Wang and Qin *EURASIP Journal on Image and Video Processing* (2017) 2017:57
DOI 10.1186/s13640-017-0207-0

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Surface editing using swept surface 3D models

Xiaohui Wang and Jingyan Qin*

Abstract

This paper presents a system for editing strip-like patches on the mesh using swept surface. The user first selects a rough cylinder-like region, the boundary of which is automatically refined. A swept surface approximation is automatically done, including extraction of the trajectory and the corresponding deforming 2D curves. The swept surface is described by a map from a real interval to both rigid body deformations and 2D curves. These 2D curves are analyzed and decomposed into many elements, which can be edited respectively, and a novel pattern analysis is also performed on these elements to extract curve patterns. These 2D curves serve as handles for controlling the geometry, and modifications to them can change the geometry directly. Thus, users can just edit the pattern or element curve to control the global geometry. We show the novelty and efficiency of our framework using varietal demonstrations.

Keywords: Swept surface, Feature editing, Pattern analysis

1 Introduction

To meet the growing needs for 3D models, modeling from existing models is becoming a practical way for generating new models rapidly and efficiently. Local surface editing is a popular way for this, which is also an important and active research topic in computer graphics.

The state-of-the-art local surface editing techniques depend on intrinsic coordinates, such as multiresolution decompositions [1] and various shape deformation frameworks. Both provide efficient and reliable functionalities for deformation, geometry transferring, and so on. However, such techniques focus on global deformation of the surface patch or geometry transferring, which is not competent for more sophisticated surface editing with rich user interaction.

Though professional modeling packages [2, 3] can provide precise control using parametric patches or subdivision surfaces, they are really sophisticated and tiresome to use, especially for inexperienced users. In users' point of view, surface editing or modeling by 2D curves is always the ideal choice. The most famous example is teddy [4], which is a huge success and is significantly improved by a recent work [5].

In the meanwhile, pattern recognition techniques develop rapidly in recent years. In the graphics field, techniques such as geometry feature (or pattern) clustering and analysis play a more and more important role. Typical work is [6], in which local featured areas are clustered and matched using geometrical descriptors. In a more recent work [7], they present a framework for discovering structural regularity including similar transformations. However, geometric features are hard to analyze. Current approaches focus on similar transformations extraction or local geometrical descriptor analysis. Both rely on simple geometric properties such as curvature. The applications are limited to simple geometric patterns with similar curvature distributions. More complicated patterns sensitive to human beings cannot be analyzed. Using such analysis for easier surface editing is even harder. To the best of our knowledge, no existing technique exploits pattern recognition for better user experience. Just think of the following scenario: you select a part and edit it, all similar parts changed, and then, the editing work is significantly simplified.

As discussed above, our goal is to design a local surface editing approach for free and convenient design, especially for inexperienced users. We particularly desire the following properties for the approach:

* Correspondence: qinjingyanking@foxmail.com
School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, China

Springer Open

© The Author(s). 2017 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

- Usability: It must be sufficiently easy to use. 2D curve editing instead of sophisticated parameter control is always preferred.
- Intelligence: It must be smart enough to handle boring repetitive work automatically, such as applying similar modification to similar parts can be done by just selecting one part, editing it, and all done.
- Functionality: It must be powerful enough for complicated surface editing.

However, it is a challenging job. Precise control and easier user interaction is somewhat a contradiction. And intelligent interaction such as handling repetitive work automatically is even harder.

Luckily, we observe that a significant amount of feature regions we can see in 3D models are cylinder-like and have the nature of sweeping, that is, they can be described by swept surface. In general, it is known as swept volume, which is generated by the motion (including deformation) of arbitrary 3D object. In this paper, we only consider swept surface generated by restricting the object to be planar curves. The swept surface is a natural, intuitive, and convenient 3D modeling method with a long history and is widely used in computer-aided design (CAD) field. It is generated by moving curve $C_1(v)$ along another curve $C_2(u)$, which are called profile curve and trajectory curve respectively, and $C_1(u)$ may be deformed during its motion along the trajectory. Swept surface can naturally decompose the shape into the trajectory and associated profile curves. If we restrict $C_1(v)$ to be planar curves, it can be described and analyzed easily using cross-planes given by local Frenet frames [8] of the trajectory. Then, editing the surface details becomes editing 2D curves, which provides both convenient and precise controlling of the shape. The most exciting thing is that the analysis of the geometry similarity becomes the analysis of a sequence of 2D curves, which is much easier. Such analysis also enables the system to supply intelligent editing as mentioned above. These 2D profile curves can be further decomposed into many elementary elements, including PCA (principle component analysis) and multi-resolution analysis, each of which can be analyzed and edited independently. This is also our main motivation. In CAD field, this is also known as generalized cylinders [9]. Modeling using generalized cylinders is done through construction of the trajectory and the profile curves (cross-section curves), respectively.

Our work is also closely related to skeleton extraction methods as it is a crucial step for extracting the trajectory of swept surface. And a severe problem is that it is hard to extract the skeleton for open mesh, as common automatic methods such as medial-axes-based are ill-posed. In [10], they present a novel method for the extraction of mesh skeleton, which is robust and can bring a direct relation between the mesh vertices and the skeleton. We further improve this method to enable it to work on open mesh.

The major contributions of this work are:

- A swept surface representation of local surface patch. It provides the fundamental technique of the whole system. It allows users to edit local surface freely and efficiently and enables better pattern analysis for intelligent editing.
- Approximating mesh surface patch by swept surface. We present a robust way of approximating surface patch by swept surface.
- Analysis of patterns using swept surface. We present a novel way for detecting repetitive patterns along the trajectory.
- A special fine property is that all the operations, including varieties of editing and even geometry transfer, do not need a reparameterization, that is, we can keep the topology of mesh while doing such editing work.

The pipeline of the system is illustrated in Fig. 1. The user first selects a rough region, which is automatically refined. Then, a swept surface representation of this region is built, including decomposition into lower level elements and analysis for repetitive patterns. Then, the user can edit the 3D model by editing 2D curves. She/he can just edit one of the patterns, and all the other similar shapes will change automatically.

The left of the paper is arranged as the following. Section 2 is a description of our swept surface representation of mesh patch. Section 3 is an informal introduction of the user editing procedure of the system, that is, how the user can use our system to edit the mesh patch. Section 4 is the creation algorithm part and explains how the swept surface representation is created. Section 5 is the decomposition step for decomposing 2D curves to lower level elements, which can be edited respectively. Section 6 explains our analysis algorithm for further facilitating the user editing. Section 7 explains when the user finishes editing of the 2D curves and how the original mesh is modified.

2 The swept surface representation

2.1 Swept surface

Roughly speaking, the swept surface representation includes the trajectory curve and the profile curves. We denote profile curves as C_1: $[0,1]\times[0,1]\rightarrow R^2$. For each $u\in[0,1]$, $\chi(v) = C_1(u, v)$ denotes a planar curve. The surface function can then be written as

$$S(u,v) = \xi\ (u) + \tau(u)C_1(u,v) \tag{1}$$

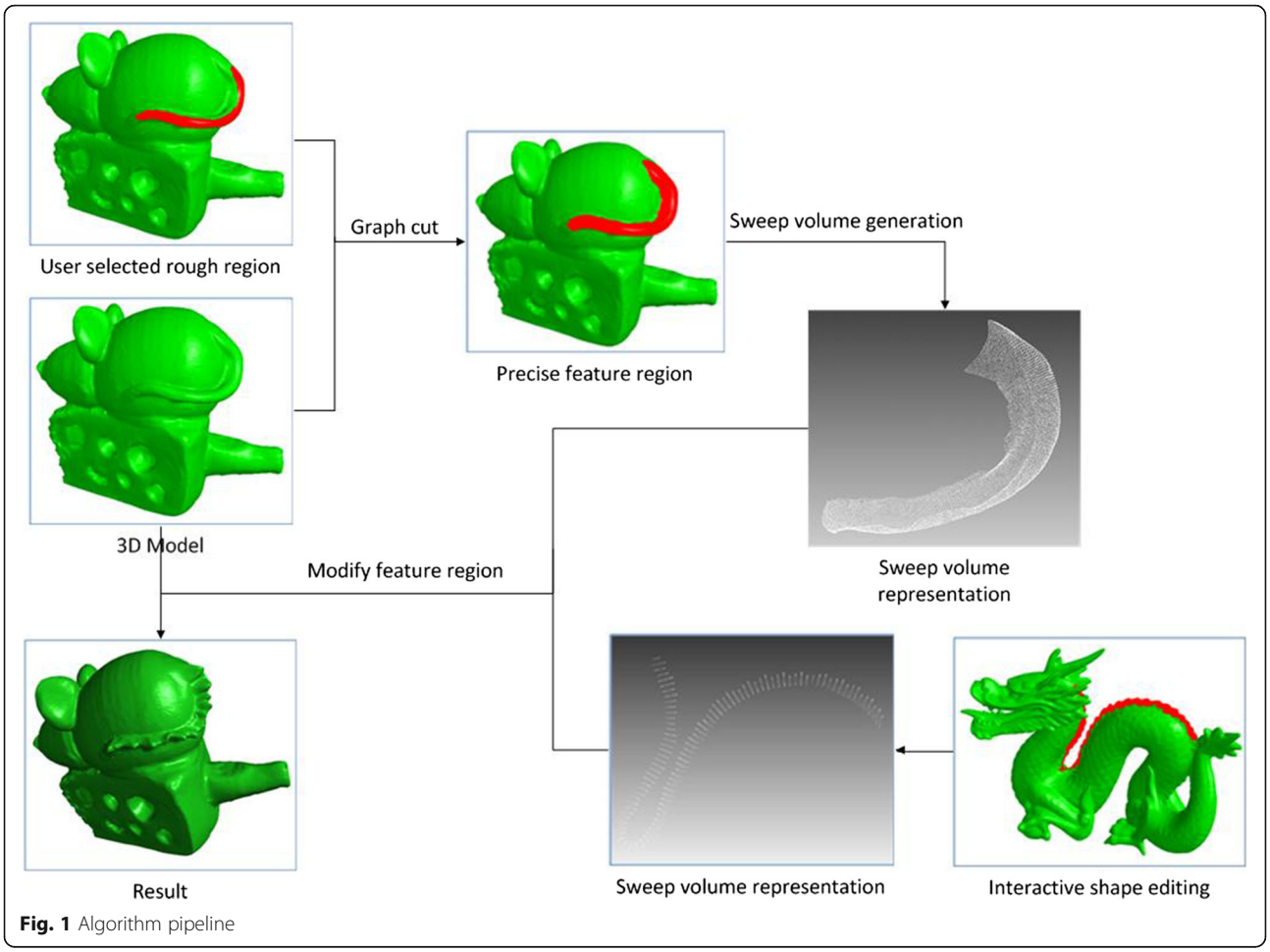

Fig. 1 Algorithm pipeline

where ξ: $[0,1] \rightarrow R^3$ denotes the trajectory and $\tau \in M_3 \times_2$ rotates the planar curve. ξ together with τ map the planar curves into 3D space. We denote the local Frenet frame by ($T(u)$, $N(u)$, and $B(u)$), where $T(u)$, $N(u)$, and $B(u)$ are tangent, normal, and binormal unit vectors, respectively, at the trajectory point $\xi(u)$. Then, generally, $\tau(u)$ maps unit vectors along X- and Y-axis to $N(u)$ and $B(u)$, respectively, and can be written as $\tau(u) = [N(u), B(u)] \in M_3 \times_2$. We also call the plane given by point $\xi(u)$ and normal $T(u)$ the cross-section plane.

2.2 Discrete swept surface patch on mesh

To clarify the subsequent statements, we suppose that the mesh has n vertices. We use $\mathbf{V} = \{1, 2 \ldots n\}$ to denote the set of its vertices. $\mathbf{E} = \{(i, j)\}$ is the set of edges, and $\mathbf{V}_E$ is the set of all the points on edges (including vertices). We identify (i, j) and (j, i) for each edge. For each point i, $N_1(i)$ means the one-ring points of i. We use v_i and n_i to denote the original position and normal for vertex i, and v_i' is the new position. Vertices of the selected region are denoted by $\mathbf{V}_S$.

As we focus on using the swept surface to represent a surface patch on a triangular mesh to be edited, the following properties of such representation is required:

- Discreteness. As the patch is to be analyzed and edited, a discrete representation is needed.
- Editing. Editing of swept surface can change the mesh patch while keeping other regions of the mesh.

As discussed above, to approximate the continuous case, the transformation can be represented as:

$$\xi_i, \quad \tau_i, i = 1, 2, \ldots, n \tag{2}$$

where n is the number of sample points on the trajectory. Parameterize the trajectory by length, then the length for each point is denoted by l_i, $i = 1, 2, \ldots, n$.

The corresponding profile curve is denoted as quaternions

$$X_i = \left\{ c_{i,j} | j = 1, 2, \ldots m_i \right\}, \quad i = 1, 2, \ldots, n \tag{3}$$

where m_i is the number of sample points on each curve. Each $c_{i,\,j}$ is a quaternion corresponding to a point denoted as (p, k, l, λ), where $p \in R^2$ is the 2D position of the point in the local frame and $(k, l) \in E$ and $\lambda v_k + (1\text{-}\lambda) v_l$ gives the 3D position of the point. We keep this

information to modify the original mesh when completing editing the 2D curves.

For $\mathbb{C}_{i,j} = (p, k, l, \lambda)$, we use $\mathbb{C}^p_{i,j}$, $\mathbb{C}^k_{i,j}$, $\mathbb{C}^l_{i,j}$, $\mathbb{C}_{i,j}\lambda$ to represent p, k, l, λ, respectively, and use $\mathbb{C}^p_{i,j}$ to denote the projected 3D position $\tau_i\mathbb{C}^p_{i,j} + \xi_i$.

3 User editing

We provide a novel user editing scheme. The main picture of editing is as follows: The user selects a rough strip region of interest (ROI), which is automatically refined to a better region smooth. Here, better mainly means the region should be dissimilar to its nearby regions; see top of Fig. 1 for example. Then, our system automatically represents the region using a swept surface. For the user, it is just a sequence of profile figures. The editing work then becomes editing a sequence of planar curves. Essentially speaking, the editing work is to replace each curve of the sequence by another curve. Of course, the user does not have to edit each curve he wants to modify as it is really a boring work. To simplify the editing work, we design a scheme of editing based on hierarchical curve decomposition and repetitive pattern analysis. Intuitively, each curve is decomposed into several sub-elements, each of which is either a real value or another curve and can be edited respectively, while the modified elements can then assemble back to the whole curve. Note that each sub-curve can still be subdivided into other low-level elements, which can also be edited to change its parent curve. With such decomposition, editing such sequence of planar curves then becomes the editing of the sequence of lower level elements, though the sequence of lower level elements may still be a sequence of simpler planar curves, which is more convenient for editing. The repetitive pattern analysis further simplifies the editing step. Here, repetitive means repetitive sub-sequence up to a scale of the interval length. A detailed description of the decomposition and analysis is given in Sections 5 and 6; here, we just enumerate several common scenarios of the editing work.

- Modify a sub-sequence of curves when the user just wants to change a small portion of the curve sequence. Usually, he does not have to change each of the curves. For example, he may make an affine transformation for one figure and use the same transformation for each of the curves. He may apply specify several transformations for some of them, and the system applies interpolated transformations for other curves automatically. He may also create new curves using similar methods to replace all them.
- Editing the sequence using a single curve. When the curve decomposition includes real values, these values can then be edited as a single real-valued function (see Fig. 2 for example). The profile figure of the cuboid is a sequence of squares. Using PCA analysis, the rotation angle for one of the principle directions is a constant or a straight line of that described by a real value function. We just replace the line by another straight line, and then, the cuboid becomes twisty.
- Editing repetitive templates. When the curves have the character of repetition, our system can further

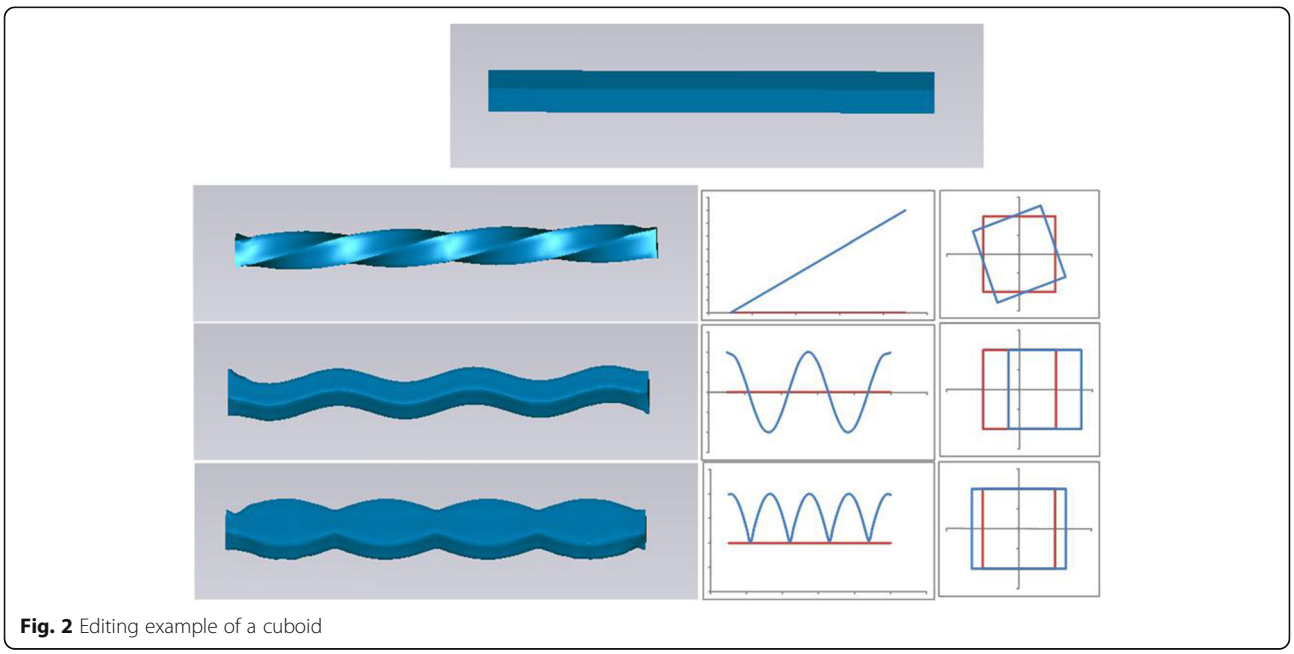

Fig. 2 Editing example of a cuboid

facilitate the editing work by an automatic analysis of such patterns. The user may just select a single pattern and replace it, and then, all the occurrences of the pattern are changed (see Fig. 5).

- Using template library. We also design a template library for replacing existing curves by other pre-designed curves. The user may replace portions of the curve sequence or patterns by pre-designed curves.
- Geometry transfer. Though it is not our main purpose, our system can still easily handle cylinder-like feature transfer. Each of the elements can be transferred respectively. This is done by replacing the sequence of target model by the corresponding element function of the source model (see Fig. 3).

4 Methods—creation of swept surface

Creation of the swept surface representation is the fundamental step of the pipeline. There are three steps:

- *Refinement of selection*. The shape of the swept surface region is crucial to subsequent analysis and editing. However, it is impossible for users to select a precise region as they need. Generally, the most important quality of the boundary is the smoothness, which means the smoothness of not only the curve itself but also the region it crosses, that is, it should not cross the boundary of dissimilar regions. To ensure this criterion, we require the smoothness of both the curve and the normal of the original mesh along the curve.
- *Extraction of trajectory*. The trajectory extraction is the pivotal step for creation of the representation. To create a meaningful swept surface, traditional skeleton is always the ideal choice. However, as the selected region is a cylinder-like open mesh, with surface details or noise, it is a challenge to extract the skeleton from such open mesh, and traditional methods such as Reeb graph, potential field, distance field, or medial surface will not give a robust and reliable solution. So, we use a variety of the method in [10]. However, there are sometimes more than one connected curves for each plane, and only the main curve is preserved.
- *Refinement of trajectory and extraction of profile curves*. The profile curves are extracted by the intersection of the cross-section planes and the mesh.

4.1 Refinement of selection

We use a graph-cut-based approach to refine the selection. Given a mesh patch of triangles, our goal is to get a refined patch with the above properties. The mesh is converted to a weighted undirected graph by taking every face of the mesh as a graph vertex, every edge shared by two faces as a graph edge, and the graph is represented as $G = (V, E)$. Denote the initial selected region as V_{init}. Before defining the edge weights, we first define the distance between neighboring triangles. Besides geometry distance, the distance definition must reflect the variability of normals, as it is the key factor for distinguishing mesh regions and is commonly used in mesh segmentation methods. To integrate the above considerations, we define the distance for neighboring triangles as follows:

$$\text{distance} = \sqrt{\omega_1 d^2_{\text{geometry}} + \omega_2 d^2_{\text{normal}}} \tag{4}$$

where d_{geometry} and d_{normal} denote the distance for geometry and normal, respectively, and ω_1 and ω_2 are the two importance factors for the two distances. The geometry distance d_{geometry} is defined as follows: bend the two triangles along their common edge to let them on the same plane, and d_{geometry} is defined as their barycenter distance on the plane. And their normal distance is defined as $d_{\text{normal}} = |n_1 - n_2|$, where n_1 and n_2 are the normals of the two triangles. And the edge weight for the two neighboring faces is defined as $w = e^{-\text{distance}}$. The next is to define the initial set of selected region V_{select} and unselected region V_{unselect} as follows:

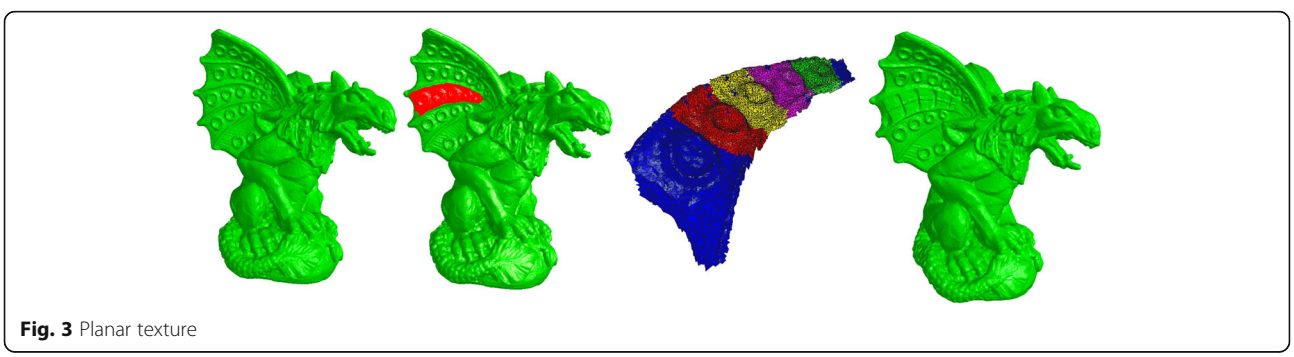

Fig. 3 Planar texture

$$V_{\text{select}} = \left\{ \begin{matrix} v \in V_{\text{init}} | v \text{ has a certain distance} \\ \text{from the boundary of } V_{\text{init}} \end{matrix} \right\} \tag{5}$$

$$V_{\text{unselect}} = \left\{ \begin{matrix} v \in V - V_{\text{init}} | v \text{ has a certain distance} \\ \text{from the boundary of } V_{\text{init}} \end{matrix} \right\} \tag{6}$$

Suppose the final selected and unselected vertices are V'_{select} and V'_{unselect}. We want to optimize the following energy function:

$$E = E_{\text{edge}} + E_{\text{vertices}} \tag{7}$$

where

$$E_{\text{edge}} = \frac{1}{2} \sum_{(u,v)\in E, u\in V'_{\text{select}}, v\in V'_{\text{unselect}}} \omega(u,v) \tag{8}$$

$$E_{\text{vertices}} = \omega_{\text{select}} \left\{ v \in V | v \in V_{\text{select}}, v \notin V'_{\text{select}} \right\} + \omega_{\text{unselect}} \left\{ v \in V | v \in V_{\text{select}}, v \notin V'_{\text{unselect}} \right\} \tag{9}$$

This is essentially a graph-cut problem. To solve it, we first build a directed graph by creating two directed edges with the same weight for each edge of G and create a source vertex connecting to the vertices in V_{select} with weight ω_{select} and a sink vertex connected from the vertices in V_{unselect} with weight ω_{unselect}. Then maximal flow algorithm is used to solve it.

We apply the following rule to further smooth the boundary. For each vertex on the boundary, the spanning angle formed by it and its two neighbors is computed. At each time, the vertex that has the smallest value of angle and of which two neighbors share an edge on the original mesh is removed by connecting its two neighbors directly on the boundary. The process is repeated iteratively.

4.2 Extraction of skeleton

The method in [10] contains two steps: geometric contraction and connectivity surgery. We extend the first step to mesh with boundary and retain the latter unchanged.

The following statements take the mesh patch as a separated mesh. We denote E and $\overline{E}$ as its interior edges set and boundary edges set, respectively.

In [10], they solve the following sparse system for mesh contraction:

$$\| W_L L V' \|^2 + \sum_i W_{H,i}^2 \| v'_i - v_i \|^2 \tag{10}$$

where L is the $n \times n$ curvature flow Laplace operator:

$$L_{ij} = \begin{cases} \omega_{ij} = \cot\alpha_{ij} + \cot\beta_{ij} & \text{if } (i,j) \in E^k \\ \omega_{ij} = 2\cot\alpha_{ij} & \text{if } (i,j) \in \overline{E} \\ \sum_{(i,k)\in E} -\omega_{ik} & \text{if } i = j_0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

For a given point i, the Laplacian constraint can be rewritten as

$$\sum_{k\in N_1(i)} \omega_{ik}(v_k - v_i) = 0 \tag{12}$$

However, this does not work for boundary points. We observe that if the patch is mirrored along its boundary, then the boundary points become inner points. We do not need to really mirror the patch, such as creating new points. Instead, we use a modified version of the Laplacian contraction, and these boundary points contract normally along the inward curvature flow in normal directions as inner points.

Suppose that i is a boundary point and its two boundary edges are $e1$ and $e2$; the normal of the plane P_i determined by $e1$ and $e2$ is $n = \frac{e1 \times e2}{||e1 \times e2||}$. For another point $j \in N_1(i)$, it mirrors by plane P_i which is given by a linear mirror operator σ_i, which has the form

$$\sigma_i(x) = A_i x + b_i \tag{13}$$

where

$$A_i = I - 2nn^T, b_i = (2v_i \cdot n)n \tag{14}$$

Then, the equation can be rewritten as

$$\sum_{(i,k)\in E} \omega_{ik}((v_k - v_i) + (\sigma(v_k) - v_i)) + \sum_{(i,k)\in\overline{E}} \omega_{ik}(v_k - v_i) = 0 \tag{15}$$

This leads to a new curvature flow matrix L', and the constraint function becomes:

$$\| W_L L' V' \|^2 + \sum_i W_{H,i}^2 \| v'_i - v_i \|^2 \tag{16}$$

The output of this step is a skeleton consisting of points and straight lines connecting them, and a byproduct is the map from each vertex on the mesh and its corresponding point on the skeleton. For a point v on the mesh edge, we use $S(v)$ to denote its corresponding point on the skeleton. Note that the skeleton may not just consist of a single curve; we take it as a graph and just choose the path with the maximal length as the initial trajectory (the vertices are treated as sample points). As not every point on the mesh is associated with the

trajectory, we denote the vertices associated with the trajectory by $\mathbf{V}_{s0}$ and use T to denote the trajectory.

4.3 Refinement of trajectory and profile curves

Our goal is to create the swept surface representation, composed of the trajectory and profile curves, as in Section 2.2. We use an iterative step to refine the trajectory and profile curves. In each step, we first use current configuration of $S(v_i)$ and $\mathbf{V}_{s1}$ to find the profile curves and refine the trajectory and then use the new trajectory to renew $S(v)$ and $\mathbf{V}_{s1}$.

- *Profile curve extraction*. The local Frenet frame at each vertex of the trajectory is calculated using discrete geometry, including the associated cross-section planes.

For each trajectory vertex t, its associated profile curve is obtained through the intersection of its cross-section plane and triangles of the mesh. We suppose each intersectant triangle intersects the plane with its two edges. Then, for each cross-section plane, its cross-section curves are 1-manifold which may have boundary points. However, there may be more than one connected component. The reason for this is that some curves which do not belong to the trajectory vertex are also crossed by the plane. To resolve this, we measure the projection of each component to the trajectory. For each component, let V be its vertex set, and we define:

$$\frac{MeanDistance(V) = \sum_{v \in V} d(S(v), t)}{V} \quad (17)$$

Here, *MeanDistance* measures the mean belonging likelihood, the less value the *MeanDistance* has, the more likelihood this component belong to the vertex t. So, we just choose the component has the minimal value of *MeanDistance*.

For each point on edges, the associated Frenet frame plane can still be calculated by a linear interpolation of its two endpoints. And the extraction of the profile curves can be processed in a similar way. To this end, $S(v)$ and $\mathbf{V}_{s1}$ can be updated. For each point $t \in T$, we denote $\gamma(t)$ the associated cross-section curve. Denote the new function is $S'(v)$, then

$$S'(v) = \operatorname{argmin}_{v \in \gamma(t)}(d(t, S(v))) \quad (18)$$

Note that if there is no point t such that $v \in \gamma(t)$, then we denote $S'(v) = \phi$ indicating that v is not controlled by the swept surface. And updated set of $\mathbf{V}_{s1}$ is defined as= $\mathbf{V}_{s1} = \{v | S'(v) \neq \phi\}$.

- *Trajectory refinement*. The trajectory is refined with respect to both the smoothness and the embedding.

  Suppose current positions of the trajectory vertices are $x_1, x_2, \ldots, x_n$, then the new positions $x_1', x_2', \ldots, x_n'$ is given by minimizing the quadratic:

$$F = \omega_1 F_1 + \omega_2 F_2 \quad (19)$$

where F_1 is a discretized tension spline energy similar to [11], which is given by

$$F_1(x_{i_1}, , x_{i_2}, \ldots, , x_{i_n}) = \sum_{k} \left[\begin{array}{l} \| 2x_{i_k} - x_{i_{k-1}} - x_{i_{k+1}} \|^2 \\ + \lambda \| x_{i_{k+1}} - x_{i_k} \|^2 \end{array} \right. \quad (20)$$

And F_2 is the embedding term, given by

$$F_2 = \sum_i \left[\| x_i' - \text{Average}\left(W_\gamma(x_i)\right) \|^2 \right. \quad (21)$$

where *Average*(·) denotes the mean position of a set of points. We further divide $\mathbf{V}$ into two subsets: constrained points $\mathbf{V}_1$ and free points $\mathbf{V}_2$. Constrained points $\mathbf{V}_1$ is defined by the vertices in final set $\mathbf{V}_{S0}$, which is controlled by the swept surface, and free points $\mathbf{V}_2 = \mathbf{V} - \mathbf{V}_1$ should not be controlled. Note that $\mathbf{V}_1 \in \mathbf{V}_S$, but we do not guarantee that every point in $\mathbf{V}_S$ is a constraint by the swept surface.

5 Decomposition of planar curves

To make the editing of sequence of planar curves practical, each planar curve can be decomposed into several independent lower level elements. The main motivation of such decomposition is to create an easier human-computer interaction, and another motivation is to facilitate pattern analysis. Formally, for any curve C, there are several elements $C_1, C_2, \ldots, C_n$, such that $C = F(C_1, C_2, \ldots, C_n)$, where F is the inverse map from lower level elements to the whole curve. Each C_i may be real value or planar curve.

We propose two decomposition schemes. One is based on affine transform, and another is based on multi-resolution curve analysis. Figure 4 is an example.

- *Affine decomposition*. Affine transform is a map of the form

$$F(A, b, x) = Ax + b \quad (22)$$

where A is a non-singular 2×2 matrix and $b \in \mathbb{R}^2$ is the translate component. An affine decomposition is to represent the curve C as A, b, C' such that $F(C') = C$, that is, each vertex of C' is affine-transformed, and then, we

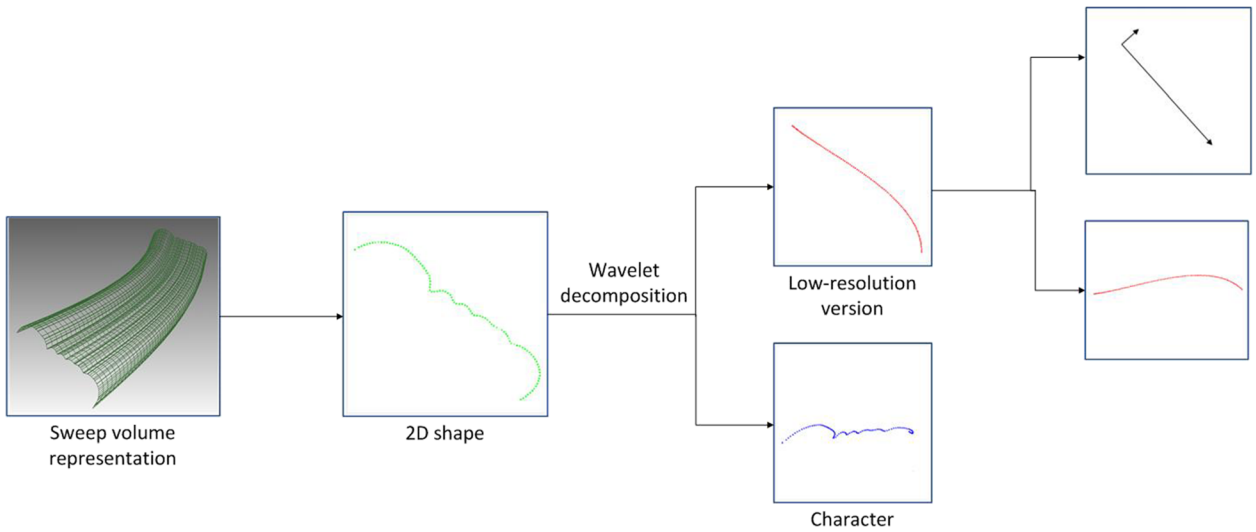

Fig. 4 Decomposition of planar curves

get curve C. Since A, b is collection of real values, the low-level elements are essentially a set of real values and another curve. The user can then just edit A, b, or C', and then, C is changed by the map F. The user can, for example, edit a sequence of one of the real components of A by just editing a single curve.

Specially, we can restrict A to be a rigid body, similar transform, that is, A is controlled by less than four real values. A special case is PCA analysis, where b is the barycenter of C, and A is composed of its two main directions. Figure 2 is a special case of such editing.

- Multi-dimensional decomposition. A curve can be further decomposed into multi-level curves using the B-spline wavelets. It has the form $C = F(C_1, C_2)$, where F is the wavelet reconstruction operator and C_1 and C_2 are the base curve and detailed curve, respectively. Each curve may be edited respectively. This decomposition can be used to handle planar geometry textures as in Fig. 3.

6 Analysis of sequence

Analysis is the key step to provide the intelligence in the system and can magically ease the editing work.

6.1 Analysis representation

The repetitive pattern analysis is performed on a sequence of elements.

For a clearer description, we start from the continuous case:

There is a function $f:[a, b]\rightarrow D$, where D is the object space, depending on the decomposed element. D can be R, the space of real numbers, or the space of planar curves. It can also be customized to other spaces. The task is to find some templates, which is also a function defined on $[a, b]$. Doing some transformations (depending on D, e.g., translation and scaling) using these templates can give f, or say, decompose $[a, b]$ to some disjoint segments, which can be clustered into several groups, each of which can be generated using a specific template. There are several criterions for a good decomposition:

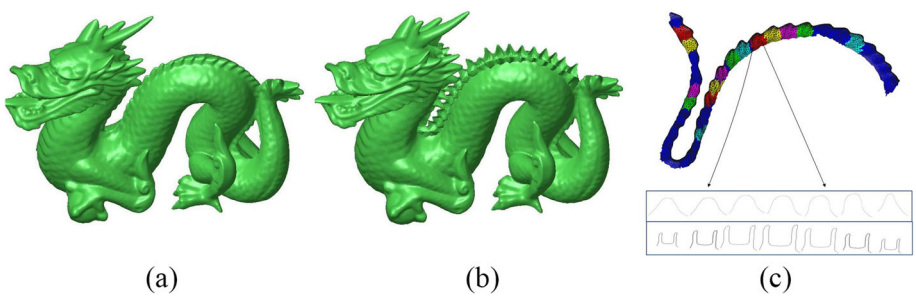

Fig. 5 Illustration for feature editing. **a** is the left image. **b** is the middle image. **c** is the right image

- The templates should be neither too long nor too short.
- There should be as less templates as possible.
- There should be as less segments of $[a, b]$ as possible.

The discrete case is almost the same. Let a be 0 and b length of the trajectory. For each sample point i, we just let $f(l_i)$ the specified associated element, where l_i is the length parameter for the point as in Section 2.2. Other values of f are assigned using linear interpolation. We call the function f the element function.

6.2 Extraction of template

After the extraction of trajectory and profile curves, we aim to find the templates of the element function f along the trajectory. Here, we present a novel pattern analysis process. The algorithm is described as follows:

In order to find templates, sets of similar segments along the trajectory are computed at first. A segment s_i is denoted as $[s_{i0}, s_{i1}]$, where s_{i0} and s_{i1} are the start and end of sample points along the trajectory. Two segments are considered similar if their distance is less than a prescribed threshold. The distance metric of segments can be defined based on the property of function f.

After that, for the sake of the criterions described in Section 6.1, for each set of similar segments $S_i = \{s_{n1}, s_{n2},..., s_{ni}\}$, the maximum coverage of non-overlapping segments of that set is computed, which is defined as follows:

$$\max\left\{\sum_{s_k\in SB}||s_{k1}-s_{k0}|\ |SB\boxtimes S_i, \forall s_m, s_n\in SB, s_m\cap s_n=\phi\right\} \quad (23)$$

The set S_i that has the maximum value of maximum coverage is then found out, and the corresponding set of non-overlapping segments is reported. The next possible template is extracted by repeating the above process after eliminating the reported segments from the trajectory.

In the case of handling planar curves, we use the centroid distance-based Fourier descriptor [12] as the shape descriptor of planar curves to compute the distance metric of segments, because it is suitable for describing planar curves. Each planar curve is then represented by a feature vector:

$$F(c_i) = (a_0, , a_1, ..., , a_{d-1}) \quad (24)$$

where $a_k(k = 0,1,..., d - 1)$ is the coefficient of Fourier descriptor of curve c_i. The difference of two curves is measured by the L_2 distance of the feature vectors. The distance of two segments is defined as the sum of distance of corresponding planar curves. To find similar segments, a k-means clustering algorithm is performed in that d-dimensional feature space to group planar curves into K clusters, named $\{C_1, C_2, ...,C_K\}$. Then, for each two curve c_i and c_j in the same cluster C_k, a segment-growing algorithm is performed by extending each segment along the trajectory until their distance reaches a prescribed threshold. The segment s_i that appears most frequently in a cluster after the growing

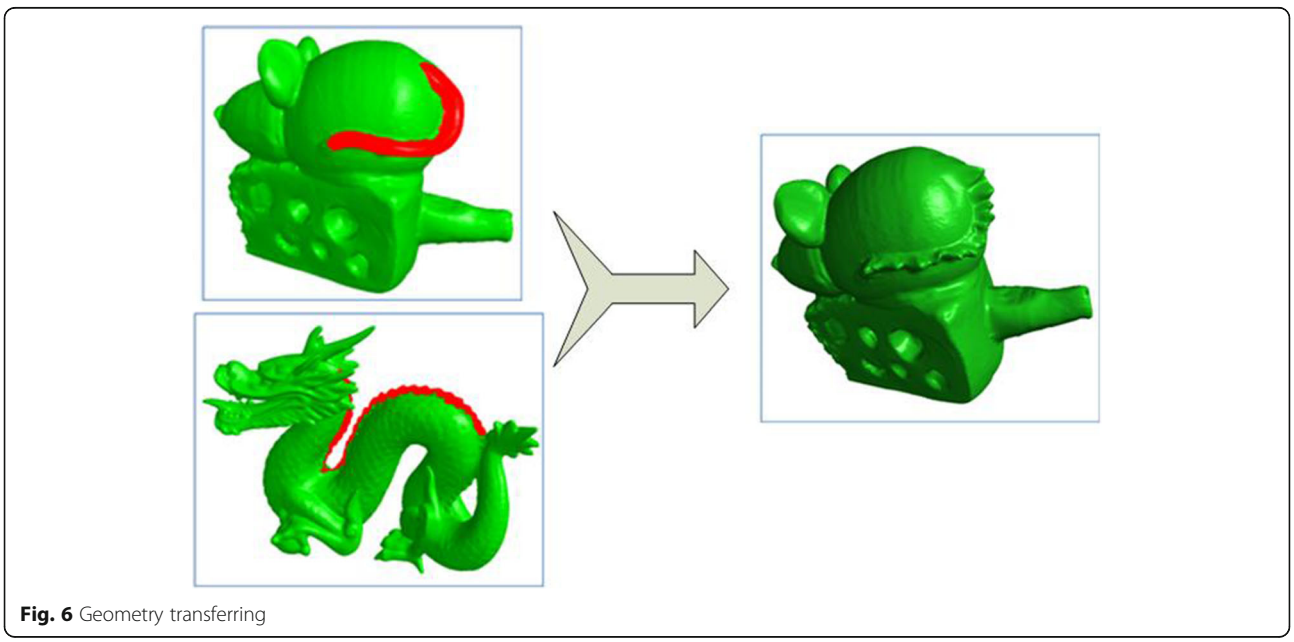

Fig. 6 Geometry transferring

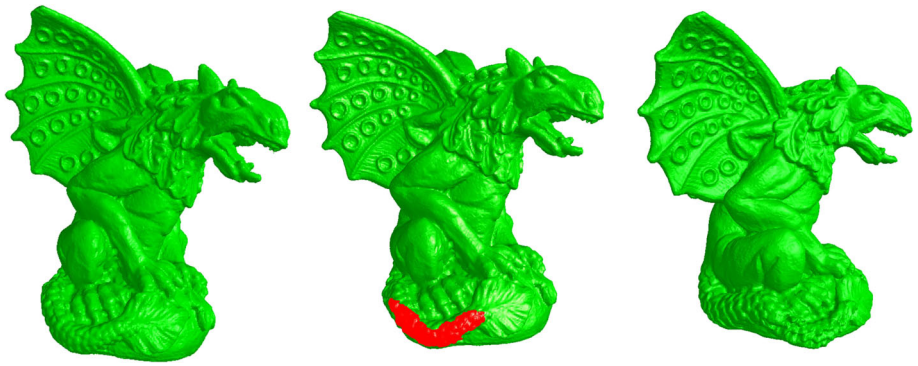

Fig. 7 Detail enhancement

process is extracted as the representative segment of cluster C_k. For each representative segment s extracted as above, a set of similar segments S_i is obtained by searching along the trajectory.

In other cases, the distance metric of segments can be defined based on property of function f. For example, if the object space D is R, the distance is defined as the L_n distance of two real functions.

7 Modification of the mesh

After profile curves being modified, the original mesh should be changed due to changed 2D curves. As mentioned in Section 2.2, regions controlled by the swept surface should be modified corresponding to the swept surface, while other regions should retain their original shapes. To keep the topology of the mesh, we only modify the positions of vertices. So points in $\mathbf{V}_1$ should be constrained by the swept surface, and points in $\mathbf{V}_2$ should be constrained by their original local coordinate, which is, in our implementation, the Laplacian coordinate.

For simplicity, the Laplacian operator is defined with uniform weights

$$\vartheta(v_i) = v_i - \frac{1}{N_1(i)} \sum_{j \in N_1(i)} v_j \tag{25}$$

The new positions of vertices are then obtained by minimizing the following quadratic energy:

$$E_{\text{total}} = \omega_1 E_{\text{swept}} + \omega_2 E_{\text{average}} + \omega_3 E_{\text{lapacian}} \tag{26}$$

$$E_{\text{swept}} = \sum_{i,j} \left| \mathbb{C}^{\lambda}_{i,j} v^{k}_{\mathbb{C}_{i,j}}{}' + \left(1 - \mathbb{C}^{\lambda}_{i,j}\right) v^{l}_{\mathbb{C}_{i,j}}{}' - \mathbb{C}^{P}_{i,j} \right|^2 \tag{27}$$

$$E_{\text{average}} = \sum_{i \in V_1} \left| \vartheta\left(v_i'\right) \right|^2 \tag{28}$$

$$E_{\text{lapacian}} = \sum_{i \in V_2} \left| \vartheta\left(v_i'\right) - \vartheta(v_i) \right|^2 \tag{29}$$

The swept term controls the shape to be consistent with the swept surface. And the Laplacian term in Equation 29 constraints uncontrolled vertices to their original shape. However, the above two terms cannot determine a smooth shape (and sometimes there are infinitely many solutions). So, we use an average term to further control the smoothness of the controlled region.

8 Results and discussion

We have implemented our system and successfully tested it on a variety of models. Figure 5 shows an illustration for feature editing. The left image is the original model, and the middle one is the edited model with the template of the dragon's backbone. Figure 6 shows a funny example of geometry transferring for jokes. In this example, the geometric texture of dragon's backbone is transferred to the model of a mouse tail. Figure 7 demonstrates detail enhancement for models. The region marked in red enhances the geometric texture in detail.

9 Conclusions

In this paper, we proposed a novel framework for mesh editing using swept surface. The user just selects a cylinder-like patch and then can edit it as 2D figures. A swept surface approximation is automatically done, including extraction of the trajectory and the corresponding deforming 2D curves. A novel pattern analysis is done, so the user can change a set of elements by just editing one of them. Demonstrations show the novelty and efficiency of this framework.

The swept surface analysis can be extended to multi-trajectories, that is, the trajectory is not restricted to a single line. As for the future work, we intend to implement a multi-grid linear solver, which can significantly improve the algorithm performance.

Acknowledgements
The authors thank the editor and reviewers.

Funding
This work was supported by the Natural Science Foundation of China (61602033), the Social Science Fund of Beijing (16YTC027), the Science and Technology Plan Project of Beijing (Z171100001217009), and the Fundamental Research Funds for the Central Universities (FRF-TP-15-027A1).

Availability of data and materials
We can provide the data.

Authors' contributions
XW did the main work of the work. JQ did the experiments.

Ethics approval and consent to participate
We approved.

Authors' information
Xiaohui Wang, female, has a PhD degree, and is from the School of Mechanical Engineering, University of Science and Technology Beijing. Her main research interests are computer vision, pattern recognition, affective computing, and interdisciplinary research across computer science and art design. Jingyan Qin, female, is a professor, has a PhD degree, and is from the School of Mechanical Engineering, University of Science and Technology Beijing. Her main research interests are interaction design, information design, and information visualization on big data.

Consent for publication
We agree.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 26 June 2017 Accepted: 16 August 2017
Published online: 24 August 2017

References

1. M. Lounsbery, T.D. DeRose, J. Warren, Multiresolution analysis for surfaces of arbitrary topological type. ACM Transaction on Graphics **16**(1), 34–73 (1997)
2. MAYA. Autodesk. https://www.autodesk.com/products/maya.
3. [3DS] MAX. Autodesk https://www.autodesk.com/products/3ds-max/.
4. T. Igarashi, S. Matsuoka, H. Tanaka, *Teddy: a sketching interface for 3D freeform design* (ACM SIGGRAPH, ACM, New York, 2007)
5. A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, *FiberMesh: designing freeform surfaces with 3D curves* (ACM SIGGRAPH, ACM, New York, 2007)
6. R. Gal, D. Cohen-Or, Salient geometric features for partial shape matching and similarity. ACM Transaction on Graphics **25**(1), 130–150 (2006)
7. M. Pauly, N.J. Mitra, J. Wallner, H. Pottmann, L.J. Guibas, Discovering structural regularity in 3D geometry. ACM Transaction on Graphics **27**(3), 1–11 (2008)
8. Manfredo P. Do Carmo. Differential geometry of curves and surfaces. Prentice Hall, 1976
9. A.S. Aguado, E. Montiel, E. Zaluska, Modeling generalized cylinders via Fourier morphing. ACM Transaction on Graphics **18**(4), 293–315 (1999)
10. A. Oscar Kin-Chung, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction. ACM Transaction on Graphics **27**(3), 1–10 (2008)
11. Y.-k. Lai, Q.-y. Zhou, S.-m. Hu, J. Wallner, H. Pottmann, Robust feature classification and editing. IEEE Trans. Vis. Comput. Graph. **13**(1), 34–45 (2007)
12. D. Zhang, L. Guojun, in *Proc. of 5th Asian Conference on Computer Vision (ACCV)*. A comparative study of Fourier descriptors for shape representation and retrieval, vol 2 (2002), pp. 652–657

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com