[SXY*11] SHAO T., XU W., YIN K., WANG J., ZHOU K., GUO B.: Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching. *Computer Graphics Forum 30*, 7 (2011), 2011–2020. 2

[TGY*09] TALTON J. O., GIBSON D., YANG L., HANRAHAN P., KOLTUN V.: Exploratory Modeling with Collaborative Design Spaces. *ACM Trans. Graph. 28*, 5 (2009), 167:1–167:10. 1, 3

[Tri13] TRIMBLE: Trimble 3D warehouse, http://sketchup.google.com/3dwarehouse/ 2013. 4, 7

[WP95] WEBER J., PENN J.: Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH* (1995), pp. 119–128. 2

[XCF*13] XU K., CHEN K., FU H., SUN W.-L., HU S.-M.: Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM Trans. Graph. 32*, 4 (2013), 123:1–123:15. 2

[XHCOB12] XU K., HAO Z., COHEN-OR D., BAOQUAN C.: Fit and Diverse : Set Evolution for Inspiring 3D Shape Galleries. *ACM Trans. Graph. 31*, 4 (2012), 57:1–57:10. 3

[XXM*13] XIE X., XU K., MITRA N. J., COHEN-OR D., GONG W., SU Q., CHEN B.: Sketch-to-Design: Context-Based Part Assembly. *Computer Graphics Forum 32*, 8 (2013), 233–245. 3

[ZCOM13] ZHENG Y., COHEN-OR D., MITRA N. J.: Smart Variations: Functional Substructures for Part Compatibility. *Computer Graphics Forum 32*, 2pt2 (2013), 195–204. 2, 3