# Semantic Shape Editing Using Deformation Handles

Mehmet Ersin Yumer* Siddhartha Chaudhuri† Jessica K. Hodgins* Levent Burak Kara*
*Carnegie Mellon University †Cornell University

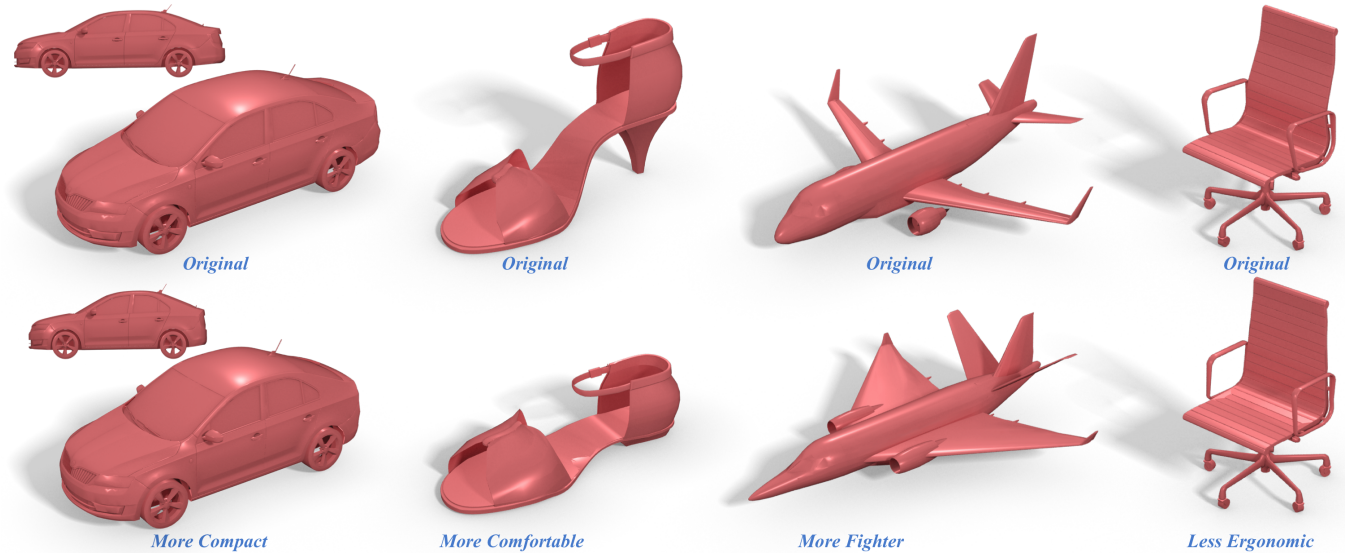


**Figure 1:** *Shapes edited using our system. Please refer to our supplementary video for real-time editing examples.*

## Abstract

We propose a shape editing method where the user creates geometric deformations using a set of semantic attributes, thus avoiding the need for detailed geometric manipulations. In contrast to prior work, we focus on continuous deformations instead of discrete part substitutions. Our method provides a platform for quick design explorations and allows non-experts to produce semantically guided shape variations that are otherwise difficult to attain. We crowdsource a large set of pairwise comparisons between the semantic attributes and geometry and use this data to learn a continuous mapping from the semantic attributes to geometry. The resulting map enables simple and intuitive shape manipulations based solely on the learned attributes. We demonstrate our method on large datasets using two different user interaction modes and evaluate its usability with a set of user studies.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Applications;

**Keywords:** semantic editing, semantic deformation, shape deformation, shape sets, crowdsourcing.

## 1 Introduction

The ability to edit existing digital objects is central to many modeling activities including shape design, shape exploration, and product customization. Intuitive and fast editing methods enable digital artists to build upon prior work, designers to explore shape variations, and engineers to respond to new product requirements. However, conventional shape design and editing technologies (*e.g.,* Maya, ZBrush, AutoCAD, SketchUp) are difficult to master because they require (1) expertise in the target product domain to ensure meaningful alterations and (2) familiarity with the geometric representation and operations to realize the intended modifications. As a result, transforming high-level modeling intentions into geometric directives is often challenging. Moreover, while studies in product design demonstrate that attribute-based user ratings can assist in consumer preference modeling [Orsborn et al. 2009], it remains difficult to customize existing shapes to reflect such preferences.

In this work, we propose a shape editing method where users edit 3D shapes using a set of high-level semantic attributes (Figure 1 and 2). We focus on *deforming* an input shape rather than composing a new shape through an assembly of existing shape parts [Chaudhuri et al. 2013]. As such, our method extends to databases that were not created using a part-based framework. Additionally, it allows users to explore variations of an input shape while retaining the underlying topological structure. Our approach is particularly useful in scenarios where the user's desires can be expressed using a set of attributes relevant to the target product, but there is no immediate means of transforming such intentions into geometric operations (*e.g., "Make this shoe more fashionable"*). Our system provides continuous deformations but does not add new components or remove existing ones from the model being edited. Hence, our approach complements existing part-based geometric modeling technologies (*e.g.,* [Chaudhuri et al. 2013]) by enabling a semantically driven interface amenable to shape exploration and customization.

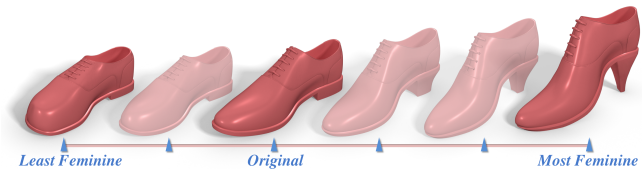


**Figure 2:** *Our system enables a continuous deformation space where design variations are explored using semantic sliders.*

To identify the relevant attributes, we survey a group of digital art professionals, educators and students to compile a rich set of attributes pertinent to the shape set. The initial set is then reduced to a compact set by a second, non-expert group of participants. Next, we crowdsource a large set of pairwise semantic comparisons between the models in the shape set and use this data to learn a continuous mapping from the attribute scores to the geometric space. Our geometric representation builds on the abstraction handle space proposed by Yumer and Kara [2014].

A direct mapping from the attribute to the geometry space is rarely useful as geometrically distant objects may share similar attribute values, causing highly discontinuous shape transitions when traversing the attribute space. To address this gap, we formulate shape deformation as a *constrained path traversal* problem in the geometric space, using the models in the shape set as regression points. Our formulation allows a shape database to be encoded as a single continuous deformation space wherein we compute different scalar functions, each for a different semantic attribute of interest.

The resulting formulation enables two modes of shape manipulation based solely on semantic descriptors. In the first mode, an input shape can be manipulated by adjusting the individual attribute levels with a group of sliders. In the second mode, an input shape can be embedded in a number of 2D scalar maps, each corresponding to a different attribute.

Our main contributions are

- A formulation for mapping semantic attributes to geometry for smooth shape editing using deformation handles without the need for low-level mesh correspondence or topological equivalence.
- A framework for learning such mappings for novel shape sets.
- Two intuitive, semantically driven shape manipulation user interfaces.
- Demonstration of our techniques on four large data sets and shape-semantic attribute data for the datasets (made publicly available).

## 2 Related Work

Our approach is closely related to prior work on shape editing, data-driven set analysis and attribute-to-geometry mapping. Below, we discuss the works that form the foundation for our approach.

**Shape deformation and editing.** To date, several deformation techniques have been proposed that aim for smoothness or detail preservation through energy function minimization (see [Botsch and Sorkine 2008] for a review). These methods primarily operate on polygonal models and take as input direct geometric directives, such as position constraints. These methods have been widely demonstrated for organic shapes. However, these approaches are not immediately suitable for man-made shapes where models often consist of many disconnected and overlapping sub-assemblies, discontinuous features, and poor triangulation.

Botsch and Kobbelt [2004] allow the user to author constraints on models and deform a region of interest using basis functions. Kraevoy *et al.* [2008] present a method for axis-aligned, non-uniform scaling of man-made shapes. Bokeloh *et al.* [2011] introduce a pattern-aware deformation method where a set of *sliding dockers* are extracted and selectively applied to suit the deformation. Gal *et al.* [2009] and Zheng *et al.* [2011] utilize an *analyze and edit* approach to shape editing, where constraints extracted from the original model are used during shape editing to produce globally plausible and logical deformations. The above methods facilitate shape deformation through information extracted from a single model, and hence are not immediately amenable to constraints emanating from a set analysis.

With the increasing availability of digital repositories, data-driven shape analysis and synthesis methods have been recently receiving a great deal of attention. A growing body of work has focused on low-level segmentation and shape matching [Golovinskiy and Funkhouser 2009; Sidi et al. 2011; van Kaick et al. 2013], part-based dataset exploration [Ovsjanikov et al. 2011], and part-based shape synthesis [Xu et al. 2012; Kalogerakis et al. 2012; Averkiou et al. 2014]. Mitra *et al.* [2013] provide an extensive overview of related techniques. More recent methods aim to decipher the geometric principles that underlie a product family in order to enable deformers that are customized for individual models, thereby expanding data-driven techniques beyond compositional modeling. Yumer and Kara [2012; 2014] present such a method for learning statistical shape deformers in the form of meta-handles that enable input models to be manipulated through abstract geometric proxies.

The above methods aim to establish a suitable abstraction between the target geometry and the means for interacting with it. However, these methods remain purely geometric and are unsuitable for modeling tasks involving semantic attributes as input.

**Semantic attributes.** Previous attribute-based shape manipulation and synthesis systems require mesh correspondences among the input data [Deng and Neumann 2008; Allen et al. 2003; Blanz and Vetter 1999]. This constraint restricts the admissible geometric variations in the input models, because the models must share a common topology and vertex correspondences must be established. In our approach, the topological and geometric variations among the members in the shape set is higher, and we do not require low-level mesh correspondence.

Attribute-based approaches have also been explored in recent applications, starting with binary attributes [Tao et al. 2009]. Relative attributes, similar to ours, have been used in image search [Parikh and Grauman 2011; Kovashka et al. 2012] and font search applications [O'Donovan et al. 2014], and semantic image color palette editing [Laffont et al. 2014]. Our approach shares similar goals, with a focus on 3D shape manipulation.

In a recent work, Chaudhuri *et al.* [2013] introduce a part-based assembly method for content creation. The approach learns semantic attributes for the parts that compose a shape. Each part group (*e.g.,* torso, head) is treated as an independent category of discrete objects. A relative attribute-based search, similar to the image-based applications mentioned above, lets users navigate the list of parts sorted with respect to an attribute. The system then constructs the final shape by assembling the individually chosen parts (*e.g.,* head, legs, arms, and torso are assembled to create an animal). This approach relies primarily on discrete object search within a prescribed set of parts, followed by part-based assembly of existing components. Similar to the image-based applications, the

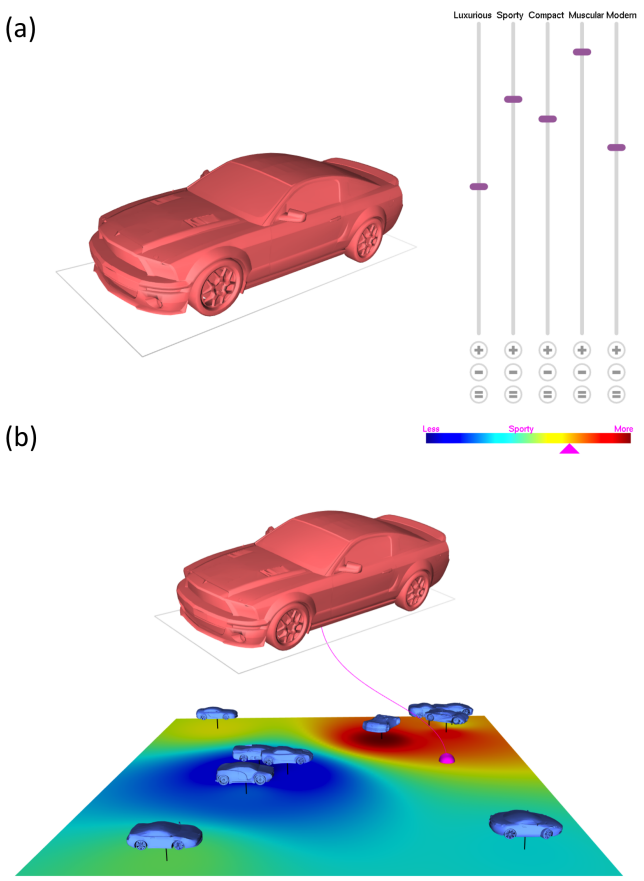


**Figure 3:** *Editing modes in our system: (a) Direct attribute editing, (b) Attribute map exploration for* Sporty.

system is tailored toward choosing entities from a fixed set rather than generating novel geometric variations.

Our approach is different from previous attribute-based systems in that when the user queries a *more* [attribute] variant of an object, the input object is geometrically altered to match the user's intent rather than being replaced by another *more* [attribute] object, without the need for topological equivalence or low-level mesh registration.

## 3 End-user Experience

Our approach provides two interaction modes: direct attribute editing and attribute map exploration. In both cases, the user starts by importing a polygonal mesh model that belongs to a semantically processed category of shapes. The imported model need not have been previously seen by our system.

**Direct attribute editing.** The interface for direct attribute editing (Figure 3(a)) provides a 3D view of the input model with a trackball interaction. Attributes pertinent to the category are presented as interactive sliders. As the user adjusts the sliders, the input shape deforms in real-time to reflect the new shape dictated by the attributes.

A geometric deformation frequently induces changes in many of the attributes simultaneously. Therefore, when one attribute is adjusted in the interface and the geometry is deformed, the remaining attributes also change from their original settings. The user has the option to constrain such changes using the three buttons

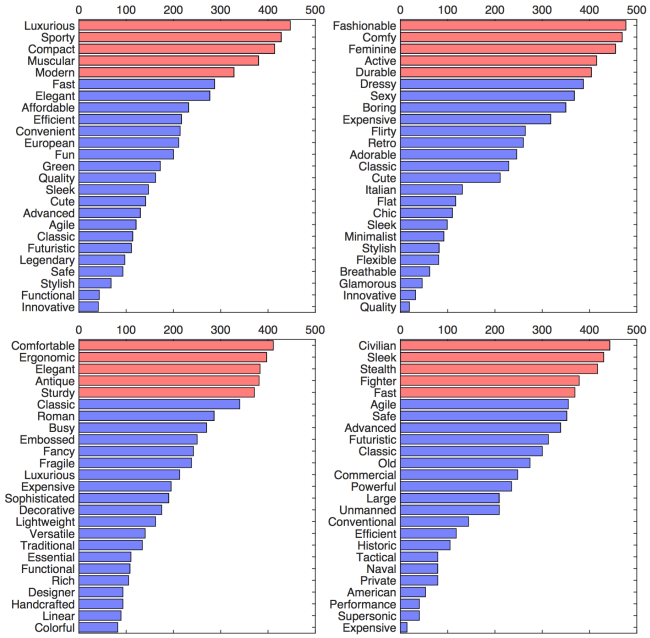


**Figure 4:** *Attributes for* Cars *(top-left),* Shoes *(top-right),* Chairs *(bottom-left), and* Airplanes *(bottom-right) sets. Over 1000 users were asked to select the most relevant five attributes for each category. The horizontal axis shows the number of users who selected that attribute. The highest ranking attributes shown in red.*

associated with each attribute. When the [+ / - / =] button is activated underneath an attribute, that attribute is constrained to only [increase / decrease / remain about the same] when other attributes are adjusted. For instance, while increasing a car's sportiness, the system can be constrained such that its compactness can only increase. This capability is made possible through a constrained path traversal formulation (Section 6.1).

**Attribute map exploration.** This mode provides an exploratory interface for shape deformation (Figure 3(b)). A color-coded 2D map is displayed for the selected attribute, with the colors denoting varying attribute strengths (red: high, blue: low). The map is created by smoothly embedding the high-dimensional shape space into 2D. It allows the user to explore different styles of shapes with similar attribute strengths, which is not possible with the 1D slider. Representative shapes are displayed at various map locations to guide the exploration. The user can *grab-and-slide* an input shape on the map, and the shape is deformed in real time to match the attribute strength at the corresponding location.

## 4 Attribute Collection

We conducted our user studies with two different kinds of users: (1) Experts: product design professionals and visual artists, (2) General public: Amazon Mechanical Turk (AMT) and in-person participants.

### 4.1 Formative User Study I: Attribute Discovery

We asked fifty experts to each provide the ten attributes that they believe are most relevant to the target product category. The overall purpose of this study was also explained, encouraging the experts to consider attributes that they think consumers would find useful for product rating. We consolidated the attribute lists, filtered out

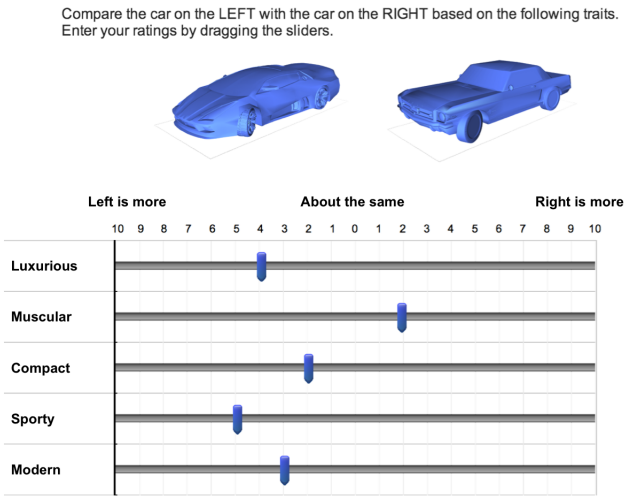


**Figure 5:** *A user response from the attribute rating study.*

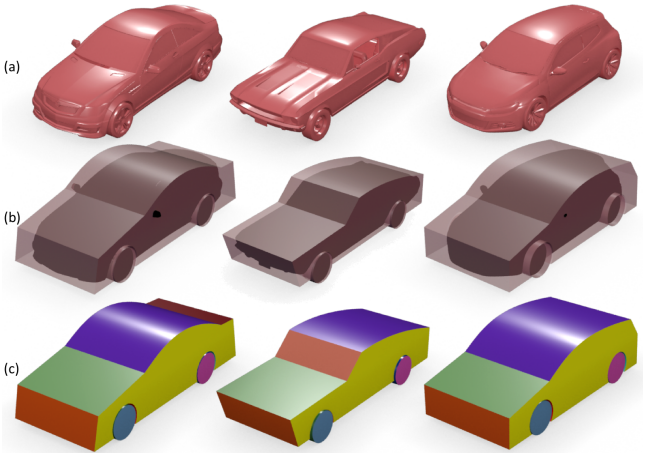


**Figure 6:** *(a) Examples from the* Cars *set. (b) Deformation handles. Note the differences in topologies. (c) Correspondence computed across the handles (matching handles colored similarly).*

synonyms, and used the resulting list to deploy an AMT study. We asked 1000 participants to choose the five attributes they would find relevant for themselves and for other consumers when rating products from each category. Figure 4 shows the resulting distribution of attributes for *Cars* and *Shoes*. We followed the well-known strategy of conflict by Schelling [1980] and asked users to evaluate others' views as well: the experts were asked to provide attributes that the general public would consider relevant, and the general public was asked to select attributes that others would consider relevant.

### 4.2 Formative User Study II: Attribute Rating

In this study, we asked participants to compare pairs of models rather than asking them to rate individual models on an absolute scale (Figure 5). The user is asked to evaluate a pair of model images for each available attribute, where slider positions indicate the relative magnitude of the user's rating.

For a dataset of 100 shapes, there are nearly 5000 possible comparisons. To make the task manageable, our system selects 95 random pairs for each task from a uniform distribution during runtime. We aggregate all tasks performed by all users to arrive at the final pairwise ratings separately for the general public (AMT) and the experts. We limit each AMT participant to a maximum of 20 tasks in order to prevent user domination in the data. The final data are collected from more than 2500 unique AMT participants and from more than 300 unique experts for the *Cars* and *Shoes*, and more than 1300 unique AMT participants for the *Chairs* and *Airplanes*. These exclude rejections per the criterion explained below. We develop our system and report results primarily based on the larger AMT dataset. User ratings are publicly available.[1]

**User reliability.** Crowdsourcing user studies require special care when user ratings are subjective or qualitative [Kittur et al. 2008]. We use two methods to filter out unreliable responses: (1) **Consistency:** For each task (95 pairwise comparisons), we duplicate five randomly chosen questions by reversing the presentation order of the models. The user sees the resulting set of 100 questions in random order. The five control questions, with five sliders each, give us 25 consistency checks. Each check requires the user to consistently identify the shape with the higher attribute strength. Users who fail five or more checks are discarded from the study. (2) **Diligence:** We expect most AMT respondents to complete a task within a similar period of time. We discard users who complete the tasks more than ten times faster than the mean completion time.

[1]Polygonal mesh models and associated semantic attribute ratings: *http://www.meyumer.com*

## 5 Attribute Learning

After the attribute ratings are collected, we embed all of the training shapes in a common deformation space, compute an attribute prediction function using the training data, and establish an inverse map from the attributes to the geometric changes in the target shape.

### 5.1 Shape Features

We compute shape features on a set of abstract surfaces that represent each shape (Figure 6) instead of using low-level polygonal mesh structures. This approach enables us to embed the input shapes in a common feature space that facilitates continuous deformation and shape synthesis. We first compute the co-constrained abstraction of each shape using a previous method [Yumer and Kara 2014], which produces a set of deformation handles (Figure 6(b,c)). A deformation handle can be an instance of the following surface types: sphere, cylinder, cone, open quadric. Open quadrics are used when deformation handles cannot be reliably represented by the former primitives.

These surface types are parameterized as follows:

- Sphere: $r, p_i | i \in \{x, y, z\}$
- Cylinder: $r, p_i, \theta_i | i \in \{x, y, z\}$
- Circular Cone: $r, \beta, p_i, \theta_i | i \in \{x, y, z\}$
- Quadric: $k_1, k_2, p_i, \theta_i | i \in \{x, y, z\}$

where $r$ represents the radius associated with the type, $\beta$ represents the conic angle, and $k_1$, $k_2$ represent orthogonal constant curvatures. These parameters form the intrinsic parameters. $(p_i, \theta_i)$ represent the absolute position and orientation of the surface relative to the axis-aligned bounding box of the shape, and they form the extrinsic parameters. A shape's feature vector consists of each surface's intrinsic parameters followed by a set of relative fea-

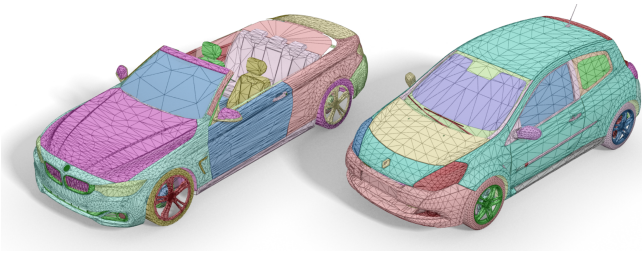

**Figure 7:** *Example shapes from the* Cars *dataset. Each shape may consist of many surface patches (colored differently), and the shapes may be topologically dissimilar.*

tures computed between the extrinsic features of all surface pairs (*e.g.,* $p_x^m - p_x^n$, $\theta_z^m - \theta_z^n$, where $m$,$n$ are two different handles).

We use a sparse vector encoding scheme to represent each feature vector. In this representation, each entry into the vector is preceded by an integer id uniquely demarcating the corresponding feature value. Sparse encoding allows shapes with different numbers of handles and handle types to be incorporated within the same formulation (Figure 6(c)). Each deformation handle attains a unique integer id based on the compatible segmentation used during shape abstraction. With this, handles that are common between two or more models attain the same id (*e.g.,* the green engine hood in Figure 6), while a handle unique to a particular model attains a unique id not shared by any other handle (*e.g.,* the trunk in the first column of Figure 6). For each model, our sparse vector representation thus encodes id-intrinsic parameter pairs for each of the deformation handles. The relative features between pairs of handles are encoded similarly by computing the integer id for each deformation handle pair and record the corresponding feature value associated with that pair. Note that if two or more models share similar handles $h_A$ and $h_B$, the id computed to represent the $(h_A, h_B)$ pair will be common across these different models.

**Previously unseen shapes.** Let $\mathcal{K} = \{(Y_i, L_i)|i = 1, \ldots, |\mathcal{K}|\}$ be the set of handles $(Y_i)$ and corresponding types $(L_i)$ computed for the set of input shapes (Figure 6(c)). These handles and their correspondances across the shape set are computed using the prior formulation [Yumer and Kara 2014]. Let $d = \{(y_i, l_i)|i = 1, \ldots, |d|\}$ be a previously unseen shape's handles $(y_i)$ and handle types $(l_i)$. We train a polynomial-kernel SVM classifier [Cortes and Vapnik 1995] on $\mathcal{K}$ and use the resulting classifier to assign a handle type $(l_i)$ to each surface $(y_i)$ of the new shape. Lastly, based on the acquired handle types, we compute the intrinsic and extrinsic parameters that results in the shape's feature vector.

**Discussion of shape features.** Previous approaches such as [Chaudhuri et al. 2013; Deng and Neumann 2008; Allen et al. 2003] rely on features computed from the mesh representation. For geometrically and topologically different shapes (Figure 7), such features either restrict these approaches to ranking or search applications, or require that vertex-level correspondences are established a priori. Our approach eliminates the need for low-level correspondences, thereby allowing geometrically and topologically different shapes to be used for learning (Figure 7).

## 5.2 Attribute Prediction

We learn a function to map the shape features to the predicted attribute scores in a two-step process. First, the relative comparison scores for the training shapes are converted to absolute attribute scores with associated reliability estimates. Second, the absolute scores are extended to a continuous scoring function for the entire feature space. This approach is a slight departure from common practice, where the scoring function is learned directly from non-numerical relative comparisons (e.g. [Freund et al. 2003; Parikh and Grauman 2011]). We found that our approach efficiently learns to model the high nonlinearity of the collected data without requiring a large number of training comparisons.

### 5.2.1 Absolute Attribute Scores for Training Shapes

We model the absolute attribute score ${}_aP_i \in \mathbb{R}$ for attribute $a$ and shape $i$ as a normal (Gaussian) distribution:

$$ {}_aP_i \sim N({}_a\mu_i, {}_a\sigma_i^2) = \frac{1}{{}_a\sigma_i\sqrt{2\pi}} e^{-\frac{(x-{}_a\mu_i)^2}{2\,{}_a\sigma_i^2}} \tag{1} $$

Because users compare shapes using a numerical scale, the relative scores may also be represented as distributions. If the (unknown) absolute scores are independently normally distributed, their pairwise differences can also be modeled as normal distributions [Steel et al. 1960]: ${}_aP_i - {}_aP_j \sim N({}_a\mu_{ij}, {}_a\sigma_{ij}^2)$, where ${}_a\mu_{ij}$ and ${}_a\sigma_{ij}^2$ are the (known) mean and variance of user-provided relative scores comparing shapes $i$ and $j$. Our modeling assumption stands up to sanity checks: the distribution moment-based skewness test [Thode 2002] shows that the pairwise comparisons conform well to normal distributions (with $\tilde{9}0\%$ of the data for all datasets passing the normality check). We observe that

$$ {}_aP_i - {}_aP_j \sim N({}_a\mu_{ij}, {}_a\sigma_{ij}^2) = N({}_a\mu_i - {}_a\mu_j, {}_a\sigma_i^2 + {}_a\sigma_j^2) \tag{2} $$

To compute the absolute attribute scores of all shapes, the overdetermined system of linear equations given by all ${}_a\mu_i - {}_a\mu_j = {}_a\mu_{ij}$ and ${}_a\sigma_i^2 + {}_a\sigma_j^2 = {}_a\sigma_{ij}^2$ pairs is solved in a least squares sense to maximize the data likelihood.

### 5.2.2 Scoring Function

After we compute absolute attribute scores for the training shapes, we map the set of features extracted from the shape deformation handles to these scores. The mapping can be extended to the entire feature space to yield a function that predicts attribute strengths for previously unseen shapes.

We modify Shepard's method [Shepard 1968; Lewis et al. 2010] to map the sparse feature vector of an arbitrary shape, $\mathbf{x}_s$, to a predicted attribute strength:

$$ \widetilde{f}_a(\mathbf{x}_s) = \sum_{t \in \mathcal{T}} \frac{w_t(\mathbf{x}_s)}{\sum_j w_j(\mathbf{x}_s)} f_a(\mathbf{x}_t) \tag{3} $$

where $\widetilde{f}_a(\mathbf{x}_s)$ is the predicted value of attribute $a$ for a shape with features $\mathbf{x}_s$, and $\mathcal{T}$ is the set of training shapes (*i.e.,* shapes for which the attribute strengths, $f_a(\mathbf{x}_t)$, are known). The weight function $w_t(\mathbf{x}_s)$ for data point $t$ is

$$ w_t(\mathbf{x}_s) = {}_ar_t \|\mathbb{1}_s \cdot \mathbb{1}_t \cdot (\mathbf{x}_s - \mathbf{x}_t)\|^{-p} \tag{4} $$

where $\mathbb{1}_i$ is the indicator function vector for the sparse feature vector of shape $i$ and its elements are equal to 1 for the features of a deformation handle if the shape possesses that handle (as determined by the id in the sparse vector representation) and 0 otherwise. $p$ is a positive real number. We choose $p > 1$, resulting in a globally

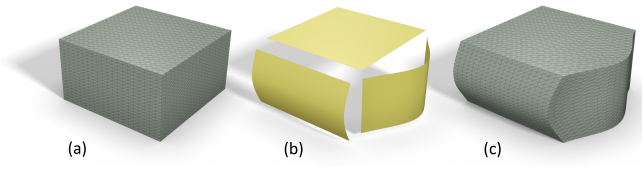


**Figure 8:** *Conformal deformation: (a) initial deformation handles, (b) a new attribute value alters the position in the feature space, hence dictating new handle parameters, (c) modified handles are optimized to establish watertight boundaries.*

continuous prediction function [Lewis et al. 2010][2]. Lastly, we also augment the weight function with a reliability constant ${}_a r_t$ defined at known data points. Because we model the attribute score at a known data point (each training shape) with a normal distribution (Equation 1), we select the distribution mean as the corresponding predicted value $f_a(\mathbf{x}_t) = {}_a\mu_t$ and define reliability proportional to the inverse of the variance ${}_a r_t = 1/\,{}_a\sigma_k^2$.

For the sparse feature vector of an arbitrary shape, $\mathbf{x}_s$, and the feature vectors of the training shapes $\mathbf{x}_t, (t \in \mathcal{T})$, the indicator function in Equation 4 allows only the handles in the training shapes that also appear in $\mathbf{x}_s$ to contribute to the attribute score of $\mathbf{x}_s$.

While RankSVM [Parikh and Grauman 2011; Chaudhuri et al. 2013] provide an alternative to our attribute scoring function, we favor our approach because of the strongly multi-modal nature of our problem. We show example shapes generated by the two methods in Section 7. Additionally, multi-modal, nonlinear approaches (*e.g.,* Gaussian Mixture Models (GMM), RankBoost [Freund et al. 2003]) could be used as an alternative to Equation 3. However, we have found our approach to provide significantly faster ($>$100x) training times over GMM. Further discussions regarding the rationale behind using our method over RankSVM and GMM can be found in Section 7.1.

### 5.3 Deformation

The attribute prediction function in Equation 3 represents a continuous scalar field over the shape space: the attribute is the potential, and the shape features are the coordinates in this space. Given a target attribute value, we use this space as a way to deform an input shape. In this section, we describe our deformation algorithm.

For an input shape, we first compute its abstract deformation handles (Figure 8(a)) using [Yumer and Kara 2014]. The intrinsic and extrinsic parameters of these handles are combined to form the shape's feature vector as described in Section 5.1.

When users interact with our system, they use semantic attributes as control parameters for deforming the shape. When an attribute is modified, our model computes the corresponding change of shape features. (This mechanism is detailed in the next section.) Recall that the features are simply parameters of handle surfaces (Figure 8(b)). We can use the updated parameters to determine new positions for all vertices of the handle surfaces, represented as polygon meshes.

The new handle configurations, however, also necessitate the computation of new handle boundaries because the feature vector does not explicitly encode boundary information. To ensure connected, watertight boundaries, we jointly optimize the final vertex positions of all handle surfaces as follows:

$$\underset{p_i}{\text{minimize}} \sum_{\{\mathcal{V}_h,\mathcal{E}_h\}\in\mathcal{H}} \left( \sum_{i\in\mathcal{V}_h} |p_i - f_i| + \lambda \sum_{j\in\mathcal{E}_h} -log\left(\frac{\beta_j}{\pi}\right)\right) \quad (5)$$

where $\mathcal{H}$ is the set of deformation handles of the shape. $\mathcal{V}_h, \mathcal{E}_h$ are the vertices and *internal* edges of the surface mesh associated with handle $h$. $p_i$ and $f_i$ are the current and target positions of vertex $i$. $\beta_j$ is the dihedral angle between the polygons that share edge $j$. $\lambda$ is a constant that controls the contribution of the smoothness term. We use the new positions of the surface handle vertices as the boundary conditions for an anisotropic 3D cage deformation method described in [Yumer and Kara 2014] (Figure 8(c)). The volumetric deformations computed for the tetrahedral mesh associated with the cage are then used to deform the embedded original mesh geometry, thereby completing the deformation process.

A target shape contains deformation handles possessed by no other training shape, those unique handles remain on the target shape and deform only due to the compatibility constraints arising from Equation 5. Section 7 will show examples of such scenarios.

## 6 Semantic Editing Interfaces

Our system has two interaction modes: (1) direct attribute editing (2) attribute map exploration. Both modes utilize the attribute prediction function given in Equation 3 to deform the shape toward the desired attribute value. However the underlying navigation mechanisms differ for the two approaches.

### 6.1 Direct Attribute Editing

The continuous attribute prediction function given in Equation 3 is not convex. However, by design, the local maxima and minima can only occur at the data points (models in the user studies) used in the construction of the function. For each attribute, this function is defined over the same geometric space $\mathbf{x}_s$; thus, an input shape's location in this space is readily known. For direct attribute editing, we compute a path for the shape in this space for each attribute in real time and map these paths to the corresponding sliders.

**Unconstrained attribute editing.** We require a continuous path between the shape variations with minimum and maximum attribute value. We employ a relaxed solution as a tradeoff between deformation smoothness and attribute monotonicity. Let $(x_s, f_a(x_s))$ be the shape's current position in the deformation space and the corresponding attribute value for attribute $a$. Let set $\mathcal{T} = \{(x_t, f_a(x_t))|k = 1, \ldots, |\mathcal{T}|\}$ be the set of shapes (features and corresponding attributes) used in the construction of Equation 3. Let set $\mathcal{T}_{low} = \{(x_t, f_a(x_t))|f_a(x_t) < f_a(x_s), x_t \in \mathcal{T}\}$ be a subset of $\mathcal{T}$ that contains shapes with attribute scores less than $f_a(x_s)$, and $\mathcal{T}_{high} = \mathcal{T} - \mathcal{T}_{low}$. We build two $k$-nearest neighbor graphs; $\mathcal{G}_{low} = \mathcal{T}_{low} \cup \{(x_s, f_a(x_s))\}$ and $\mathcal{G}_{high} = \mathcal{T}_{high} \cup \{(x_s, f_a(x_s))\}$. We then use Dijkstra's algorithm to find the shortest path connecting the shape with the minimum attribute value in $\mathcal{G}_{low}$ to $x_s$, and subsequently $x_s$ to the shape with the maximum attribute value in $\mathcal{G}_{high}$. Figure 9(a-b) illustrates the steps. The resulting path thus only passes through a subset of the models in $\mathcal{T}$. Both $\mathcal{G}_{low}$ and $\mathcal{G}_{high}$ may consist of disconnected islands (*e.g.,* $\mathcal{G}_{low}$, depicted in white edges in Figure 9(a), has two islands). In such cases, the resulting path incorporates only the islands that contain the current shape $x_s$. The two ends of this path map to the maximum and minimum slider positions of the corresponding attribute in the user interface.

A literal traversal of the raw, shortest path in Figure 9(b) typically results in sudden, visually discontinuous geometric changes. To

[2] $p = 2.3$ for all results presented in this paper

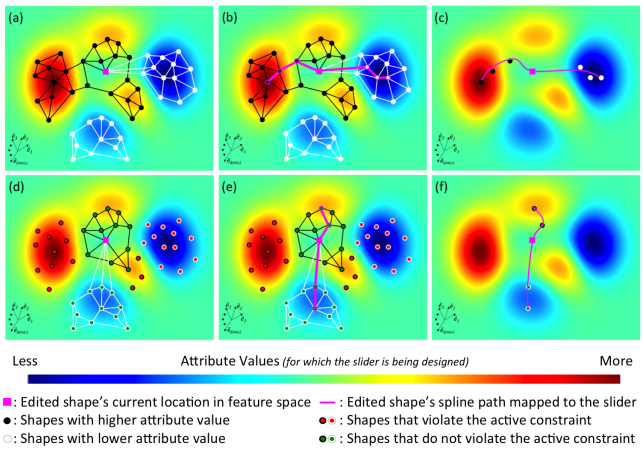


**Figure 9:** *(a) Current shape, higher- and lower- attribute clusters with resulting k-NN graphs ($N = 3$ for illustration purposes). (b) Computed exact path. (c) Smooth spline path. (d-f): (a-c) for constrained editing.*

mitigate this effect, we replace this path with a $G1$ continuous piecewise cubic spline regressed over the constituent points (Figure 9(c)). Because the traversal along this new path is still in the geometric space, we traverse the identical path in all other attribute prediction functions, read off the corresponding attribute scores, and map the results onto the slider set in the user interface. This approach helps the user explore a variety of deformations, with real-time updates on all attributes. Please refer to our supplementary video for interaction examples. Note that the traversal path is a function of the current attribute being edited as well as the current coordinates of the target shape in the feature space. As a result, the final deformation depends on the order in which the attributes are adjusted. For instance, given the original target shape, sliding $Attribute_A$ to its maximum and then sliding $Attribute_B$ to its maximum may not result in the same end shape if $Attribute_B$ had been maximized before $Attribute_A$.

**Constrained attribute editing.** For constrained editing, we eliminate the shapes that violate the constraints set by the user from $\mathcal{T}_{low}$ and $\mathcal{T}_{high}$. We then follow the same procedure explained above for unconstrained attribute editing using the reduced shape set. For example, if the user constrains attribute $b$ to *stay about the same* while he/she is editing the shape through other attributes, we eliminate shapes from $\mathcal{T}_{low}$ and $\mathcal{T}_{high}$ whose $b$ values are more than $10\%$ (determined empirically) lower or higher than the current shape's $b$ value. If the user activates the + button for attribute $b$, we only include the shapes that have a higher $b$ value then the edited shape's current configuration. Figure 9(d-f) illustrates this idea. Multiple constraints can also be activated by the user.

### 6.2 Attribute Map Exploration

Our second editing mode involves a *grab-and-slide* approach where the user explores variations of an input shape in a $2D$ height map constructed for each attribute. The colors in the map dictate the attribute's level (red: high, blue: low, see Figure 3(b)).

Let $\mathcal{X} = \{x_i | i = 1, \ldots, |\mathcal{X}|\}$ be the features of the shapes used when constructing Equation 3. We first compute a 2D embedding of $\mathcal{X}$ using locally linear embedding [Roweis and Saul 2000], resulting in $\mathcal{Y} = \{y_i | i = 1, \ldots, |\mathcal{Y}|\}$, where $y_i$ is the 2D coordinates of shape $i$ with feature vector $x_i$ (Figure 10(a)). We compute a scalar field in the embedded space $\mathcal{Y}$ (Figure 10(b)) using Equation 3, while replacing the feature vectors $x_i$ with $y_i$ in Equation 4.

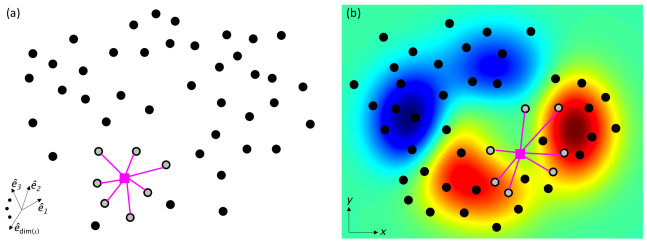


**Figure 10:** *(a) Shapes in the original feature space. (b) Shape embedding and the resulting attribute map. Nearest neighbors are preserved in the embedding.*

For a new shape $s$ imported by the user for editing, we first compute the reconstruction weights of its $k$-nearest neighbors in the feature space as follows:

$$\begin{aligned} \underset{W_i}{\text{minimize}} \quad & \sum_{i \in \mathcal{K}} (x_s - W_i x_i)^2 \\ \text{subject to} \quad & \sum_{i \in \mathcal{K}} W_i = 1 \\ & W_i > 0 \qquad \forall\, i \in \mathcal{K} \end{aligned} \tag{6}$$

where $\mathcal{K}$ is the $k$-nearest neighbors of shape $s$ and the $W_i$ are the reconstruction weights (Figure 10). The embedding of shape $s$ is

$$y_s = \sum_{i \in \mathcal{K}} W_i y_i \tag{7}$$

As the user slides shape $s$ on the map attaining new embedded location $y'_s$, we compute the target feature vector $x'_s$ as follows:

$$x'_s = (1 - c_s) x_s + c_s \sum_{i \in \mathcal{K}} \frac{||y_i - y'_s||^{-p}}{\sum_j ||y_j - y'_s||^{-p}} x_i \tag{8}$$

where $p$ is a positive real number, $c_s = |y'_s - y_s| / C$ and $C$ is a normalization constant corresponding to the maximum distance that the shape can travel on the map.

The map visualized by the user represents a 2D projection of a much higher dimensional space. Hence, visually dissimilar models might appear proximate. Although this might at times be confusing for the user, the users have usually found it straightforward to navigate this space as they explore (similar to the *Design Galleries* [Marks et al. 1997]). Our user studies (Section 7) show that the users found the map interaction useful and inspirational.

## 7 Results and Discussion

We present results from our system using AMT comparative user studies with *Cars* (131 models), *Shoes* (127 models), *Chairs* (61 models), and *Airplanes* (53 models) shape sets. For each set, the top five attributes shown in Figure 4 were used. In the *Shoes* set, both the boots and sandals were contained in the same training pool (*i.e.,* they were not parts of different shape categories). Our supplementary document contains further details of these shape sets and noteworthy differences between experts' and AMT subjects' attribute ratings.

Given a shape set, we first compute the co-constrained abstractions using [Yumer and Kara 2014]. This step takes less than a minute per shape on an i7, 8-core machine and is performed only once for

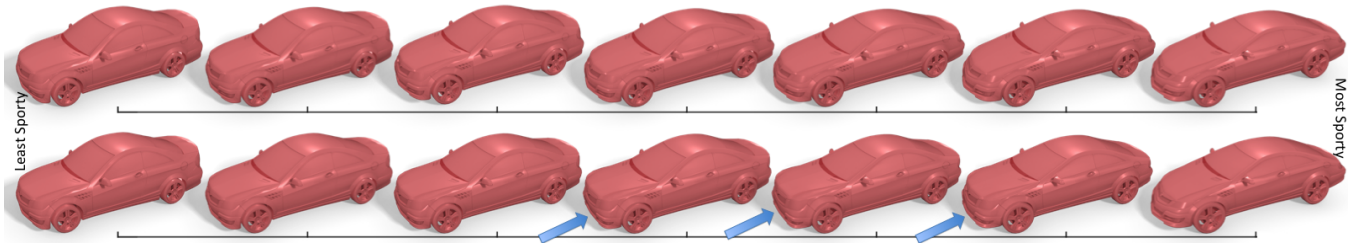


**Figure 11:** *Examples ranging from the least to the most* sporty *configuration. Top: Our system, Bottom: RankSVM. With RankSVM, the transitions can be unexpected and abrupt as shown by the blue arrows (please see the video highlighting this phenomenon).*

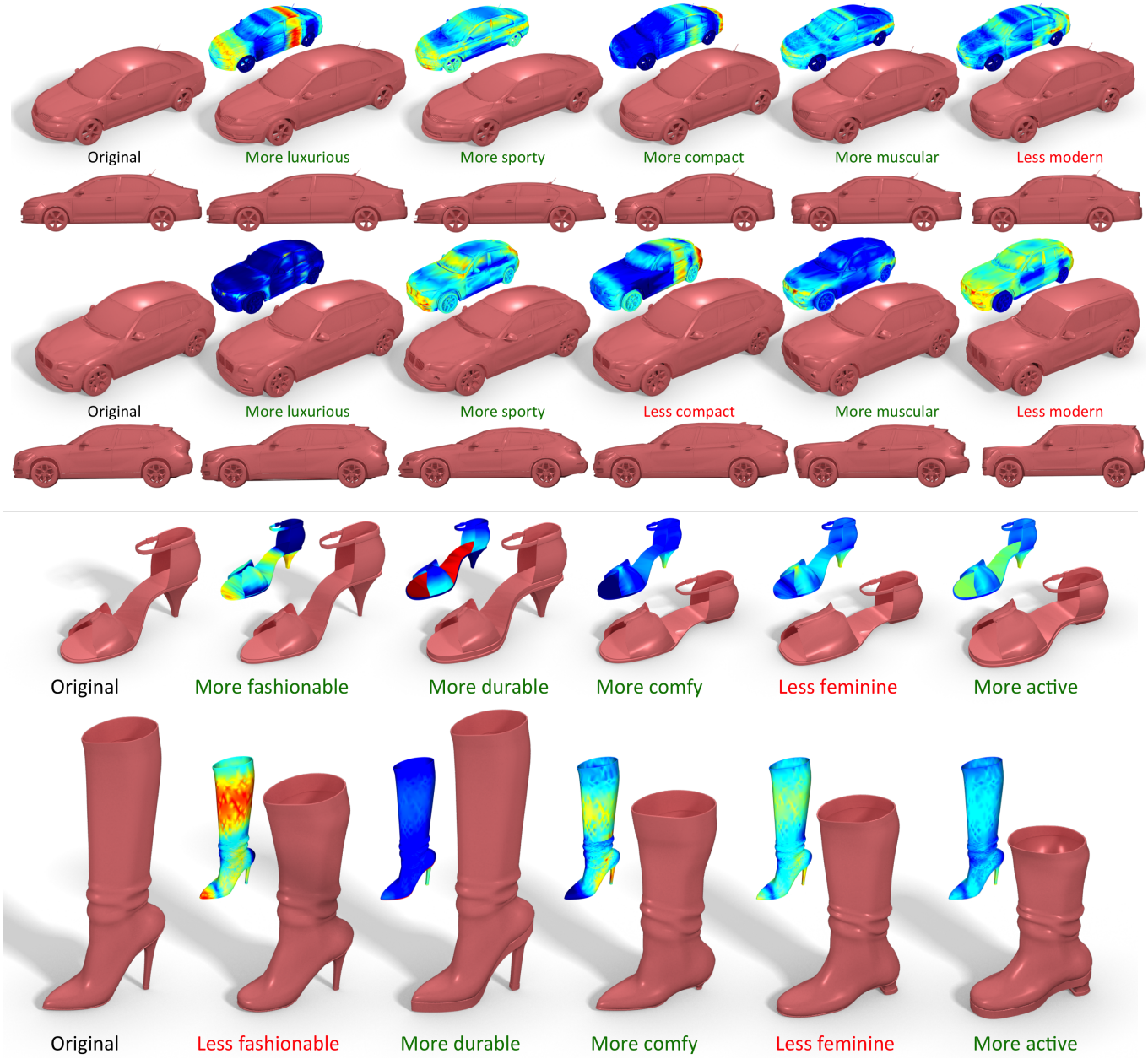


**Figure 12:** *Unconstrained editing examples with two different cars and two different shoes as starting points. Color maps indicate the amount of local deformation (normalized scale + normalized shear) for each variant: Min. deformation-* 0 1 *- Max. deformation.*

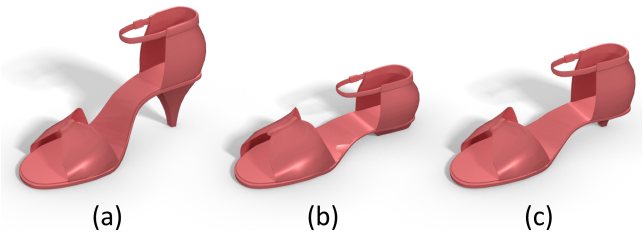


**Figure 13:** *(a) Original. (b) Unconstrained editing by increasing* Comfy *value, resulting in* 50% *decrease in* Fashionable. *(c) Constrained editing by increasing* Comfy *value while keeping* Fashionable *about the same. Note the changes in the heel and the toe.*

a dataset. Learning the mapping from the deformation space to the attribute set takes two minutes for each dataset. For an unseen input shape, we first compute its deformation handles (<1min per shape). Once the deformation handles are computed, shape editing in our system takes place at interactive rates.

**Unconstrained and constrained editing.** Figure 12 shows unconstrained editing examples from the *Cars* and *Shoes* shape sets, whereas Figure 18 shows unconstrained editing examples from the *Chairs* and *Airplanes*. In these examples, the user is adjusting the attribute noted for each variant. Figure 13 demonstrates how constraining *fashionable* alters the result obtained when the user changes the *comfy* attribute. Figure 14 shows a similar comparison for a car model. Figure 19 shows further examples of constrained shape deformation, where multiple constraints are also used.

**Pairwise ranking quality.** We compared the pairwise rating from our system to the AMT data. For a pair of shapes $(P, Q)$ with attribute scores computed by our system to be $f_a(P) > f_a(Q)$, our system's rating was marked successful if the same relationship held for the mean AMT attribute scores $f_a^{AMT}(P) > f_a^{AMT}(Q)$ for the two shapes and unsuccessful otherwise. We used a leave-one-out method: leaving one shape out from training, and testing all shape pairs containing it. This test was repeated for all semantic attributes and the results were averaged. The ranking error with our method is 11.2% for the *Cars*, 9.5% for the *Shoes*, 9.7% for the *Chairs* and 10.4% for the *Airplanes*. The ranking error with the constrained GMM method (Section 5) is 10.7%, 8.9% , 8.3%, and 9.1%, respectively for the same datasets with more than 100 times longer learning time.

### 7.1 Comparison of the Learning Approach

Our approach to learning an attribute prediction function as a continuous scalar field (Section 5.2.2) is decoupled from the initial estimation of absolute attribute scores for training shapes from relative comparisons (Section 5.2.1). We favor this approach over ranking methods based on linear projections (*e.g.,* RankSVM [Parikh and Grauman 2011; Chaudhuri et al. 2013]) because of the strongly multi-modal nature of our problem. In particular, significantly different geometric configurations may have similar attribute strengths as demonstrated in our supplementary document and video. This phenomenon is at least in part due to our choice of shape features, which are exceptionally appropriate for deformations, but are not *linearly* correlated with attributes. Chaudhuri et al. [2013] have no such constraints for their application; hence, they use an expanded feature space with additional nonlinear features to obtain satisfactory results with RankSVM. In Figure 11, and in the supplementary video, we demonstrate that the scoring function produced by RankSVM produces less meaningful deformations than our approach.

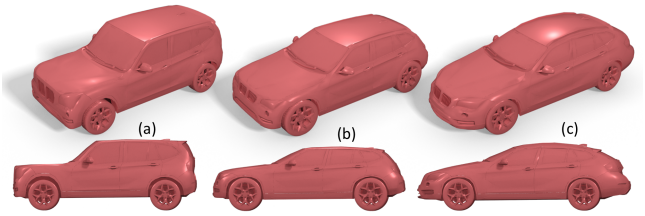


**Figure 14:** *(a) Unconstrained editing: decreasing* Modern. *(b) Constrained editing: decreasing* Modern + *constrained increasing of* Sporty. *(c) Unconstrained editing: increasing* Sporty.

**Table 1:** *Percent of participant responses that match our system's ordering.*

| Cars attribute | Match | Shoes attribute | Match |
|---|---|---|---|
| Luxurious | 79.4% | Fashionable | 86.2% |
| Sporty | 89.3% | Comfy | 94.5% |
| Compact | 94.2% | Feminine | 92.8% |
| Muscular | 91.3% | Active | 87.3% |
| Modern | 87.6% | Durable | 84.9% |
| **Chairs attribute** | **Match** | **Airplanes attribute** | **Match** |
| Comfortable | 74.5% | Civilian | 72.1% |
| Ergonomic | 83.6% | Sleek | 77.8% |
| Elegant | 73.9% | Stealth | 16.3% |
| Antique | 27.3% | Fighter | 62.4% |
| Sturdy | 86.2% | Fast | 81.4% |

One could also argue that a multi-modal, nonlinear approach (*e.g.,* Gaussian Mixture Models (GMM), RankBoost [Freund et al. 2003]) could be used to learn an alternative prediction function to Equation 3. We first experimented with a constrained GMM model where the relative attribute ratings collected from the users were used as constraints following the approach of Yumer *et al.* [2014]. We obtained results similar to our approach using constrained GMM, but with significantly longer training times (>100x) due to the large number of constraints, and the number of Gaussians approaching the number of shapes. Additionally, because we collected numerical relative comparisons, we did not need a recourse to RankBoost with a loss function based on non-numerical comparisons. Our method is similar to Mixture of Experts approaches [Yuksel et al. 2012] with a fixed number of Gaussian process expert learners. This commonality explains the similarity of the results with a GMM where the number of Gaussians approaches the number of shapes involved in the user study.

### 7.2 Evaluation User Study I

We conducted a user study on AMT to assess how well our system's shape scoring matches the scores given by the participants. As shown in Figure 15, participants were presented with three versions of a shape and were asked to rank the shapes with respect to the prescribed attribute (most to least). The three versions were presented to the participant in a random order and corresponded to the attribute's *maximum*, *minimum*, and *middle* slider positions. Each participant was presented with ten such questions. Eight of the questions were unique, while the remaining two were duplicates, with the triplets presented in a different order. We rejected a participant if the participant did not pass the diligence criterion described in Section 4.2 or if the participant's responses to the two duplicate questions did not match. After excluding 27% of the participants based on the rejection criteria, we compiled the results from over 1200 questions for each of the five attributes of all four datasets.

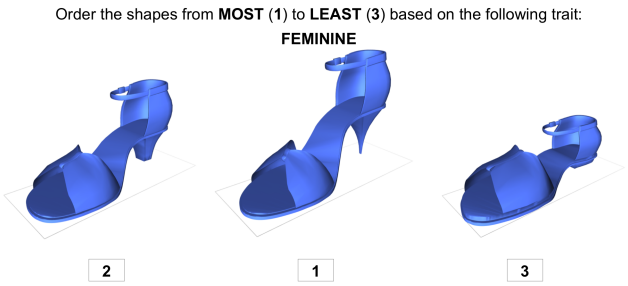


**Figure 15:** *A question in user study I and the participant's correct answer.*

A match between our system and a participant was marked successful only if the participant's ordering of the shapes exactly matched ours. The rate of successful matches is shown in Table 1, where the baseline (chance) success rate is $1/6 = 16.7\%$

### 7.3 Evaluation User Study II

We administered a second, in-person study to evaluate how well our system was received by the participants using the systems created with the data from the two larger datasets (*Cars* and *Shoes*). Each participant was given a brief description of the system. We then asked the participants to explore variations of five different shapes to arrive at final shapes to their satisfaction. We restricted the participants' interactions to one minute per shape. By doing so, our primary goal was to measure participants' satisfaction with the models they generated in a time-restricted session. At the end, participants were asked to complete a survey involving evaluations on a Likert scale.

The results collected from 41 non-expert participants (*i.e.,* no prior professional experience or training related to shape modeling or design) are summarized in Figure 16. The majority of the participants were satisfied with their models and did not find the system cumbersome despite their short interactions with our system.

### 7.4 Limitations and Future Work

Our approach produces variations of existing shapes rather than discovering new solutions in the design space. During deformation, the maximum extent that a shape can be deformed using a prescribed attribute depends on the training shapes that carry the extreme value of the attribute. As new shapes are introduced to the training set or existing ones are removed, the amount and nature

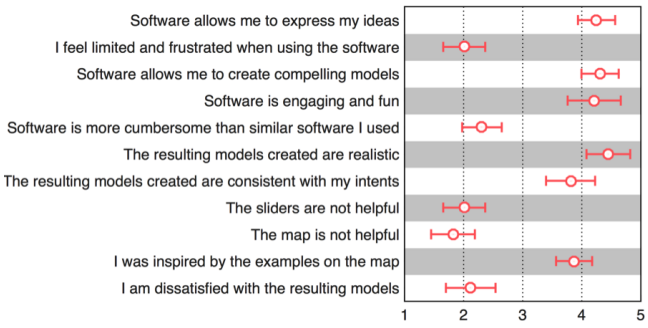


**Figure 16:** *Results of user study II. 1 - strongly disagree, 3 - neutral, 5 - strongly agree. Error bars indicate one standard deviation.*

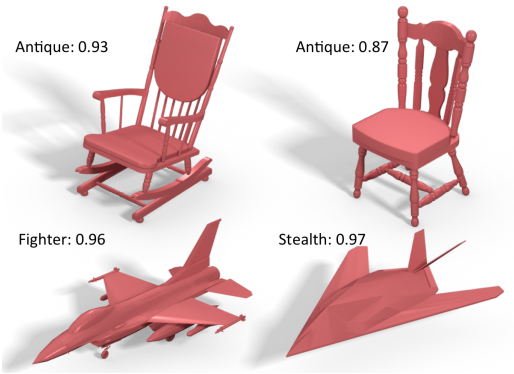


**Figure 17:** *Examples of original shapes from the three semantic attributes where our method exhibits limitations.*

of the deformation an input shape undergoes may change. Additionally, during constrained editing, the user may *overconstrain* the deformation such that no appropriate training shapes may remain for the construction of the deformation path (Section 6.1). While we have not encountered such a scenario, an insufficient number of training shapes or an abundance of controllable attributes may result in such failure modes. Further work is necessary to quantify the probability of overconstraining as a function of these two parameters.

Our method utilizes the deformation handles introduced by Yumer and Kara [2014]. Hence, it can only capture and deform details that are volumetrically significant in the shape's geometry. Figure 17 shows that *Antique* chairs are mostly differentiated by finer details they exhibit on the legs and backrest components. Our method fails to capture these details as demonstrated in Figure 18 (making the original chair more *Antique* does not significantly alter the geometry). Moreover, the clustering of deformation handle surfaces also plays a key role in our system. For instance, *Stealth* airplanes (Figure 17) consist of unique abstraction surfaces not shared by other airplanes in the database. Our system fails to cluster the abstraction surfaces of such airplanes together with the corresponding deformation handles of other airplanes. This results in insignificant deformations when the original airplane is made more *Stealthy* (Figure 18).

Our current attribute rating, learning, and deformation methods require that input shapes are deformed in their entirety without the option to modify *parts* of the shape. For instance, although our method does a fair job in deforming a civilian airplane into a fighter (Figure 19), there are many additional parts on a fighter airplane (Figure 17). Additionally, our approach only considers geometry and ignores other semantically relevant information such as appearance (*e.g.,* color and texture). A formulation that incorporates such channels of information may help establish a richer and more robust mapping between product features and semantic ratings.

## 8 Conclusion

We introduce an approach for semantic shape editing using attribute ratings learned from pairwise shape comparisons. Our approach enables users to manipulate 3D shapes using a set of attribute sliders or by navigating a set of attribute maps. The mapping from attributes to geometry is dictated by ratings crowdsourced from the general public. As such, our method can accommodate new shapes and new user ratings, thereby allowing it to adapt to evolving shape preferences, different user bases, and the availability of new semantic attributes.

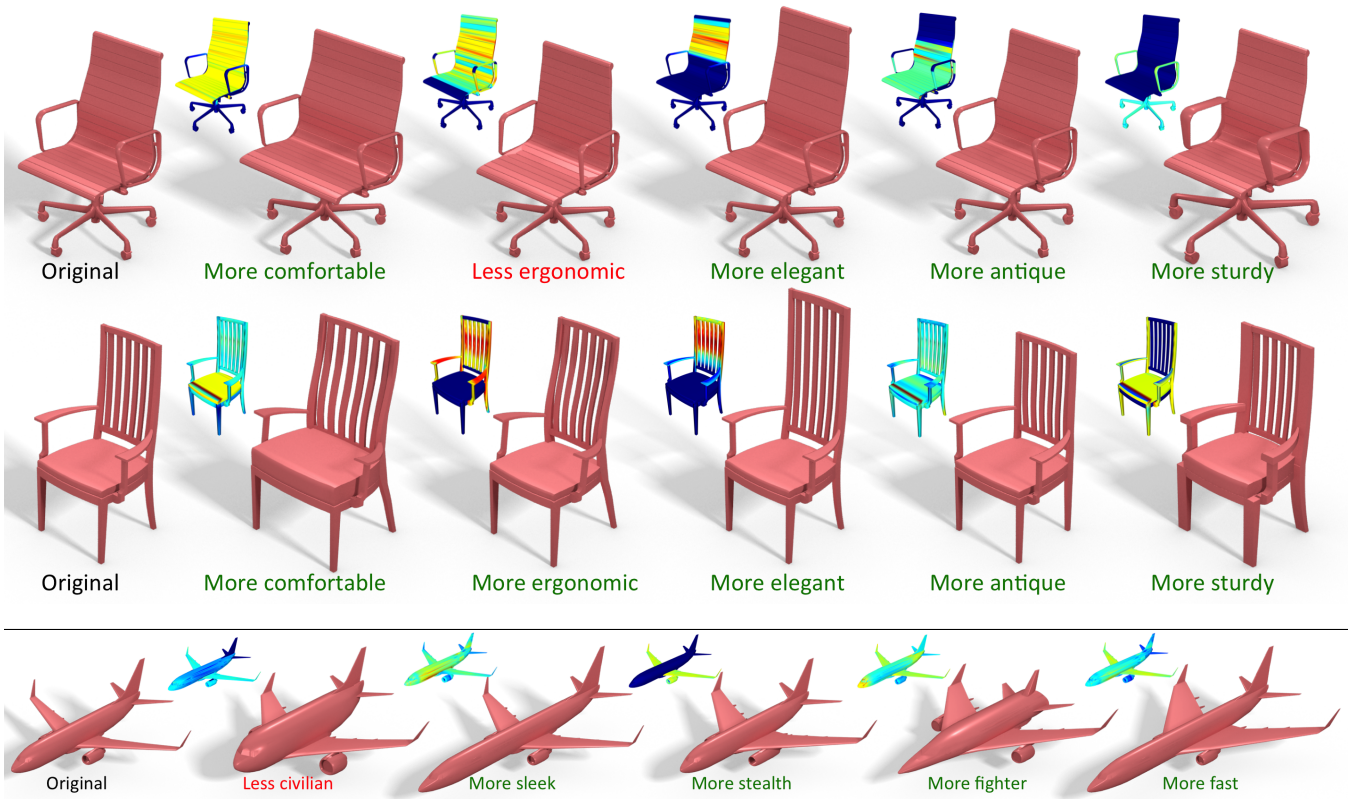


**Figure 18:** *Unconstrained editing examples from the chairs and airplanes datasets. Color maps indicate the amount of local deformation (normalized scale + normalized shear) for each variant: Min. deformation-* 0 1 *- Max. deformation.*

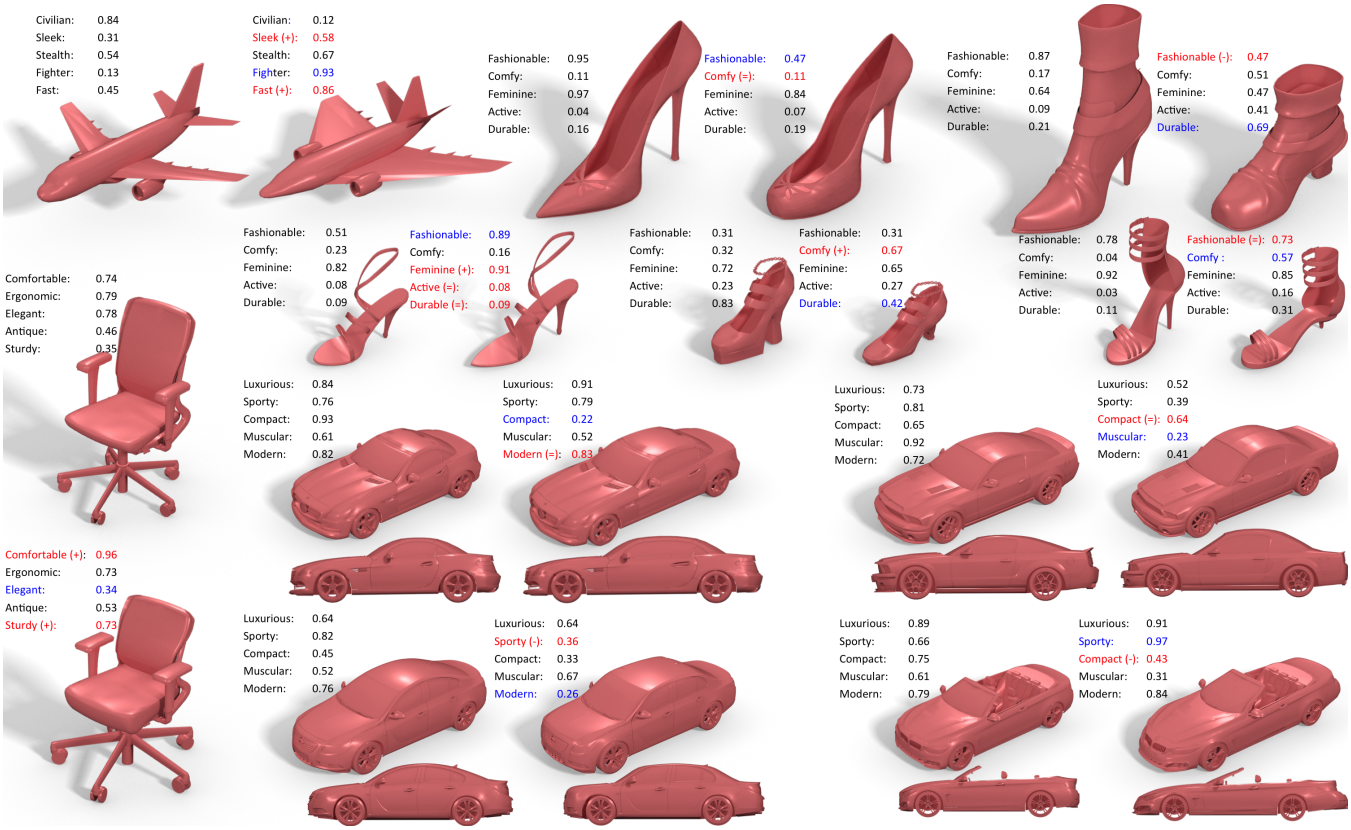


**Figure 19:** *Shapes semantically edited with our system. Blue: Edited semantic attribute. Red: Constrained attributes during editing.*

## Acknowledgments

We thank Niloy J. Mitra, Scott Hudson, Elizabeth Carter, and the anonymous reviewers for the valuable discussions. This work is partially funded by NSF CMMI - 1235427.

## References

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM Trans. Graph.*, vol. 22, 587–594.

AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In *CGF*, vol. 33(2), 125–134.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *ACM SIGGRAPH*, 187–194.

BOKELOH, M., WAND, M., KOLTUN, V., AND SEIDEL, H.-P. 2011. Pattern-aware shape deformation using sliding dockers. In *ACM Trans. Graph.*, vol. 30, 123.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for realtime modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE TVCG 14*, 1, 213–230.

CHAUDHURI, S., KALOGERAKIS, E., GIGUERE, S., AND FUNKHOUSER, T. 2013. AttribIt: Content creation with semantic attributes. In *ACM UIST*, 193–202.

CORTES, C., AND VAPNIK, V. 1995. Support-vector networks. *Machine Learning 20*, 3, 273–297.

DENG, Z., AND NEUMANN, U. 2008. *Data-Driven 3D Facial Animation*. Springer.

FREUND, Y., IYER, R., SCHAPIRE, R. E., AND SINGER, Y. 2003. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research 4*, 933–969.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iWires: an analyze-and-edit approach to shape manipulation. In *ACM Trans. Graph.*, vol. 28, 33.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. Consistent segmentation of 3D models. *Computers & Graphics 33*, 3, 262–269.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph. 31*, 4, 55.

KITTUR, A., CHI, E. H., AND SUH, B. 2008. Crowdsourcing user studies with Mechanical Turk. In *ACM SIGCHI*, 453–456.

KOVASHKA, A., PARIKH, D., AND GRAUMAN, K. 2012. Whittlesearch: Image search with relative attribute feedback. In *IEEE CVPR*, 2973–2980.

KRAEVOY, V., SHEFFER, A., SHAMIR, A., AND COHEN-OR, D. 2008. Non-homogeneous resizing of complex models. In *ACM Trans. Graph.*, vol. 27, 111.

LAFFONT, P.-Y., REN, Z., TAO, X., QIAN, C., AND HAYS, J. 2014. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Trans. Graph. 33*, 4, 149.

LEWIS, J., PIGHIN, F., AND ANJYO, K. 2010. Scattered data interpolation and approximation for computer graphics. In *ACM SIGGRAPH ASIA Courses*.

MARKS, J., ANDALMAN, B., ET AL. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. In *ACM SIGGRAPH*, 389–400.

MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. 2013. Structure-aware shape processing. In *Eurographics STARs*, 175–197.

O'DONOVAN, P., LĪBEKS, J., AGARWALA, A., AND HERTZMANN, A. 2014. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graph. 33*, 4, 92.

ORSBORN, S., CAGAN, J., AND BOATWRIGHT, P. 2009. Quantifying aesthetic form preference in a utility function. *Journal of Mechanical Design 131*, 6, 061001.

OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. Graph. 30*, 4, 33.

PARIKH, D., AND GRAUMAN, K. 2011. Relative attributes. In *IEEE Conference on Computer Vision*, 503–510.

ROWEIS, S. T., AND SAUL, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–2326.

SCHELLING, T. C. 1980. *The strategy of conflict*. Harvard University Press.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *ACM Nat. Conf.*, 517–524.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph. 30*, 6, 126.

STEEL, R. G., TORRIE, J. H., ET AL. 1960. Principles and procedures of statistics. *Principles and procedures of statistics.*.

TAO, L., YUAN, L., AND SUN, J. 2009. Skyfinder: attribute-based sky image search. In *ACM Trans. Graph.*, vol. 28, 68.

THODE, H. C. 2002. *Testing for normality*, vol. 164. CRC Press.

VAN KAICK, O., XU, K., ZHANG, H., WANG, Y., SUN, S., SHAMIR, A., AND COHEN-OR, D. 2013. Co-hierarchical analysis of shape structures. *ACM Trans. Graph. 32*, 4, 69.

XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. Graph. 31*, 4, 57.

YUKSEL, S. E., WILSON, J. N., AND GADER, P. D. 2012. Twenty years of mixture of experts. *NNLS 23*, 1177–1193.

YUMER, M. E., AND KARA, L. B. 2012. Co-abstraction of shape collections. *ACM Trans. Graph. 31(6)*, 166:1–166:11.

YUMER, M. E., AND KARA, L. B. 2014. Co-constrained handles for deformation in shape collections. *ACM Trans. Graph. 33(6)*, 187:1–187:11.

YUMER, M. E., CHUN, W., AND MAKADIA, A. 2014. Co-segmentation of textured 3D shapes with sparse annotations. In *IEEE CVPR*, 240–247.

ZHENG, Y., FU, H., COHEN-OR, D., AU, O. K.-C., AND TAI, C.-L. 2011. Component-wise controllers for structure-preserving shape manipulation. In *CGF*, vol. 30, 563–572.