SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2019, Vol. 62 052105:1–052105:16
https://doi.org/10.1007/s11432-018-9694-8

Sketch simplification guided by complex agglomeration

Yue LIU[1], Xuemei LI[2*], Pengbo BO[3] & Xifeng GAO[4]

[1]*School of Computer Science and Technology, Shandong University, Jinan* 250101*, China;*
[2]*School of Software, Shandong University, Jinan* 250101*, China;*
[3]*School of Computer Science and Technology, Harbin Institute of Technology, Weihai* 264209*, China;*
[4]*Courant Institute of Mathematical Sciences, New York University, New York* NY 10012*, USA*

Received 13 August 2018/Accepted 30 September 2018/Published online 2 April 2019

Abstract We propose a novel method for vector sketch simplification based on the simplification of the geometric structure that is extracted from the input vector graph, which can be referred to as a base complex. Unlike the sets of strokes, which are treated in the existing approaches, a base complex is considered to be a collection of various geometric primitives. Guided by the shape similarity metrics that are defined for the base complex, an agglomeration procedure is proposed to simplify the base complex by iteratively merging a pair of geometric primitives that exhibit the minimum cost into a new one. This simplified base complex is finally converted into a simplified vector graph. Our algorithm is computationally efficient and is able to retain a large amount of useful shape information from the original vector graph, thereby achieving a tradeoff between efficiency and geometric fidelity. Furthermore, the level of simplification of the input vector graph can be easily controlled using a single threshold in our method. We make comparisons with some existing methods using the datasets that have been provided in the corresponding studies as well as using different styles of sketches drawn by artists. Thus, our experiments demonstrate the computational efficiency of our method and its capability for producing the desirable results.

Keywords sketch, simplification, base complex, iterative merging

Citation Liu Y, Li X M, Bo P B, et al. Sketch simplification guided by complex agglomeration. Sci China Inf Sci, 2019, 62(5): 052105, https://doi.org/10.1007/s11432-018-9694-8

1 Introduction

Sketches are drawings comprising short strokes and that do not contain considerable amount of detail. This style of drawing is extensively used in different fields of application such as graphic design, artistic logo or format design, computer-aided design in engineering, and two-dimensional (2D) animation. Sketches are usually quickly drawn on a paper by artists who express their inspirations or an original design concept; thus, a sketch often contains a large number of intersecting strokes in disorder. Sketches that are to be used for high-level editing, such as high-quality delineating for animation and sketch-based three-dimensional (3D) modeling, need to be graphically cleaned up. In practice, to obtain clean and simple line drawings from sketches, one has to often manually draw lines by tracing the sketches, which can be obtained at a dramatic time and labor cost. Thus, an automatic and robust sketch simplification technique is urgently required.

A sketch that is drawn using electronic painting tools, such as tablets and stylus pens, is a vector graph. Vector graphs have an extensive area of application because of their prominent advantages such

* Corresponding author (email: xmli@sdu.edu.cn)

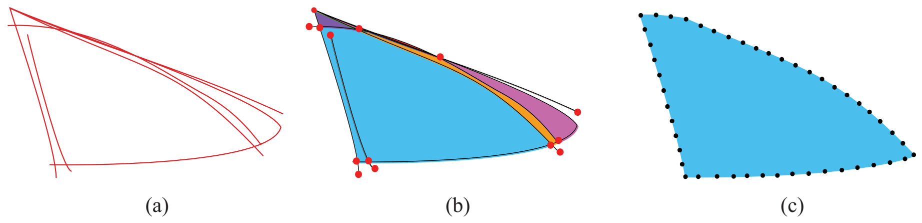

Figure 1 (Color online) (a) Drawing a triangle with seven strokes; (b) the base complex of (a) comprises black curves, red nodes, and differently colored patches; (c) the largest blue patch in (b) is a polygon surrounded by dense boundary points.

as resolution independency, storage saving, and easy editing. The objective of sketch simplification is to reduce the complexity of rough sketches and to produce relatively clean line drawings while maintaining the fidelity of the simplified sketches with respect to the original drawings.

The techniques that have been previously used simplify the sketches by grouping and merging the existing strokes based on geometric information, such as distances [1] or additional semantic information [2], and various parameters often have to be manually tuned to achieve satisfying results. In contrast to these prevailing methods, we propose a vector sketch simplification method based on a geometric structure that is extracted from an input vector graph, i.e., a base complex, which can be described as a collection of graphic primitives including nodes, curves, and patches, as depicted in Figure 1(b). Specifically, we convert a vector sketch into a base complex and resolve its simplification using an agglomeration procedure that iteratively merges two neighboring graphic primitives that are guided by shape similarities, which are defined for various graphic primitive pairs.

The used agglomeration process is inspired by an iterative contraction approach that is commonly used in 3D mesh simplification algorithms. The simplification of the base complex can be gradually accomplished by performing reductions in the number of graphic primitives and slight modifications in the shape of the base complex. The requirement of shape similarity ensures a minimum change in shape after each step of the agglomeration, which avoids undesired deformations and ensures consistency with the original shape. Our method performs efficiently and only a single threshold is used to control the level of simplification with respect to the actual requirements. The simplified base complex is finally converted into a vector graph as the simplified output.

We can summarize our contributions as follows:

- As an intermediate representation of the input vector graph, a geometric structure called a base complex is created, which inherits the crucial geometric information of a vector graph;
- Based on the base complex, the iterative decimation algorithm of 3D mesh simplification is adapted to 2D sketch simplification, which is guided by the shape similarities of graphic primitives in a base complex;
- In contrast to the existing approaches, which depend on many parameters, the simplification of a base complex can be easily controlled using a single parameter.

2 Related work

In this section, we review the related studies on vector graph processing with a focus on the problem that we have attempted to address, which can be described as sketch simplification. Besides, because our concept of sketch simplification is inspired by mesh surface simplification, we also discuss published studies on iterative algorithms that have been used for mesh simplification.

Vector graph simplification. To simplify an input vector sketch comprising a large number of strokes, some existing methods group the strokes, replacing each group with a smooth parametric curve. Further, the stroke grouping is usually guided by an analysis of the geometric properties of the strokes such as their degree of proximity, continuity, or parallelism. For example, the method presented in [1] considers stroke proximity and continuity and specifies a single distance-based parameter to control sketch simplification. Similarly, the method presented in [3] utilizes a dynamic threshold to control the stroke simplification process. In [4], the parts of strokes to be simplified are determined using two types of stroke density metrics. The stroke density is controlled by the omission of curves; however, the result of simplification depends on the sequence of painting. In [5], the input sketches are partitioned into differently sized boxes to accomplish local ordering; subsequently, piecewise B-spline curves are fitted to the globally ordered data points.

To obtain better stroke groupings, some studies use the regions of a sketch as guidance for grouping. For example, a novel high-level semantic analysis of strokes, which is presented in [2], succeeds in obtaining sensible stroke groups when assisted by semantic information of the regions. However, the semantic regions of a sketch are not easy to identify. Even though this approach involves a mutual refinement process of stroke and region interpretations, sketch simplification can still be realized by stroke grouping.

Machine learning approaches have also been investigated for sketch simplification. The study presented in [6] uses a learning-based method to classify strokes; after defining three stroke features, a trained neural network is used to determine whether the pairs of strokes belong to a same group. A disadvantage of this method is that it is designed to refine the painting style of a specific designer. In [7], a supporting vector machine (SVM) is used to train stroke pairing; further, unlike in [6], the stroke pairs are divided into three categories. This method does not require manual annotations; however, a large amount of input sketches and corresponding clean line drawings are required to train the SVM. The study in [8] presents a method of sketch simplification using a convolutional neural network (CNN). However, this method is designed to address raster images; after a CNN has been used to perform a series of convolution operations on a raster image, the results have to be processed into vector graphs using other vectorization tools.

Except for [8], all the cited methods have been designed for vector graphs. To address the input sketches in raster forms, the method described in [9] uses both the Garbor and the Kalman filters to extract clean lines from the sketch. The method described in [10] establishes over-segmented clean line drawings from the input sketch with high fidelity and further merges successive lines if they can be approximated well using a single curve. This global optimization process presents a tradeoff between fidelity and simplicity. The approach in [11] uses the Delauney triangulation of a point set to identify the regions of a graph and to extract its skeleton for further processing, which shares similarities with [12, 13].

Sketch simplification plays an important role in sketch-based 3D design and modeling systems, which often starts from 2D curves. Sketch-based modeling systems help those with sufficient drawing skills to easily manipulate 3D shapes without having to understand the underlying details of modeling. Such systems provide a lightweight, gesture-based interface to approximate 3D polyhedral modeling, in which sketch simplification can be considered to be a pre-processing phase. The simplification processing that is applied in the modeling systems is simple, and a cleaned line drawling is usually obtained by interactive operations. More details on these procedures are available in [14–20].

Iterative mesh simplification. The iterative mesh simplification method is among the most popular within the various categories of existing mesh simplification algorithms. Its main procedure involves repeated application of local operators that eliminate one mesh element in a single step (a vertex, an edge, or a face), thereby iteratively simplifying a mesh. This simplifying operation is associated with an error function that measures the amount of error introduced by a simplification operation.

For example, vertex decimation which was proposed in [21], deletes a single vertex and re-tessellates the resulting hole. The method in [22] is a simpler variant of mesh optimization, in which the edges are collapsed to generate levels of details and present a progressive mesh representation. Various similar methods relying on different approximation errors have been proposed such as [23–27]. This kind of mesh simplification method exhibits the benefit that it can generate a tradeoff among computational efficiency, geometric fidelity, and topological generality.

The basic concept behind our method is similar to that of iterative mesh simplification [24]. However, the problem of sketch simplification is quite different than the problem of mesh simplification, which indicates that simplification operators and cost functions that are suitable for sketch simplification have to be reinvented. To solve this problem, we introduce the base complex for sketch simplification.

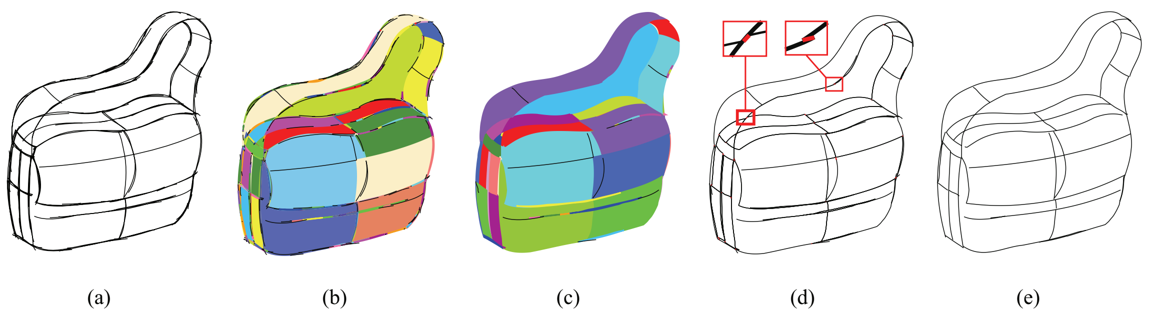

Figure 2 (Color online) (a) Input vector sketch; (b) base complex of the input vector graph containing nodes, curves, and patches; (c) simplified result obtained after merging pairs of primitives under the constraint of shape similarity measurements; (d) two types of unsmoothing cases are marked in red boxes; one is the curve confusion at an intersection, the other is a zigzag shape; (e) the obtained simplified graph.

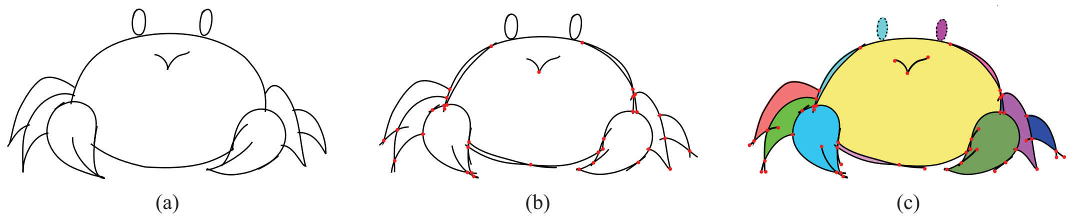

Figure 3 (Color online) (a) Simple input vector graph; (b) intersections between various strokes are shown as red dots; (c) base complex extracted from (a), where the patches are set in different colors; the black solid lines show the open curves; the black dotted lines show the closed curves, and nodes are represented as red dots.

3 Method

The overview of the steps used in our method is depicted in Figure 2. An input vector sketch (Figure 2(a)) comprises a large set of polynomial curves such as Bézier curves. Unlike the current stroke-grouping methods, our method explores the solution space of sketch simplification by considering a vector graph as a 2D mesh (Figure 2(b)). By extending the concept of a base complex from the surface [28] and volumetric meshes [29] to the 2D space (Figure 2(b)), valid primitive pairs that are identified from the base complex are sorted to minimize the distortion of sketch simplification. This procedure provides a good balance between the simplicity of the sketch and the fidelity with respect to the original shape (Figure 2(c)).

3.1 Base complex

Definition. A base complex is a collection of primitives, i.e., nodes, curves, and patches. A node is the intersection between two strokes, or the endpoint of a stroke. A curve comprises a set of line segments. A curve bounded by two nodes is referred to as an open curve, and a closed curve is topologically a circle. A patch is a closed region bounded by curves. An example of a base complex is depicted in Figure 3(c).

Extraction. Given a sketch that contains strokes (Figure 3(a)) (e.g., Bézier curves and straight line segments), we first compute all the stroke intersections, as depicted in Figure 3(b). The strokes are further separated into pieces by nodes, and each piece is represented using a curve in the base complex.

To find patches, we construct a graph $G = (V, E)$, where V denotes a set of vertices and E denotes a set of edges [30]. The vertex set V corresponds to a set of nodes in a base complex. An edge in E corresponds to a curve between two nodes. A patch can be defined as a closed path in G, which does not contain any other closed paths. A patch can be represented by a sequence of ordered vertices and each edge in E is adjacent to a maximum of two patches. The base complex is constructed by finding all patches in G, as depicted in Figure 3(c).

3.2 Base complex simplification

The initially obtained base complex is intricate, and we simplify it by iteratively applying a collapse operator, which acts on primitive pairs by merging them into a single primitive. Three classes of primitive pairs can be found in the base complex, namely the curve-curve, the curve-patch, and the patch-patch pairs. The process of merging primitive pairs eliminates at least one edge at every step and guarantees a monotonic decrease of graph complexity. By quantifying the shape variation in the process of simplification, the collapse operation attempts to minimize it in each step of the collapse procedure.

The primitive pair collapse operator is inspired by the contraction operation in iterative mesh simplification algorithms [24]. However, in mesh simplification, only vertex pairs are considered for contraction operation, whereas three different classes of primitive pairs are considered for the simplification of a base complex. Furthermore, the cost function in mesh simplification is a quadric error metric for the mesh, while our method relies on the measurement of shape similarity between various geometric primitives.

Our simplification pipeline can be summarized as follows:

(1) Identify all the valid pairs of primitives in the base complex;

(2) Rank all the pairs in an ascending order of the cost of shape variation;

(3) Remove the top-ranked pair if the stopping criteria is unsatisfied; otherwise, exit;

(4) Re-evaluate the cost for the neighboring elements that are affected by the removal operation and reorganize the array of pairs;

(5) Goto step (1).

The stopping criteria is a user-specified shape similarity value ϵ. Suppose we have a similarity measurement function for a primitive pair O_i and O_j, denoted by $d(O_i, O_j)$. We construct a new primitive O_{ij_new} by merging the primitive pair O_i, O_j. The cost w of the shape variation while collapsing O_i, O_j is defined by $w = \min(d_1, d_2)$, where $d_1 = d(O_i, O_{ij\_\mathrm{new}})$ and $d_2 = d(O_j, O_{ij\_\mathrm{new}})$ are the similarities between O_{ij_new} and the original primitive pair O_i and O_j. The shape variation is defined by measuring the shape similarity of the newly introduced primitive and the original primitive pair before the collapse operation. In the following text, we explain the method that is used to compute the shape similarity as well as the procedure of identification and removal of valid primitive pairs in detail.

3.2.1 *Shape similarity measurement*

A considerable amount of work has been conducted on similarity measurement of 2D objects in various research areas such as planar object recognition, shape matching and retrieval. One of the central concerns in similarity measurement is the shape feature representation that is required for performing shape similarity measurement. Among these available similarity measurement methods, we choose to use the Fourier transform (FT) method to analyze the shape of a graphic primitive. The shape feature of a primitive obtained by the FT method is called Fourier descriptor (FD), and the used shape similarity d is the Euclidean distance of FDs for the two primitives. The reasons for using an FT are given as follows.

(1) The advantage of using an FD is the simplicity of its application in a computational environment, which makes the process of finding shape descriptors using computer programs an easy task. Besides, it is based on a well-developed mathematical theory and supported by theoretical and experimental studies that confirm the utility of FDs as feature representations of a planar shape.

(2) The shape representation based on the FT is invariant to translation, scaling, and rotation, and it is also compact and easy to derive.

(3) Despite the FT being sensitive to boundary noise, with the representations behaving poorly in the presence of noise or distortion of the boundary, the boundaries of the vector geometric primitives are almost smooth, and noise-generated impact on the performance of the method can be neglected.

The contour-based FT method is a powerful tool for conducting shape analysis. It transforms a shape boundary to its corresponding shape feature, as described in detail in [31, 32]. In the FT method, the contour of a 2D object is considered to be a closed curve comprising successive boundary points with the coordinates (x_t, y_t), $0 \leqslant t < N$, where N is the total number of points on the boundary. An example of this coordinate chain is depicted in Figure 4(a). For an object with a closed contour, a periodic function

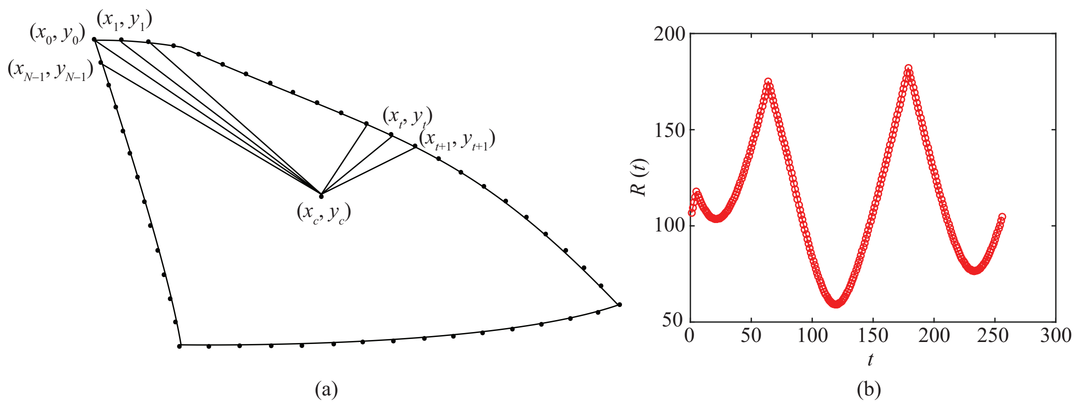

Figure 4 (Color online) (a) Contour of a triangle and its central point; (b) periodic function $R(t)$ of the triangle created using 256 sampling points on the contour in (a). The horizontal axis indicates the sampling points, whereas the vertical axis indicates the distance from the sampling point to the central point.

$u(t)$ that considers the boundary points as the input variable can be defined to describe the object. This periodic function is further expanded in a Fourier series $u^*(t)$. The normalized Fourier transformed coefficients of the expansion are used as shape features and are referred to as the FD. This object is uniquely represented by its Fourier coefficients.

In general, a shape signature $u(t)$ is a one-dimensional (1D) function which uniquely describes a 2D object shape. Among the various shape signatures that are used, the centroid distance is chosen as the shape contour signature in our work because of its good performance, as described in [33, 34]. The centroid distance function $R(t)$ is expressed as belows, which is depicted in Figure 4(a).

$$R(t) = \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2}, \tag{1}$$

where $x_c = \frac{1}{N}\sum_{t=0}^{N-1} x(t)$ and $y_c = \frac{1}{N}\sum_{t=0}^{N-1} y(t)$. The function $R(t)$ can be written as a Fourier series in the form of an exponential summation.

$$R(t) = \sum_{n=0}^{N-1} a_n \exp\left(\frac{\mathrm{j}2\pi nt}{N}\right), \quad 0 \leqslant t \leqslant N-1, \tag{2}$$

where $a_n = \frac{1}{N}\sum_{t=0}^{N-1} R(t)\exp(\frac{-\mathrm{j}2\pi nt}{N})$, $n = 0, 1, \ldots, N-1$ are the Fourier coefficients. According to [31], the normalized shape feature vector can be expressed as

$$\mathrm{FD} = [f_1, f_2, f_i, \ldots, f_{N/2}], \tag{3}$$

where $f_i = \frac{|a_i|}{|a_0|}$. a_i denotes the i-th component of Fourier coefficients. The deviation of the primitive α and β is the Euclidean distance between the feature vectors $\{f_i^\alpha\}$ and $\{f_i^\beta\}$, which is calculated as follows:

$$d(\alpha, \beta) = \|\mathrm{FD}_\alpha - \mathrm{FD}_\beta\| = \sqrt{\sum_{i=1}^{N/2} \left(f_i^\alpha - f_i^\beta\right)^2}. \tag{4}$$

A smaller d indicates a higher similarity or a lower difference between α and β, and vice versa.

3.2.2 *Removal of the primitive pairs*

A sketch contains only curves; however, its base complex contains nodes, curves, and patches. To eliminate curves in a sketch, we design collapse operations on the base complex to merge and remove valid primitive pairs. These collapse operations can be classified as the operations of a curve-curve (CC) pair, a curve-patch (CP) pair, or a patch-patch (PP) pair type.

Valid pair identification. While any combination of primitives that are either curves or patches can be considered to be a pair, a pairing is valid only when the two primitives are immediate neighbors

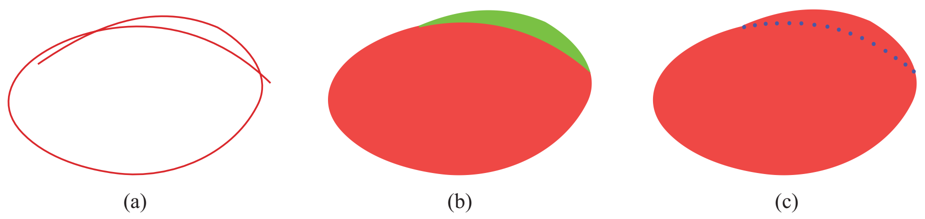

Figure 5 (Color online) (a) Input vector graph; (b) the red patch has a common edge with the green patch, so that they can be paired; (c) a new patch is obtained after the patch-patch merging. The public edge of the two patches is marked by a blue dashed line.

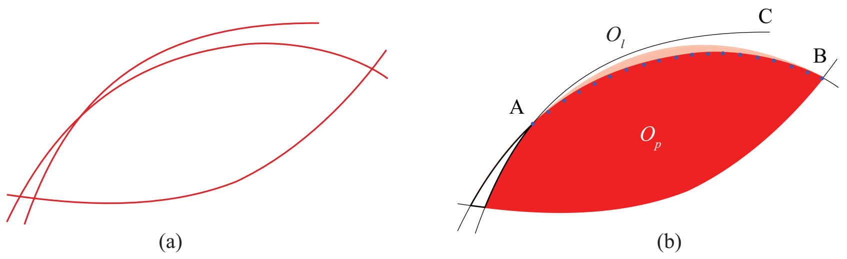

Figure 6 (Color online) (a) Input vector graph; (b) the point A is the common point of the curve O_l and the patch O_p. The blue dashed curve is the affected edge E_{AB} on O_p, and E_{AB} is expanded outwards because of O_l.

to each other. Two patches are neighboring when they share at least one curve, whereas a patch and a curve are neighboring if they share at least one node. However, it is less straightforward to define the neighborhood of a curve, and we use both connectivity and geometric distance to identify its neighbors. If the minimum distance $D(l,m) = \min_{i,j} \|p_i^l - p_j^m\|$ between the curves l and m is less than $d_\varepsilon = \alpha D$ and the angle of the tangent vectors at the two points connected by the minimum distance is smaller than d_θ, the two curves can be considered to be neighbors. The diagonal of the bounding box of a sketch is denoted by D.

Valid pair removal. The removal of a PP is to form a new patch by eliminating the shared curves (Figure 5). The removal of a CC and a CP is not as simple; therefore, we will explain it in detail.

The removal of a CP involves the deletion of the curve c of the pairing, which is followed by the deformation of the original patch o. The curve serves as a modifier for the shape of the patch, essentially causing a slight deformation to an adjacent edge e belonging to the patch, as demonstrated in Figure 6. The deformed edge can be computed by fitting a cubic Bézier curve to the samples of c and o. Similarly, the removal of a CC is performed by fitting a cubic Bézier curve to the two curves of the pairing with a least-square method, as depicted in Figure 7.

3.3 Conversion from a base complex to a vector graph

The graphic primitives in a base complex can be separated into descriptive and non-descriptive elements. A descriptive graphic primitive has a perceptually meaningful existence by depicting what the artist wants to express, whereas a non-descriptive graphic primitive expresses little information related to the artist's interpretation. Because of simplification, the amount of non-descriptive graphic primitives in the simplified base complex decreases while that of the descriptive graphic primitives increases. More specifically, the redundant primitives are removed and the remaining graphic primitives are sufficiently compact to describe what the artist wanted to depict, as illustrated in Figure 8. In practical applications, vector graphics are often represented in the form of line drawings; therefore, we use the curves and edges of all the patches of the simplified base complex to create a simplified vector graph. Because both a curve and a patch comprise discrete sampling points, each curve or edge of a patch is fitted onto a piecewise cubic Bézier curve [35] to obtain a practical form.

While converting a base complex into a vector graph, two types of unsmoothing cases have to be addressed, curve confusion and zigzag shape, as depicted in Figure 2(d). Note that the representation of

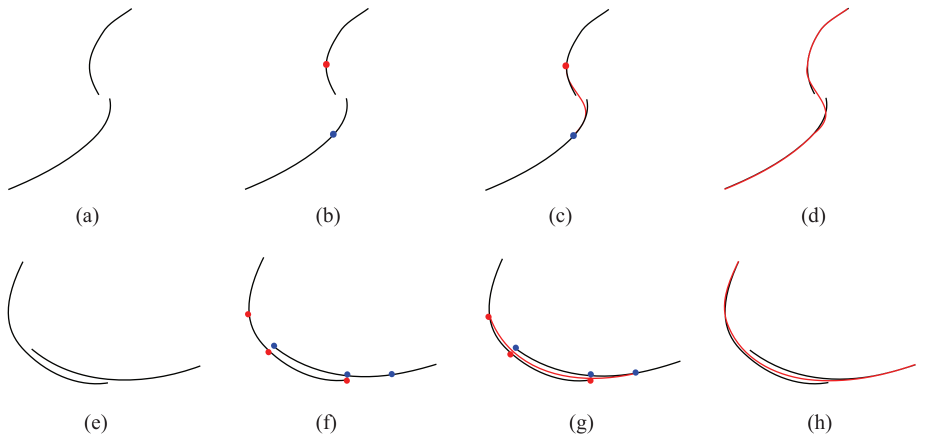

Figure 7 (Color online) Two representative cases of a curve-curve pair merging, where (a)–(d) present the merging process of two broken curves and (e)–(h) present the merging process of two overlapped curves. Both the processes are achieved by fitting the two nearest line segments into a Bézier curve and then connecting the remaining line segments to create a new curve.

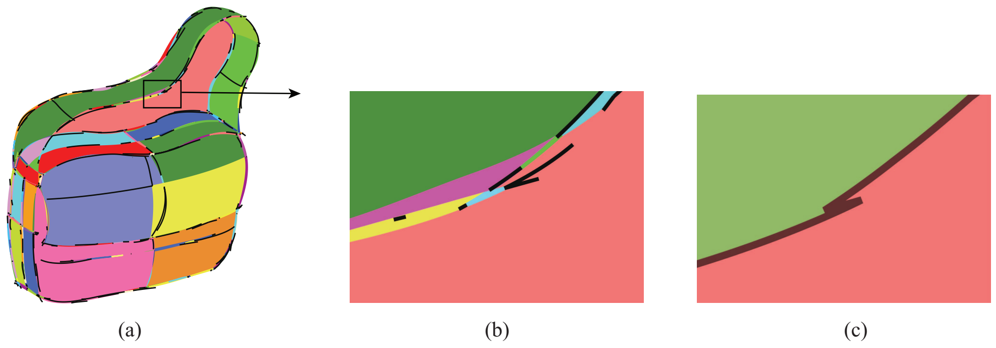

Figure 8 (Color online) The initial base complex (a), one part on detail (b) and its simplified base complex (c).

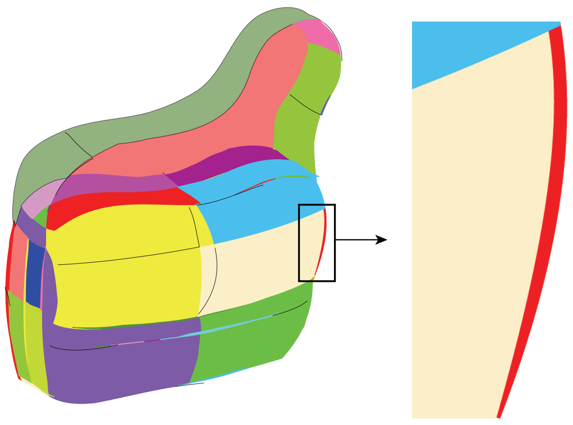

Figure 9 (Color online) Border patch.

some patches using a sequence of ordered vertices is redundant in the results obtained by the merging algorithm because it may contain some nodes that belong to the patches that are deleted during PP merging. To identify and resolve the unsmoothing cases, we apply a smoothing algorithm aided by the redundant nodes and patches. The detailed procedure will be presented in the following.

In addition, if the input graph is rough and exhibits relatively wide borders, which is a common consequence of repeatedly using plenty of strokes to depict the boundary of an area, a slender patch (we call it the border patch) may remain on the border after simplification, as depicted in Figure 9. This occurs because the weight of this border patch is larger than the threshold ε; therefore, it is preserved

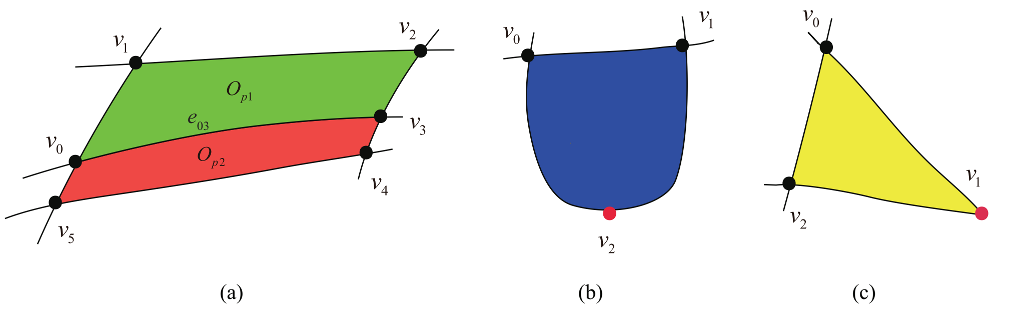

Figure 10 (Color online) (a) Vertices v_0–v_5 are vertices of the first type, each with a degree larger than 2. The edge e_{03} is adjacent to both the patch O_{p1} and O_{p2}, whereas the other edges are only adjacent to one patch. (b) Vertex v_2 is an auxiliary vertex added to the blue patch, and it belongs to the second type of vertices. (c) Vertex v_1 is of the third type.

without being merged into the neighboring patches. Perceptually, a border patch represents one edge of the patch adjacent to it, and it is easy to find because of its small width and two long edges of similar length. However, while converting a base complex into a vector graph, bi-edges may occur. To solve this problem, one of the two edges can easily be eliminated by processing the CC merging operation once more for further simplification.

Smoothing operation. We first create a clear representation of the patch. A patch comprises at least 3 vertices and is expressed as $O_p = \{v_0, v_1, \ldots, v_{n-1}\}$, where v_i are its vertices. Between a pair of vertices v_i and v_j, there exists exactly one edge e_{ij}, and it is considered to be adjacent to a maximum of two patches.

As depicted in Figure 10, these vertices may be separated into three types. The first type of vertices contains those of a degree greater than 2. The second type of vertices include the auxiliary vertices as defined in [30]. Thus, if more than one edge exists between two junctions or if a patch is a loop without any crossing points, an auxiliary point with a degree of 2 is added. The degree of the vertices of the third type is equal to 2; however, they can be found at locations exhibiting sharp turns or corners.

The smoothing algorithm first uses the patch data to locate the short edge e_{ij}, which is defined as

$$(l_{e_{ij}} < d_\varepsilon) \wedge (l_{e_{ij}}/l_{O_p} < \beta), \tag{5}$$

where l_{O_p} is perimeter of patch O_p, $l_{e_{ij}}$ is length of edge e_{ij}, and β is a constant that is set to 0.04. After merging simplification, the base complex may contain large patches with redundant nodes, where some nodes that belonged to small patches before merging procedure will exist on the created large patch.

As depicted in Figure 11, if a short edge e_{ij} exists on the patch O_{p1}, we can obtain information about neighboring edges e_{jh} and e_{ki} on both sides of e_{ij}. We determine the edge e_{ij} to be located in a zigzag shape if the three edges e_{ki}, e_{ij}, and e_{jh} are adjacent to the same patches. Further, the two neighbor edges e_{jh} and e_{ki} can be merged into a new curve by the CC merging operation described in Subsection 3.2.

If the three edges (e_{ki}, e_{ij}, and e_{jh}) are instead adjacent to different patches, as depicted in Figure 11(b), we are faced with curve confusion. In this case, the edges e_{ki}, e_{ij}, and e_{jh} are adjacent to the patches O_{p1} and O_{p4}, O_{p1} and O_{p3} as well as O_{p1} and O_{p2}, respectively. The solution involves finding the other pair of neighboring edges, named e_{mi} and e_{jn}, on the opposite patch O_{p3}. Then the four edges e_{ki}, e_{jh}, e_{mi}, and e_{jn} are paired with each other in six combinations, $e_{ki}e_{jh}$, $e_{ki}e_{mi}$, $e_{ki}e_{jn}$, $e_{jh}e_{mi}$, $e_{jh}e_{jn}$, and $e_{mi}e_{jn}$. Next, the unit tangent vectors of these four edges at the endpoints of edge e_{ij} are calculated and expressed as $\boldsymbol{t}_{ki}$, $\boldsymbol{t}_{jh}$, $\boldsymbol{t}_{mi}$, and $\boldsymbol{t}_{jn}$. A valid combination of the two edges can be merged into one curve if it satisfies the following relation:

$$T = \frac{|\boldsymbol{t}_1 \cdot \boldsymbol{t}_2|}{\|\boldsymbol{t}_1\|\,\|\boldsymbol{t}_2\|} < \gamma, \tag{6}$$

where $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ represent any two of the described unit tangent vectors. In our experiments, the value of γ is set to 0.174.

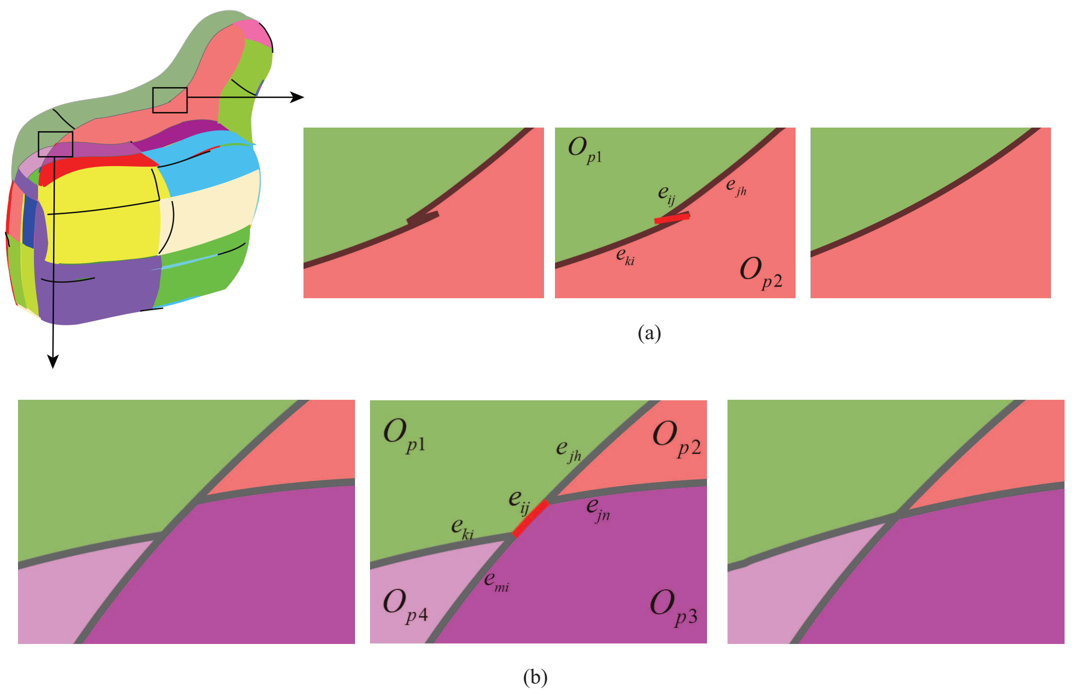

Figure 11 (Color online) (a) The leftmost picture depicts the result of merging simplification. The zigzag curve problem is presented in more detail on the right, where the short edge e_{ij} is marked in red. The edge e_{ij} and both of its two neighboring edges (e_{ki} and e_{jh}) that are found using one of the adjacent patches O_{p1} and O_{p2}. The edges e_{ki} and e_{jh} can be merged into a new curve by curve-curve merging. (b) Details of the curve confusion problem at a junction. The short edge e_{ij} located on the confusion junction is marked in red. Two valid curve pairs e_{ki} e_{jn} and e_{jh} e_{mi} are merged into the corresponding new curves.

3.4 Technical details

According to the three merge operations described in Subsection 3.2, the calculation of weights is discussed as follows.

Patch-patch merging. $O_{\text{new}} = O_{p1} \cup O_{p2}$, where O_{new} is a patch. Two shape deviations are calculated as $d_1(O_{p1}, O_{\text{new}})$, $d_2(O_{p2}, O_{\text{new}})$, and the weight is $w = \min(d_1, d_2)$. Curve-curve merging is accomplished in a similar manner.

Curve-patch merging. $O_{\text{new}} = O_p \cup O_l$, where O_{new} is a patch and the weight is $w = d_1(O_p, O_{\text{new}})$.

Aided by Figure 4, the calculation procedure for the FD of a patch is easy to understand. For the FD of a curve O_l, the calculation process is the same as that of a patch. A curve can be considered to be a slender patch with a considerably small width. If n ordered points exist on a curve O_l, it can be expressed as $\{p_0^{O_l}, p_1^{O_l}, \ldots, p_{n-2}^{O_l}, p_{n-1}^{O_l}\}$; for an infinitesimally narrow patch, $\{p_0^{O_l}, p_1^{O_l}, \ldots, p_{n-2}^{O_l}, p_{n-1}^{O_l}, p_{n-2}^{O_l}, \ldots, p_1^{O_l}, p_0^{O_l}\}$. We have devised a simple test to verify whether the FD that is calculated in this manner is appropriate. As depicted in Figure 12(a), eight lines are drawn from right to left. The degree of bending of these curves increases for each drawn curve. We first calculate the FDs of these eight lines. Using the rightmost line as a reference, we calculate the shape similarities with other lines $d(l_1, l_i)$, $i = 2, 3, \ldots, 8$. As depicted in Figure 12(b), the degree of deviation of the seven shape similarities increases in order, which indicates that this method of FD calculation is reasonable.

4 Experiments and discussions

To evaluate the validity of our algorithm, we verify it using datasets provided by previously published studies [1, 2, 6], an image simulation from [8], as well as using some vector graphs of different styles, including three cartoon art graphs and three product design graphs, as presented in Figure 13.

It is necessary to highlight some differences between our method and the method in [2] because both

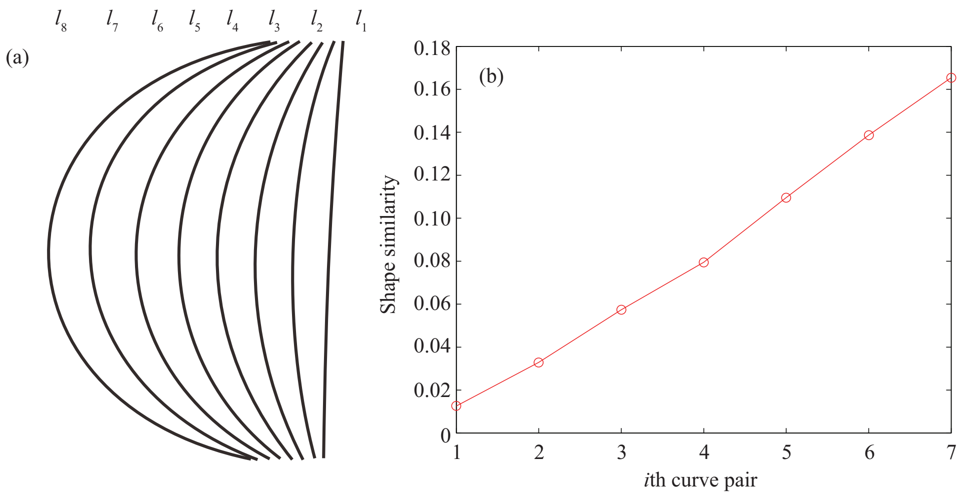

Figure 12 (Color online) (a) Eight lines are drawn from right to left with a progressive increase in the bend angle; (b) shape similarities of the lines.

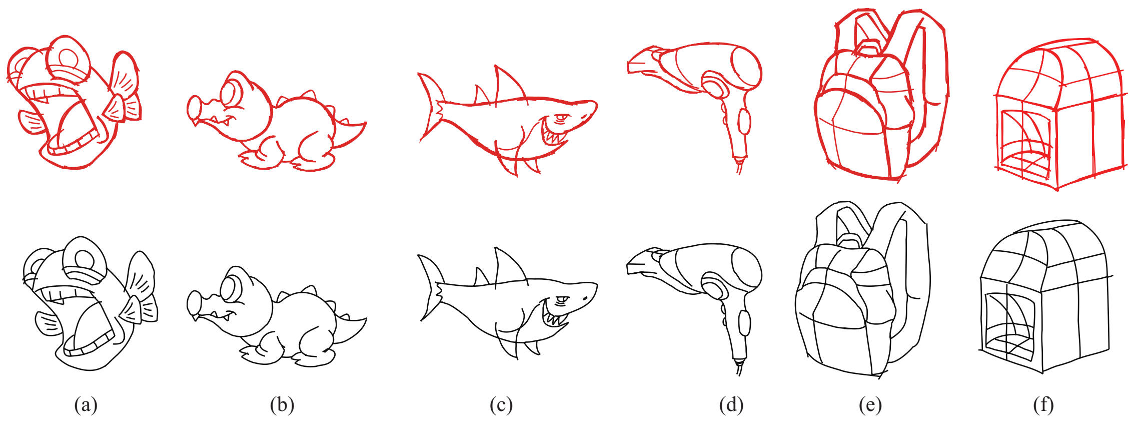

Figure 13 (Color online) Graphs in the first row are input sketches. Their simplified versions are shown in the second row.

the studies use a similar method for region merging. Most importantly, the objectives of these two studies are different. The method in [2] is a stroke grouping method in which region merging led by region detection is only used to address the over-segmentation problem (a "trapped ball" method), and the merged regions are further used to improve or guide the stroke grouping operations. An example of this approach is depicted in Figure 14. The objective of our study is to directly reduce the number of graphic primitives to achieve the simplification of a base complex. Additionally, the regions in [2] are in raster forms, and a large amount of calculation is required for a region of high resolution. In contrast, a patch in our method is a vectorized polygon, and only its boundary has to be considered, as denoted by the blue patch in Figure 1(c).

Some comparisons with other existing algorithms are given in the following paragraphs, where the parameter setting and performance of our algorithm are also discussed.

Comparisons. The results of comparisons with the methods presented in [1, 2, 6, 8] are depicted in Figure 15. The graphs in the first row are the input sketches, whereas those in the second row are results from existing methods ([1] in Figure 15(a), [6] in Figure 15(b) and (c), [2] in Figure 15(d) and (e), and [8] in Figure 15(f)). The graphs that are illustrated in the last row are results of our method. First, in terms of visual effects, our results are as good as those obtained by other methods, and some of them are even better, such as that shown in Figure 15(a). Regardless of the differing styles of the sketches, our method provides concise and simplified results. Second, in terms of processing time, our algorithm is more efficient. The running times that are required for the two sketches in Figure 15(d) and (e) are

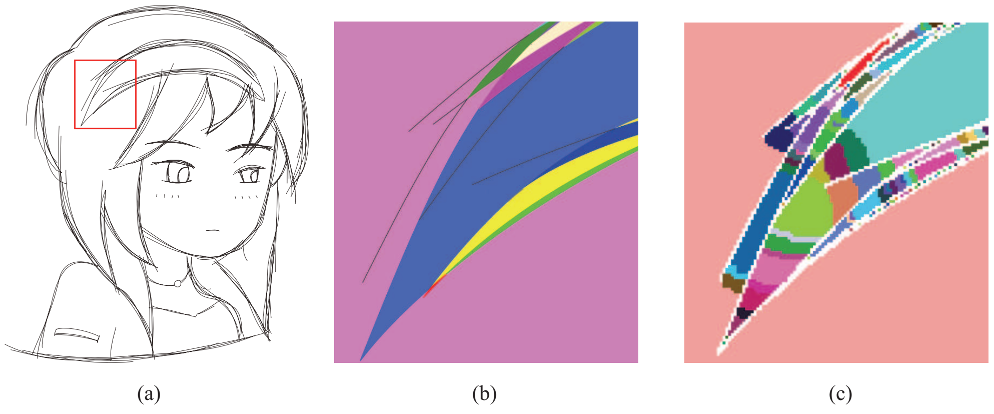

Figure 14 (Color online) (a) Input sketch of a girl, where the red rectangle denotes an example area of region detection; (b) patches in the base complex of our method; (c) initial over-segmented regions obtained by "trapped ball" method in [2].

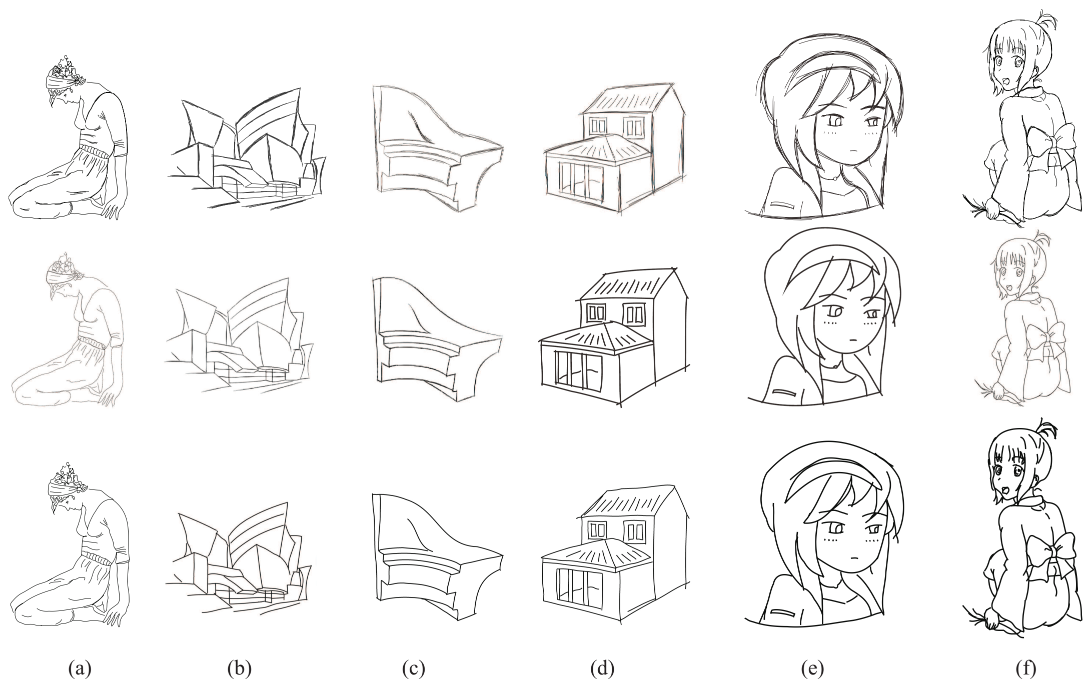

Figure 15 The graphs in the first row are the input sketches, where (a) is obtained from [1], (b) and (c) from [6], (d) and (e) from [2], and (f) is a vector graph simulated from [8]. The second row shows the results of the compared methods, and the third row presents the results of our method.

16 and 33 s, respectively, while the method in [2] requires 2.3 and 3.8 min for the same sketches. The running time of our method on some of the other sketches in Figure 15 is even shorter because their complexity is lower than that of sketches in Figure 15(d) and (e).

In addition to producing a concise output, another advantage of our algorithm is its capability to satisfy the high fidelity requirement, i.e., it is able to maintain as much useful strokes of the input vector graph as possible. In Figure 16, we show some detailed parts of the graphs at which this effect is shown. For example, by applying the method in [10] to the Figure 13(b), we cannot obtain complete connection between the teeth of the animal, regardless of the manner in which we adjust the operation thresholds. In contrast, our method retains the curve connecting the two patches and obtains a more complete shape. Similarly, some of the details from the results of [2] are shown in the second row of Figure 16(c).

The inputs of the learning-based method described in [8] are presented in raster form, and their outputs are also raster images. They perform a simple vectorization by the publicly available Potrace software in the final step. Their method is robust and effective and is considerably suitable to be used in the field of

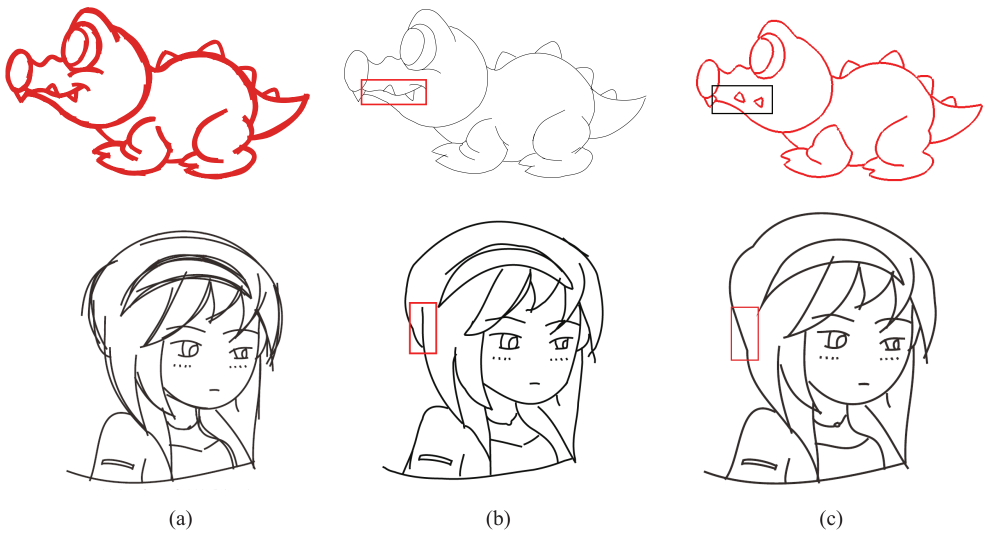

Figure 16   (Color online) (a) Input sketches; (b) outputs of our method; (c) output of the method from [10] in the first row and that of the method from [2] in the second row. The parameter options of [10], i.e., the maximal number of open curves, the minimal length of open curves, and the minimal region size, are set to 12, 10, and 7, respectively.

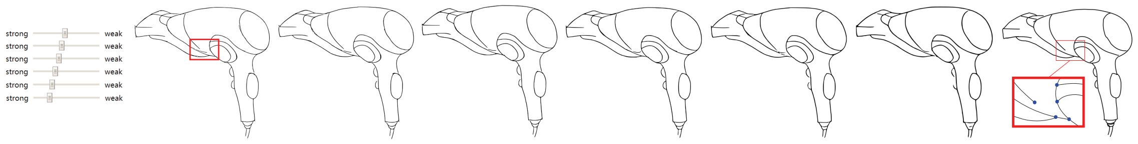

Figure 17   (Color online) Six results obtained using a different simplification degree (from weak to strong) of the [8] online demonstration. The final graph is the result of our method. Despite minor deviations in the sequence, the part highlighted by the red rectangle is not simplified as accurately as in the result of our method. The blue dots show the topological nodes.

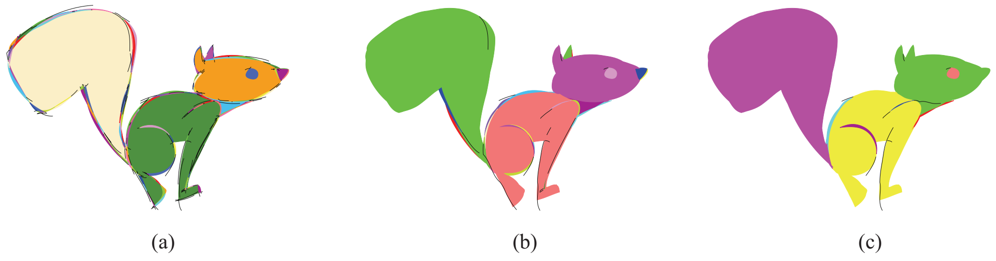

Figure 18   (Color online) (a) Initial base complex, containing 198 patches and 176 curves; (b) by setting $\varepsilon = 0.015$, we obtain a simplified base complex with 28 patches and 15 curves; (c) by setting $\varepsilon = 0.05$, we obtain a simplified base complex with 8 patches and 11 curves. The nose and an ear of the squirrel are merged into the neighboring patch, which does not occur in (b).

comics or manga; however, it cannot be used for product design that requires 3D editing. The limitation is that aside from the common problem in machine learning which has strong dependency on the quality and quantity of the training data, it does not consider the topological structure (e.g., [30]) of the graph, thereby neglecting connectivity in some crucial parts. However, our approach maintains the topological structure information of the graph, as shown using an example in Figure 17.

Parameters. The merging algorithm of our method is completed when the current minimum weight is greater than the parameter ε, which controls the simplification level of the graph. If a larger ε is set, some details of the input graph may disappear. The larger the value of ε is, the higher will be the degree of simplification of the vector graph, further, more amount of information will be lost. The results obtained using different thresholds are shown in Figure 18.

In addition to being able to control the base complex simplification level by setting the parameter ε, one can also specify the number of graphic primitives that should be left in base complex such as the

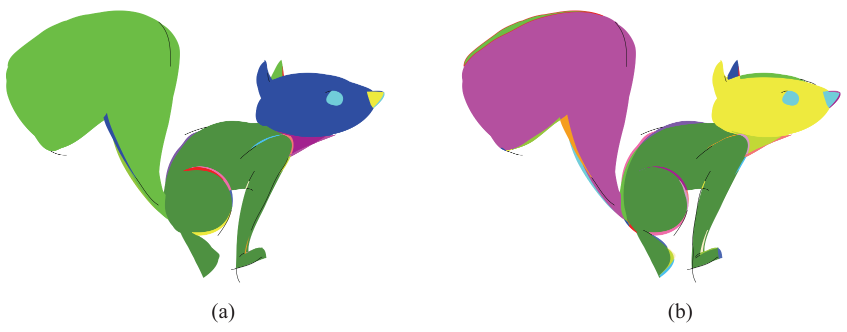

Figure 19 (Color online) (a) By setting $N = 128$ and $\varepsilon = 0.015$, 41 patches and 16 curves remaine in the base complex after merging; (b) by setting $N = 1024$ and $\varepsilon = 0.015$, 23 patches and 14 curves remaine in the base complex after merging.

Table 1 Parameter settings and running time

Input	ε	α	Time (s)
a	0.018	0.012	6.332
b	0.025	0.012	8.118
c	0.013	0.012	7.652
d	0.014	0.012	3.488
e	0.014	0.013	5.085
f	0.015	0.012	1.317

number of patches or the number of curves. However, because the number and shape of the graphic primitives vary with different vector graphs, it is more convenient to control the simplification by setting ε. By conducting many experiments, we have conclude that the optimal value of ε is generally between 0.01 and 0.03, and is usually around 0.015.

While calculating the FD, the number of boundary sampling points is set as N. As reported by [36], the shape characteristics of a graph can be represented with 128 sampling points. The accuracy of the shape representation in our method can be adjusted by varying the number of sampling points. If more sampling points are chosen, more details of the shape can be preserved. Consequently, the matching result will be more accurate. To optimally balance the accuracy and time, we set $N = 256$. A change in N will affect the value of ε. If we commence by setting the simplified base complex to $N = 256$ and $\varepsilon = 0.015$ and by subsequently changing the value of N to 128, the value of ε should be increased to obtain a similar simplified result. Conversely, if N is set to 512, ε should be decreased 0.015 to get a similar simplified result. This is because when the value of N is large, the accuracy of features is high and the corresponding error is small, which indicates that the d calculated by Euclidean metric will be small; Further, a small threshold ε should be chosen. An example of this effect is shown in Figure 19.

Other parameters in our algorithm, such as $d_\varepsilon = \alpha D$ which indicates the average bandwidth of the input sketch, are fixed; specifically, $\alpha = 0.012$. As already described, during the curve-curve merging, we use the distance and angle constraints to identify curve pairs, which is a procedure similar to the one in [1]. However, when the minimum distance is shorter than d_ε and when the angle of two tangents at the shortest distance is smaller than d_θ, the two curves can be paired up but not necessarily merged. This is because of the restriction of ε; if the shape of the potential new curve changes considerably as compared to the two initial curves, the cost of this step would become larger than ε, and they would not be merged. The value of α only affects valid pairs of curves but does not affect merging. Therefore, a larger α can be set to obtain more possible valid curve pairs, without any concern about this resulting in undesired shape changes during simplification.

Performance. The running time of our algorithm primarily depends on the size and complexity of the input graphs or the number of primitives in the base complex, and it ranges from a few seconds to several minutes. The degree of simplification of the vector graph can be controlled by modifying the dominant parameter ε. Examples of the running time for several sketches under different parameter settings (shown in Figure 13) are presented in Table 1 .

Limitations. As mentioned in Subsection 3.2, the PP merging is performed by simply combining two patches with a common edge into a new one. These patches blindly expand, without shrinking in shape, which leads to the appearance of boundary patches. This phenomenon is especially noticeable if a considerably rough graph is used. Although boundary patches can be rectified during post-processing by performing CC re-merging, if the boundary patches are considerably wide, the results of the re-merging simplification are not desirable. In addition, it may happen that two curves which should perceptually be merged but do not merge during the merging procedure because a fixed experimental value of ε is used and $w > \varepsilon$. We will study and address these limitations of our method in future work.

5 Conclusion

We have proposed a vector graph simplification method based on the agglomeration of primitives in the base complex of a vector graph. A base complex is essentially a collection of graphic primitives, i.e., nodes, curves, and patches. And the core of our method is an algorithm that iteratively merges two primitives of the base complex, guided by the shape similarity measurement. A single threshold is used to control the simplification of the base complex.

Our algorithm is fast and simple, and it can be applied to arbitrary vector graphs. We would like to implement the functionality which would allow users to edit sketches on-the-fly in the future. Instead of stopping our simplification process using a hard threshold, we would like to enhance the robustness of our approach by also incorporating the intention of the user [37].

Acknowledgements This work was partly supported by National Natural Science Foundation of China (Grant Nos. 61572292, 6160227, 61672187), NSFC Joint Fund with Guangdong (Grant No. U1609218), and Shandong Provincial Key Research and Development Project (Grant No. 2018GGX103038).

References

1 Barla P, Thollot J, Sillion F X. Geometric clustering for line drawing simplification. In: Proceedings of the 16th Eurographics Conference on Rendering Techniques, 2005. 183–192
2 Liu X, Wong T T, Heng P A. Closure-aware sketch simplification. ACM Trans Graph, 2015, 34: 1–10
3 Shesh A, Chen B Q. Efficient and dynamic simplification of line drawings. Comput Graph Forum, 2008, 27: 537–545
4 Grabli S, Durand F, Sillion F X. Density measure for line-drawing simplification. In: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, 2004. 309–318
5 Pusch R, Samavati F, Nasri A, et al. Improving the sketch-based interface. Visual Comput, 2007, 23: 955–962
6 Orbay G, Kara L B. Beautification of design sketches using trainable stroke clustering and curve fitting. IEEE Trans Visual Comput Graph, 2011, 17: 694–708
7 Ogawa T, Matsui Y, Yamasaki T, et al. Sketch simplification by classifying strokes. In: Proceedings of the International Conference on Pattern Recognition, 2016. 1065–1070
8 Simo-Serra E, Iizuka S, Sasaki K, et al. Learning to simplify: fully convolutional networks for rough sketch cleanup. ACM Trans Graph, 2016, 35: 1–11
9 Bartolo A, Camilleri K P, Fabri S G, et al. Scribbles to vectors: preparation of scribble drawings for CAD interpretation. In: Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2007. 123–130
10 Favreau J D, Lafarge F, Bousseau A. Fidelity vs. simplicity: a global approach to line drawing vectorization. ACM Trans Graph, 2016, 35: 1–10
11 Parakkat A D, Pundarikaksha U B, Muthuganapathy R. A Delaunay triangulation based approach for cleaning rough sketches. Comput Graph, 2018, 74: 171–181
12 Zou J J, Yan H. Cartoon image vectorization based on shape subdivision. In: Proceedings of the International Computer Graphics, 2001. 225–231
13 Bo P B, Luo G N, Wang K Q. A graph-based method for fitting planar B-spline curves with intersections. J Comput Des Eng, 2016, 3: 14–23
14 Wang Y T, Wang L Y, Deng Z G, et al. Sketch-based shape-preserving tree animations. In: Proceedings of the Computer Animation and Social Agents, 2018
15 Guo X K, Lin J C, Xu K, et al. CustomCut: on-demand extraction of customized 3D parts with 2D sketches. In: Proceeding of the Eurographics Symposium on Geometry Processing, 2016
16 Shesh A, Chen B. Smartpaper: an interactive and user friendly sketching system. In: Proceedings of the Computer Graphics Forum, 2004
17 Ku D C, Qin S F, Wright D K. Interpretation of overtracing freehand sketching for geometric shapes. In: Proceedings of the Computer Graphics, Visualization and Computer Vision, 2006. 263–270

18 Schmidt R, Wyvill B, Sousa M C, et al. Shapeshop: sketch-based solid modeling with blobtrees. In: Proceedings of the ACM SIGGRAPH, San Diego, 2007

19 Bae S H, Balakrishnan R, Singh K. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In: Proceedings of the ACM Symposium on User Interface Software and Technology, 2008. 151–160

20 Grimm C, Joshi P. Just DrawIt: a 3D sketching system. In: Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, 2012. 121–130

21 Schroeder W J, Zarge J A, Lorensen W E. Decimation of triangle meshes. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, 1992. 65–70

22 Hoppe H, DeRose T, Duchamp T, et al. Mesh optimization. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 1993. 19–26

23 Guéziec A. Surface simplification with variable tolerance. In: Proceedings of the 2nd International Symposium on Medical Robotics and Computer Assisted Surgery, 1995. 132–139

24 Garland M, Heckbert P S. Surface simplification using quadric error metrics. In: Proceedings of the 24th International Conference on Computer Graphics and Interactive Techniques, 1997. 209–216

25 Cohen J, Manocha D, Olano M. Simplifying polygonal models using successive mappings. In: Proceedings of the Conference on Visualization, 1997. 395–402

26 Kobbelt L, Campagna S, Seidel H P. A general framework for mesh decimation. In: Proceedings of the Graphics Interface Conference, 1998. 43–50

27 Hoppe H. View-dependent refinement of progressive meshes. In: Proceedings of the 24th International Conference on Computer Graphics and Interactive Techniques, 1997. 189–198

28 Tarini M, Puppo E, Panozzo D, et al. Simple quad domains for field aligned mesh parametrization. ACM Trans Graph, 2011, 30: 1

29 Gao X F, Deng Z G, Chen G N. Hexahedral mesh re-parameterization from aligned base-complex. ACM Trans Graph, 2015, 34: 142:1–142:10

30 Noris G, Hornung A, Sumner R W, et al. Topology-driven vectorization of clean line drawings. ACM Trans Graph, 2013, 32: 1–11

31 Kauppinen H, Seppanen T, Pietikainen M. An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. IEEE Trans Pattern Anal Mach Intell, 1995, 17: 201–207

32 Loncaric S. A survey of shape analysis techniques. Pattern Recogn, 1998, 31: 983–1001

33 Zhang D S, Lu G J. A comparative study on shape retrieval using Fourier descriptors with different shape signatures. In: Proceedings of Asian Conference on Computer Vision, 2002

34 Zhang D S, Lu G J. Study and evaluation of different Fourier methods for image retrieval. Image Vision Comput, 2005, 23: 33–49

35 Glassner A. Graphics Gems. Orlando: Academic Press, 1990

36 El-ghazal A, Basir O, Belkasim S. Farthest point distance: a new shape signature for Fourier descriptors. Signal Process-Image Commun, 2009, 24: 572–586

37 Durand F. Where do people draw lines?: technical perspective. Commun ACM, 2012, 55: 106