Co-Constrained Handles for Deformation in Shape Collections

Mehmet Ersin Yumer Levent Burak Kara

Carnegie Mellon University

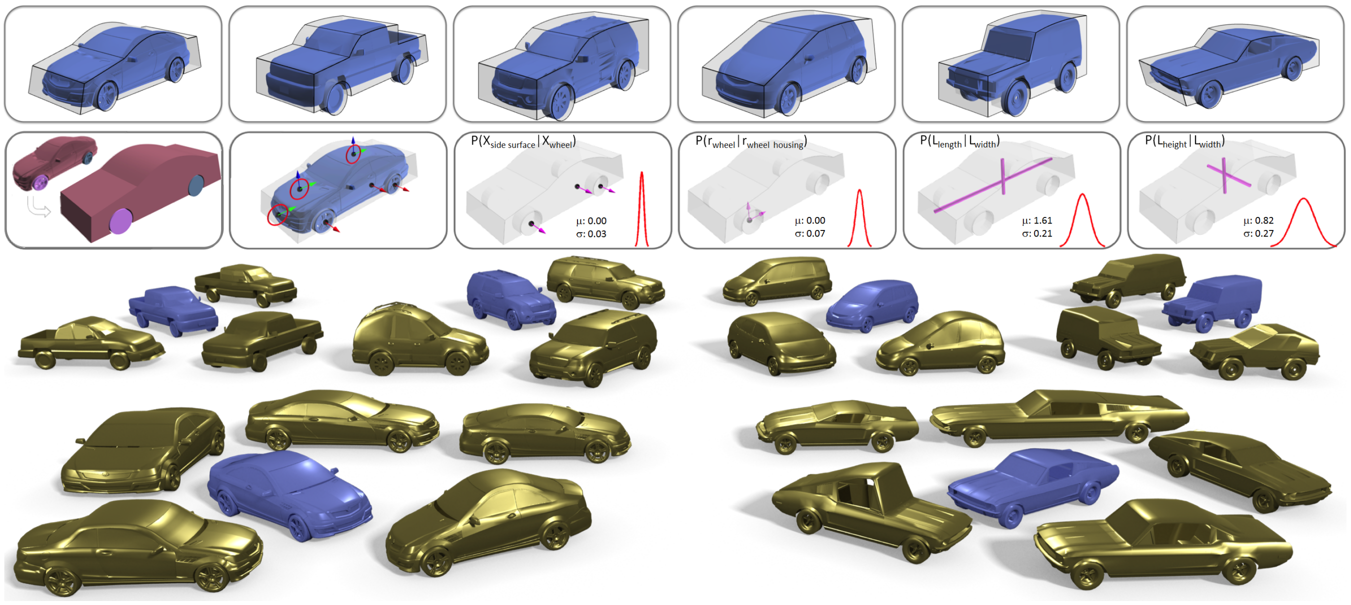

Figure 1: *Top: Co-constrained handles (light gray) learned from a dataset of 19 car models. Middle left to right: Co-constrained abstraction of a car, the set of shape manipulators on the handles, and the four most significant soft constraints learned in the form of conditional probability distributions. Bottom: A variety of deformed shapes (gold) generated using the constrained handles.*

Abstract

We present a method for learning custom deformation handles for an object, from a co-analysis of similar objects. Our approach identifies the geometric and spatial constraints among the different parts of an object, and makes this information available through abstract shape handles. These handles allow the user to prescribe arbitrary deformation directives including free-form surface deformations. However, only a subset of admissible deformations is enabled to the user as learned from the constraint space. Example applications are presented in shape editing, co-deformation and style transfer.

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

Keywords: co-constraints, 3D shape deformation, shape sets, co-deformation, constraint learning, style transfer.

Links: DL PDF

1 Introduction

Man-made shape deformation is a central task in digital content creation. Intuitive and fast deformation paradigms enable digital artists to build upon prior work, product designers to explore shape variations, and engineers to incorporate new functionality into prior designs. However, man-made objects are particularly difficult to manipulate in meaningful ways, as these models often embody a large set of tacit constraints stemming from functional, aesthetic, structural, and fabrication considerations. Moreover, due to the eventual manufacturing and assembly methods, these models are often constructed in multiple sub-assemblies, giving rise to many disconnected and intersecting geometric elements. As a result, translating the high-level design intentions into geometric directives is often challenging, and moreover, such intentions cannot be reverse-engineered reliably from a single instance of the designed artifact.

As a result, unlike organic shape modeling which may be governed by well-understood energy minimizers, man-made shape manipulation requires a deeper analysis prior to editing. Previous approaches to man-made shape deformation analyze a single model and present the user a set of on-object handles [Gal et al. 2009], or part-based bounding-volume handles [Kraevoy et al. 2008; Zheng et al. 2011]. However, such single-model analysis is only useful for extracting low-level geometric relationships readily computable from a given model. We contrast this with the observation that many design constraints common to a product family may be implemented in geometrically unique ways on the individual models of the set (Figure 2). We therefore argue that an analysis of man-made objects for deformation purposes requires an understanding of the set of shapes that the model belongs to.

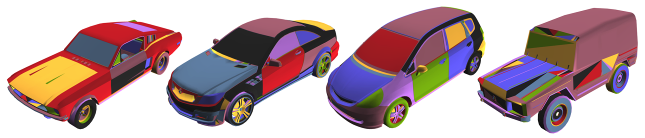

Figure 2: *Models from the same shape family may exhibit many topological and geometrical differences. Above, the wheel housings are either purely cylindrical, sharp cornered, or free-form.*

Co-Constraints. We address the problem of learning meaningful constraint spaces for a family of shapes. Figure 1 illustrates the idea. Given a collection of compatibly segmented models [Sidi et al. 2011; Golovinskiy and Funkhouser 2009] as input, an abstract shape proxy is computed using the techniques described in [Yumer and Kara 2012]. Initially, each distinct segment of a model is abstracted separately, giving rise to multiple abstract segments in a model. We then apply feature extraction and clustering to the individual surfaces of these abstract segments in preparation for a statistical shape analysis. The subsequent analysis learns the intrinsic and extrinsic shape variations across the similarly clustered surfaces in the form of probabilistic constraints. This results in a refined abstract model we call *co-constrained abstraction*, whose eventual surfaces serve as deformation handles (Figure 1, top and middle rows). The user deforms the object using through these handles. When a surface is deformed, the system works to preserve the learned constraints by automatically adjusting all other handles through an optimization scheme. Our work differs from those involving deformation spaces [Sumner et al. 2005; Kry et al. 2002]. Instead of computing a space spanned by the different deformations of a canonical model, we reveal the shape constraints from a set of topologically and geometrically distinct models. This allows any model in the shape collection to be deformed according to the identified constraints, without undesirably inheriting the shape characteristics unique to the other models in the set.

Shape Deformation. Our handles are the surfaces of co-constrained abstractions (Figure 1). The user interacts with these handles through manipulators or free-form sketch input to apply a deformation. As shown in Figure 3, our handles are not on-model handles, as man-made objects often consist of many disconnected parts. An important feature of our approach is thus the deformation transfer from the handles to the embedded object. We accomplish this using a physically-based deformation method introduced in Section 5. This method deforms the underlying model such that the boundary of the volume may undergo an arbitrary free-form deformation. Prior to deformation, we compute an anisotropic stiffness field that maps original shape features into a propensity for deformation. This helps salient shape features of the original model to remain intact relative to the regions devoid of features.

Utility. Our method is geared toward applications involving guided shape editing, where the learned statistics are enforced as soft and hard constraints. Since our method can register all handles of the models in the shape set, it can be used to automatically propagate the deformations applied to a single model to the entire shape set concurrently (*i.e., co-deformation*). Likewise, the inter-model relationships enable style transfer, where the shape characteristics learned from a database can be applied to arbitrary models.

Contributions. Our primary contributions are as follows:

- A method to induce shape constraints in a collection through shape abstraction.
- Synthesis of co-constrained shape manipulators in the form of handles for each shape with established links throughout the shape collection.

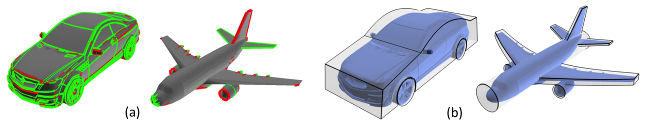

Figure 3: *(a) Part boundary (green) and crease edges (red). For complex man-made shapes, these handles can be overwhelming or fail to reveal salient deformation handles. They may be insufficient in cases of large smooth areas (airplane). (b) Our handles computed as co-constrained abstractions provide salient handles.*

- Abstract volumetric style transfer by deforming a shape automatically using the statistics from a dataset that represents a target style.
- Co-deformation of multiple shapes that belong to the same dataset through single shape editing.

2 Related Work

2.1 Shape deformation

Deformation of organic shapes. To date, most deformation techniques fall into this category. These methods primarily operate on a polygonal model, and focus on preserving surface details and smoothness through an energy function [Botsch and Sorkine 2008]. Similar to our volumetric deformation involving variable elasticity to account for man-made shape details, [Popa et al. 2006] use variable elasticity for material aware deformation. Sorkine *et al.* [2007] aim to achieve locally smooth and globally rigid deformations. For an overview of earlier organic shape deformations refer to [Botsch and Sorkine 2008]. Chao *et al.* [Chao et al. 2010] used nonlinear elasticity with improved artifact handling over linear models. In these methods, the deformation seeks to minimize an energy function describable in terms of the bulk or differential properties of the model. In the domain of man-made shapes, however, these functions often fail to capture the tacit shape constraints commonly expected after a manipulation.

Deformation of man-made shapes. Digital models of man-made shapes exhibit many disconnected and intersecting sub-assemblies, discontinuous features and poor triangulation (Figure 2). Early works by Botsch and Kobbelt [2004] allow the user to author constraints on a complex model and deform it using a basis function resulting from the constraints. Kraevoy *et al.* [2008] presents a method for axis-aligned, non-uniform scaling of man-made shapes. They embed the model into a rectangular prism and when resized, the *vulnerable* parts of the model deform less compared to the rest of the model. In this work, we use a vulnerability formulation similar to theirs to instantiate an anisotropic stiffness field. Using this field as input, we describe a new algorithm in Section 5 for free-form deformations, in addition to the axis-aligned non-uniform scalings described in [2008]. Bokeloh *et al.* [2011] introduces a pattern aware deformation method where they extract a set of *sliding dockers* which can be added or removed to suit the applied deformation, they also introduce a new algebraic model for parametric analysis and editing of shapes that exhibit patterns [Bokeloh et al. 2012]. Xu *et al.* [2009] define deformation degrees-of-freedom at the model joints, which are computed from an analysis of slippable motions. This enables a realistic deformation of shapes that exhibit dynamic joints.

In an inspiring work, Gal *et al.* [2009] utilize the sharp features on a model as deformation handles, and use an *analyze and edit* approach to propagate the deformations to the entire model. This method performs well on mechanical shapes with well-defined

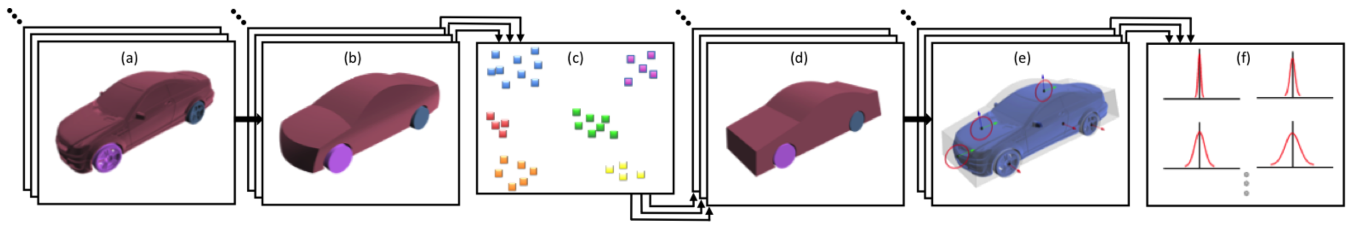

Figure 4: *Key steps of our approach. (a) Input is a dataset of segmented models (only one model is shown), (b) Initial segment abstractions are created. (c) Distinct surfaces of the segment abstractions, aggregated from all the models in the dataset, are clustered and constraints are computed. (d) This results in a co-constrained abstraction for each model. (e) Handles are instantiated on the surfaces of the co-constrained abstractions. (f) The handles encode the current configuration with respect to the shape collection (normalized Gaussian distribution of learned soft constraints), also serve as a means to propagate constraint information to compute additional constraints for deformation.*

sharp features that can serve as handles. Most relevant to our method, Zheng *et al.* [2011] extend the *analyze and edit* method to volumetric component controllers. They generate controllers for the parts of the model using volumetric primitives, and use an optimization step that resolves inter-model constraints. Our method differs from these methods in that we compute the relevant constraints from a family of shapes. Our volumetric constructs that lead to the deformation handles can accommodate more complex manipulations such as free-form surface deformations, whereas component controllers in [Zheng et al. 2011] are limited to non-uniform resizing.

2.2 Shape sets

Co-segmentation and shape matching. Shape segmentation is of primary importance when extracting low-level geometric relationships. Golovinskiy and Funkhouser [2009] segment the models in a set through shape alignment primitive clustering. Sidi *et al.* [2011] use unsupervised clustering based on initial per-shape segmentation, whereas Huang *et al.* [2011] exploit joint segmentation for generating consistent segmentations of single manifold objects. Wang *et al.* [2012] introduce a semi-supervised approach that utilizes limited user input with descriptor space clustering. In recent work, Van Kaick *et al.* [2013] introduce a hierarchical approach to segmentation. This approach provides multiple layers of correspondences for a set of shapes instead of a single segmentation. In this work, we utilize a modified version of [Huang et al. 2011] to segment the models in a collection. Each segmented model may exhibit a number of segments common to other models, as well as segments that may be unique to itself. These segmented models constitute the input to our system.

High level co-analysis. There exists a large body of works that utilize a co-analysis of shapes for high level shape operations. Xu *et al.* [2010] describe a style-based shape clustering method, from which a style transfer approach is demonstrated. Kalogerakis *et al.* [2012] present a generative model learned from a shape set. The set can be expanded with newly synthesized instances emanating from the same family. Using a genetic algorithm, Xu *et al.* [2012] synthesize novel shapes from an analysis of existing shapes. Yumer and Kara [2012] introduce a co-abstraction method where the models in a collection are geometrically simplified to the maximum extent possible, while preserving the distinguishing characteristics of each model. Kim *et al.* [2013] introduce a method to construct cuboid model templates in large collections. Ovsjanikov *et al.* [2011] compute a descriptor space to define a set of prismatic templates for the shape parts, and use the manipulable templates to efficiently browse the shape collection. Recently, Fish *et al.* [2014] introduced a a system where configurations of a shape family are learned as geometric distributions, referred to as the meta-representation. They assume previously known co-segmentation, and their guided editing is limited to relatively rigid transformations and uniform resizing. However, our system enables a framework where not only part correspondence, but also abstract surface correspondences are computed. Such correspondence then enables us to introduce a finer shape editing system where rigid transformations and uniform resizing, but also non-uniform resizing, and arbitrary free-form deformation is enabled.

Proposed approach. All previous shape deformation works focus on editing singular shapes, whereas we take a different approach and interlace information extracted from a shape set into singular and collective shape editing. Our analysis of the shape collection results in a set of handles that encapsulate the constraints among the segments of the models. We compute a set of abstract handles for the identified segments in the model. The shapes of these handles (and thus the volume they enclose) are unique to each individual model. However, the nature of the deformations enabled by such a handle is common across all instances of that same handle computed on the other models. This decoupling of handle shapes (unique to a model) versus handle operations (common to all instances on every model) allows each model in the collection to be surrounded by custom deformers, while the results of the deformations remain congruent to the constraints dictated by the collection.

3 Overview

Figure 4 shows the major units of our approach. The input to our method is a set of *compatibly* segmented, and aligned models [Sidi et al. 2011; Golovinskiy and Funkhouser 2009][1]. This is less restrictive than a consistent segmentation, where each label has to appear in all the models. As such, not all bicycles in a collection are required to have pedals, but the ones that have pedals should compatibly have the same segment label for their pedals; *i.e.,* they should be semantically similar. From this input, our method identifies a number of *handles* for each model, together with a set of constraints that govern these handles' behaviors. Geometrically, each handle corresponds to a distinct *surface*, which is obtained from a shape abstraction algorithm. To manipulate a model, the user either interacts with a manipulator instantiated on the handle or sketches the new silhouette curves in 3D to reshape the handle (Figure 5). Finally, the deformed handles drive a feature-sensitive volumetric deformation algorithm. This suitably deforms the original geometry embedded in the closed volume bounded by the handles.

[1]Compatible segmentation does not require the models to all share the same semantic parts. However, it requires all semantically equivalent parts to be labeled similarly.

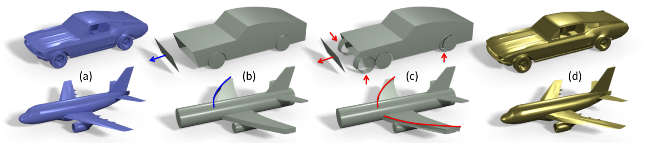

Figure 5: *(a) Input models. (b) User prescribed deformation (top: translation, bottom: silhouette sketching). (c) Constraints resolved by the system, forming the full set of boundary conditions for our volumetric deformation system. (d) Final models.*

	Constrained Mean-Shift	Semi-Supervised GMM
Cars	89%	93%
Airplanes	92%	94%
Robots	84%	89%
Bicycles	91%	93%
Lamps	87%	89%

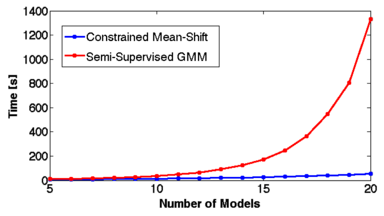

Figure 6: *Constrained Mean-Shift vs. GMM. Left: Average accuracies over different datasets. Clustering is performed with 15 models in the each dataset. Right: Time-complexity vs. number of models in the clustering averaged over the five different datasets.*

4 Handle Synthesis and Constraint Learning

4.1 Preprocessing

Segmentation. To generate the compatible segmentation, we use a guided version of the unsupervised joint shape segmentation [Huang et al. 2011]. The method presented by Huang *et al.* [2011] is optimized for closed, uniformly meshed manifolds. Since man-made shapes are composed of many disconnected parts that often do not form a single connected topology, we implement the following alterations: We adopt shape signatures from [Yumer et al. 2014], and skip the initial patch creation, and instead create cuts through the crease edges of the model. Combined with the already existing separate parts, this procedure results in a better initial segmentation for man-made objects. The resulting segmentation is cleaned-up through a user-interface that collects user assistance in cases of sub-optimal segmentations[2] We also compute the global symmetries associated with the model [Mitra et al. 2006].

Segment abstraction. We use the method presented by Yumer and Kara [2012] to create an abstraction for *each segment* (five for the car model in Figure 4; one for the body, and one for each of the four wheels). We do not create a hierarchy of abstractions as they originally proposed, but we increase the initial coverage percentage to 90% so that the computed abstraction faithfully represents its target segment. We treat each segment of a model separately. Hence, for a single model the number of abstractions equals the number of segments on the model.

4.2 Surface Clustering

Each processed model embodies multiple abstracted segments, each of which is composed of a multitude of distinct surfaces. The goal in this step is to collect all the surfaces originating from all the abstractions, and identify a clustering of these surfaces. Each cluster will allow geometric statistics to be computed for the constituent surfaces which, in turn, will lead to the shape constraints. Prior to clustering, all models are uniformly scaled such that their axis-aligned bounding box diagonal has a length of unity.

The cluster analysis is performed in two steps: (1) Seed surface clusters originating from equivalent[3] segments, and (2) Global clustering that links the seed clusters, thereby establishing the constraints between semantically dissimilar segments.

Seed surface clusters. In the first step, we compute a seed clustering of the surfaces across the models in the collection through a coarse alignment of equivalent segments' abstractions. We anisotropically align the bounding boxes of the abstractions, *i.e.,* we translate and non-uniformly scale the abstractions until the positions and sizes of their bounding boxes match. Next, we compute the following pairwise distances among the constituent surfaces:

$$D_{IJ} = D_{JI} = \sum d_{iJ}/n_i + \sum d_{jI}/n_j \qquad (1)$$

where D_{IJ} is the mean of distances from surface I to surface J, d_{iJ} is the minimum distance from point i on surface I to surface J, n_i is the number of points uniformly sampled on surface I. Note that D_{IJ} is conceptually similar to the mean Hausdorff distance. If, $D_{IJ} \leq 0.1\sqrt{2Area(I) + 2Area(J)}$, I and J are clustered together. This provides a seed clustering of the abstraction surfaces associated with the geometrically similar segments in the collection. Once the surface clusters are computed, we revert to the original, non-scaled versions of the surfaces, and extract a feature vector describing the attributes of each surface in the cluster. This vector is composed of: Three principal component vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ encoding the distribution of the vertices sampled from the surface, average local area-weighted outward normal vector of the surface $\mathbf{n}$ (this normal can be uniquely determined since these surfaces are the boundaries of closed volumes), and the first three principal components $\kappa_1, \kappa_2, \kappa_3$, of a 16-bin normal curvature histogram computed over the surface. For the latter, a *PCA* basis is computed from the normal curvature histogram from all the surfaces in the seed cluster. Top three principal vectors are used as projection bases to compute the reduced dimensional representation of the curvature signature of a surface. $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{n}, \kappa_1, \kappa_2, \kappa_3$ collectively give rise to a 15-dimensional feature vector.

Constrained Mean-Shift Global Clustering. This step takes the seed clusters as input and creates a set of larger clusters that join the surfaces originating from semantically dissimilar segments. We use the mean-shift algorithm [Cheng 1995] to compute the final

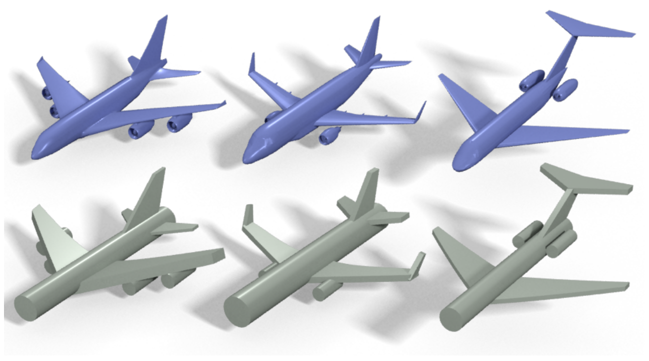

Figure 7: *Top: Input models. Bottom: Co-constrained abstractions whose surfaces serve as deformation handles.*

[2]This clean-up was required for 16% of all models presented in this paper. We believe the recent method described in [van Kaick et al. 2013] may be a suitable alternative for initial part segmentation.

[3]The initial part segmentation is assumed to carry the desired information in terms of functional or semantic similarity.

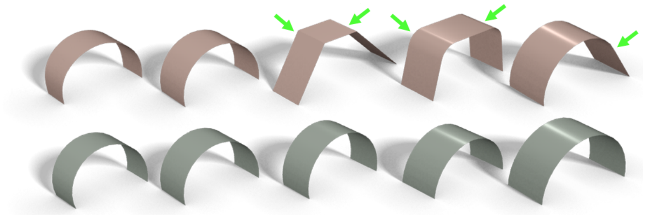

Figure 8: *Top: Initial segment abstractions of five different wheel housings. Bottom: Co-constrained abstractions. Note the change in the number of surfaces for some of the segment abstractions.*

clusters:

$$m(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in N(\mathbf{x})} K(\mathbf{x}_i - \mathbf{x})\mathbf{x}_i}{\sum_{\mathbf{x}_i \in N(\mathbf{x})} K(\mathbf{x}_i - \mathbf{x})} \quad (2)$$

$\mathbf{x} \in R^{15}$ is the surface feature vector, $m(\mathbf{x})$ is the mean-shift, $K(\mathbf{x}_i - \mathbf{x}) = e^{-c||\mathbf{x}_i - \mathbf{x}||^2}$ is the Gaussian kernel estimator, and $N(\mathbf{x})$ is the neighborhood of $\mathbf{x}$. For a given surface, if the mean-shift determines any one of the surfaces in a seed cluster to be a neighbor using the L^2 norm, then $N(\mathbf{x})$ includes all of the surfaces belonging to that cluster. This alleviates the well-known isotropic kernel disadvantage associated with the mean-shift algorithm. Figure 6 shows the accuracy and processing time associated with this constrained mean-shift variant we use compared to the compute-intensive, semi-supervised Gaussian mixture model (GMM). As shown, the accuracy gain by GMM does not justify the associated time complexity. Hence, we use the constrained mean-shift approach for the results presented in this paper. Details of the semi-supervised GMM and a more detailed analysis of this comparison can be found in the supplementary document.

Our features are not designed for conventional shape segmentation, we do not want to group the planar face and the radial surface of the car wheel. Instead, our clusters favor surfaces belonging to similar families (*e.g.,* planar, cylindrical, spherical), and that share similar relative positions (through $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$) and orientations (through $\mathbf{n}$) along any of the three global axes. Note that, it is intended to merge some of the semantically different clusters in this step. For instance the side surface handle of the body of the car, and the side surfaces of the tires on the same side. This allows the system to automatically suggest intelligent editing in real time during user interaction; while the user modifies the side surface of the body, tire side surfaces are also modified in plausible directions (see supplementary video).

4.3 Co-constrained Abstraction and Handles

We analyze the identified clusters to extract a set of constraints underlying the shape collection (Figure 7). The first set of constraints aims to establish the statistically dominant surface *family* for a cluster of surfaces. The second set aims to identify the statistical *degrees of freedom* of each surface cluster to form the constrained and free deformation handles. Also, planar, cylindrical, and spherical surfaces are constrained to remain as such after any deformation.

Surface family constraints. Once the surface clusters are formed, we compute the probability of the surfaces within a cluster belonging to a planar ($\mathbb{P}_p$), a spherical ($\mathbb{P}_s$), or a cylindrical ($\mathbb{P}_c$) category:

$$\mathbb{P}_p = \sum_i^n C_i^p/n \quad \mathbb{P}_s = \sum_i^n C_i^s/n \quad \mathbb{P}_c = n_c/n$$

where n is the number of surfaces in the cluster, $C_i^p = 2(\sigma_2 - \sigma_3)/(\sigma_1 + \sigma_2 + \sigma_3)$, and $C_i^s = 3\sigma_3/(\sigma_1 + \sigma_2 + \sigma_3)$. In C_i^p and C_i^s, $\sigma_1 > \sigma_2 > \sigma_3 > 0$ are the *PCA* components of the point set formed by the vertices sampled from surface i. n_c is the number of surfaces in the cluster, which has originated from a basic cylinder primitive in the segment abstraction [Yumer and Kara 2012]. If none of the above probabilities is greater than 0.9 (empirically determined and used for all presented examples), we deem the variations among the surfaces of that cluster are too significant to warrant a categorization. As such, the surfaces in that cluster are treated as polynomial surfaces of quadratic or cubic degree, determined by the initial segment abstraction. Otherwise, all surfaces in the cluster are converted to the surface category with the highest probability. This is done for each surface individually by re-fitting a parametric model of the detected family to the original model points giving rise to the segment abstraction surfaces. We call this combined set of new abstractions the *co-constrained abstraction*. Figure 8 shows the transition from a segment abstraction to a co-constrained abstraction.

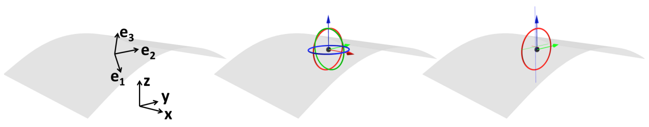

Figure 9: *Handle manipulators. Left: Principal component vectors on the surface. Middle: Manipulator with all nine degrees of freedom; the user selects translation versus scaling using a key press. Right: Co-constraints disables certain degrees of freedoms. When user desire, the manipulator will depict a range indicator to mark the deformation ceiling dictated by the shape collection.*

Degree of freedom constraints. Once the co-constrained abstraction is computed, we establish the appropriate degrees of freedom for each constituent surface (handle). Each surface is initially assigned axis-aligned three translational t_x, t_y, t_z, three rotational $\theta_x, \theta_y, \theta_z$, and three scaling s_x, s_y, s_z degrees of freedom. For t_x, we extract the x component of the center of mass positions of the surfaces within a cluster. If the standard deviation is less than a user-prescribed threshold (currently 0.05 for a unit-normalized bounding box), we prohibit t_x from being manipulable on the basis that the observed variation is too limited to warrant that degree of freedom. Constraints on t_y and t_z are computed similarly. For θ_x, we project each surface to the yz plane and compute the angular variation between the first principal components $\mathbf{e}_1$ among the projected surfaces. If the standard deviation is less than a threshold (currently $10°$), we constrain θ_x from being manipulable. Constraints for θ_y and θ_z are computed similarly. For scaling s_x, s_y and s_z, we find the span of each surface along the scaling axis. If the standard deviation is less than a percentage of the maximum span (currently 5%), we constrain s_x from being manipulable.

The default setting is either to fully constrain a degree of freedom, or to make it fully manipulable with no ceiling on the manipulation extent. The user has the option to turn all constraints off, thereby allowing all degrees of freedom of a surface to be manipulable. A third option is to allow all degrees of freedom to be manipulable, but for a selected degree of freedom, show the admissible 'range' of the manipulation as a visual reference. For translational degrees of freedom, for instance, the manipulator displays a transparent bar showing the $\pm 3\sigma$ range arising from the above statistics (Figure 9).

The above apply to planar or free-form surfaces. For the radial surfaces of co-constrained cylinders, we additionally compute the standard deviations in their radii r and the lengths of their axes l. Similarly, for spherical surfaces, we compute the variation in r.

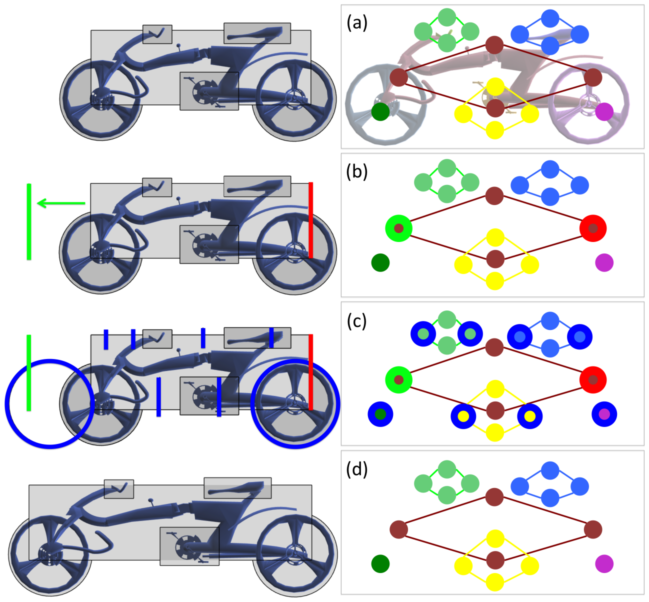

Figure 10: *(a) Contact graph of handles. Surfaces into the page are omitted for viewing simplicity. (b) User edited handle in green, system anchored handle in red. (c) System chosen additional constrained handles in blue, for which the positions are computed by Equation 4. (d) Resulting deformations.*

4.4 Correlation Among Deformation Handles

We compute a number of pairwise statistics that estimate the conditional probability distributions among the identified surface clusters. The resulting information is used to: (1) Visually query the 'severity' of a prescribed handle configuration, with respect to the statistics originating from the shape collection, and (2) Appropriately update the surrounding handles in response to an edit applied to a particular handle. During deformation, the former gives feedback to the user on how severely the pending surface modification will affect the relationship between that surface and all other surfaces in the model. The latter enables an automatic adjustment of all handles such that the resulting deformed shape complies maximally with the constraints learned from the database.

We estimate the conditional feature probability of cluster i, given cluster j as follows:

$$\mathbb{P}(k_i|k_j) = \mathbb{P}(k_i, k_j)/\mathbb{P}(k_j) \tag{3}$$

where k takes values from: $t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_x, s_y, s_z, r, l$. We model $\mathbb{P}(k_i, k_j)$ and $\mathbb{P}(k_j)$ as univariate Gaussian distributions from the observed data, resulting in an estimated Gaussian for the conditional probabilities $\mathbb{P}(k_i|k_j)$.

Soft constraints. During the deformation process, the user is continuously informed about the conditional probabilities that are violated by more than a predefined deviation from the mean of the corresponding feature. This passive feedback mechanism, which we refer to as soft statistical constraints, enables users to situate the edits they are performing with respect to the collection of shapes. Such information aids the user in two ways: The user might either (1) prescribe edits that do not violate the statistical constraints to ensure realistic models that stay within the variations among the dataset, or (2) The user might deliberately apply exaggerated deformations by that violate the associated soft constraints.

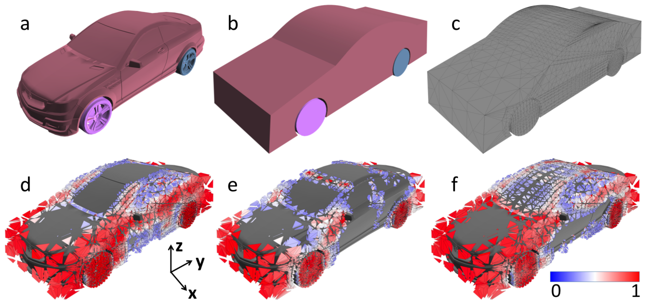

Figure 11: *(a) Segmented model. (b) Co-constrained abstraction. (c) Tetrahedral mesh domains computed using the surfaces of (b) as boundaries. (d-f) Element vulnerability in x, y, and z directions. Elements with vulnerability less than 0.3 are not depicted.*

Handle anchoring. An anchor is a handle that is automatically kept fixed to prevent rigid body transformations, while another handle is being manipulated. This alleviates the need for the user to mark the fixed handles every time another handle is manipulated. For each handle, its anchor is identified as follows: If the handle has a globally symmetric pair, it is selected as the anchor. Otherwise, a handle that maximizes the Euclidean distance from the handle under consideration in the 4D space defined by $[g, n_x, n_y, n_z]$, is selected, where g is the normalized geodesic distance ($0 \leq g \leq 1$), and $[n_x, n_y, n_z]$ is the outward normal (Figure 10(b)).

Constraint propagation. We use the statistics to update the handles for which the user has not prescribed any edits. Note that, independently constraining adjacent handles may result in ill-conditioned configurations for shared handle edges. To prevent this, we utilize the contact graph of the handles (Figure 10). To select the handles to be updated by the system, we propagate the information from the user edited and anchored handle nodes in a breath first search manner (BFS). To avoid adjacent handle constraining, for any pair of connected nodes, if one node is constrained, we ensure its neighbor is not. Note that the contact graphs of the individual segments are disjoint (Figure 10(a)). If none of the handles in a segment is edited by the user, the system randomly selects a node in such segments, and propagates to additional nodes in the corresponding contact graph using BFS. Moreover, if the user edits two adjacent surfaces and this results in an ill-conditioned configuration, then the latest edited surface overrides the first, and the user is notified. Once the handles to be constrained are selected, we solve the following optimization problem:

$$\begin{aligned} \underset{s_i}{\text{minimize}} \quad & \sum_i^N \frac{1}{\sigma_i} |\mu_i - s_i| \\ \text{subject to} \quad & k_j = u_j \quad \forall\, k_j \in \mathbb{U}. \end{aligned} \tag{4}$$

where N is the number of relevant statistics (see Equation 3) pertaining to the user edited and anchored handles. σ_i and μ_i are the standard deviation and mean of the random variable i, and s_i is its current value computed on the model in question. In Equation 4, the summation represents the total deviation of the constrained handles from the dataset mean. $\mathbb{U}$ is the set of degrees of freedom corresponding to the user edited handles, k_j are the degrees of freedom as defined in Equation 3, and u_j are the values resulting from the user prescribed edits. The updates resulting from this optimization generate additional constraints for the volumetric deformation (Figure 10(c), Figure 5). The equality constraints of the optimization in Equation 4 ensure that user intentions are accurately reflected in the deformation constraints. We use sequential quadratic programming to solve the resulting system. Once the propagation is complete, the constraints on the surfaces are updated according to the new po-

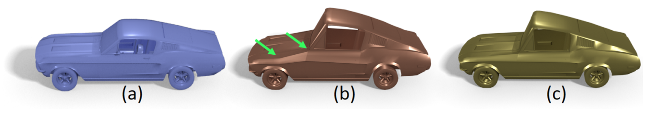

Figure 12: *(a) Input model. (b) Deformation created by tilting the top surface using a homogeneous-isotropic material, note the undesirable artifacts on the door and hood. (c) Using our material.*

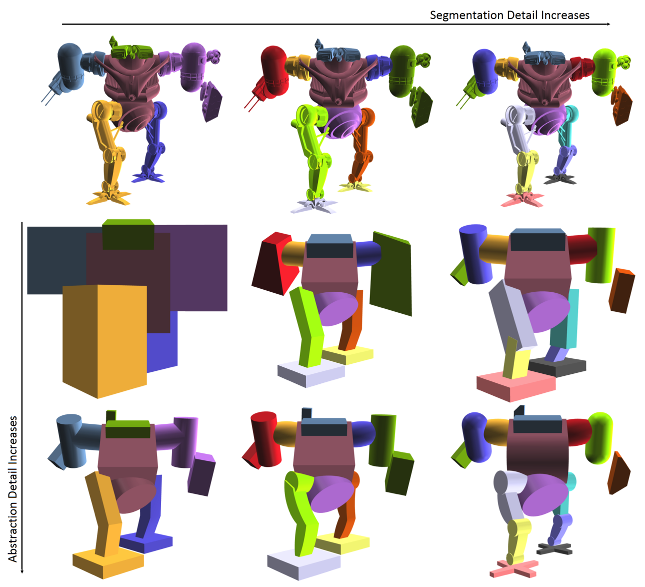

Figure 13: *Handles as a function of segmentation (left-to-right) and abstraction (top-to-bottom) granularity.*

sitions of the handles for subsequent editing sessions if the entire shape has undergone rigid body translations or rotations.

5 Volumetric Deformation

Meshing. Our volumetric deformation is based on a finite element model, using a structure-aware, non-homogeneous and anisotropic material. We generate a tetrahedral mesh inside the volume bounded by a co-constrained abstraction using the method described in [Si and Gärtner 2011]. Figure 11(c) shows the mesh generated for the abstraction in Figure 11(b). Each co-constrained abstraction volume in a model is independently meshed in this way.

Vulnerability. We first triangulate all non-triangular surface polygons. Next, for each triangle, we compute its vulnerability v_x, v_y, v_z along the three principal directions using the slippage and normal curvatures introduced in [Kraevoy et al. 2008]. In our method, we transfer the computed vulnerabilities to the tetrahedral mesh elements such that each tetrahedron gets assigned the maximum directional vulnerability among the original surface triangles intersecting the tetrahedron.

Non-homogeneous, anisotropic material. Using the vulnerabilities assigned to the tetrahedrons, we design a material matrix with the following properties:

$$E_{xx} = v_x \qquad E_{yy} = v_y \qquad E_{zz} = v_z$$
$$G_{xy} = G_{yz} = G_{xz} = max(E_{xx}, E_{yy}, E_{zz})/2$$
$$E_{xy} = E_{yz} = E_{xz} = 0$$

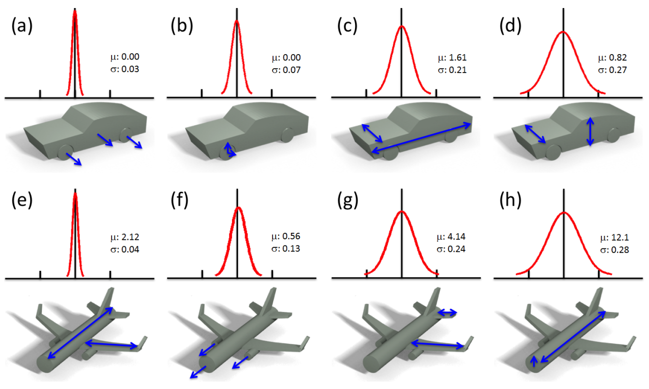

Figure 14: *Four most significant (*i.e.*, minimum variance) statistics learned from car and airplane datasets given by Equation 3. Plots are mean-shifted for visual comparison. Cars: (a) co-planarity of wheel cap center and body side, (b) wheel housing radius relative to wheel radius, (c) car length given its width, and (d) car width given its height. Airplanes: (e) fuselage length relative to wing span, (f) co-planarity of radome and engine tips, (g) wing span relative to horizontal stabilizer span, (h) fuselage length relative to fuselage cross-section radius.*

where v_x, v_y, v_z are the vulnerabilities in the three major directions, and E and G are the elastic and shear modulus, respectively. By design, our material is anisotropic and is not volume preserving ($E_{ij} = 0$ for $i \neq j$). We assemble a global stiffness matrix from the local stiffness matrices computed with this material using 4-node tetrahedral element shape functions [Norrie and De Vries 1978].

Boundary conditions. For each segment in the original model, tetrahedral mesh creation and stiffness matrix computation for its co-constrained abstraction results in the linear system:

$$K\delta = 0 \tag{5}$$

where K is the global stiffness matrix computed above, and $\delta \in \mathbf{R}^{3n_t}$ is the concatenated vector of tetrahedral mesh vertex displacements. n_t is the number of vertices in the tetrahedral mesh. A handle deformation prescribes the displacements for the vertices on this surface. This constrains the corresponding entries in δ. Similarly, vertices of the anchor surface (Section 4.4) are identified in δ and set to have zero displacements. The residual boundary forces are then used to solve the rest of system of equations. Note that K is sparse and is computed only once for each segment. This enables a fast solution in which a linear set of sparse equations with a precomputed K is solved for a specified deformation only once.

Deformation propagation. Following mesh deformation, we compute the new positions for the original model vertices embedded in the mesh using tetrahedral barycentric coordinates. For parts of the original model that fall outside the meshed volume, we use the as-rigid-as-possible deformation technique [Sorkine and Alexa 2007] such that parts of the model that reside inside the tetrahedral domain are treated as known displacements.

6 Results and Discussions

Co-constrained abstraction surfaces as handles. The notion of *co-constraints* exploits shape set information in deformation and editing processes. Resulting handles embody not only geometric features of the shape itself, but also information about the family of shapes it emerges from. For instance, in Figure 8, our method

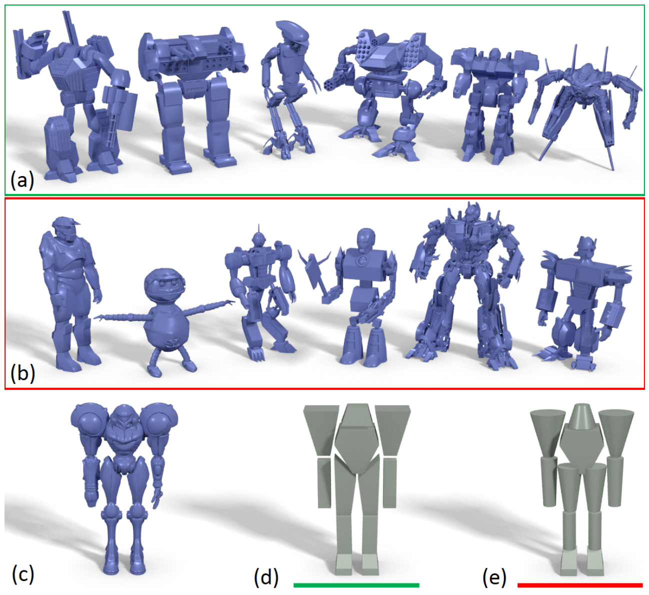

Figure 15: *Model in (c) yields deformation handles in (d) when conjointly analyzed with models in (a). Similarly, it yields deformation handles in (e) when analyzed with (b).*

identifies the dominant circularity of the surface through an analysis of the shape set. For each model, this constraint is enforced in the synthesis of co-constrained abstractions and their surface handles. This helps the system and the user to clearly understand the intended relationship between the wheel-housing and the wheel, for all the models in the collection.

Volume deformation. Our volumetric deformation is made structure-aware through the use of *vulnerability* introduced in [Kraevoy et al. 2008]. However, our deformation method is more flexible and enables arbitrary free-form handle deformations, in addition to non-uniform resizing. In our implementation, our handles are amenable to free-form deformation through the use of curve sketching. The user can activate a handle, and sketch a new silhouette for the handle from an orthographic view.

Handle hierarchy. Two main factors contribute to the complexity of the created handles: the initial volumetric segmentation, and the subsequent initial abstraction. First, the volumetric segmentation dictates the number of disjoint parts and enables independent deformation, scaling, and rigid-body transformation of these parts. Second, both the segmentation granularity, and initial segment abstraction level dictates the number of co-constrained abstraction handles that are created. Figure 13 shows the hierarchy of handles enabled through this flexibility. Each coarse-level handle can be independently decomposed into several more detailed handles in the subsequent level. This enables greater flexibility in selecting different levels of detail as the handles of different components.

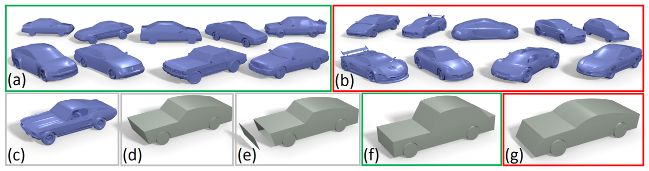

Figure 16: *Model in (c) yields deformation handles in (d) when analyzed with all models in (a) and (b). (e) User input. (f) System updates remaining handles based on statistics in (a). (g) System updates remaining handles based on statistics in (b).*

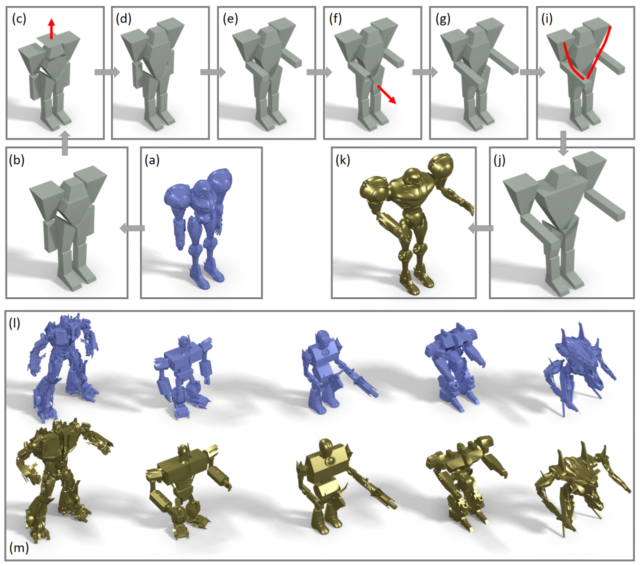

Figure 17: *Co-Deformation: user input on a single model's handles are propagated to the entire dataset to deform all the models. (a-b) Source model and its deformation handles, (c-h) a series of user input and system auto-completes, (i-j) Deformed model and its handles' final configuration, (l) Target models in the dataset, (m) Source deformation imposed on the target models.*

Dataset statistics. The shape collection allows the identification of statistical variability in the position, rotation, and scale of a given surface associated with the co-constrained abstractions. We compute these from the surface clusters (Section 4), and impose the learned constraints onto the manipulators of the handles. These constraints prevent users from making implausible edits. For instance, a car's left and right side panels are symmetrically configured (as learned from the dataset). Hence, shearing one side relative to the other will create an infeasible design, which is naturally prohibited by the constraints. When desired, these constraints can be turned off, allowing arbitrary deformations to be prescribed through an unconstrained manipulator. At any time, the user can view a handle's current configuration in relation to the learned statistics through a gauge (please see the accompanying video). Figure 14 shows a set of minimum-variance conditional probabilities learned from the car and airplane datasets. Such probabilities signal a strong coupling between the associated handle variables. Note that the algorithm uses solely the geometric properties of the handles and does not have access to the semantic descriptors used herein.

Effect of different datasets. The input dataset has a significant impact on the resulting handles and shape deformation. Figure 15 illustrates how different datasets can produce different handles for the same model. Figure 15(a) shows that if the dataset consists of objects having mostly rectangular limbs, the resulting handles tend to be rectangular in nature (Figure 15(d)). Similar discussions for the case of cylindrical limbs (Figure 15(b and e)).

Note that the conditional probabilities computed from the dataset can be used to automatically compute an *optimal* configuration for all the handles in a model. This is particularly useful for explicitly altering a single handle or for automatically reconfiguring all the handles to make a model consistent with a dataset.

Figure 16 illustrates the idea for explicit handle alteration. We com-

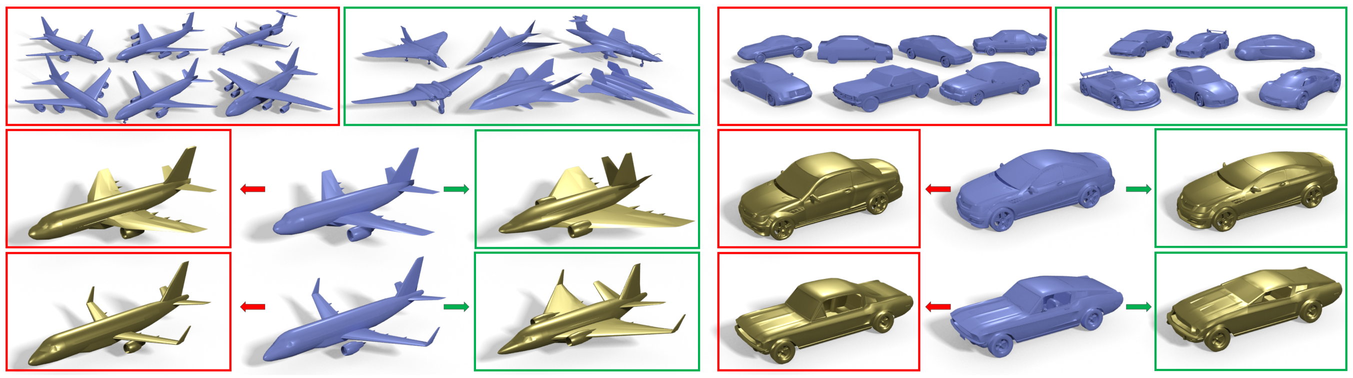

Figure 18: *Abstract style transfer shown on two models per two categories. Left: airplanes. Right: cars. The deformation results are automatically generated by the system, where the deformation handles are modified based on the statistics of corresponding dataset. Note how passenger airplanes can imitate the shape of fighter jets, and how regular cars can be modified into sports cars, or classical sedan cars.*

pute the optimal configuration of all the other handles by minimizing the following functional:

$$\begin{aligned} \underset{s_i}{\text{minimize}} \quad & \sum_{i}^{N} \frac{1}{\sigma_i} |\mu_i - s_i| + \lambda \cdot \sum_{j}^{M} D_j \\ \text{subject to} \quad & k_l = u_l \quad \forall\, k_l \in \mathbb{U}. \end{aligned} \tag{6}$$

Equation 6 is similar in spirit to Equation 4. Here, N is the number of statistics computed for the model under consideration (Equation 3) and involves the largest subset of handles present on the model that will not cause ill-conditioning (*i.e.,* no adjacent handles in the subset). M is the number of free-form handles in this subset, and D_j is the distance between handle j and the corresponding free-form handles on all other models (Equation 1). $\mathbb{U}$ is the set of degrees of freedom corresponding to the handles where edits are prescribed by the user, k_l are the degrees of freedom as defined for Equation 3, and u_l are the values resulting from the user prescribed edits. In Equation 6, the first term in the minimization promotes affine similarity between the model's co-constrained abstraction surfaces and that of the dataset's mean, and favors handles exhibiting small variance. The second term accounts for the geometric proximity between the free-form handles. λ controls the relative weight between these terms.

Co-Deformation. Clusters learned from the abstraction surfaces serve as a means to transmit a model's deformation to all other models in the shape collection, which we call *co-deformation*. Figure 17 illustrates the idea. This is achieved by exploiting surface correspondences in the co-constrained abstractions of the models in the dataset. Hence, a deformation applied to a handle can be fluidly transferred to all other models, thereby allowing a batch-editing of the shape collection using a single model.

Abstract style transfer. Equation 6 can also be utilized for abstract style transfer, by omitting the constraints in the optimization problem since there will be no user edits in this scenario. Particularly, an input model can be automatically deformed via its handles by making the statistics of the model similar to those computed from the dataset. Figure 18 demonstrates the idea. In effect, this can be viewed as a process similar to that shown in Figure 16, except without any user prescribed alteration to the handles.

Editing results. Figure 19, and Figure 1 shows various additional editing examples achieved using our system. In all cases, the original models are shown in blue and the deformed versions in gold. The shape collection from which the co-constraints are learned contain more input models than what is depicted. The robots set contains 14 models, the airplane set 18, the bicycle and lamp sets each contain 13 models[4]. Note that in all contexts, the deformed models exhibit both plausible as well as unrealistic/unnatural edits. Indeed, these correspond to cases where the identified co-constraints are either active, or are turned off making the model freely editable.

Preliminary user study. We administered a small user study to benchmark our method against existing software suites. The time to create the result with the commercial packages was five times of the time to create with our software, in average for all users. Please refer to the supplementary material for the details of this preliminary user study.

Performance. Our approach consists of a pre-processing and deployment phase. Pre-processing involves: (1) Computing an initial abstraction for each segment in each of the models, (2) Constructing the surface clusters and identifying the co-constraints from the collection, (3) Meshing each co-constrained abstraction volume on each of the models, and (4) Computing the material and stiffness matrix K associated with each mesh, which is computed only once for a co-abstraction volume. These steps are currently not at interactive speeds (*e.g.,* less than a minute to compute the initial abstractions for one model on average). When deployed with the above information, the deformation takes place at interactive speeds. The usage is interactive even in cases when one model's deformation is transferred to the other models in the database simultaneously.

Limitations. In our implementation, we do not alter the input model's topology. As a result, if the original model's surface polygonization is coarse, or the polygons are configured such that the model becomes non-conducive to deformations in certain directions, our software may produce unexpected results. For instance, if a coarsely polygonized part of a model undergoes bending, the final result will exhibit kinks along the deformed surface. Our method can only decode shape constraints that are computable from the abstraction geometries. In product design, there are a additional layout, aesthetic, ergonomic and manufacturing constraints that are critically important. If such relevant attributes are not represented in the geometry, our method might fail to capture these constraints.

Future Work. The techniques introduced in this work are initial steps towards leveraging implicit information of shape collections for shape deformation purposes. Our method depends on the volumetric abstractions, through which the statistics are computed. Future directions might include computing statistics on details that can only be represented by surface height maps, curves, or textures, which are not easy to represent with volumetric abstractions.

[4]Please refer to the supplemental material.

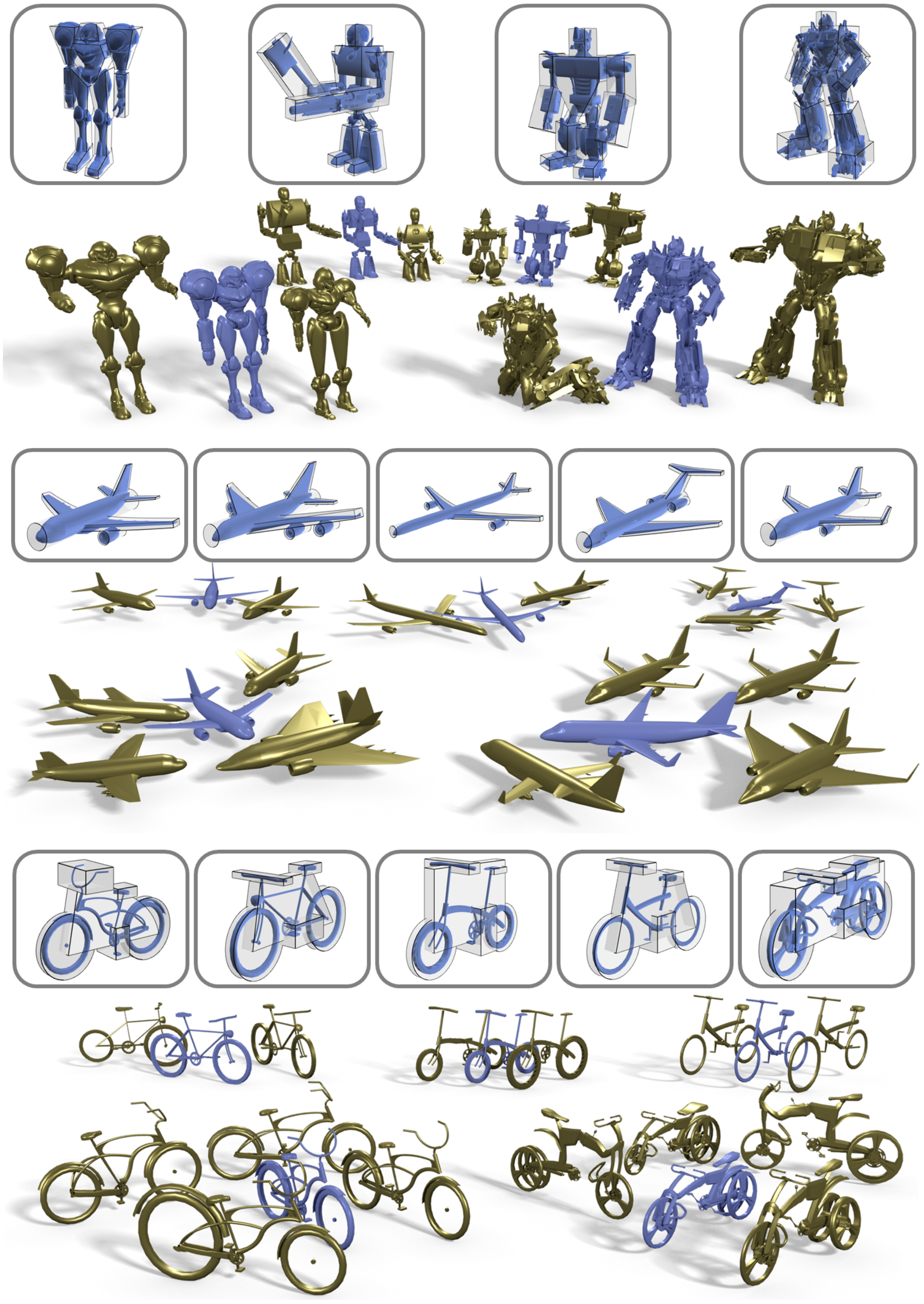

Figure 19: *Example handles and co-constrained deformations. Original models in blue. Deformed models in gold. See supplemental material for results on an additional dataset.*

7 Conclusion

We address the problem of shape editing where the underlying shapes are expected to deform in ways that make certain edits more preferable over others. We describe a new deformation method that learns the implicit shape constraints in a product family from example models. The decoupling of handle shapes (unique to a model) versus the handle operations allows each model in the collection to be surrounded by custom deformers, while the results of the deformations remain congruent to the constraints dictated by the collection. The results show that the proposed approach works well for models comprised of many topologically disconnected parts, and can accommodate a wide variation of topological and geometrical differences in the input models.

Acknowledgments

We thank the anonymous reviewers for their valuable input and suggestions. We thank Eric Anderson and Niloy Mitra for useful discussions. The 3D models are collected from Artist-3D, Archive-3D, Trimble (Google) 3D warehouse, and Turbosquid. We thank the creators of these models for making their work publicly available. This material is based upon work supported by the National Science Foundation under Grant Nos. CMMI-0846730, CMMI-1235427.

References

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

BOKELOH, M., WAND, M., SEIDEL, H.-P., AND KOLTUN, V. 2012. An algebraic model for parameterized shape editing. *ACM Transactions on Graphics 31*, 4, 78.

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

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. on Vis. and Comp. Graph. 14*, 1, 213–230.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Transactions on Graphics 29*, 4, 38.

CHENG, Y. 1995. Mean shift, mode seeking, and clustering. *IEEE T. Pattern Analysis and Machine Intelligence 17*, 8, 790–799.

FISH, N., AVERKIOU, M., VAN KAICK, O., SORKINE-HORNUNG, O., COHEN-OR, D., AND MITRA, N. J. 2014. Meta-representation of shape families. In *ACM Transactions on Graphics*, vol. 33(4), 34.

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

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

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. In *ACM Transactions on Graphics*, vol. 30, 125.

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

KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. 2013. Learning part-based templates from large collections of 3d shapes. *ACM Trans. on Graphics 32*.

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

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proc. of the Symposium on Computer Animation*, 153–159.

MITRA, N., GUIBAS, L., AND PAULY, M. 2006. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics 25*, 3, 560–568.

NORRIE, D. H., AND DE VRIES, G. 1978. *An introduction to the finite element analysis*. Academic Press New York.

OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. 2011. Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics 30*, 4, 33.

POPA, T., JULIUS, D., AND SHEFFER, A. 2006. Material-aware mesh deformations. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, 22–22.

SI, H., AND GÄRTNER, K. 2011. 3d boundary recovery by constrained delaunay tetrahedralization. *International Journal for Numerical Methods in Engineering 85*, 11, 1341–1364.

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 Transactions on Graphics 30*, 6, 126.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Sym. on Geometry Processing*, 109–116.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. In *ACM Transactions on Graphics*, vol. 24, 488–495.

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. on Graphics 32*, 4.

WANG, Y., ASAFI, S., VAN KAICK, O., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Active co-analysis of a set of shapes. *ACM Transactions on Graphics 31*, 6, 165.

XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. 2009. Joint-aware manipulation of deformable models. In *ACM Trans.on Graphics*, vol. 28, 35.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z.-Q. 2010. Style-content separation by anisotropic part scales. *ACM Transactions on Graphics 29*, 6, 184.

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

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

YUMER, M. E., CHUN, W., AND MAKADIA, A. 2014. Co-segmentation of textured 3d shapes with sparse annotations. In *Computer Vision and Pattern Recognition. CVPR 2014.*, IEEE.

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.