Shape Design From Exemplar Sketches Using Graph-Based Sketch Analysis

Günay Orbay

Levent Burak Kara[1]
e-mail: lkara@cmu.edu

Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213

We describe a new technique that works from a set of concept sketches to support the exploration and engineering of products. Our approach allows the capture and reuse of geometric shape information contained in concept sketches, as a means to generate solutions that can concurrently satisfy aesthetic and functional requirements. At the heart of our approach is a graph-based representation of sketches that allows the determination of topological and geometric similarities in the input sketches. This analysis, when combined with a geometric deformation analysis, results in a design space from which new shapes can be synthesized, or a developing design can be optimized to satisfy prescribed objectives. Moreover, it facilitates a sketch-based, interactive editing of existing designs that preserves the shape characteristics captured in the design space. A key advantage of the proposed method is that shape features common to all sketches as well as those unique to each sketch can be separately identified, thus allowing a mixing of different sketches to generate a topologically and geometrically rich set of conceptual alternatives. We demonstrate our technique with 2D and 3D examples. [DOI: 10.1115/1.4007147]

1 Introduction

Early design activities frequently involve the generation of a wide variety of concepts [1]. Designers commonly record such ideas in the form of conceptual sketches, which are recognized to be critically important for product design and development [2–5]. These sketches help designers assess different concepts and generate new ones early on, but rarely get utilized in the digital phases of the design process. This is mainly because current software is severely limited when such informal representations are concerned. In product form design specifically, while concept sketches embody a rich set of geometric information regarding the shape of the design, most of this information merely serves as static visual references, rather than providing a means to expedite the development and engineering of the product.

Furthermore, product form design is heavily affected by downstream engineering considerations, as the final product is required to satisfy both aesthetic and functional requirements. Typically, candidate designs generated by the styling teams have to be evaluated and validated by the engineering teams. This necessitates an iterative process involving multiple parties, which can significantly impact the design cost and duration. As a result, only a handful of concepts may be adequately developed, while many are prematurely eliminated.

In this work, we attempt to enhance the passive usage of conceptual sketches. We propose a new method that converts the geometric information stored in the sketches into a computationally suitable form. The proposed method does so by automatically identifying emerging shape ideas from the commonly appearing patterns, and concept specific design features. These patterns and features serve as means to computationally encode the shape ideas contained in the sketches to construct a design space. This design space allows (1) synthesis of novel forms through nonlinear mixing of input sketches, (2) a style-preserving free-form exploration of the design space through interactive sketching, and (3) production of design solutions that satisfy prescribed engineering objectives and constraints.

[1]Corresponding author.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received May 27, 2011; final manuscript received July 1, 2012; published online October 2, 2012. Assoc. Editor: Karthik Ramani.

2 Related Work

We group the previous studies on computational conceptual design systems in three categories. We give a summary of example-based design approaches and related difficulties in practical applications. We then discuss generative design methods and associated challenges. Finally, we review the major sketch-based design systems that support conceptual design and geometric modeling.

2.1 Example-Based Design Methods. Example-based design methods are primarily based on geometric shape interpolation techniques [6,7]. The first application to industrial design was proposed by Chen and Parent in 1989 [8]. Wang [9] extended the idea and combined shape interpolation with geometric transformation. Hsiao and Liu proposed using similar techniques for Computer Aided Design (CAD) models [10] or for scan-digitized geometries [11]. Similarly, Chen et al. [12] used image interpolation to study the mapping between car shapes and their descriptive characteristics. Kang and Lee [13] used mesh-based interpolation on different 3D ship hull forms to generate new forms.

The majority of these techniques require input models to be geometrically preregistered to resolve the mapping problem. Hence, an automatic identification of such correspondences has also been a popular research topic [7]. Despite these advances, the strict correspondence requirement limits example-based design approaches to mainly topologically equivalent geometries. Moreover, most approaches work from existing models, or require significant effort for model creation. In conceptual design, these methods have found limited usage as the content at these stages is incomplete, exploratory, and may exhibit large intermodel variations. Our work, on the other hand, is designed to work on geometrically and topologically different conceptual sketches. A key advance is a graph-based representation that enables the identification of a form common to all sketches, as well as features unique to each sketch.

2.2 Generative Design Methods. Generative methods focus on determining a parametrization and a set of generative rules for a product, using a set of existing designs. New designs can be synthesized from these learned rules. Cagan and Agarwal proposed using shape grammars as a language for defining template topologies through geometric rules [14]. McCormack et al. used this approach to study brand identity [15]. Orsborn et al. proposed

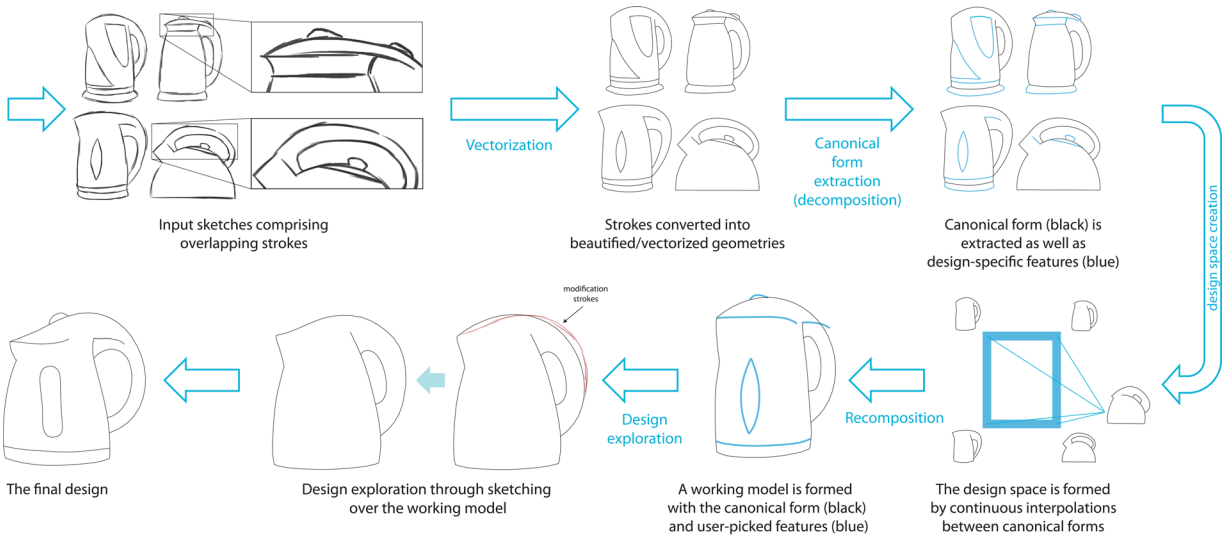

Fig. 1 Proposed method works from user-provided conceptual sketches. It then develops these sketches into a design space. The design space is either manually explored or used in integrated shape optimization.

combining different vehicle classes using shape grammars [16]. Although useful as a generative language, current shape grammar techniques require the rules to be determined and defined manually. In this work, we aim to circumvent rule specification using an automatic geometric deformation analysis, which lends itself to a generative design process.

Among the class of generative methods, genetic algorithms (GA) have been a popular choice for design [17–21]. Smyth and Wallace used GAs to synthesize aesthetic forms through manual visual inspection [17]. Frazer et al. utilized GAs to synthesize envelope designs for sky scrapers [18]. Bezirtzis et al. extended these applications to 3D models [19]. Wannarumon et al. used similar techniques together with prescribed aesthetic metrics for jewelry design [20]. Although GAs are successful in generating novel product forms, they require the model parametrization to be specified a priori. Moreover, parameter quality dictates the variety of designs that can be generated. This parameterization typically requires manual intervention.

2.3 Sketch-Based Design Systems. Recent years have seen the development of many sketch-based computer interfaces for CAD modeling. New methods allow the creation of curves and surfaces via intuitive free-hand sketching. Earlier studies were focused on creating primitive geometries [22,23] using sketch input. Recent studies [24–28] attempt to enable free-form surface creation from simple strokes and gestures. One group [24–26] focuses on creating initial blobs on which more features can be added while producing smooth surfaces. Another group [27,28] suggests creating curves in 3D which are later utilized as construction curves for free-form surfaces. A comprehensive analysis of existing work on sketch-based modeling can be found in Ref. [29]. These sketch-based methods primarily focus on using the sketch input for geometry creation rather than using the shape ideas in the geometries to generate new design concepts.

The above approaches aim to support product design by generating design alternatives, by learning design preferences from experience, and by creating 3D geometries from sketch input. However, these studies provide little or no means to learning the shape ideas contained in the conceptual design sketches and using them to aid the design process. In this work, we aim to provide support for shape design using explicitly encoded geometric information extracted from conceptual design sketches. In contrast to prior work, our approach is designed to be useful in the early design stages where concepts are articulated in the form of simple sketches. This allows working from a set of designs by automatically identifying a common form and concept specific design features, which then serve as a design space.

3 Method Overview and User Input

In this work, we introduce a new technique that works from a set of concept sketches to support the design and engineering of products. From a set of sample user-drawn sketches collected from a digital pen interface, our approach learns a generative shape model from which novel designs can be synthesized, or a developing concept can be engineered to satisfy prescribed objectives. At the heart of our approach is a graph-based representation of design sketches that enables the extraction of a *canonical shape* common to all sketches. The canonical shape serves as a means to establish a design space that captures the geometric variations among the input sketches, as well as features unique to each design. Our approach consists of two main steps: (1) Canonical form identification and detail extraction using topological matching and (2) design space construction and exploration (Fig. 1). In the first step, the input sketches are analyzed to identify features common to all sketches, from which a canonical form is created. For this, our method represents each sketch as a graph that enables a topological comparison. In the second step, a deformation analysis identifies the geometric variations among the canonical forms of the sketches. This analysis produces a design space from which new designs can be synthesized using nonlinear interpolation and extrapolation. Once created, the user may impose geometric constraints on the design space, transfer unique features from the original sketches onto the synthesized design, or modify the resulting designs with their pen strokes.

A sketch-based interface is used as a graphical front-end to our method. All interactions between the user and the system are through a pressure sensitive tablet. The input to the system is a set of sketches drawn by the user. The sketches are composed of strokes that are sampled at a constant rate through the tablet. Depending on the speed of the stylus, the distance between consecutively sampled points may vary proportionally. Our method is designed to work for sketches consisting primarily of contour curves, edges and character lines, without any shading or texture strokes. A typical sketch is shown in Fig. 1. Users may interact with the system through pen strokes to modify the curves for design exploration and to define geometric constraints.

4 Canonical Form Identification and Detail Extraction Using Topological Matching

Our system first beautifies the input sketches into vectorized drawings and then analyzes these drawings to identify the commonalities and differences between these drawings. Three steps

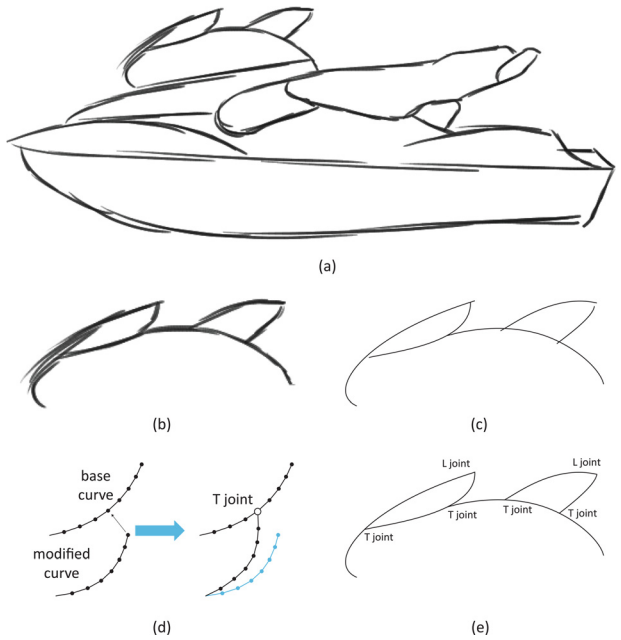

Fig. 2 Geometric representation of sketches. (*a*) Input sketch. (*b*) A subset of stroke groups, each defining an individual curve. (*c*) Vectorized curves. (*d*) T and L joints defined by the user. (*e*) Resulting line drawing.

contribute to this process: (1) A trainable sketch vectorization algorithm that converts raw sketches into vectorized drawings, (2) A graph-based representation of the vectorized drawings, and (3) A graph matching algorithm that identifies a topology common to all drawings (which we call as the *canonical* form), as well as topologies unique to each drawing.

4.1 Preprocessing and Geometric Representation. Users draw their sketches on a pressure sensitive tablet which samples the coordinates of the stylus at a constant rate. Typical sketches may exhibit multiple oversketched strokes. Once a sketch is completed, it is converted into a vector drawing using a trainable sketch vectorization algorithm we have previously developed [30]. A typical vectorization is shown in Fig. 1. This method operates on stroke-level features that encode the geometric relationships between the input strokes. This approach uses a training sketch to learn a neural network that can parse the training sketch into the intended stroke groups. The trained network is then applied to parse future sketches into unique curve groups. Finally, a parametric curve fitting algorithm beautifies each stroke group into a single curve. In this work, the results of this curve fitting are used to represent each curve as a single polyline. In our implementation, we used equal distance sampling when converting the parametric curves into polylines. Although we choose polylines as our underlying representation, our approach is similarly applicable to curves defined parametrically. To facilitate discussions, we refer to the resulting vectorized sketches as *line drawings* throughout the paper.

4.2 Graph Representation of Line Drawings. At the end of the previous step, input sketches are converted into line drawings that consist of multiple polylines (Fig. 2(*c*)). The next task is to analyze these vectorized sketches for distinctive *features* and a *canonical form* that is common to all designs. To this end, we have found a pairwise analysis of the curves to provide useful information. Specifically, the curves whose end points are proximate to other curves to be particularly critical. To encode these relationships, we define two types of curve interactions: (1) end point to end point interaction (L joints) and (2) end point to intermediate point interaction (T joints).

To facilitate the identification of such curve interactions, we ask the user to demarcate those joints after the sketch is vectorized. Figure 2(*d*) demonstrates the idea. To form L joints, the user simply picks two curves at their end points that are to define the joint. Our system treats the first curve as fixed and uses affine transformations to transform the second curve onto the first curve. The distant end of the second curve is kept intact during this process. To define T joints, the user picks the curve that forms the base for the T joint and then picks the second curve. Through similar transformations, the end point of interest on the second curve is repositioned to the closest point of the base curve. The user repeats this process until all joints are defined. Note that, during this process, the system only uses uniform scaling and rotation type transformations thus do not alter the shape of the curves represented by their curvature profiles.

We exploit the above types of joints to identify the similarities and differences between different line drawings. As illustrated in Fig. 3(*a*), sketches may contain curves that have multiple joints of different types. To encode this information, we use a graph-based representation of sketches which we call *sketch graphs*. In this representation, curves and joints appear as different types of nodes in the graph, and are connected by different types of links. With our representation:

(1) Different joint types can be distinguished.
(2) A pair of curves may contain multiple joints of different types.
(3) Different line drawings can be compared efficiently using unique node and link labels.

Note that this representation is different than a trivial representation in which the joints appear as nodes and the curves appear as links. With such a representation, a curve can have at most two joints and two joints can have at most one curve between them. Similar difficulties would exist if the joints were encoded as links and the curves were encoded as nodes.

Figure 3 demonstrates our representation on a drawing composed of 9 curves and 12 joints of different kinds. For each curve

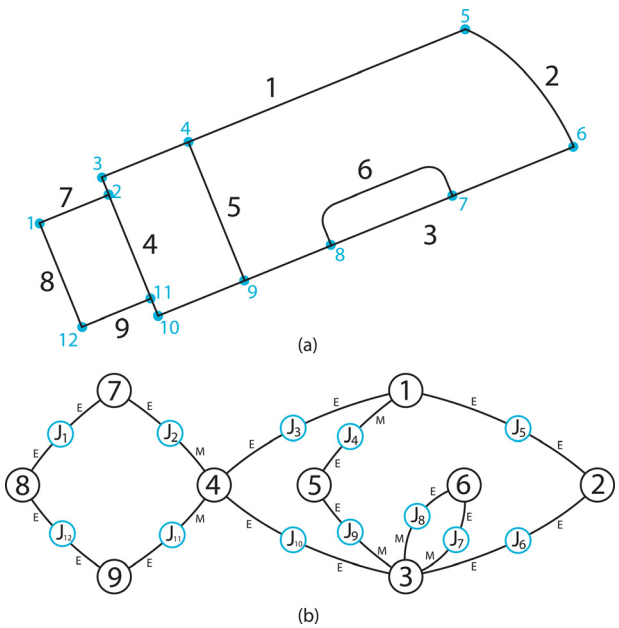

Fig. 3 (*a*) A line drawing of a flash drive. (*b*) Corresponding graph representation. *M* denotes that the curve-joint link corresponds to a midpoint connection on the curve, *E* denotes an end point connection. The graph has 9 curve nodes and 12 joint nodes.

and joint, there is a unique node in the graph. For each joint between a pair of curves, the corresponding curve nodes are connected to the joint node. Multiple joint nodes may exist between two curves nodes. The graph links connecting different types of joints (i.e., T or L joint) have different labels. If a joint is located along the interior of a curve, the graph link between this joint and the curve is labeled as M denoting a *middle point*. Similarly, if a joint is located at an end point of a curve, the graph link is labeled as E denoting an *end point*. It should be noted that this choice of connections implies that curve nodes can only be connected to joint nodes, and vice versa. Thus, having a separate joint node in the graph allows a curve to be connected to multiple joints in the graph.

4.3 Topological Graph Matching. After vectorization and joint formation, all sketches are converted into sketch graphs as described above. In this phase, the aim is to find a canonical form common to all sketches, as well as design-specific features. For this purpose, we seek to partition each sketch graph into two subgraphs, corresponding to a *base graph* representing the canonical form, and a set of *detail graphs* unique to each sketch. These graphs and their sizes are unknown a priori with only one constrain that the base graphs of each sketch are topologically identical. We use a graph-subgraph isomorphism algorithm to determine the topologically identical (i.e., having exactly the same node and link structure) subgraphs as candidates for the canonical form. Among those, we choose the one which is also geometrically similar in all designs using a set of pairwise geometric similarity features.

4.3.1 Graph Decomposition and Recomposition Model. We initially assume that each graph in a given set is the union of an unknown base subgraph and an unknown detail subgraph

$$\begin{aligned} G_1 &= B_1 \cup d_1 \\ G_2 &= B_2 \cup d_2 \\ &\vdots \end{aligned} \tag{1}$$

where G, B, and d denote the sketch graph, the base and the detail subgraphs, respectively (Fig. 4(*a*)). Here, the base subgraphs in all sketch graphs are topologically identical. Our method will later use this decomposition to synthesize new designs as a combination of a geometrically modified base, and a set of multiple modified detail graphs

$$G_{\text{synthesis}} = f(B_1, B_2, \ldots) \cup g_1(d_1) \cup g_2(d_2) \cup \ldots \tag{2}$$

where f and g denote functions that modify the shapes of geometric curves associated with a group without altering the topology (Figs. 4(*b*) and 4(*c*)).

4.3.2 Identification of the Base Subgraph. This is the determination of the canonical form and involves the following two steps: (1) Topological enumeration of all candidate subgraphs and (2) identification of the subgraph that captures the highest level of geometric commonality shared by the input sketches. Our sketch graphs are attributed relational graphs (ARG) as they encode different types of attributes on graph nodes and links. We use the VF2 method [31] to efficiently identify graph-subgraph isomorphisms.

In this problem, we consider two graphs $G_1(N_1,L_1)$ and $G_2(N_2,L_2)$, where N and L denote the nodes and the links, respectively. Let (u,v) represent an edge between nodes u and v. We seek a nodal mapping $M : N_1 \rightarrow N_2' \subseteq N_2$ having the following property: $(u,v) \in L_1 \Leftrightarrow (M(u), M(v)) \in L_2' \subseteq L_2$ where L_2' contains all links formed by the nodes in N_2'. Note that M is a bijective mapping. This formulation aims to identify all subgraphs of G_2 that are isomorphic to G_1. Thus, the resulting subgraphs of G_2 are of the same size as G_1.

For a set of graphs, the size of the largest common subgraph can be at most as big as the smallest graph in the set. The size of the subgraph representing the canonical form, on the other hand, is unknown[2]. To identify the canonical form, we first set the smallest graph as the *reference* graph G_1, while the remaining graphs are called *target* graphs. We then enumerate all subgraphs of G_1 such that each identified subgraph embodies a group of curves that are directly or indirectly connected (i.e., devoid of islands)

$$\begin{aligned} \text{Subgraph}(G_1) = \{S_n^i\} &\text{such that} \\ n &\in \{1, 2, \ldots, N_{\text{curves}}\} \\ \forall n : i &\in \left\{ 1, \ldots, \binom{N_{\text{curves}}}{n} \right\} \end{aligned} \tag{3}$$

where S represent a subgraph of G_1. N_{curves} is the number of curves in the corresponding reference sketch. n denotes the number of curves contained in the subgraph, and i indexes the subgraphs with n curves. We calculate candidate matches between each subgraph S of G_1, and the set of target graphs through pairwise comparisons. For each S, we compute a number of candidate subgraph mappings from S to target graphs using the VF2 method. We denote the resulting candidate mapping functions that have n number of curves by M_n.

Each subgraph mapping aims to identify the similarity between the two graphs that are matched. At the heart of this analysis are three geometric features that measure the match between a pair of curves as described below.

4.3.3 Curve Dissimilarity Features. To establish a congruent basis for comparison, each sketch is uniformly scaled and center-fit within a unit square. The similarity between two graphs is based on the similarity between the curves in the two graphs. For two curves (each belonging to a different graph), we use the following three features. Figure 5 shows the parameters used in these features.

The first feature measures the mismatch between the locations of the curves' centroids in their respective sketches

$$\varepsilon_s(c_1, c_2) = \frac{||\mathbf{d}_1 - \mathbf{d}_2||}{\sqrt{2}} \tag{4}$$

where $\mathbf{d}$ is the distance vector measured from the center of the sketch to the centroid of the curve.

The second feature measures the alignment difference between the two curves

$$\varepsilon_a(c_1, c_2) = \frac{||\mathbf{u}_1 \times \mathbf{u}_2||}{||\mathbf{u}_1|| \cdot ||\mathbf{u}_2||} \tag{5}$$

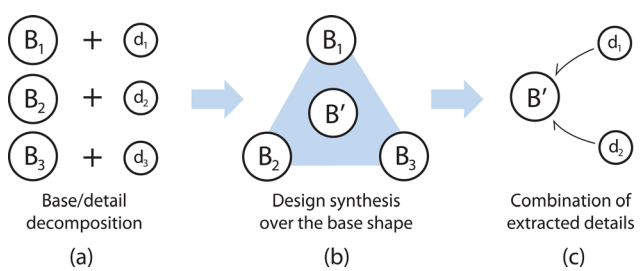

Fig. 4 (*a*) The sketch graphs are decomposed into base and detail subgraphs. (*b*) The base graphs are used to define a design space in order to synthesize new shapes from geometric variations. (*c*) The resulting shape is then recombined with previously decomposed details.

[2]The size of the canonical form can be at most the size of the largest common subgraph.

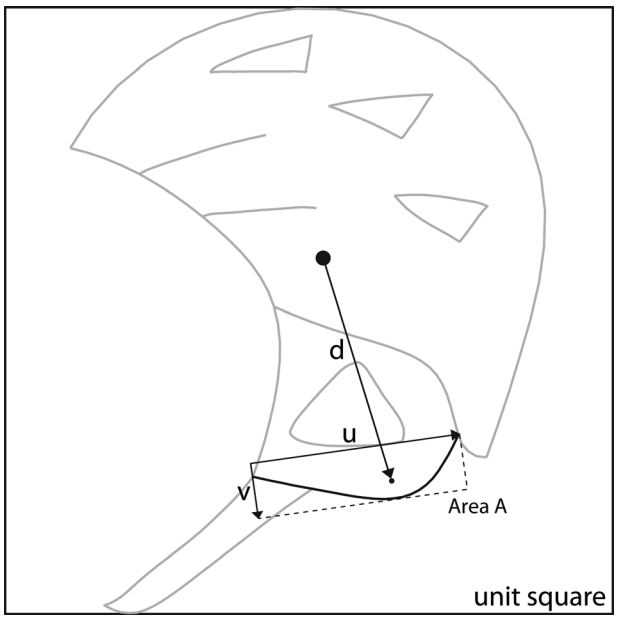

Fig. 5 The basic geometric properties used for pairwise dissimilarity features. Each vectorized sketch is uniformly scaled into a unit square prior to feature calculation

where **u** represents the first principal direction of the curve's points computed using principal component analysis. The length of these vectors estimates the spread of the curves in the drawing plane.

The third feature measures the difference between the oriented bounding box areas of the curves

$$\varepsilon_r(c_1, c_2) = 1 - \left(\frac{\min(A_1, A_2)}{\max(A_1, A_2)}\right)^{\frac{1}{2}} \tag{6}$$

where A is the area of the bounding box aligned with the first and second principal component vectors **u** and **v**. All three features are designed such that they are bounded between [0,1], and are zero when the curves are identical. We define pairwise curve similarity as

$$\sigma_{\text{curve}}(c_1, c_2) = 1 - \frac{1}{6}(4\varepsilon_s(c_1, c_2) + \varepsilon_a(c_1, c_2) + \varepsilon_r(c_1, c_2)) \tag{7}$$

The weights of the features were determined experimentally using a set of test sketches. These tests have shown the spatial dissimilarity feature to be more critical than the other features.

4.3.4 Overall Graph Similarity. As described earlier, the goal in graph matching is to identify the canonical form, which represents the highest level of geometric commonality between the graphs. The above features help quantify the similarity between the matched graphs during this process.

A distinction between topological and geometric similarity is necessary. Topologically, while the largest common subgraph could be chosen as the canonical form, this may not result in the highest level of shape similarity. Instead, we seek to maximize the geometric similarity between the graphs as computed using the above features. This similarity may be maximized for subgraphs containing fewer number of curves than that of the largest common subgraph. Hence, we decompose our analysis into multiple levels in which similarity matching is performed separately for groups of subgraphs each delineated with a unique topological size.

For the ith subgraph of the reference graph G_1 containing n number of curves $\left(S_n^i\right)$, we calculate the overall graph similarities using the described curve features. Let S_n^i be described as $\left(Nc_n^i, Nj_n^i, L_n^i\right)$, where Nc and Nj are the curve and joint nodes of S_n^i, and L_n^i are the links. Let $G_k \in \{G_2, \ldots, G_K\}$ be a target graph. Let M_n^j be the jth mapping between S_n^i and G_k. Let c_t $t \in \{1, .., n\}$ be a curve in S_n^i. The overall graph similarity of the mapping M_n^j is calculated as

$$\sigma_{\text{match}}(S_n^i, G_k \mid M_n^j) = \sum_{t=1}^{n} \eta\left(\sigma_{\text{curve}}\left(c_t, M_n^j(c_t)\right), n\right) \tag{8}$$

where $\eta(.,.)$ is a nonlinear scaling function (Fig. 6), we define as

$$\eta(x, n) = 1 - \sqrt{1 - x^n} \tag{9}$$

This function enables a comparison between mappings containing different number of curves. Note that without this adjustment, the cumulative similarity score σ_{match} would favor mappings with a large number of constituent curves (n), even if the similarity between individual curve pairs (σ_{curve}) is weak. η suppresses weak curve similarities as the number of curves increases.

The graph representing the canonical form $S_{\text{canonical}}$ is then identified as

$$\text{Score}(S_n^i) = \frac{1}{K}\sum_{k=1}^{K} \max_j \sigma_{\text{match}}\left(S_n^i, G_k \mid M_n^j\right) \tag{10}$$

$$S_{\text{canonical}} = \arg\min_{n,i} \text{Score}(S_n^i) \tag{11}$$

The above decision rule identifies the canonical subgraph exhibiting the largest level of geometric commonality between all sketches. Once this subgraph is determined, the focus moves to feature identification. For this, the canonical graph is subtracted from all graphs $\{G_1, \ldots, G_K\}$. For each graph, the remaining topologies are clustered into islands of fully or partially connected subgraphs using a greedy clustering algorithm. Each subgraph identified in this way forms a feature of the corresponding sketch and is made available to the designer during shape exploration and synthesis.

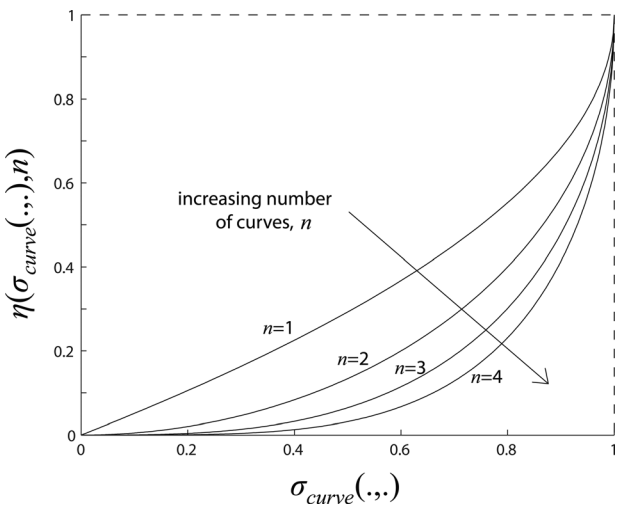

Fig. 6 A nonlinear scaling function that is used to combine curve scores in order to compare graph matches with different sizes

5 Design Space Construction and Exploration

At the end of the previous step, each sketch is vectorized into curves, converted into a sketch graph, and decomposed into a base graph (canonical form) and a number of feature graphs. Here, the focus is on the canonical form, and its geometric variations among the input exemplars. In this step, our system learns these variations in the canonical form to construct a design space.

We use a deformation-based shape interpolation method using inverse kinematics [32], which allows the synthesis of shapes from a set of registered exemplar shapes. This method is suitable when large deformations exist between the exemplars and is not limited by the distortions that typically arise with the use of linear interpolation techniques.

5.1 Preparation for Deformation Analysis. As discussed in Sec. 2, the shape interpolation problem requires a one-to-one mapping between the constituent geometries. Our method automatically determines a correspondence between the canonical form in each design as a byproduct of graph matching. The resulting match has a one-to-one mapping on the curve and joint nodes. However, for curve matches, a direction ambiguity still remains, which is resolved using the joint information in the graph representations. For instance, in Fig. 7, curve 1 in line drawing 1 and curve 2 in line drawing 2 form joint 1 at different locations on the curves. The joint on the former curve is located at the curve's starting point, whereas the joint on the latter curve is located at its end. For curves which do not form any L joints, we compute the correct alignment through a simple test between the vectors formed between the end points of the curves.

Identifying the correct curve alignment is sufficient to ensure that the individual line segments within each polyline is also mapped correctly to the corresponding line segment in the matching curve. This is due to the fact that all curves in our system are uniformly sampled using the same number of points, thus ensuring that each polyline curve has the same number of line segments. However, T joints can invalidate polyline correspondence when the matched joints are located at different points along the curve's arc length. To alleviate this issue, we split these curves at the T joints and resample each resulting curve to the same number of points as others, and accordingly update the joint information.

5.2 Deformation-Based Shape Interpolation. We first describe how to interpolate two shapes using a deformation analysis and then we extend it to multiple shapes. At this stage, all canonical forms among the exemplar shapes have a unique correspondence. A typical polyline curve pair is illustrated in Fig. 8 with the underlying line segments. Since the task is to deform one curve into the other, we pick one to be the reference curve and the other to be the target curve.

We start by calculating a transformation that would take one line segment in one curve and transform it into the corresponding line segment in the other curve. To compute a unique transformation, we define a third vertex for each line segment by rotating it 90 deg, and form a triangle (Fig. 8(*c*))

$$\mathbf{v}_3^j = \mathbf{v}_1^j + \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (\mathbf{v}_2^j - \mathbf{v}_1^j) \tag{12}$$

where $\mathbf{v}$ denotes the position vector (a 2×1 column vector) that stores the x and y coordinates of a vertex. Once all line segments are converted into triangles, we define an affine transformation that maps the jth triangle in the reference curve onto the corresponding triangle in the target curve (Fig. 8(*d*))

$$\Phi_j(\mathbf{p}) = \mathbf{T}_j \mathbf{p} + \mathbf{t}_j \tag{13}$$

$\mathbf{T}_j$ is a 2×2 matrix including rotation and scale transformations and $\mathbf{t}_j$ is a vector that encodes the translation component of the affine transformation. $\mathbf{p}$ is a point on the reference triangle to be transformed (suitably its vertices). To determine $\mathbf{T}_j$ in Eq. (13), the translation component must be eliminated. This is achieved by subtracting the equation for the third vertex, from the equations for the original two vertices. From the two remaining equations, the transformation matrix for the jth line segment is determined as follows:

$$\mathbf{T}_j = [\,\mathbf{v}_1^j - \mathbf{v}_3^j \quad \mathbf{v}_2^j - \mathbf{v}_3^j\,].[\,\bar{\mathbf{v}}_1^j - \bar{\mathbf{v}}_3^j \quad \bar{\mathbf{v}}_2^j - \bar{\mathbf{v}}_3^j\,]^{-1} \tag{14}$$

Here, $\bar{\mathbf{v}}$ denotes the coordinate vectors of the *reference* shape (C_1), while $\mathbf{v}$ denotes the coordinate vectors of the *deformed* shape (C_2). It should be noted that $\mathbf{T}_j$ is linear in the coordinates of the deformed shape.

Once the transformation matrices for all line segments are calculated, we reshape each matrix into a column vector and combine the vectors into a feature vector $\mathbf{f}$. The linear relationship between the actual coordinates of the deformed curve and the feature vector is

$$\mathbf{f} = \mathbf{G}\mathbf{x} \tag{15}$$

where $\mathbf{x}$ is a vector storing the global coordinates of the deformed curve in the form of $x = [x_1, x_2, x_3, \ldots, y_1, y_2, y_3, \ldots]^T$, $\mathbf{f}$ is the feature vector and $\mathbf{G}$ is a sparse block diagonal matrix whose coefficients only depend on the coordinates of the reference curve C_1. Note that $\mathbf{x}$ is a $2(n+m) \times 1$ column vector and $\mathbf{G}$ is a $4m \times 2(n+m)$ matrix where n is the number of original vertices (excluding the newly created ones) and m is the number of original line segments. Feature vector $\mathbf{f}$ is the deformation that represents the deformed shape C_2 in terms of the reference curve C_1, whereas $\mathbf{G}$ is the linear operator that maps the geometry space to a feature space.

Given a reference shape and a feature vector $\mathbf{f}$, a deformed shape can be calculated by an inverse application of $\mathbf{G}$ on $\mathbf{f}$.

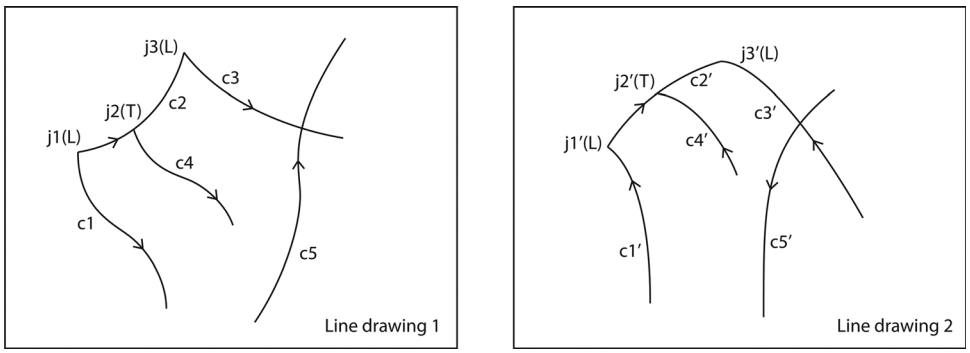

Fig. 7 Curve directions are corrected using the corresponding joint information

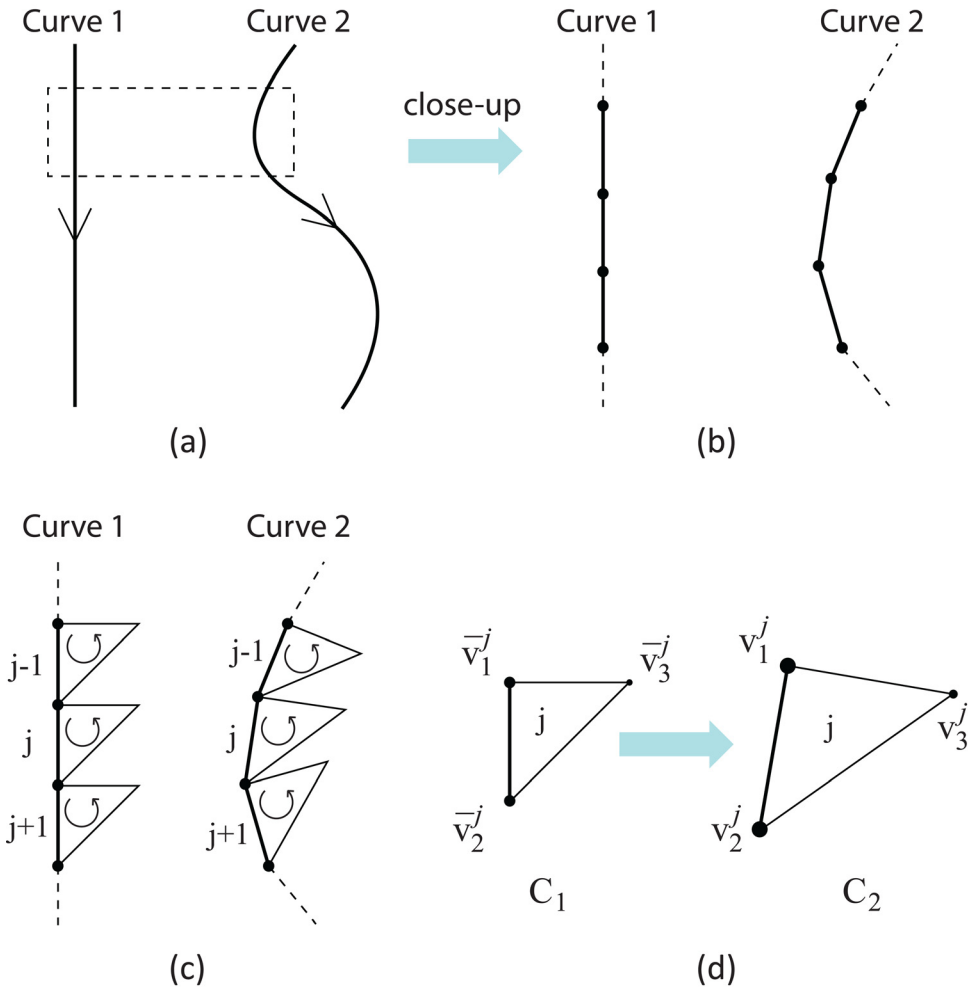

Fig. 8 Transformation between exemplars: (*a*) Each line segment on both curves is converted to a triangle. (*b*) *j*th triangle in C_1 then undergoes an affine transformation to produce the corresponding triangle in C_2.

However, since **G** was constructed without the translation component of the affine transformation, coordinates of at least one point must be prescribed to ground the shape in space. In general, an arbitrary number of vertices on the deformed shape may be constrained, which means that their coordinates are known. The remainder of the deformed shape can then be solved in the least-squares sense using the following minimization:

$$\mathbf{x} = \arg\min_{\mathbf{x}} ||\overline{\mathbf{G}}\mathbf{x} - (\mathbf{f} - \mathbf{c})|| \tag{16}$$

where **c** is the vector obtained by multiplying the constrained coordinate(s) with the associated columns of **G**. **c** can also be viewed as a feature vector that forces the free coordinates to align with given constraints. Moreover, $\overline{\mathbf{G}}$ is matrix **G** with the columns associated with the constrained coordinates removed. Solving Eq. (16) produces the new deformed shape **x** which deviates minimally from the original deformed shape giving rise to **f**, while precisely satisfying all prescribed constraints. In practice, the constraints on the geometry are defined by drawing modification strokes as shown in Fig. 1.

5.3 Design Space Construction With Multiple Exemplars. In cases of multiple exemplars, we arbitrarily choose one exemplar as the reference shape. The remaining exemplars form the set of deformed shapes. To facilitate subsequent calculations, we also add a copy of the reference shape to the set of deformed shapes. This allows the deformed shapes to encompass all exemplars available to the system.

The feature vectors between the reference shape and each of the deformed shapes are calculated[3]. A design space is then defined as a weighted combination of the feature vectors $\mathbf{f}_i$ using weights w_i as follows:

$$\mathbf{f}(\mathbf{w}) = \sum_{i=1}^{l} w_i \mathbf{f}_i \tag{17}$$

where l is the number of deformed shapes. A feature vector synthesized in this way can be used to compute a new shape using Eq. (16). Similar to interpolating absolute coordinates, an arbitrary combination (linear as above, or nonlinear) of the feature vectors may result in distorted deformations in cases where large bending and rotations are involved between the reference shape and the deformed shapes. This is because the feature vectors contain the cumulative deformations arising from distinct rotation and scaling transformations, which become indistinguishable when combined. To alleviate this issue, we decouple the rotation deformation from other deformations. Transformation matrix $\mathbf{T}_j$ of the *j*th line segment can be decomposed into a rotation ($\mathbf{R}_j$) and scaling-shear ($\mathbf{S}_j$) component via polar decomposition [33] as

$$\mathbf{T}_j = \mathbf{R}_j\mathbf{S}_j \tag{18}$$

[3]Note that one of the feature vectors will be formed by identity transformation matrices as the reference shape and one of the deformed shapes are identical.

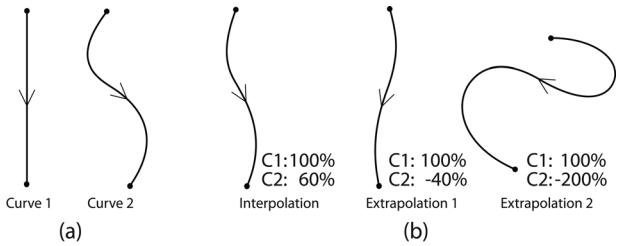

Fig. 9 Shape synthesis using a nonlinear design space: (*a*) While deformation from the first exemplar to the second (*b*) can be interpolated with either (left) positive or (middle) negative amounts, a nonlinear design space can also handle (right) extrapolations involving large deformations

With this decomposition, a rotation and scale-aware weighting of $\mathbf{T}_j$ can be calculated as

$$\mathbf{T}_j = \exp\left(w \log\left(\mathbf{R}_j\right)\right).w\mathbf{S}_j \tag{19}$$

where w is the weight, and exp and log are the matrix exponential and matrix logarithm operators, respectively. An example interpolation is shown in Fig. 9. This approach can produce interpolations as well as extrapolations involving large deformations.

Using Eqs. (17) and (19), a weighted transformation matrix is calculated as follows:

$$\mathbf{T}_j = \exp\left(\sum_{i=1}^{l} w_i \log\left(\mathbf{R}_{ij}\right)\right).\sum_{i=1}^{l} w_i \mathbf{S}_{ij} \tag{20}$$

where $\mathbf{R}_{ij}$ and $\mathbf{S}_{ij}$ are the rotation and scaling-shear matrices of the jth triangle of the ith exemplar, respectively. Note that a feature vector $\mathbf{f}$ is formed by stacking the $\mathbf{T}$ matrices of an exemplar into a column vector. Similar to Eq. (16), geometric constraints can be imposed, and the resulting geometry is calculated as

$$\mathbf{x} = \arg\min_{\mathbf{x},\mathbf{w}} ||\overline{\mathbf{G}}\mathbf{x} - (\mathbf{f}(\mathbf{w}) - \mathbf{c})|| \tag{21}$$

This expression generates designs that deviate minimally from the design space, while satisfying the prescribed constraints. Note that this minimization concurrently determines the final shape, and the optimum weights of the exemplars that produce it.

In our system, the user may control the weights of the exemplars directly through a widget. In this case, the provided weights are used in Eq. (20) to calculate the resulting feature vector, and Eq. (16) is used to generate the final geometry.

However, the key utility of the proposed system is realized when the exemplar weights are determined automatically using Eq. (21). This corresponds to the case where the user imposes geometric constraints through the graphical user interface by picking, dragging, and sketching over the input exemplars. In this case, any modification concentrated to a particular region can be extended to the whole shape through the design space. This allows the synthesized designs to be bear the characteristics of the exemplars in the design space. In effect, the exemplars provide the set of meaningful deformations that are encoded in the design space. This approach helps any alteration to a working design to result in shapes that are globally congruent with the exemplars in the design space.

Figure 10 demonstrates the idea. The first four modifications shown in Fig. 10(*b*) are the user's sketch modifications to the first exemplar. The first modification forces a concavity on the right leg. Since the second exemplar happens to exhibit this feature, the resulting deformation produces a shape that inherits the characteristics of the second exemplar. Likewise, the second modification intends to create a convex right leg. This deformation is achieved through a negative contribution of the second exemplar, which helps deform the other legs in similar ways. Subsequent deformations illustrate the resulting deformations and the calculated exemplar contributions when arbitrary modifications are applied.

5.4 Scale Invariance. The deformation gradient formulation is scale dependent as the scaling component of the feature vectors also encode the size changes among the input sketches. As a result, as the weight of a larger sketch is increased, the overall scale of the synthesis also increases. We introduce a modification to the feature space formulation to alleviate this dependency. This modification allows the size of the resulting shape to be determined from the geometric constraints only.

The first modification is to the feature vectors of the exemplars. We normalize the scaling component of the deformation gradients for each sketch with respect to the reference sketch

$$\psi_i' = \psi_i \frac{||\mathbf{I}||}{||\psi_i||} \tag{22}$$

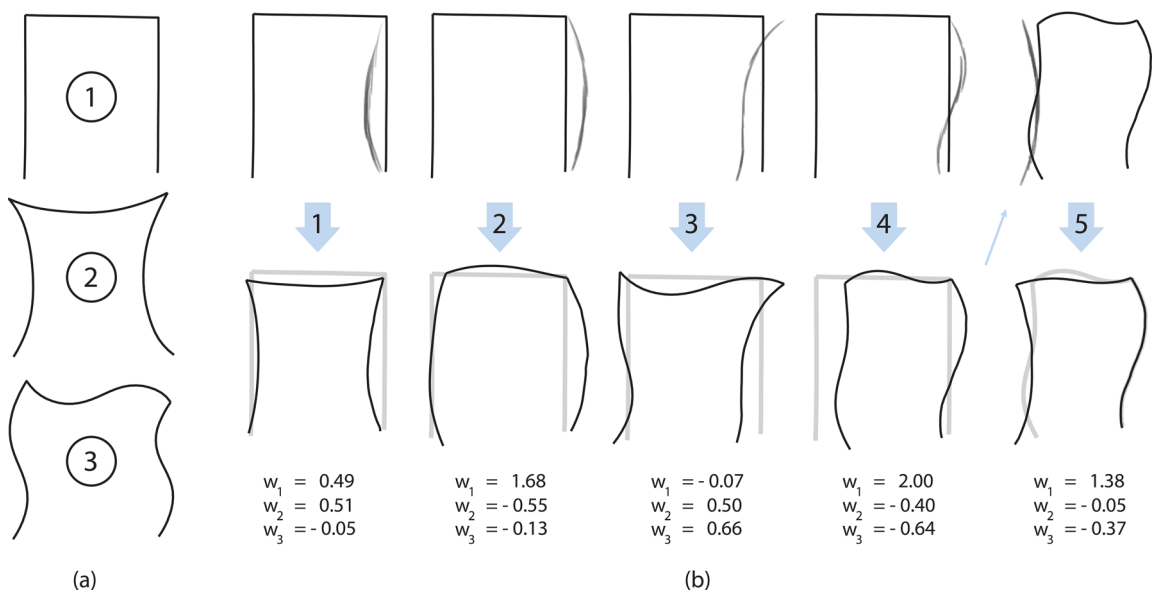

Fig. 10 Sketch modification using automatic weight calculation propagates local modifications to the remainder of the sketch using the geometric deformations learned from the exemplars

where ψ_i' and ψ_i are the normalized and original scaling feature vectors, which are constructed the same way as the features vectors. **I** is the scaling feature vector which is constructed from identity transformation matrices, and $\|.\|$ is the matrix norm operator. This normalization scales the exemplars to the scale of the reference sketch such that the scale transformation between each exemplar to the reference shape is an identity matrix. This normalization also helps preserve the overall size of the reference shape when only one point is constrained. Note that at least two points are required to uniquely define the size. In these cases, we use Eq. (16) with the normalized feature vectors.

For cases in which the user constrains more than one point and assigns the exemplar weights manually (i.e., $\mathbf{f}(\mathbf{w})$ is known), we introduce a free scaling parameter to Eq. (16) as

$$\mathbf{x} = \arg\min_{\mathbf{x},\alpha} ||\overline{\mathbf{G}}\mathbf{x} - (\alpha\mathbf{f}(\mathbf{w}) - \mathbf{c})|| \tag{23}$$

where α is the unknown scaling parameter. This scaling parameter introduces an additional degree of freedom that allows a uniform scaling of the final shape in cases where the exemplar weights are externally prescribed.

In cases where the user constrains more than one point and lets the system calculate the set of weights, the introduction of α is no longer necessary. This is because the presence of the reference shape in the set of deformed shapes allows the associated weight computed by Eq. (21) to serve as the free scaling parameter.

5.5 Design Space Exploration and Shape Optimization. Our system presents the canonical form of the reference design as the initial shape (i.e., a working design). For design exploration, the user may impose geometric constraints and attach one or more of the identified features to the working design. The geometric constraints can be added and modified by picking/moving gestures of the stylus, or by directly sketching over the working design. In case of sketch modifications, our system automatically determines the portions of the shape to be modified using a proximity check. It then moves the points selected for modification onto the modification strokes. These points are subsequently treated as constrained points. The user may also pick one or more design-specific features and attach them to the working model. For each added feature, the user first picks a point on the feature and then picks an attachment point on the working design. The user may also define further constraints on the features.

Shape variations can be achieved through a manual or automatic control of the exemplar weights. In the first case, the user adjusts the weights of each exemplar through a widget, which interactively changes the shape. In the second case, the user specifies the geometric constraints. The resulting shape is then determined according to Eq. (21).

Our approach is also conducive to constrained shape optimization when geometric objective functions are specified. Current shape optimization techniques typically require parameterized geometries. Often times, this parameterization is not trivial and has to be established by the designers. In our approach, the graph-based sketch analysis and the subsequent design space construction alleviates this need. During optimization, the exemplar weights serve as the optimization parameters, whose combination leads to an optimum design. Section 7 demonstrates this capability on different examples.

6 Computational Complexity Analysis

The computational complexity of our system is dictated primarily by the canonical form identification and detail extraction step, as graph matching is an nondeterministic polynomial-time (NP)-complete problem.

Let n_e = total number of exemplars, n_c = maximum number of curves in an exemplar, and n_j = maximum number of joints in an exemplar. As described earlier, we calculate and evaluate all possible graph matches between a reference sketch and all other sketches. In the graph representation, the number of nodes, N, is the summation of number of curves and joints (i.e., $n_c + n_j$). We assume the worst case scenario of all nodes connecting to one another. Thus, the number of connected subgraphs of the reference graph to be matched with other graphs is $2^N - 1$. For each subgraph, the VF2 algorithm determines graph matches at a cost of $O(N!N)$ [31]. Combined, the worst case complexity of this step is $O(N!N2^N)$.

The design space exploration step is far less demanding. Equations (16) and (21) both require the solution of a linear system which can be efficiently done using QR, Cholesky matrix decompositions or similar methods. Our implementation works at interactive speeds that allow users to work fluently. Detailed discussion of the computational complexity of this step can be found in Ref. [32].

7 Example Design Cases

Secs. 7.1 and 7.2, we demonstrate (1) canonical form identification, detail extraction, shape exploration, and synthesis and (2) constrained shape optimization under engineering objectives.

7.1 Canonical Form Identification, Detail Extraction, Shape Exploration, and Synthesis. We demonstrate our method with four design cases. The first three examples are user-drawn and are simplified with the stroke clustering and vectorization method described earlier. The last example is created with a sketch-based 3D modeling interface, thus did not require curve vectorization.

The first example demonstrates the design of a mug from four sketches as shown in Fig. 11. These designs have a common container part with different handle types. The handle of the third design is attached to the mug at only one point whereas other designs have handles attached at two points. As a result, the canonical form does not include the handle. A new design is synthesized through exemplar weight control (i.e., using Eq. (16)), and through a series of geometric constraint definitions and modifications.

The second example demonstrates a car body design from four side view sketches as shown in Fig. 12. These designs have a common set of curves defining the shape of the body as well as design-specific features such as different headlights and character lines. For instance, the first design has two pairs of headlights whereas the fourth design has no visible headlights. The canonical form thus excludes the headlights and several character lines. As shown in Fig. 12, a number of new designs are synthesized by the user through exemplar weight control (i.e., using Eq. (16)). By attaching individual features from different exemplars, and by defining and modifying several geometric constraints (Fig. 12, column 3), three new designs are developed.

The third example demonstrates the automatic exploration of the design space through sketch-based modifications of three different hair dryers as shown in Fig. 13. Notice that several parts, such as the nozzle of the second exemplar, are not included in the canonical form and are identified as design-specific features instead. The user sketches modifications to the canonical form. The weights of the exemplars are determined using Eq. (21). As the user makes local modifications, the remainder of the drawing is adjusted automatically using the learned deformations.

The fourth example involves 3D wireframes, whose skinned surfaces are also shown in Fig. 14. Three handheld devices which have different handle configurations are automatically analyzed to reveal the canonical form and the design-specific features. The main body is identified as the canonical form, while the handles are distinguished as features. After attaching the handles from the first and the third designs, and a series of user-guided manipulations, the final design is achieved.

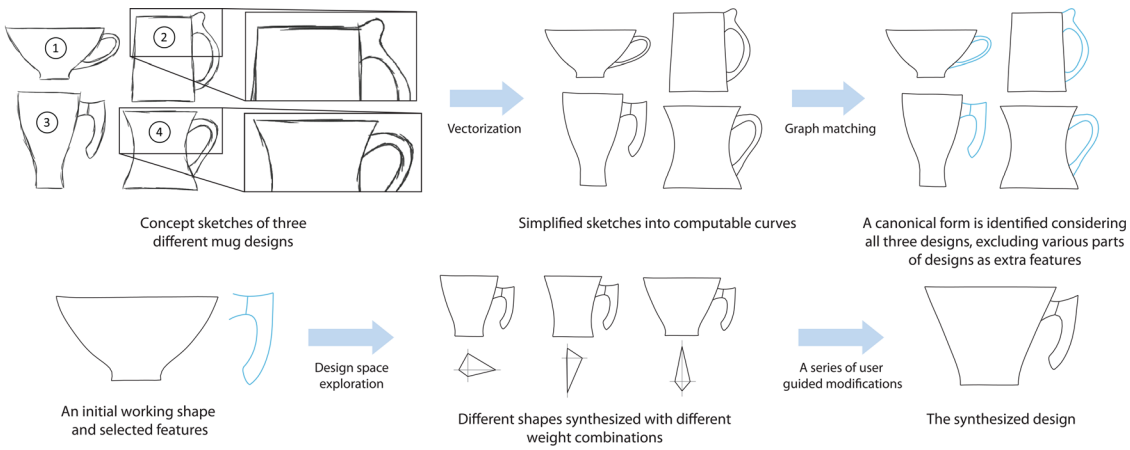

Fig. 11 Design of a new mug from four different designs

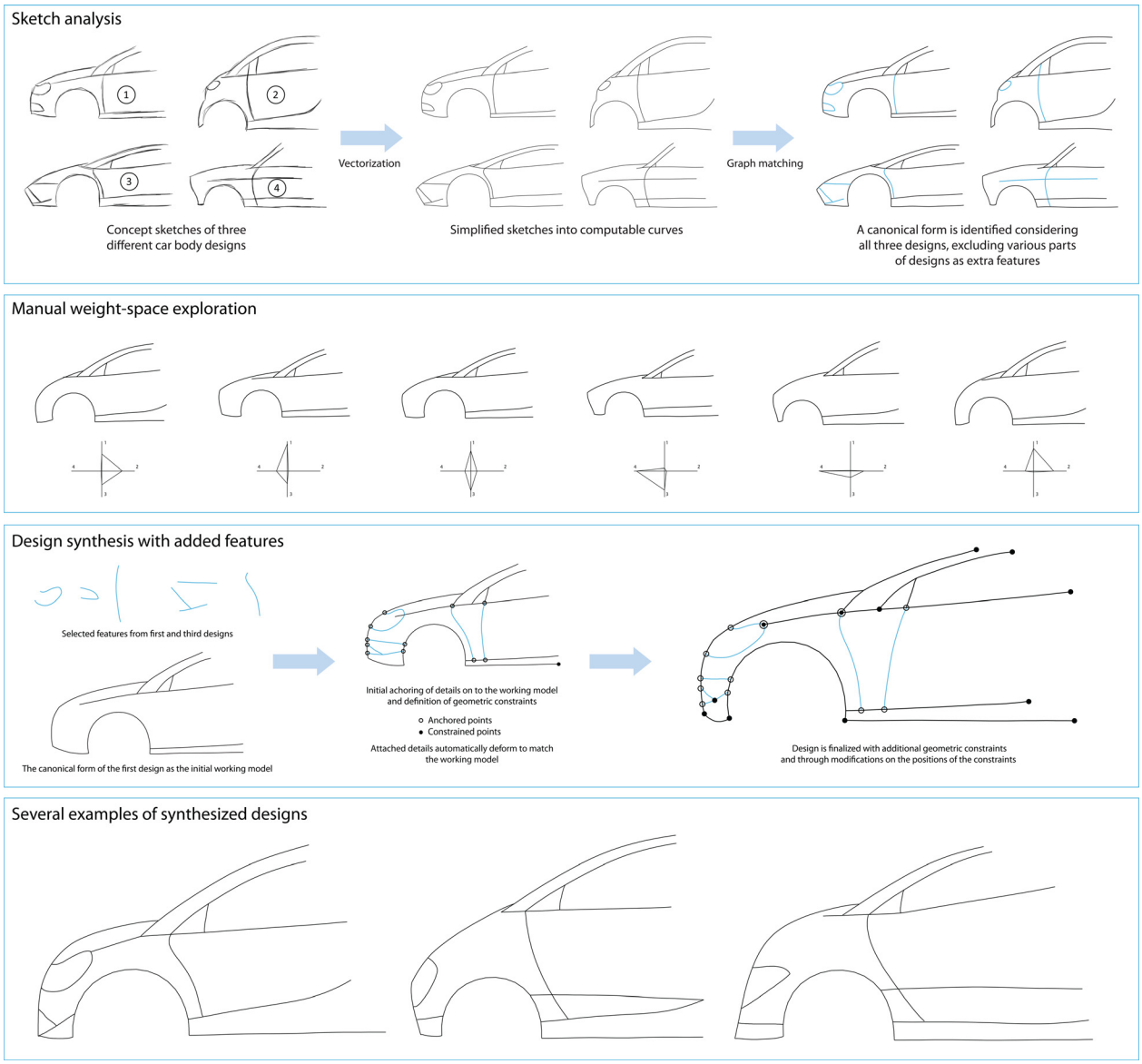

Fig. 12 Design of a new car from four different designs

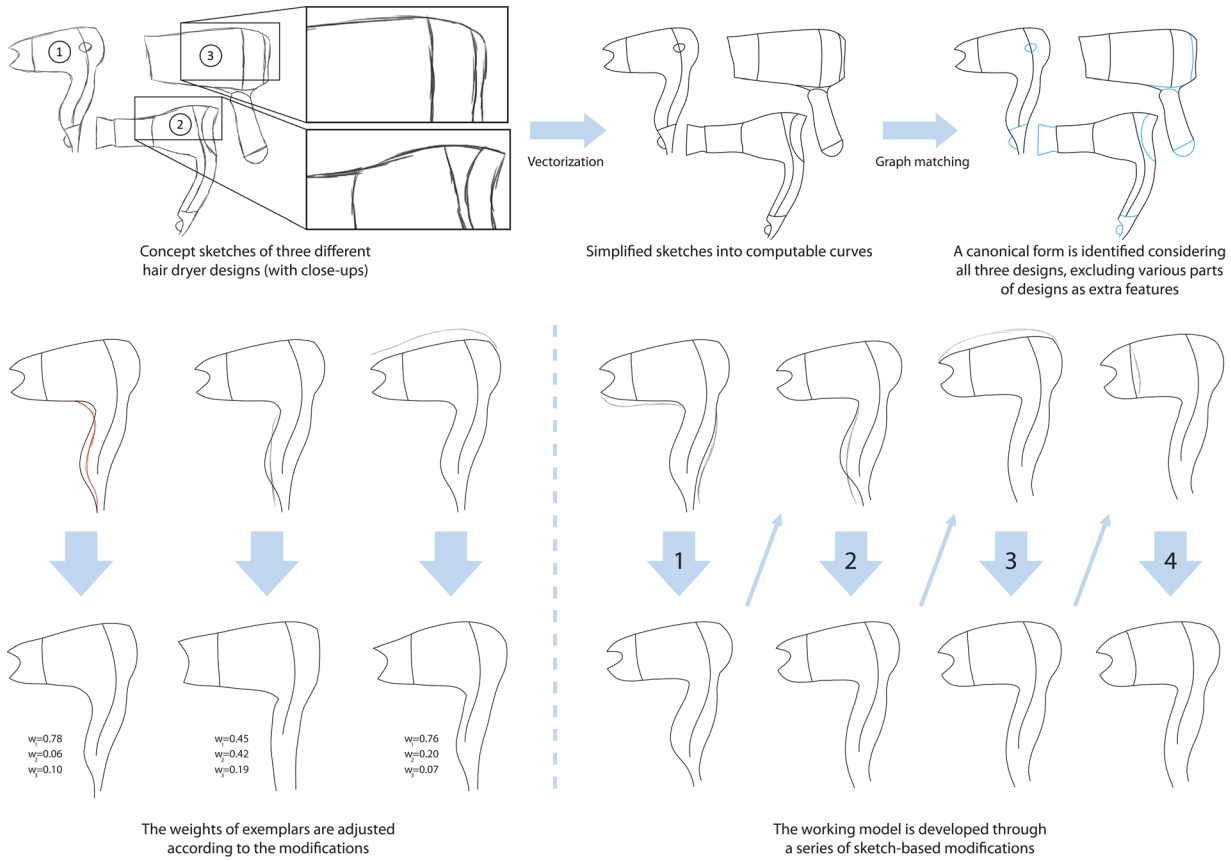

Fig. 13 Design of a new hair dryer from three different designs

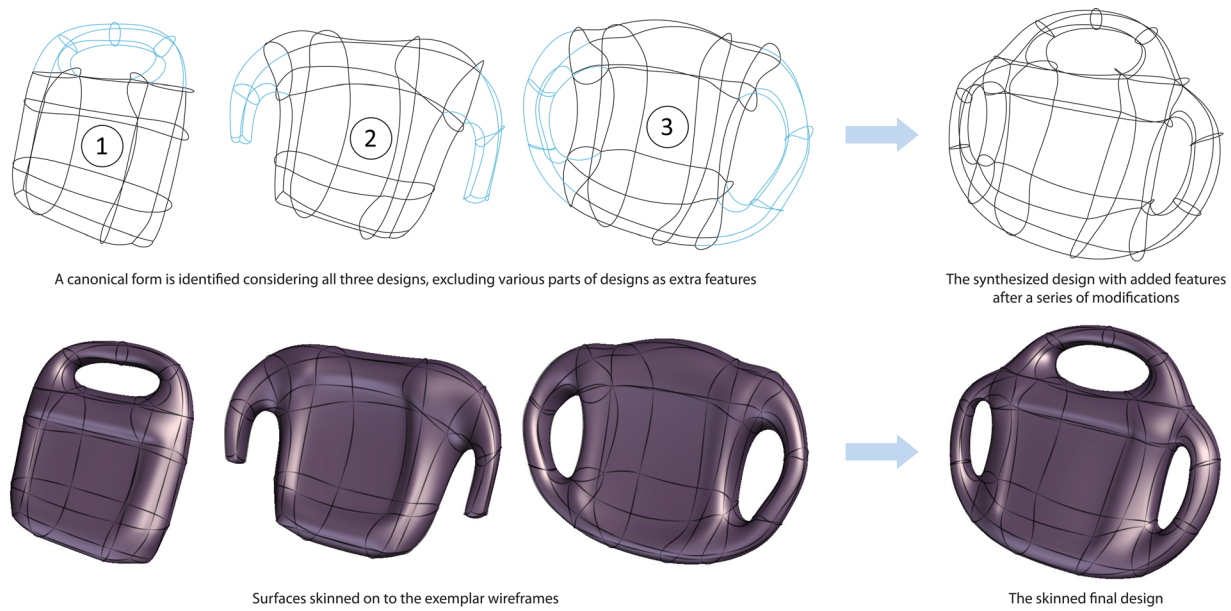

Fig. 14 Synthesis from 3D wireframes. (Top row) Analysis of three handheld device designs represented as 3D curve networks for canonical form (dark) and feature (light) identification. The three designs and the selected features are combined to produce a new design. (Bottom row) Surfaced exemplars and the final design.

7.2 Shape Optimization

7.2.1 Shape Optimization Under Engineering Constraints. We demonstrate shape optimization under engineering constraints on a bottle design example. Three bottles are sketched and beautified as shown in Fig. 15. To illustrate the extent of the design space, Fig. 15(*c*) shows two shapes that are synthesized using exemplar weight control. Note that the weights also exhibit negative values, which still result in plausible solutions

Using this design space, a minimum-weight bottle enclosing a prescribed liquid capacity is to be designed. The constraints are shown in Fig. 15(*d*). The sum of exemplar weights is kept at unity while the upper and lower bound for the weights are set at 1.2 and −0.5, respectively. The radius of the bottle mouth is constrained,

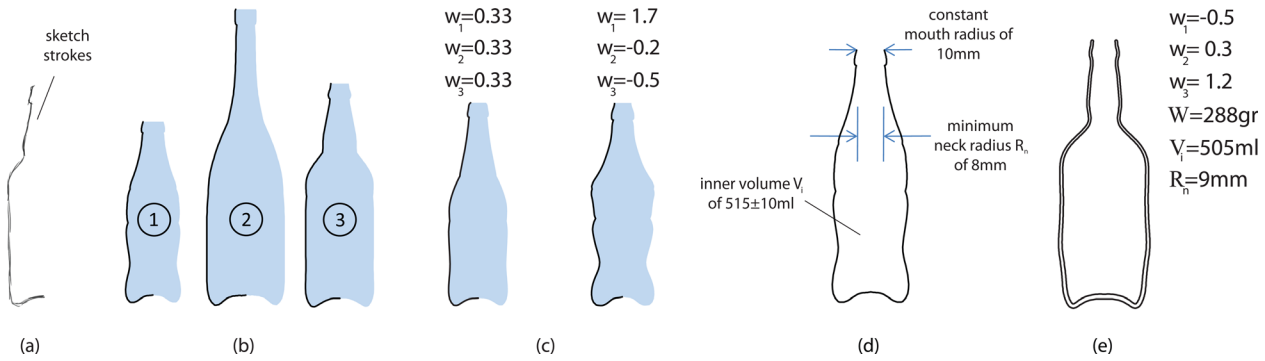

Fig. 15 (*a*) and (*b*) Bottle design consisting of soda, wine, and beer bottles as exemplars. The bottle is formed by revolving the sketched contours. The size variation among the exemplars should be noted. (*c*) Example design obtained by exemplar weight control. (*d*) Geometric constraints. (*e*) Optimum design and resulting exemplar weights.

while the height is controlled by a scaling parameter. A skin profile whose thickness increases linearly toward the bottom is adopted. There are four design parameters: three exemplar weights and the free scaling parameter. The objective function to be minimized is the weight of the bottle, which is easily calculated by revolving the thickness profile around the vertical axis. Figure 15(*e*) shows the resulting design. This design resembles mostly the third exemplar, as the resulting exemplar weights indicate. The negative weight associated with the first exemplar suggests that the first exemplar originally promotes a small liquid volume to bottle weight ratio, which the objective function is trying to reverse.

7.2.2 Shape Optimization Under Spatial Constraints. Figure 16 shows the design of a computer mouse. A mouse which embodies the internal structures, while fitting to the human hand is sought. Two design spaces are defined for the top and side views, each consisting of three exemplars. For consistency, each exemplar uses the same weight in the two design spaces. The objective function for the optimization is defined as the total area of interference between the mouse, and the internal and external spatial constraints. The sum of exemplar weights is kept at unity while the upper and lower bounds for the weights are set at 1.2 and −0.2. There are five optimization parameters: three exemplar weights, one position (the position of the mouse relative), and one scale parameter (freely scales the resulting shape). The resulting shape satisfies the internal and external constraints as shown in Fig. 16(*f*). The resulting shape has nearly equal contributions from the three designs.

7.2.3 Shape Optimization for Aesthetics. This example demonstrates the synthesis of curvature variation minimizing designs. Curvature minimization is a fundamental objective commonly used for aesthetic curve design [34]. We define a length-weighted, minimum curvature variation over a set of curves as follows:

$$\kappa_{\text{var}} = \sum_{i=1}^{N} L_i \int_0^1 \left\| \frac{\vec{\kappa}_i(s)}{ds} \right\|^2 ds \tag{24}$$

where i is the index of the curve, N is the total number of curves, L_i is the length of the ith curve, $\kappa_i(s)$ is the curvature, and s is the normalized chord length parameter. The term L_i scales the curvature variation integral of a curve by its length to favor curvature minimization of the longer curves over the short ones.

Three mouse designs with curvature profiles are shown in Fig. 17. Note that, the exemplars show fluctuations in the curvature profiles, which indicate a lower aesthetic quality. A design space is created using the three exemplars, and a design that has the minimum curvature variation is sought. The objective function to be minimized is the above functional, and the optimization parameters are the weights of the exemplars. The weights of the exemplars are constrained such that the size of the weights vector is one. Starting from a weight set having equal contributions from each exemplar, the optimum solution is calculated using sequential quadratic programming. The resulting design is shown in Fig. 17 with the associated weights. Note that, the curves of the optimum design show significantly smaller variations in the curvature profiles compared to the exemplars. This is achieved by mixing the curvature profiles of the exemplar curves with the right proportions such that their curvature fluctuations cancel each other. Numerically, the exemplar designs have total curvature variations of 3.33, 3.62, and 3.24, while the optimum design has a total curvature variation of 1.40 as computed from the above functional.

8 Discussions

We demonstrated our system with design examples covering user-guided design exploration, and shape optimization under various constraints. Although, these examples were successfully designed using our system, there exists cases that cannot be handled with our method. During our experiments, we have identified a few major causes for these inefficacies. We discuss them here by evaluating the two consecutive steps of our method, the first influences the second.

The first is sketch vectorization in which a hand-drawn sketch is converted into a set of parametric curves. For this purpose, we use our prior work on understanding and beautifying sketches

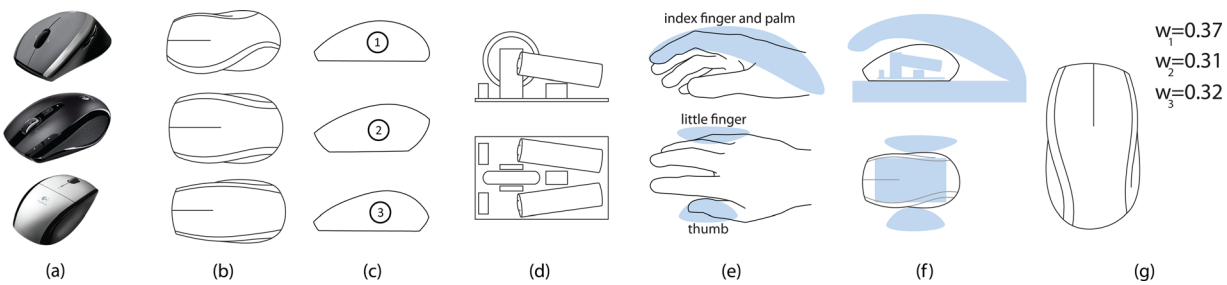

Fig. 16 Mouse design from top views and side views. Internal and external constraints are imposed. Optimization aims to minimize interference with these constraints.

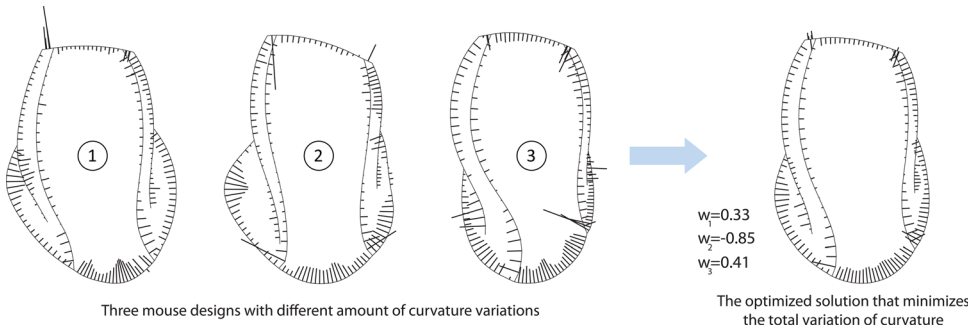

Fig. 17 Minimizing curvature variation. A design which has the least variation of curvature is sought.

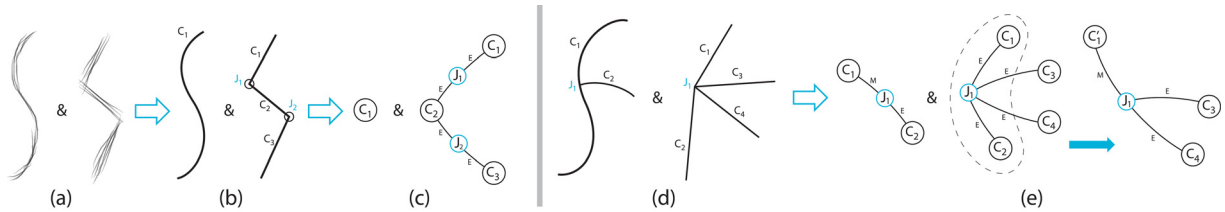

Fig. 18 The vectorization of sketches directly impact the resulting graph structure. (*a*) Strokes that define continuous or disconnected sets (*b*) are converted into single or multiple curve segments (*c*) resulting in different graph structures. (*d*) Matching of geometrically similar, (*e*) yet topologically different curves is possible by simplifying graphs through forming compound curves.

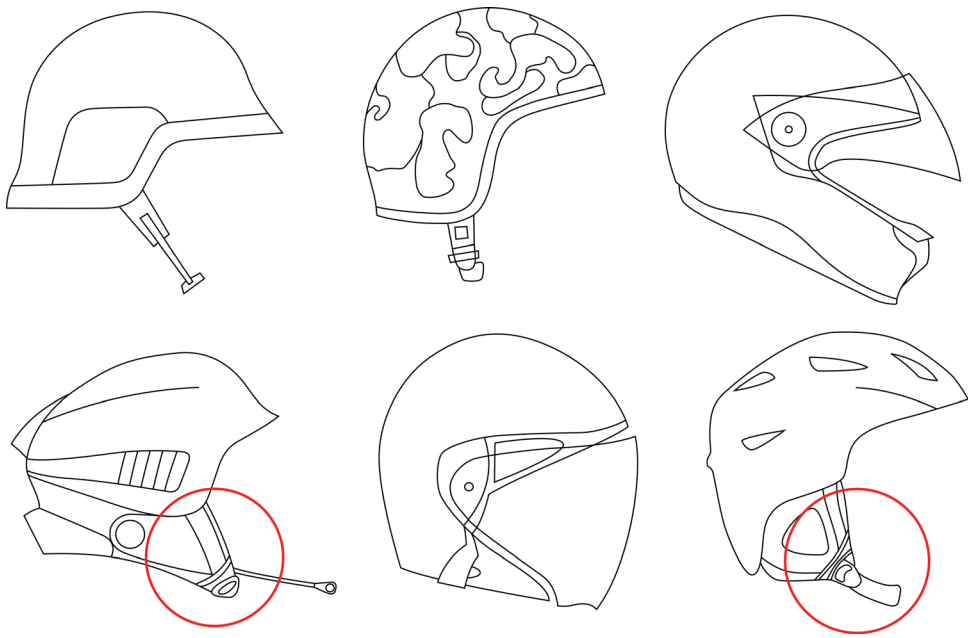

Fig. 19 Visual similarities (shown in red circles) among complicated geometries is difficult to detect with corresponding curves at the local level.

using a trainable stroke clustering algorithm and curve fitting. This algorithm is designed to analyze and learn the artists' stroke drawing preferences to automatically identify geometrically independent groups of strokes each of which defines a continuous curve. It does so using a set of pairwise stroke features that encode how proximate and aligned the strokes are and how likely it is for them to form a smooth, continuous curve if grouped. As a consequence, the algorithm is sensitive to breaking points which form intended discontinuities between the curves that are closely positioned at their end points. Figures 18(*a*) and 18(*b*) shows grouping of strokes that resulted in a smooth curve and zigzagged curves. Although this vectorization process is reliable in itself [30], the results directly impact the robustness of the following analyses as it may result in different sketch graph topologies for geometrically similar shapes (Fig. 18(*c*)). As a result, acceptable topological solutions may share relatively lower geometric similarity, and geometrically similar solutions might not be identified with topological matching.

In our experiments, we identified the most problematic cases which results in visually unacceptable matches. When studied at the local level, these cases typically contain geometrically similar curves or curve groups that are topologically different due to different numbers of breaking points. Figures 18(*d*) and 18(*e*) illustrates a simple example case. As a result of the vectorization process, the "S" shaped curve on the left was produced as a single curve, while a similar configuration on the right contains two curves that roughly correspond to the S curve. Although, the similarity may be obvious to the eye, the resulting graphs are different (Fig. 18(*e*)). However, attaching the first and the second curves (on the right) results in a compound curve and a topologically suitable graph which can then be matched with the other graph. Such potential curve pairs are typically curves that are connected at their end points forming slight kinks rather than sharp corners. In our experiments, we found that considering compound curves to be a major improvement on the robustness of the graph matching step.

On the other hand, visual similarities that appear at a global scale might not always have a topological matching solution that is suitable for the design space formation. Figure 19 shows six helmet design representing significantly different structures. Although, similar patterns are noticeable to the human eye (Fig. 19, see demarcated straps), a curve-to-curve correspondence matching solution does not exist. In such situations, our method begins to generate unacceptable results as the complexity of the exemplar drawing increases. This difficulty is partially a consequence of the curve-based representation of the geometry, and may be alleviated through other suitable forms of representations. One potential form might be a simpler abstraction of the drawings that can represent visually similar patterns with topologically identical sets of curves. Given such abstractions, our formulations are suitable to use the simpler representation to create and use the resulting design spaces.

9 Conclusion and Future Work

We describe an exemplar based shape synthesis and exploration method that uses the differences between the exemplars to form a design space. The contributions of our work is twofold. First, our approach allows design sketches to form a library of design ideas, from which novel design solutions can be synthesized. Second, it allows geometric constraints to be fluidly incorporated into the design process, thereby allowing such knowledge to be useful during the exploratory phases of design. Our approach alleviates the need for designers to manually parametrize their models and instead computes a natural parameterization defined by the deformation differences among the exemplars.

Our future directions include further studies on the inherent ambiguity of geometric correspondence. For this, geometric registration methods that utilize additional information beyond curves, such as *regions*, as well as multilevel abstraction methods that analyze input sketches at varying granularity are needed.

Acknowledgment

This research was supported by National Science Foundation Grant Nos. CAREER CMMI-0846730 and CMMI-1031703. All sketches used in the paper were drawn by the authors. The designs are courtesy of their respective owners.

References

[1] Ulrich, K. T., and Eppinger, S. D., 2008, *Product Design and Development*, 4th ed., McGraw-Hill, Irwin.
[2] Yang, M. C., 2003, "Concept Generation and Sketching: Correlations With Design Outcome," Proceedings of 2003 ASME Design Engineering Technical Conferences, Chicago, IL, Sept. 2–6.
[3] Ullman, D. G., Wood, S., and Craig, D., 1990, "The Importance of Drawing in the Mechanical Design Process," Comput. Graph., **14**(2), pp. 263–274.
[4] Schütze, M., Sachse, P., and Rmer, A., 2003, "Support Value of Sketching in the Design Process," Res. Eng. Des., **14**(2), pp. 89–97.
[5] Shah, J. J., Vargas-Hernandez, N., Summers, J. D., and Kulkarni, S., 2001, "Collaborative Sketching (C-Sketch)—An Idea Generation Technique for Engineering Design," J. Creat. Behav., **35**(3), pp. 168–198.
[6] Lazarus, F., and Verroust, A., 1998, "Three-Dimensional Metamorphosis: A Survey," Visual Comput., **14**(8), pp. 373–389.
[7] Alexa, M., 2002, "Recent Advances in Mesh Morphing," Comput. Graph. Forum, **21**, pp. 173–196.
[8] Chen, S. E., and Parent, R. E., 1989, "Shape Averaging and Its Applications to Industrial Design," IEEE Comput. Graphics Appl., **9**(1), pp. 47–54.
[9] Wang, H., 1995, "An Approach to Computer-Aided Styling," Des. Stud., **16**(1), pp. 50–61.
[10] Hsiao, S. W., and Liu, M. C., 2002, "A Morphing Method for Shape Generation and Image Prediction in Product Design," Des. Stud., **23**(6), pp. 533–556.
[11] Hsiao, S. W., and Chuang, J. C., 2003, "A Reverse Engineering Based Approach for Product Form Design," Des. Stud., **24**(2), pp. 155–171.
[12] Chen, L. L., Wang, G. F., Hsiao, K. A., and Liang, J., 2003, "Affective Product Shapes Through Image Morphing," Proceedings of the 2003 International Conference on Designing Pleasurable Products and Interfaces, ACM, p. 16.
[13] Kang, J. Y., and Lee, B. S., 2009, "Mesh-Based Morphing Method for Rapid Hull Form Generation," Comput.-Aided Des., **42**, pp. 970–976.
[14] Cagan, J., and Argawal, M., 1998, "A Bland of Different Tastes: The Language of Coffeemakers," Environ. Plan. B: Plan. Des., **25**, pp. 205–226.
[15] McCormack, J. P., Cagan, J., and Vogel, C. M., 2004, "Speaking the Buick Language: Capturing, Understanding, and Exploring Brand Identity With Shape Grammars," Des. Stud., **25**, pp. 1–29.
[16] Orsborn, S., Cagan, J., Pawlicki, R., and Smith, R. C., 2006, "Creating Cross-Over Vehicles: Defining and Combining Vehicle Classes Using Shape Grammars," Artif. Intell. Eng. Des. Anal. Manuf., **20**, pp. 217–246.
[17] Smyth, S. N., and Wallace, D. R., 2000, "Towards the Synthesis of Aesthetic Product Form," ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Proceedings of DETC'00, Baltimore, MD, pp. 218–224.
[18] Frazer, J. H., Frazer, J. M., Liu, X., Tang, M. X., and Janssen, P., 2002, "Generative and Evolutionary Techniques for Building Envelope Design," Proceedings of 5th International Conference on Generative Art, Milan.
[19] Bezirtzis, B. G., Lewis, M., and Christeson, C., 2007, "Interactive Evolution for Industrial Design," Proceedings of the 6th ACM SIGCHI Conference on Creativity and Cognition, ACM, Washington, DC.
[20] Wannarumon, S., Bohez, E. L. J., and Annanon, K., 2007, "Aesthetic Evolutionary Algorithm for Fractal-Based User-Centered Jewelry Design," Artif. Intell. Eng. Des. Anal. Manuf., **22**(1), pp. 19–39.
[21] Bentley, P. J., and Wakefield, J. P., 1997, "Conceptual Evolutionary Design by Genetic Algorithms," Eng. Des. Autom. J., **3**(2), pp. 119–131.
[22] Zeleznik, R. C., Herndon, K. P., and Hughes, J. F., 1996, "Sketch: An Interface for Sketching 3D Scenes," *SIGGRAPH 96 Conference Proceeding*, pp. 163–170.
[23] Igarashi, T., and Hughes, J. F., 2001, "A Suggestive Interface for 3D Drawing," Proceedings of ACM Symposium on User Interface Software and Technology, UIST'01.
[24] Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H., 1997, "Interactive Beautification: A Technique for Rapid Geometric Design," UIST '97: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, ACM, New York, NY, pp. 105–114.
[25] Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M., 2007, "Fibermesh: Designing Freeform Surfaces With 3D Curves," ACM SIGGRAPH 2007, ACM Transactions on Computer Graphics, San Diego.
[26] Karpenko, O. A., and Hughes, J. F., 2006, "Smoothsketch: 3D Free-Form Shapes From Complex Sketches," ACM Trans. Graph., **25**(3), pp. 589–598.
[27] Kara, L. B., and Shimada, K., 2007, "Sketch-Based 3D-Shape Creation for Industrial Styling Design," IEEE Comput. Graphics Appl., **27**(1), pp. 60–71.
[28] Bae, S. H., Balakrishnan, R., and Singh, K., 2008, "Ilovesketch: As-Natural-As-Possible Sketching System for Creating 3D Curve Models," Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, ACM, New York, NY, pp. 151–160.
[29] Olsen, L., Samavati, F. F., Sousa, M. C., and Jorge, J. A., 2009, "Sketch-Based Modeling: A Survey," Comput. Graph., **33**(1), pp. 85–103.
[30] Orbay, G., and Kara, L. B., 2010, "Beautification of Design Sketches Using Trainable Stroke Clustering and Curve Fitting," IEEE Trans. Vis. Comput. Graph., **17**(5), pp. 694–708.
[31] Cordella, L. P., Foggia, P., Sansone, C., and Vento, M., 2001, "An Improved Algorithm for Matching Large Graphs," Proceedings of the 3rd IAPR-TC-15 International Workshop on Graph-based Representations, Citeseer, Ischia, Italy, pp. 149–159.
[32] Sumner R. W., Zwicker, M., Gotsman, C., and Popovic, J., 2005, "Mesh-Based Inverse Kinematics," ACM Trans. Graph., **24**(3), pp. 488–495.
[33] Shoemake, K., and Duff, T., 1992, "Matrix Animation and Polar Decomposition," Proceeding of Graphics Interface '92, pp. 259–264.
[34] Moreton, H. P., and Séquin, C. H., 1992, "Functional Optimization for Fair Surface Design," Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, ACM, Chicago, IL, pp. 167–176.