Computers & Graphics 38 (2014) 69–77

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

CAD/Graphics 2013

Sketch2Jewelry: Semantic feature modeling for sketch-based jewelry design

Long Zeng [a], Yong-jin Liu [b,*], Jin Wang [c], Dong-liang Zhang [d], Matthew Ming-Fai Yuen [a,1]

[a] Department of Mechanical Engineering, Hong Kong University of Science and Technology, Room 2124B, Academic Building, Hong Kong
[b] Department of Computer Science and Technology, TNList, Tsinghua University, Beijing, China
[c] Department of Mechanical Engineering, Zhejiang University, Hangzhou, Zhejiang, China
[d] International Design Institute, Zhejiang University, Hangzhou, Zhejiang, China

ARTICLE INFO

Article history:
Received 5 August 2013
Received in revised form
5 October 2013
Accepted 19 October 2013
Available online 31 October 2013

Keywords:
Sketch-based modeling
Feature-based modeling
Semantic feature
Semantic retrieval
Jewelry design

ABSTRACT

Sketch-based modeling has attracted considerable attention in recent years. In this paper, we propose a semantic feature modeling system for sketch-based jewelry design, called *blue* Sketch2Jewelry. The newly devised semantic feature class encodes specific domain knowledge (jewelry design knowledge in this paper) and supplies prolific semantic information. The advantage of using semantic features is to narrow down the searching space in sketch-based feature retrieval and benefits the parameter selection from input sketches for feature instantiation and feature placement. Thus, the inaccuracy and ambiguity problems of freehand sketch inputs are alleviated within Sketch2Jewelry, compared to previous commercial feature-based modeling tools, e.g. SolidWorks, which are limited to fake sketches (i.e. not real freehand inputs). Since semantic features are high-level building blocks, together with sketch inputs, the proposed Sketch2Jewelry system can significantly improve the jewelry design efficiency. In addition, Sketch2Jewelry allows non-experts to sketch a complex jewelry model naturally and efficiently with design-by-feature. Examples are provided to demonstrate its usefulness.

1. Introduction

Many state-of-the-art computer aided design (CAD) systems, e.g. SolidWorks, UG, Pro/E, CATIA, etc, are feature-centered systems, i.e. each model is designed-by-feature. They improve the design efficiency by reusing parts stored in a database. However, with the scale of database increasing, there are two major problems: (1) it is difficult to find a required component from a large database; (2) it costs much time to adjust its location and orientation to match its neighbors.

The key observation in this work is that jewelry models can be always decomposed into components, belonging to some basic types. As shown in Fig. 1, the ring model is decomposed into six components: a ring body, a jewel, and four prongs, of three types. In addition, freehand sketching is a natural tool for modeling, by mimicking the traditional pen-paper functions to make a design process smooth and creative. In this paper, we propose a Sketch2Jewelry system, where the modeling blocks are semantic features. A semantic feature class is defined as a group of abstract data and rules [1]. To support sketch inputs, the semantic feature representation is extended in three aspects. Firstly, it contains at least one representative shape. Each shape is represented by Fourier descriptors and shape parameters. Secondly, it has four oriented relations. An oriented relation records the frequently used relationships between feature categories, which defines a subspace of semantic feature classes (detailed in Section 3.2). For example, JEWEL is usually surrounded by PRONG (shown in Fig. 1) that can be described by an oriented relation of the JEWEL class. Thirdly, it contains procedural rules to compute shape parameters and to generate shapes in over-determined or under-determined situations.

The streamline to handle an input sketch in Sketch2Jewelry is shown in Fig. 2. Firstly, the searching space is derived from the oriented relations of the sketch's spatial neighboring semantic features. The top-ten similar semantic feature classes of the input sketches are given in a candidate list. Then, the request semantic feature class is selected and its number of key points is used to guide the key point detection on the sketch. Subsequently, the shape parameters are extracted based on the correspondence between the key points of the selected semantic feature class and the sketch. Finally, the selected semantic feature class is instantiated and positioned in a location by minimizing the alignment error with input sketches.

Compared to state-of-the-art feature-based CAD systems and existing sketch-based modeling systems, Sketch2Jewelry supplies a sketch interface for semantic feature modeling in a jewelry

* Corresponding author. Tel.: +86 10 62784141; fax: +86 10 62782406.
E-mail addresses: wgz.282@gmail.com (L. Zeng), liuyongjin@tsinghua.edu.cn (Y.-j. Liu), meymf@ust.hk (M.-F. Yuen).
[1] Tel.: +852 23587189; fax: +852 23581543.

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.017

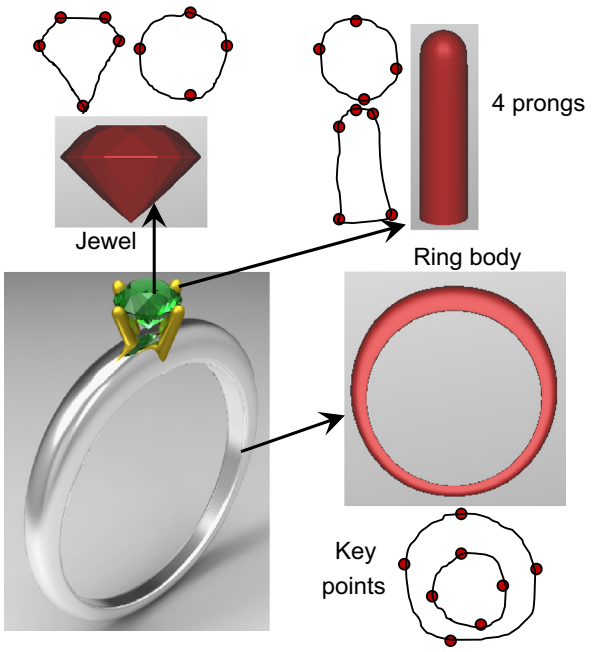


Fig. 1. A jewelry model is constructed by a ring body, a jewel, and four prongs, built by one or two simple sketch inputs.

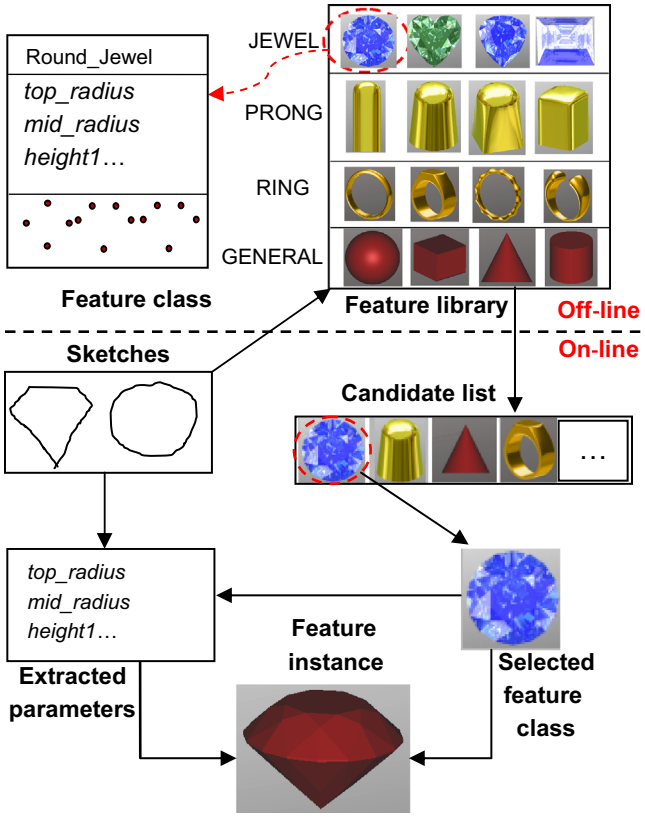


Fig. 2. Overview of Sketch2Jewelry.

domain. The feature retrieval, instantiation, and placement with sketches are done automatically, with the assistance of rich semantic information. We show in this paper that Sketch2Jewelry is an easy-to-use system, even for non-experts. For example, the ring shown in Fig. 1 can be designed in Sketch2Jewelry using less than 1 min. More examples are given in this paper to demonstrate its efficiency.

2. Related work

Sketch2Jewelry mainly takes both advantages of feature-based modeling and sketch-based modeling. Here, we review related work in these two areas.

2.1. Feature-based modeling

A *feature* is an identifiable aspect of a product that is mappable to a generic shape and functionally significant for some product life-cycle phase [2]. A feature model is a data structure that represents a part or an assembly in terms of its constituent features. There are two basic approaches to create a feature model: feature recognition and design-by-features. Both techniques have a predefined feature library which encodes the specific domain knowledge. They are complement with each other. In feature recognition, features are recognized from a geometric model which can convert the current huge geometric database into feature models. Each feature can be efficiently reused in new feature models. In design-by-features, a feature model is created increasingly by instantiating predefined feature classes and positioning them in proper locations. In feature-based modeling systems, features as high-level entities are used to enhance the design efficiency in a specific application domain.

A *semantic feature* is semantic-oriented definition, unlike the above feature definition which is still geometry-oriented. In Bronsvoort's group [3], a semantic feature is defined as a declarative class, where semantics and validity conditions are explicitly specified. In Yuen's group [4–6], a semantic feature is defined as a group of abstract data with attributes. Here, it is further extended as a group of abstract data and rules. *Abstract data* consists of shape data and semantic data, which can be trimmed for specific application domain. *Rules* usually encode domain-depended knowledge of a specific semantic feature.

2.2. Sketch-based modeling

A large body of literature had been published to supply effective sketch-based modeling tools, which has been demonstrated to be natural and efficient in the viewpoint of user interaction [7]. Our presented work is motivated by the success of two state-of-the-art sketch-based modeling methods [8] and the feature-based modeling method [5]. The major concern of sketch-based geometric modeling is how to interpret the input sketches to obtain a 3D model. Lipson and Shpitalni [9] interpret them as the projections of infinite 3D models and choose the best one according to optimization rules. SmoothSketch [10] and EasyToy [11] consider them as model's contours that are inflated into 3D round models. SilSketch [12] and FiberMesh [13] treat the sketches as handles for local model deformation. KnitSketch [14] mainly concerns the conceptual design process with a 2D pattern. We refer the interested reader to the survey paper for a comprehensive discussion [15].

Sketch-based model retrieval: The input sketches are adopted as indexes to retrieve similar parts/models from a large model database. Such systems have similar components and only vary in the process of how to match the 2D shape signatures with the 3D models. The Princeton search engine computes Fourier descriptors based on centroid distances which are rotation and translation invariant [16]. The Purdue search engine [17] adopts Fourier descriptor, Zernike moments, and 2.5D spherical harmonic descriptors as classifiers to retrieve 3D models. The NTU search engine [18] retrieves similar models based on light field descriptors. Pu et al. [19] retrieves 3D CAD models using sketches. More recently, Eitz et al. [20] and Liu et al. [21] investigate several local

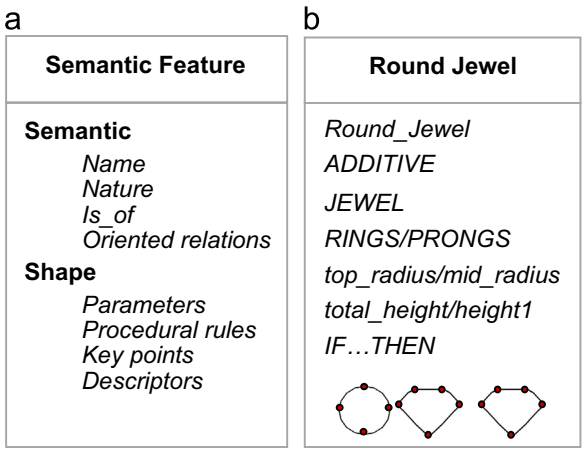


Fig. 3. Template class of semantic features and a specific example.

descriptors, which allow users to refine the retrieved models stroke-by-stroke[2].

Part-in-whole modeling: Recently, this modeling methodology is flourishing in sketch-based modeling [26] since it is in accordance with designers' thinking process: an arbitrary complex model can be constructed by assembling them with parts. Shin and Igarashi [22] and Lee and Funkhouser [23] both interpreted sketches as contours, used to retrieve parts which have similar contours in certain view directions. Rivers et al. [24] represented a model by a CSG tree of silhouette cylinders which are infinite extrusion of contours in the viewing direction. Schmidt et al. [25] procedurally defined a model with a BlobTree by sketch-based composition operators. In all the above methods, the sketches still act as guidance to position the retrieved parts into proper position of a scene. We adopt a similar idea in Sketch2Jewelry. In addition, the sketches in Sketch2Jewelry are further used to extract parameters to instantiate a feature class. The most similar work is done by Yang et al. [26], which instantiates parameterized objects by sketches, too. Their difference is given in more detail in Section 5.2.

3. Semantic feature modeling for jewelry design

In this section, a general template class of all semantic feature classes for jewelry design is given. Its key components are detailed, including oriented relations, procedural rules, shape descriptor and key point detection.

3.1. Semantic feature representation

A *semantic feature* is defined as a group of abstract data and rules [1]. Its contents can be grouped into semantic data and shape data, as shown in Fig. 3. *Semantic data* records engineering-related information, such as name, nature, is_of, and oriented relations.

- *Name* is used to identify a semantic feature instance, i.e. an entity instantiated from a specific semantic feature class.
- *Nature* is a feature property indicating whether it is ADDITIVE or SUBTRACTIVE to a feature model. It is useful when features are merged according to a CSG tree. The round jewel example in Fig. 3(b) has ADDITIVE property.
- *Is_of* is a relation indicating the feature category where this feature class belongs to.
- *Oriented relations* are subspaces of semantic feature classes (detailed in Section 3.2).

Shape data describes the geometric shape of a representative model in a semantic feature class. *Parameters*, together with *procedural rules* (Section 3.3), specify how to construct the feature shape. As the round jewel shown in Fig. 4, left, it is described by four parameters: *top_radius*, *mid_radius*, *height1,* and *total_height. Key points* and *descriptors* supply data for feature retrieval and feature instantiation (detailed in Section 3.5).

The semantic data and shape data can be augmented or trimmed for special application requirements.

3.2. Oriented relations

Oriented relation is a concept used to capture the patterns of local relationships of jewelry components. For example, a jewel is usually supported by prongs or seats. These kinds of relationships are important jewelry design knowledge. Such knowledge is helpful in resolving sketch ambiguity by narrowing down the solution space. To explore these patterns, a jewelry database (over 430 jewelry products) from Jewellery CAD/CAM Ltd[3] is analyzed and we conclude that

(a) Most of the frequently used jewelry components can be grouped into several categories, according to their functionalities.
(b) The often used relationships among categories are stable. For example, JEWEL follows with CUTTER in most cases.
(c) One category relates to more than one other categories spatially.

The oriented relations of a semantic feature class represent these patterns, as detailed below.

Feature categories in jewelry domain: A feature category is a subspace containing semantic feature classes, which have the same functions. Based on our current analysis, they are classified into six categories.

- JEWEL is a type of precious stones, used for decoration.
- SEAT is a type of circular shapes to compound bezel structures which are used to support JEWEL.
- PRONG is a type of elongated shapes to compound head structures which are used to support JEWEL.
- RING is a type of torus-like shapes to fit a customer's finger.
- CUTTER is a kind of hole-shapes where jewels are positioned.
- GENERAL contains semantic feature classes for general usage, e.g. sphere.

Each category contains many semantic feature classes with similar functionalities, but their shape parameters are different. Some exemplar models are given in the feature library shown in Fig. 2. These feature categories are application-dependent.

Oriented relations of a semantic feature class: A sketch around a semantic feature has different interpretation depending on their relative spatial location. Thus, in our case, the surrounding of a semantic feature is coarsely partitioned into four equal fan-areas, as the jewel shown in Fig. 5. In each fan-area, the compatible feature categories are listed. This is hard coded according to our previous analysis of the jewelry database. To blur the boundary between each fan-area, we also consider its two neighboring fan-areas. Thus, if an input sketch located in the third area, the system will infer that a PRONG feature or a RING feature may be inserted

[2] A *stroke* is continuous movement of a pen. A sketch can contain more than one stroke.

[3] http://www.jewelcadpro.com

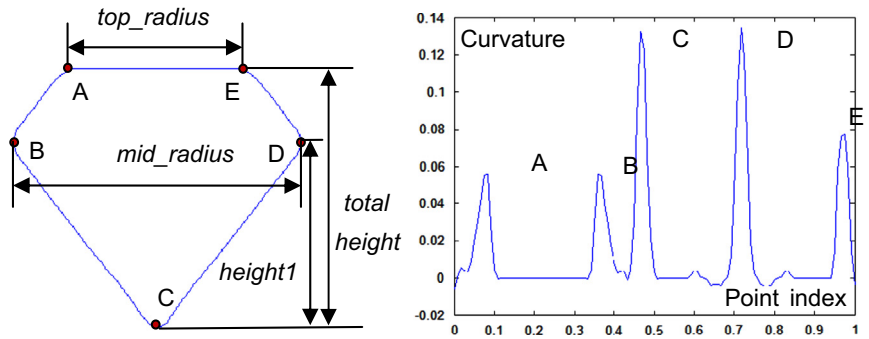
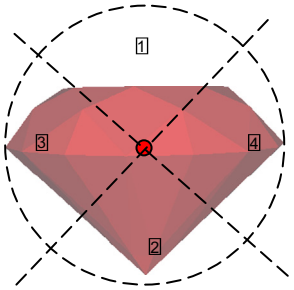


Fig. 4. Round jewel's key points and shape parameters (left), and the curvature profile of the front contour image (right).



Area	Oriented relation
1	NULL
2	RINGS/PRONGS
3	PRONGS/GENERAL
4	PRONGS/GENERAL

Fig. 5. Oriented relations around the round jewel.

according to the oriented relations listed in Fig. 5, right. In such a way, the searching space is narrowed down.

3.3. Procedural rules

The procedural rules are a sequence of commands, taking the key points of a sketch and its correspondence (computed in Section 3.5) as input. The procedural rules of a semantic feature will: (1) define the computation of shape parameters explicitly; (2) check the validity of the input parameters; and (3) construct the feature shape using the input parameters. The rules for the round jewel shown in Fig. 4 are listed in Fig. 6 and are explained below.

- Line 1 to line 3 compute the shape parameters of the round jewel, where *Dist*() is a function to compute the distance between two key points.
- Line 4 to line 14 check the completeness and validity of the shape parameters. Sketches usually supply insufficient information for feature instantiation. However, the lost data usually can be derived from the input sketches according to domain knowledge to generate a reasonable shape.
- Line 15 imports the round jewel and resizes according to the computed shape parameters.

The computation and validation of shape parameters are all explicit expressions, to improve efficiency of the sketch-based semantic feature modeling.

```
1.  top_radius = Dist(A, E)                          } Compute
2.  mid_radius = Dist(B, D)                          } parameters
3.  …
4.  IF only mid_radius is invalid THEN               } Validate
5.      top_radius = 0.8*mid_radius                  } parameters
6.      total_height = mid_radius                    } parameters
7.      height1 = 0.8*mid_radius                     } parameters
8.  END                                              } parameters
9.  …                                                } parameters
10. IF top_radius>mid_radius THEN                    } parameters
11.     mid_radius=1.25*top_radius                   } parameters
12. IF height1>total_heigth THEN                     } parameters
13.     height1=0.8*total_height                     } parameters
14. END                                              } parameters
15. Import("Circular_jewel", shape parameters)
```

Fig. 6. Procedural rules of the round jewel shown in Fig. 4.

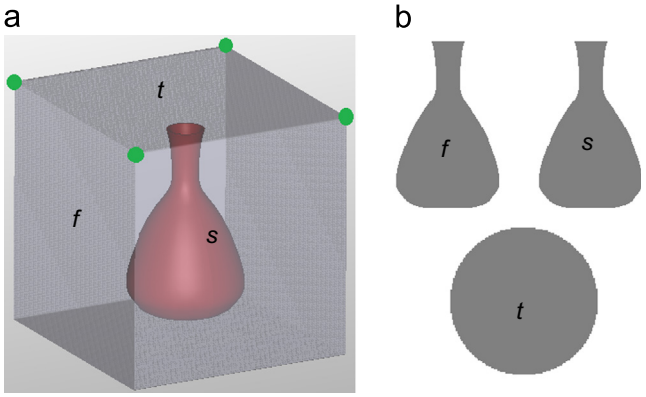


Fig. 7. Model projection configuration (f/s/t are the front/side/top views, respectively).

3.4. Shape descriptors

Shape descriptors are feature vectors of representative models of a semantic feature class. They are used to support sketch-based feature retrieval. Each representative model of a semantic feature is first normalized and oriented as suggested by Liu et al. [21] and rendered from three standard views (front/side/top). For example, the vase model (shown in Fig. 7a), the three projection images are shown in Fig. 7b. For each image, its contours are extracted using a standard contour extraction algorithm of OpenCV. However, the result may be noised, as shown in Fig. 8, left. To facilitate the subsequent key point detection process, each point is filtered by averaging within a filter-window [27] (the width of the filter-window in Sketch2Jewelry is five-point width). The results are satisfied, as shown in Fig. 8, right.

100 sample points are resampled on a filtered contour by equal arc-length. Then, the Fourier descriptor based on centroid distance [27] is adopted to encode the contour (10 Fourier coefficients is used in our experiment).

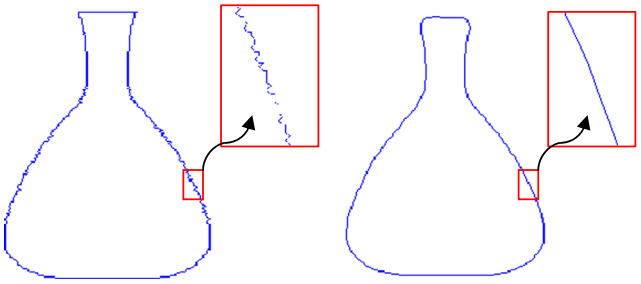


Fig. 8. Compare the feature contours without (left) and with (right) smoothing.

3.5. Key points detection

For a sampled contour $C=(x(u),y(u))$, $u\in[0,1]$, the curvature at parameter u is computed by:

$$\kappa(u)=\frac{\dot{x}(u)\times\ddot{y}(u)-\dot{y}(u)\times\ddot{x}(u)}{(\dot{x}(u)^2+\dot{y}(u)^2)^{3/2}} \quad (1)$$

where $\dot{x}$ and $\ddot{x}$ are first and second derivatives, respectively. The curvature profile $\psi(C)$ along the contour C is

$$\psi(C)=\{\kappa(u)|u\in[0,1]\} \quad (2)$$

One example of the curvature profiles of the round jewel is shown in Fig. 4, right.

To find the key points of a contour, a set of candidate key points are obtained from two sources. The first source is the most left, right, bottom, and top points of the contour. The second source is those points with maximum curvature locally, i.e. a local window (twice width of that of the filter-window) on the curvature profile, denoted as *curvature-window*. In a curvature-window, if the local maximal curvature is twice as larger as the local mean curvature, then it is considered as a candidate key point. Finally, the real key points are obtained by a merging process. That is, if two candidate key points are near each other in a given threshold, they will be merged.

In Fig. 4, left, the front feature contour of the round jewel feature has five key points. They are detected from its curvature profile (Fig. 4, right). However, four of them (A, B, C, and D) are overlapped with the most left/right/top/bottom points and are therefore merged.

4. Sketch-based modeling with semantic features

In Sketch2Jewelry, sketches are used to retrieve user requested semantic feature, to instantiate the selected semantic feature, and to position the instantiated semantic feature. We show in this section that the semantic feature retrieval and feature instantiation can benefit from the proposed semantic feature representation in Section 3.

4.1. Feature retrieval with semantics

The computation process of shape descriptor introduced in Section 3.4 is applied to input sketches, too. The computation maps 100 sample points on a stroke to one point in another 10-dimensional space (since 10 Fourier coefficients are adopted in Section 3.4), denoted as a Fourier space, as the v_0 shown in Fig. 9. A feature representative model is approximated with three projection images. Each image contains at least one contour which is presented by a Fourier point in the Fourier space. Thus, a feature class, f_i, is represented by a set of Fourier points, e.g. $f_i=\{v_1^i,v_2^i,v_3^i\}$, as shown in Fig. 9. Thus, the most similar semantic feature to one

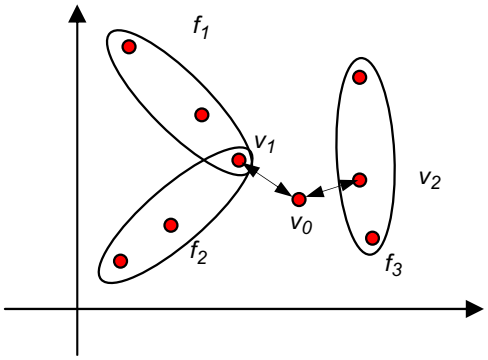


Fig. 9. Similarity metric computed as minimum point distance in a Fourier space (red dots are Fourier points; f_i is the ith feature; v_0 is the Fourier point of the input stroke). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

stroke is the one containing a Fourier point whose distance to v_0 is smallest.

Stroke-by-stroke retrieval: Stroke-by-stroke based feature retrieval means each input stroke has its own descriptors which are used to filter the unwanted features. As the strokes increasing, the matching results are gradually refined. To find the top-ten similar features, each input stroke is smoothed, averaged, and transformed into a Fourier point first (i.e. the Fourier point v_0 shown in Fig. 9). Then, for each feature, the L_2 norm is computed between its constituted points and v_0. The minimum distance is accumulated to this feature's dissimilarity value. Finally, the top-ten features with a smaller dissimilarity value are selected from the feature library. Fig. 10 shows an example of stroke-by-stroke retrieval. In Fig. 10a, the first circular stroke cannot distinguish the prong feature from the round jewel feature since they are both circular viewed from the top. However, if sketching is continued, the second stroke (Fig. 10b) will immediately distinguish the prong feature from the round jewel, which ranked as top one.

In Sketch2Jewelry, the stroke sequence is not important. It inherits all the benefits of sketch-based modeling. The users do not need to care about the drawing order in the three standard views. The user can refine the retrieval results stroke-by-stroke in any order and any place.

Program 1. SearchSpace(pStroke).

```
Input: pStroke //input sketch
Output: F    //list of feature categories
1. Find the nearest feature instance, nearF of
   pStroke
2. Indexes of occupied fan-area of nearF
3. For each id of Indexes
4. Push its oriented relations to F
5. End
```

Computation of searching space: Using the oriented relations of semantic feature classes, the searching space can be narrowed down according to Program 1. In line 1, the nearest feature instance is found by the centroid of the bounding box of existing feature instances and the input stroke. In line 2, the indexes of the occupied fan-areas may be greater than one since a stroke may cross more than one fan area. For example, if the round jewel shown in Fig. 5 is the nearest feature instance and the stroke is located in the second and fourth fan-areas, then the oriented relations RING/PRONG/GENERAL will be inserted to the feature categories list F (line 3 to line 5).

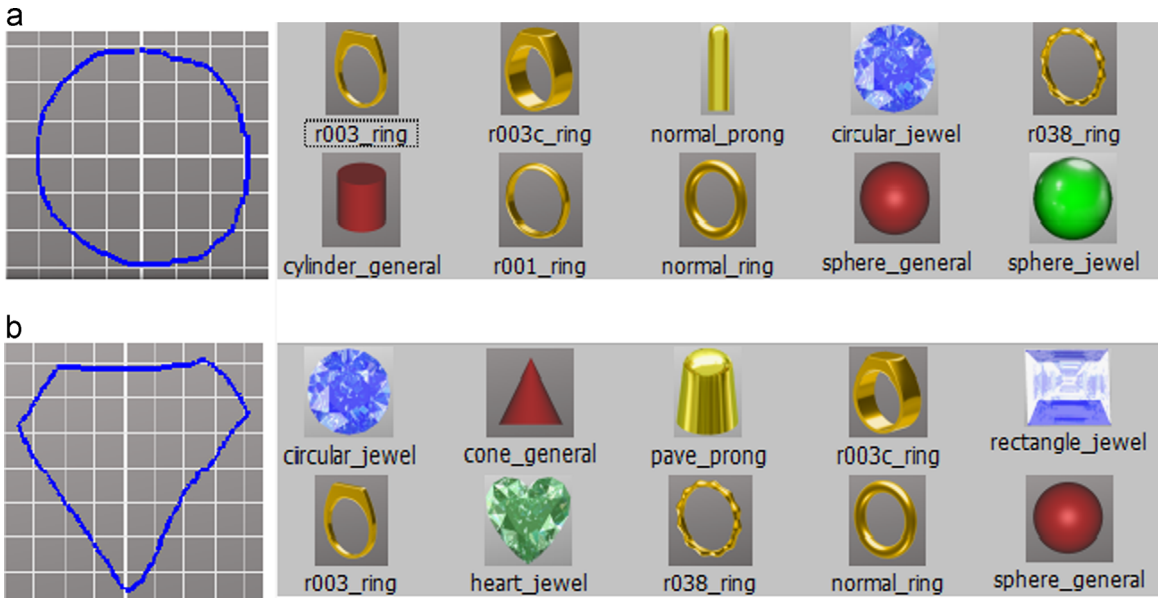


Fig. 10. Feature retrieval stroke-by-stroke.

All the feature categories in list F define the size of the searching space. At the very beginning, F is initialized to contain all feature categories in a jewelry domain. It is updated according to Program 1 after each stroke. Thus, at any time, the size of the searching space can be computed by

$$|S| = \sum_{i \in |F|} |F_i| \tag{3}$$

where $|F|$ is a number of feature categories in the list F, $|F_i|$ is the number of semantic feature classes in the subspace of F_i and $|S|$ is the total number of semantic feature classes in the searching space.

4.2. Feature instantiation with semantics

To instantiate a selected semantic feature class, the shape parameters are extracted from input sketches by detecting key points on sketches and by finding their correspondence, with the help from the rich semantics of the selected semantic feature class.

Key points detection on sketches with semantics: The key points detection technique given in Section 3.5 is not satisfied since freehand sketches are noisy. The popular Douglas–Peucker algorithm [28] and its variation [29] can also identify key points, but its number of key points depends on a given threshold. However, in our case, the number of key points can be obtained in advance from the selected semantic feature class. Thus, our task is to choose a given number of key points from an input sketch, as shown in Program 2.

Program 2. KeyPointFromSketch(N, pStroke).

```
Input: N //given number of key points
  pStroke //input stroke
Output: keys  //list of N key points
1. For each point of stroke
2.  Compute curvature, stored in curvlist.
3. End
4. For each point in curvlist
5.  winWidth=1 at ci
6.  do {
7.    IF curvlist[ci] is maximum THEN
8.      winWidth=winWidth+1
9.    ELSE
10.      Add winWidth to widthlist
11.      Break
12.    END
13.  }while{winWidth < half of curvlist
14. End
15. Sort the widthlist in decrease order
16. Choose the former N elements
```

From line 1 to line 3, the curvature of each point is computed. Line 4 to line 14 computes the maximum window width of each point, in which this point's curvature is maximal. In line 15 and line 16, the list of maximum window width is sorted and the former N points are selected as key points. This algorithm can guarantee the same number of key points on the input stroke which is obtained. It will make easier to find the correspondence.

Key point correspondence via cross-correlation: The correspondences of two curvature profile, e.g. ψ_1 and ψ_2, are obtained by minimizing the cross correlation energy [30]. That is,

$$\min_{v \in [0,1]} E(\psi_1, \psi_2, v) \text{ and}$$

$$E(\psi_1, \psi_2, v) = \frac{\sum_{u \in [0,1]} (\kappa_1(u) - \overline{\kappa_1})(\kappa_2(u+v) - \overline{\kappa_2})}{\sqrt{\sum_{u \in [0,1]} (\kappa_1(u) - \overline{\kappa_1})^2} \sqrt{\sum_{u \in [0,1]} (\kappa_2(u) - \overline{\kappa_2})^2}} \tag{4}$$

where the $\kappa_1(u)$ and $\overline{\kappa_1}$ are the curvature at u and the average curvature of ψ_1, similar to $\kappa_2(u)$ and $\overline{\kappa_2}$; v is the phase shift between two curvature profiles. The optimal solution can be found with a standard golden section search method.

The higher the correlation energy E is, the higher the similarity between the two shapes is. One example is shown in Fig. 11, in which curvature profiles ψ_1 and ψ_2 are the input sketch contour and front feature contour of the selected block feature class, respectively. The maximum cross correlation energy value is 0.64, obtained at $v = 0.02$. This means the start point of the sketch contour corresponds to the third point of the front feature contour of the block feature.

Parameter computation with semantics: The parameters are computed based on the key points and their correspondence of input sketches. For example, the *top_radius* of the round jewel (Fig. 4, left) is computed as the horizontal-coordinate difference between key points A and E; the *total_height* is the average

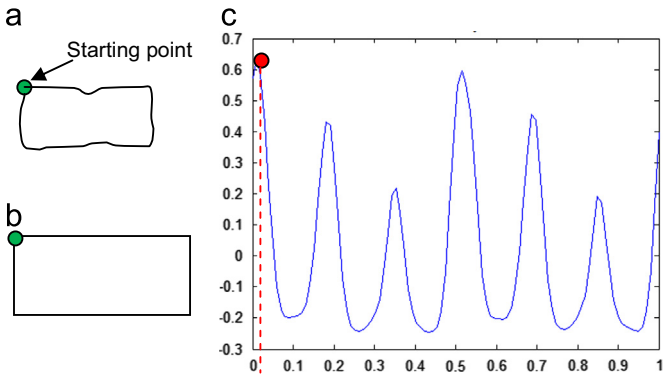


Fig. 11. Cross correlation energy between two curvature profiles.

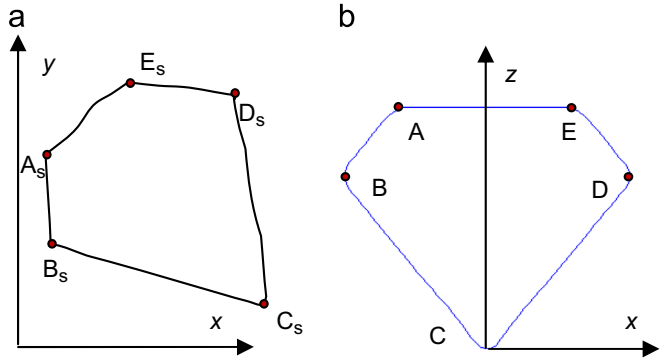


Fig. 12. The correspondences of key points between (a) a sketched contour and (b) the feature contour of a round jewel feature class.

vertical-coordinate difference between key points A and C. These are hard-coded by the procedural rules of a specific semantic feature (refer to Section 3.3). However, the over-determined and under-determined problems need to be resolved.

The over-determined problem is due to the inconsistence of sketches given in different views, for example, the key points A and E (defined in Fig. 4, left) of the round jewel in two input sketches in top and front views (Fig. 10). A simple but effective solution to the over-determined problem is to designate a primary view among the three standard views. Here, the primary view is selected as the view containing the maximum number of key points. If two views have the same number of key points, view is chosen by a priority: front, side, and top. For example, the primary view of the round feature class is the front view (Fig. 4). Meanwhile, the extracted parameters from primary view also have priority to other views. Thus, the value of the *mid_radius* is extracted from the second stroke in Fig. 10b, not the first stroke in Fig. 10a.

The under-determined problem often occurs since users may select a semantic feature class from the candidate list at any time. This is resolved by the procedural rules, as shown in Fig. 6.

4.3. Feature placement

When a selected semantic feature class is instantiated, the objective of the feature placement is to minimize the errors between their corresponding key points. As shown in Fig. 12, the key points on the image contour (Fig. 12b) should be mapped to the key points on the rotated input sketch (Fig. 12a). Since the selected semantic feature is instantiated with the parameters computed from the key points of the input sketch contour, the feature placement is actually simplified to a rigid transformation, i.e. translation and rotation.

Feature placement with rigid transformation is more stable than previous methods. For example, the one given in [31], which computes an affine matrix to deform the template model into the sketched contours, is error-prone since the sketches are noisy. Instead, sketches are imprecise in essence and only a coarse shape is described, it is not necessary to fit a template model to the sketch contour completely. Alternatively, it is reasonable to infer a set of parameters from the sketches to generate a beatified feature.

5. Results and discussion

The Sketch2Jewelry has been implemented as a prototype system. All the operations shown in the accompanying demo video are run interactively on a desktop PC running Window 7 operating system with Intel Core i5 CPU750 2.67 GHz and 3GB RAM.

An overview of the Sketch2Jewelry system is given in Fig. 2. It consists of two parts: off-line part and on-line part. For the off-line part, the main task is to construct a feature library of semantic features using the methods given in Section 3. For the on-line part, the system pre-processes the input sketches, retrieves similar features, extracts parameters and instantiates the selected feature class (detailed in Section 4). All these steps are performed interactively.

To demonstrate the main features of Sketch2Jewelry, our current feature library contains about 80 features from the jewelry industry. For example, to design the ring model shown in Fig. 1, one step is shown in Fig. 13. The user can choose to work under a workspace with multiple sub-windows or just one. If accustomed to work with a window at a time, the users can switch among the four standard views (front/side/top/3D) by using hot keys. When sketching in one view, the projections in other views are also shown interactively as a position reference, as shown in the front view of Fig. 13. Though the positions of the first stroke and second stroke have no overlap, the round jewel class can be successfully instantiated and positioned near the center of the four prongs using the primary view rule (Section 4.2).

Fig. 14 shows a complex ring model designed in Sketch2Jewelry. All the components before Boolean operations of the designed ring model are also shown in Fig. 14. Though it seems much more complex than the ring model shown in Fig. 1, it contains only six feature types. For example, the slot feature is bent and subtracted from the ring body feature. The nature of the cutter feature is always SUBTRACTIVE which cuts a hole to position the round jewels.

Fig. 15 shows an earring model, containing four hearts, three seats, three cutters, three jewels and two general features. These fifteen features belong to five feature categories. They can be easily designed in Sketch2Jewelry.

5.1. Comparison with feature-based modeling

When the same models (e.g. Figs. 1 and 14) are created in SolidWorks (the same parts are prestored in its part library), the time cost varies a lot for different users, depending on their familiarity of SolidWorks. However, in Sketch2Jewelry, all time costs are almost the same, independent of users' knowledge of our system.

This difference is mainly because of the different modeling sequences in SolidWorks and Sketch2Jewelry. In SolidWorks, there are usually three steps to instantiate a feature from library: finding a feature from library, configuring the feature's shape parameters, and adjusting the feature instance's location and orientation. Thus,

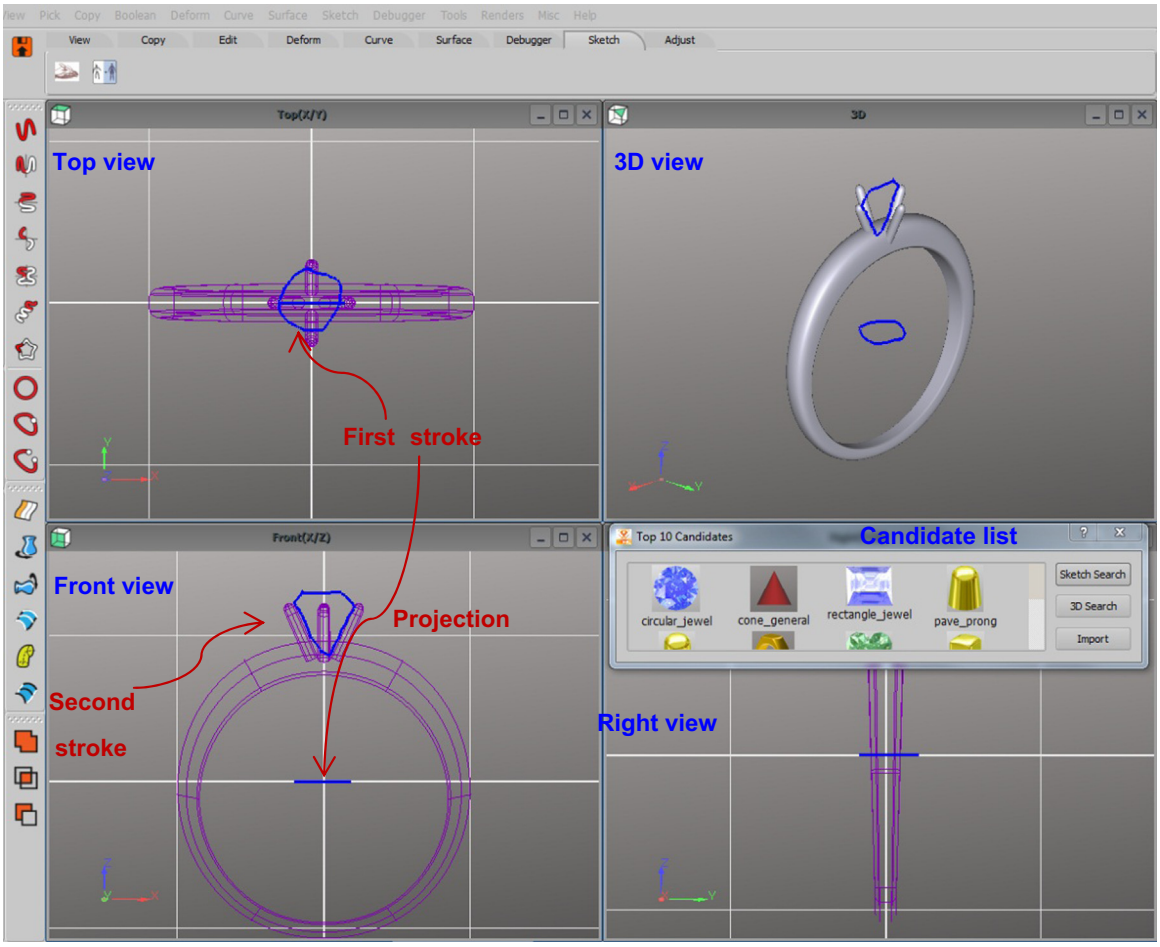


Fig. 13. Sketch the round jewel feature in Sketch2Jewelry.

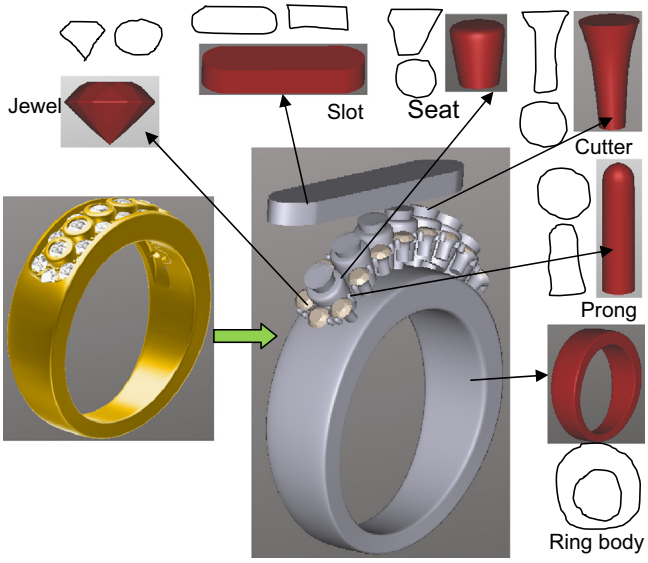


Fig. 14. A complex ring model designed in Sketch2Jewelry.

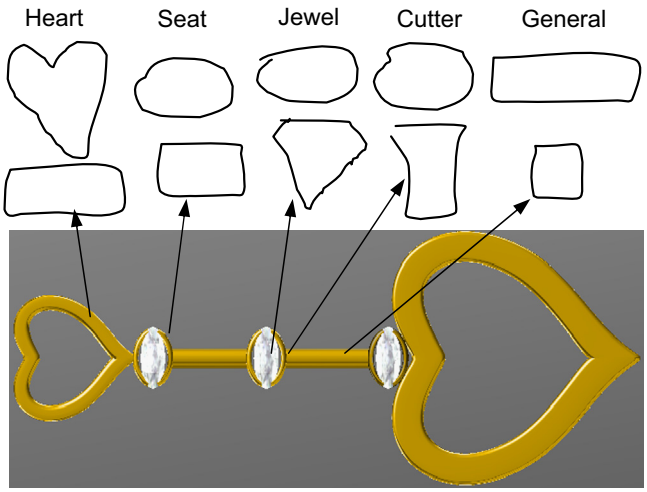


Fig. 15. A earring model designed in Sketch2Jewelry.

the total time creating a feature model is

$$T_1 = n \cdot (t_f + t_c + t_a) \tag{5}$$

where n is the number of features in the model; t_f, t_c, and t_a are the time cost of the above three steps, respectively. However, in Sketch2Jewelry, the total time cost in constructing a feature model is

$$T_2 = n \cdot t_s \tag{6}$$

where t_s is the time cost of sketching up to three silhouette contours of each feature component.

From Eqs. (5) and (6), we know the time cost of both SolidWorks and Sketch2Jewelry is proportional to the number of features. However, in Eq. (5), the more familiar with the SolidWorks the users are, the smaller the t_f, t_c, and t_a are. In Eq. (6),

the t_s is almost constant since our sketching mimics the traditional pen-and-paper input.

From a preliminary test using the models in Figs. 1 and 14, we found the time cost of Sketch2Jewelry is twice in average as fast as that of SolidWorks.

5.2. Comparison with sketch-based modeling

One close related work to Sketch2Jewelry is the sketch-based modeling of parameterized objects [26]. However, in Sketch2Jewelry:

- The way encoding model knowledge is more flexible. In [26], a specific object is described by a template which contains a parent node and a set of child nodes. Nodes are parameterized and their relative locations are hard-coded in the template. In Sketch2Jewelry, the common building blocks are encoded into the feature library. Then, a complex model is created in a design-by-feature way. It makes the design activities more creative.
- The efficiency problem mentioned in [26] is handled well. Firstly, the graph matching algorithm in [26] is replaced with a visual-based retrieval strategy and accelerated with semantics. Secondly, the searching space is narrowed down by semantics in Sketch2Jewelry. That is, the searching efficiency will not be reduced when the feature library scale increases.

6. Conclusion and feature work

Several contributions have been made within Sketch2Jewelry. First, it is a feature-centered modeling system with freehand sketch inputs, allowing users to create a model feature-by-feature. No specialized knowledge and artistic sketching skills are required. This system is easy-to-learn and easy-to-use. Second, semantic feature classes are augmented with oriented relations, procedural rules, and shape descriptors to support sketches. Third, the rich semantics are applied to semantic feature retrieval and instantiation.

In practice, some limitations need to be noted. First, we require all semantic feature classes are simple. This is because the ambiguity from sketches increases when they are complex. However, this requirement is acceptable in modeling since the finer granularity it is, the more flexibility we have. Second, coding the semantic feature classes in a specific domain is a tedious job, but only once is needed and can be reused. In the future, some sketch-based tools will be designed to easy this process. Third, the placement of an instantiated feature is not accurate. Thus, current implementation of Sketch2Jewelry is useful in illustrating design ideas. Further constraints need to be inserted and satisfied (just as SolidWorks) for accurate assembly purpose.

In addition, we intend to add more feature classes from different application domains into the feature library to verify the performance of our sketch-based system and a more comprehensive user study will be conducted.

Acknowledgments

The authors would like to thank the reviewers for their constructive comments that helped to improve this paper. The authors would also like to thank Edmond Li, from Jewellery CAD/CAM Ltd, Hong Kong, for his stimulating discussions. This work was partially supported by Jewellery CAD/CAM Ltd, the Natural Science Foundation of China (Grant nos. 61272228) and National High-tech R&D Program of China (863 Program) (Grant no. 2012AA011801). The work of L. Zeng is supported by Hong Kong Innovation and Technology Fund (ITF) UIM/210. The work of Y.J. Liu was supported in part by the Program for NCET-11-0273 and TNList Cross-discipline Foundation. The work of J. Wang was partially supported by the National Science Foundation of China (Grant nos. 61103106, 51275460).

References

[1] Zeng Long, Sketch-based semantic feature modeling with q-complex data structure [Ph.D. thesis], Hong Kong University of Science and Technology; 2012.
[2] Shah J, Mäntylä M. Parametric and feature-based CAD/CAM: concepts, techniques and applications. New York: John Wiley & Sons; 1995.
[3] Bidarra R, Bronsvoort W. Semantic feature modelling. Compu-Aided Des 2000;32(3):201–25.
[4] Au C, Yuen M. A semantic feature language for sculptured object modelling. Comput-Aided Des. 2000;32(1):63–74.
[5] Liu Y-J, Lai K-L, Dai G, Yuen M-M-F. A semantic feature model in concurrent engineering. IEEE Trans. Autom. Sci. Eng. 2010;7(3):659–65.
[6] Zeng L, Liu Y-J, Lee SH, Yuen MM-F. Q-Complex: efficient non-manifold boundary representation with inclusion topology. Comput.-Aided Des 2012;44(11):1115–26.
[7] Fu Q, Liu Y-J, Chen W, Fu X. The time course of natural scene categorization in human brain: simple line-drawings vs. color photographs. J Vis 2013;13(9):1060.
[8] Chen T, Cheng M-M, Tan P, Shamir A, Hu S-M. Sketch2photo: internet image montage. In: ACM SIGGRAPH Asia 2009; 2009. p. 124:1–10.
[9] Lipson H, Shpitalni M. Correlation-based reconstruction of a 3d object from a single freehand sketch. In: ACM SIGGRAPH 2007 courses; 2007.
[10] Karpenko OA, Hughes JF. SmoothSketch: 3D free-form shapes from complex sketches. In: ACM SIGGRAPH 2006; 2006. p. 589–98.
[11] Liu Y-J, Ma C-X, Zhang D-L. Easytoy: plush toy design using editable sketching curves. IEEE Compu Graph Appl 2011;31(2):49–57.
[12] Zimmermann J, Nealen A, Alexa M. Silsketch: automated sketch-based editing of surface meshes. In: Proceedings of the 4th Eurographics workshop on sketch-based interfaces and modeling; 2007. p. 23–30.
[13] Nealen A, Igarashi T, Sorkine O, Alexa M. Fibermesh: designing freeform surfaces with 3d curves. In: ACM SIGGRAPH 2007; 2007.
[14] Ma C-X, Liu Y-J, Yang H-Y, Teng D-X, Dai G-Z. Knitsketch: a sketch pad for conceptual design of 2d garment patterns. IEEE Trans Autom Sci Eng 2011;8(2):431–7.
[15] Olsen L, Samavati FF, Sousa MC, Jorge JA. Sketch-based modeling: a survey. Comput Graph 2009;33(1):85–103.
[16] Funkhouser T, Min P, Kazhdan M, Chen J, Halderman A, Dobkin D, Jacobs D. A search engine for 3d models. ACM Trans Graph 2003;22(1):83–105.
[17] Hou S, Ramani K. Calligraphic interfaces: classifier combination for sketch-based 3d part retrieval. Comput Graph 2007;31(4):598–609.
[18] Chen D-Y, Tian X-P, Shen Y-T, Ouhyoung M. On visual similarity based 3d model retrieval. Comput Graph Forum 2003;22(3):223–32.
[19] Pu J, Lou K, Ramani K. A 2d sketch-based user interface for 3d cad model retrieval. Comput-Aided Des 2005;2(6):717–25.
[20] Eitz M, Hildebrand K, Boubekeur T, Alexa M. Sketch-based image retrieval: benchmark and bag-of-features descriptors. IEEE Trans. Visualization Comput. Graph 2011;17(11):1624–36.
[21] Liu Y-J, Luo X, Joneja A, Ma C-X, Fu X, Song D. User-adaptive sketch-based 3d cad model retrieval. IEEE Trans Autom Sci Eng 2013;10(3):783–95.
[22] Shin H, Igarashi T. Magic canvas: interactive design of a 3-d scene prototype from freehand sketches. In: Proceedings of graphics interface; 2007. p. 63–70.
[23] Lee J, Funkhouser T. Sketch-based search and composition of 3d models. In: Proceedings of the fifth eurographics conference on sketch-based interfaces and modeling; 2008. p. 97–104.
[24] Rivers A, Durand F, Igarashi T. 3d modeling with silhouettes. In: ACM SIGGRAPH 2010; 2010. p. 109:1–8.
[25] Schmidt R, Wyvill B, Sousa MC, Jorge JA. Shapeshop: sketch-based solid modeling with blobtrees. In: ACM SIGGRAPH 2006 courses, SIGGRAPH '06. New York, NY, USA: ACM; 2006.
[26] Yang C, Sharon D, van de Panne M. Sketch-based modeling of parameterized objects. In: ACM SIGGRAPH 2005 sketches; 2005.
[27] Zhang D, Lu G. A comparative study of curvature scale space and fourier descriptors for shape-based image retrieval. J Vis Commun Image Represent 2003;14(1):39–57.
[28] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: Int J Geograph Inform. Geovisualization 1973;10(2):112–22.
[29] Lewis JP, Anjyo K. Identifying salient points. In: ACM SIGGRAPH ASIA 2009 sketches, SIGGRAPH ASIA '09, ACM; 2009. p. 41:1–1.
[30] Liu Y-J, Chen T, Chen X-Y, Chang TK, Yuen MMF. Planar shape matching and feature extraction using shape profile. In: Proceedings of 5th international conference on advances in geometric modeling and processing. Springer-Verlag; 2008. p. 358–69.
[31] Kraevoy V, Sheffer A, van de Panne M. Modeling from contour drawings. In: Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling; 2009. p. 37–44.