

A sketch-based interactive framework for real-time mesh segmentation

Abstract Meaningful mesh segmentation plays a more and more important role in various graphics applications, such as texture mapping, shape retrieval, and high-quality metamorphosis. This paper proposes a sketch-based interactive framework for real-time mesh segmentation. With an easy-to-use tool, the user can freely segment a 3D mesh while needing little effort or skill. In order to meaningfully segment the mesh, two dimensionless feature sensitive metrics are proposed, which are independent of the model and part size. We show that these metrics give the clear physical meaning to illustrate discrete differential geometric features, such as the curvature tensor and the curve length of gaussian image. Finally, we discuss three kinds of boundary smoothing methods, and present two fast topology-invariant, geometry-invariant adjustment algorithms based on convex hull features and angle features. Compared with previous methods, our method can easily achieve multi-level segmentation in just one session.

Keywords Sketch-based mesh segmentation · Dimensionless feature sensitive metric · Convex hull fitting · Angular feature smoothing

1 Introduction

Mesh segmentation (or decomposition, partition), is a useful tool for modeling [1], texture mapping [2], cross-parameterization [3], metamorphosis [4], shape retrieval [5] and collision detection [6] in computer graphics. Currently, triangle mesh is one of the most common 3D data representations due to its advantages in expressive ability and speed. However, triangle mesh does not capture high-level structures, such as semantic features. In order to obtain a meaningful object segmentation, it is vital to extract these semantic features from low-level data representations. But the concept of meaningful segmentation heavily depends on the concrete application context

and varies among individuals due to subjectivity. As a result, it is still a challenging problem in computer graphics. Now a great interest has been gained in this research community and much progress has been achieved.

Most existing mesh segmentation methods concentrate on the automatic decomposition of a surface into meaningful shape features. One important goal of these methods is to avoid over-segmentation and obtain small patches by orders of magnitude. The final outputs are typically less than 10 parts. In these methods, some control parameters are still needed to fine tune for different scenes. So it is still hard to achieve a strict-meaningfully automatic segmentation, due to this problem's intrinsic difficulties.

Our main motivation is based on the fact that in the real applications, the final outputs are usually very small number of parts (5~8, typically < 10). So the user prefers to decide the output result by his simple and intuitive assignment, rather than by abstract parameter-tuning. Furthermore, in many important applications of mesh segmentation, such as texture mapping and cross-parameterization, a good user-oriented adjustment interface is necessary for high-quality and visually appealing results. Therefore, an intuitive and simple tool plays a crucial role in this task. However, existing automatic methods can not provide such an intuitive interface. We notice that the emerging sketch-based interactive techniques bridge the gap between the user's subjective intention and the complex 3D objects. With this convenient interactive mode, the user freely draws a few loose strokes to identify interested areas or parts, then our system gives the desired segmentation results. No special training is needed for the non-expert users.

1.1 Contribution

The contribution of this paper is three-fold. Firstly, we introduce a sketch-based segmentation system with a simple and intuitive interface for efficiently performing mesh segmenting operation. Next, we propose two angle-

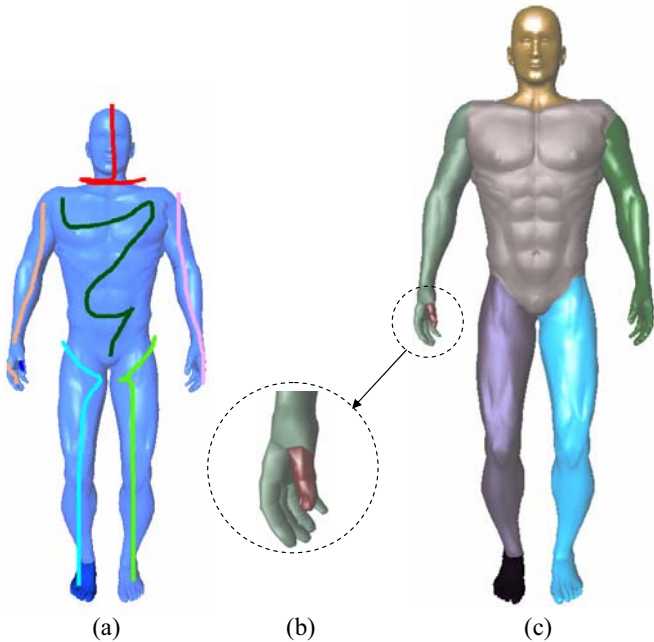


Fig. 1 An example of our method. (a) The user draws a few freehand sketches on the model. (c) Then the system gives the desired segmentation. Sketch-based interactive mode can easily implement multi-level segmentation just in one session, such as the tiny thumb zoomed in (b) vs. the huge body.

based feature sensitive metrics, which are independent of the model and part size. These metrics can clearly illustrate the discrete differential geometric features, such as the curvature tensor and the curve length of gaussian image, which provide our method a solid foundation. Finally, we discuss three kinds of boundary smoothing methods and introduce two fast topology-invariant, geometry-invariant refinement algorithms based on the convex hull features and the angle features. It is worth noting that our segmentation system not only improves the user’s segmenting experience, but also easily achieves multi-level segmentations just in one session, compared with previous automatic segmentation algorithms. Take the case in Fig. 1 for instance, the user freely segments the Android model into several parts of any size, ranging from long legs to any user-specified tiny finger, just in one interactive session. This kind of flexibility makes our system satisfy the requirements of common applications.

1.2 Definitions and scheme overview

We firstly introduce some definitions used frequently throughout this paper and then give the scheme overview.

Definition 1.2.1 *Vertex-based Part Segmentation:* Let $M = (T, G)$ be a 2-manifold triangular mesh. $T = (V, E, F)$ is a graph where V denotes the set of vertices, E denotes the set of edges and F denotes the set of facets, and G is the geometry associated with each vertex in V . P_1, P_2, \dots, P_n is a vertex-based segmentation of mesh

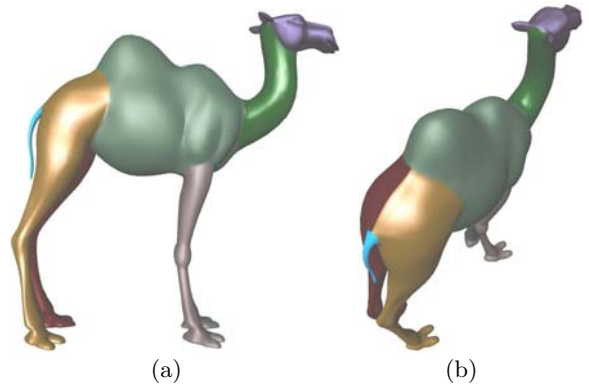


Fig. 2 Another example of our method. From the front view (a) and the side view (b), it can be seen that our system gives a natural segmentation for the Camel model.

M iff (1) $\forall i, 1 \leq i \leq n, P_i \in V$; (2) $\bigcup_{i=1}^n P_i = V$; (3) $\forall i, j, i \neq j, P_i \cap P_j = \emptyset$.

Definition 1.2.2 *Facet-based Part Segmentation:*

P_1, P_2, \dots, P_n is a facet-based segmentation of mesh M iff (1) $\forall i, 1 \leq i \leq n, P_i \in F$; (2) $\bigcup_{i=1}^n P_i = F$; (3) $\forall i, j, i \neq j, P_i \cap P_j = \emptyset$.

The rest of the paper is structured as follows. Section 2 gives a brief review of related work. Section 3 describes the vertex-based feature sensitive metric and its physical meaning in detail. Section 4 describes the facet-based feature sensitive metric. Section 5 discusses the methods for boundary refinement. Section 6 presents some results. Section 7 concludes and discusses future work.

2 Background and related work

Mesh segmentation: Segmentation, to divide an object into distinct and meaningful parts, is also a classic subject in computer vision. Specifically, in image editing, segmentation is an important preprocessing step for the subsequent composition step. Due to the intrinsic cognitive difficulty, this quite easy task for human eye is still a long way for the computer to perform in the same easy way. So recently in this field, many convenient interactive methods have been developed to assist the computer (also the user) to produce high-quality segmentation results [7]. With these methods, the user’s experience is significantly improved, contrast to per-pixel specifying methods or the lasso tool.

For 3D mesh segmentation, the main difference from image segmentation is that model’s geometric and topological information is given beforehand, and we need not consider other factors such as occlusion and lighting variance. Nevertheless, this does not make mesh segmentation an easy task. Actually, it is another challenging work. Triangle mesh is organized in low-level structures,

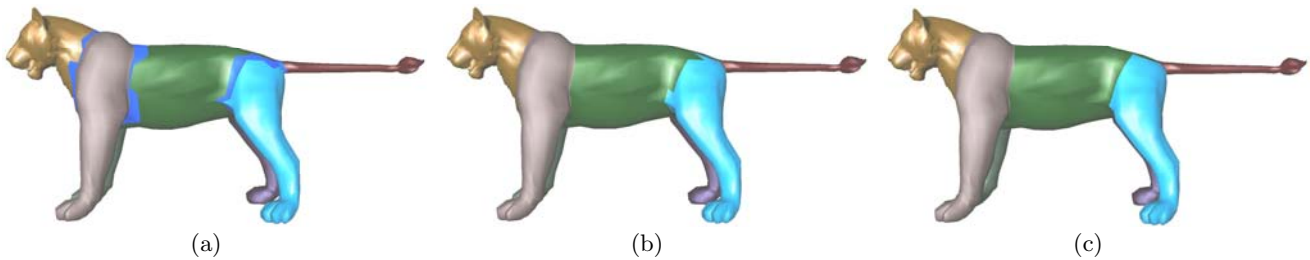


Fig. 3 Three steps in our method. (a) The segmentation result after vertex-based diffusion. Note that the blue part is the boundary facets before applying facet part Strategy 2 in Section 4. (b) The result after applying all the facet part strategies. (c) The result after boundary refinement.

while the segmentation is related to the high-level shape reasoning and understanding, which is still an open problem both in computer vision and computer graphics. Furthermore, 3D models lack a regular parameterization domain, which is harder to deal with than images.

Despite the intrinsic difficulty, much progress [6, 8–15, 19] has recently been achieved in this field. These methods are mainly based on psychological studies of human vision: (1) human’s perception for shape is partly based on decomposition and complex objects can be regarded as a combination of meaningful simple components [16]; (2) human perception inclines to divide an object into parts along negative minima of principle curvatures [17]. These studies provide useful clues for mesh segmentation.

Based on these computable hypotheses, various techniques have been proposed. Generally speaking, these segmentation methods can be classified by the following characteristics: whether it is based on local feature (such as length, curvature [9]) diffusion or global primitive approximation (such as blowing bubbles [19], cylinder tubular [10], convex hull [6, 15]); whether it is based on similar property clustering or distinct feature identification (such as the boundary detection in [11]); whether it is bottom-up hierarchical or top-down hierarchical; whether it is vertex-based or facet-based. See [20] for a good survey on mesh segmentation. [21] gives a comparative review for recent techniques.

Among these approaches, the local diffusion scheme usually achieves faster speed than the global approximation scheme, while the latter achieves finer results than the former. In our system, we take the local diffusion scheme similar to [14, 22] as the startup. We also introduce two scale-independent metrics and discuss their clear physical meanings. With the two feature sensitive metrics, our k -way segmentation system is efficient and effective. Moreover, We take careful boundary refinement into consideration. Contrasted with initial local growing scheme, our boundary refinement method adopts global fitting scheme, thus achieves a good balance between the speed and the effect of the overall procedure. Since the number of the mesh’s facets is usually greater than that of the vertices, the computation scale of the vertex-

based scheme is smaller than the facet-based one. In our scheme, we first adopt a vertex-based diffusion procedure and then take a facet-based procedure for refinement.

Sketch-based interactive interface: Sketch-based interactive mode effectively bridges the gap between current 2D human-computer interface and complex 3D models [23–26]. Sketch-based interface shows powerful abilities in editing, manipulating, and animating existing 3D meshes. It is worth noting that sketch is not only “skeleton” (uncrossed curve), but also can include diverse 2D feature components with 3D sensibleness, such as point, (directed) curve, region, cutting plane etc. In this paper, we only take the undirected curve as the sketch component, since it has already satisfied most requirements. Secondly, different from the image, which has only one front view in 2D plane, 3D model is rendered on the screen with current view, and back parts may be occluded by front parts. A simple strategy to deal with it is to navigate the model to the appropriate position and then begin drawing sketches. More complex schemes carefully take the Z-buffer into consideration.

3 Generating vertex parts with angle-based diffusion metric

In the interactive step, the user draws a few freehand sketches on the 2D screen plane. These sketches are projected onto the 3D mesh to produce some seed vertices. Every such vertex is assigned an index due to the sketch it belongs to. Then we diffuse these initial vertices to complete segmentation for the whole mesh. The natural way is to consider 1-ring neighborhood diffusion, which needs certain metric to decide feature distance between two neighbor vertices. Compared with the curvature and angle information, the Euclidean length is not a rather relevant factor in the case of 1-ring neighborhood. So we introduce an angle-based diffusion metric, which is naturally independent of the model and part size. In the following, we give a clear illustration to this metric.

As illustrated in Fig. 4a, we mark a vertex as the source v_s with its normal n_s , and mark a 1-ring neighborhood vertex as the target v_t with its normal n_t . Since

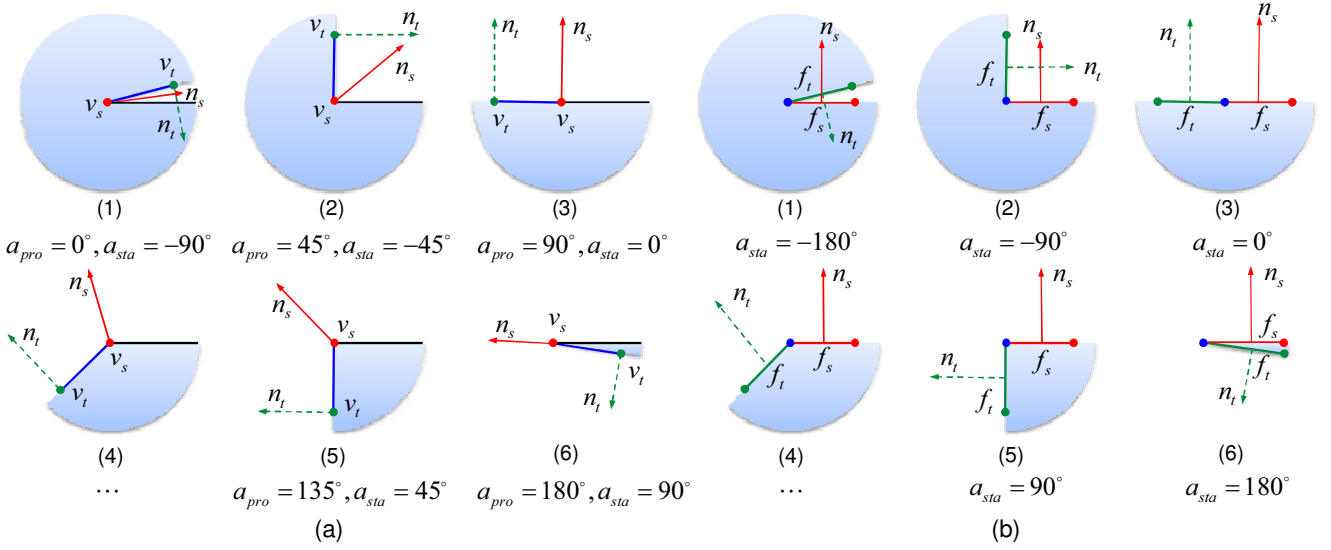


Fig. 4 The illustration of silhouette image for (a) angle-based feature sensitive metric for the vertices and (b) angle-based feature sensitive metric for the facets. Detailed explanations are given in Section 3 and Section 4.

the normal n_s (or n_t) is computed by averaging the 1-ring neighborhood facets' normals, we regard this as a *state* because it reflects the local state of a vertex's 1-ring neighbors. Similarly, we consider the directed edge e_{st} connecting the source v_s and the target v_t as a *procedure*, which represents the transferring procedure from the source to the target. To compute the sensitive distance between v_s and v_t , we need to consider two aspects: the magnitude of the procedure and the magnitude of the state's change. Specifically, we define the angle a_{pro} from the source normal n_s to the connecting edge e_{st} as the procedure value, and the angle a_{sta} from the source normal n_s to the target normal n_t as the state value. From the silhouette image shown in Fig. 4a, due to the concave shape, we will classify the source and the target from Fig. 4a-1 to Fig. 4a-3(not including) into two different convex parts (in other words, further distance), while classify the source and the target from Fig. 4a-4 to Fig. 4a-6 into the same convex part (or closer distance), due to the convex shape. Furthermore, we need to pay more attention to deal with the former case, in order to complete the segmentation. From Fig. 4a-1 to Fig. 4a-6, the angle a_{pro} arises from 0° to 180° ; the angle a_{sta} arises from -90° to 90° . As for the first case, the feature distance trend decreases when the angle a_{pro} increases, which acts like the cosine curve trend in this angle region. In the second case, due to the indetermination of the positive and negative angle in 3D space (if extra reference lacks), we focus on the relatively important region $-90^\circ \sim 0^\circ$. In this angle domain, the distance trend also decreases when the angle a_{sta} increases, which acts like the absolute sine curve ($|\sin(x)|$).

Now, we formulate the angle a_{pro} :

$$\cos(a_{pro}) = \frac{n_s \cdot e_{st}}{\|e_{st}\|} \quad (1)$$

Interestingly, this item is similar to the discrete directional curvature [27, 28], i.e.,

$$k_d = k_p^1 \cos^2(\theta) + k_p^2 \cos^2(\theta) \xrightarrow{\text{discrete}} 2 \frac{n_s \cdot e_{st}}{\|e_{st}\|} \quad (2)$$

where k_p^1 and k_p^2 are the two principal curvatures.

Then, we formulate the angle a_{sta} :

$$|\sin(a_{sta})| = \|n_s \times n_t\| \quad (3)$$

Again interestingly, we find that the curve length of gaussian image [22] is similar to the second angle, because this formula below has the same trend curve as the angle a_{sta} in $-90^\circ \sim 90^\circ$.

$$\int_{G^*} ds^* \xrightarrow{\text{discrete}} \|n_s - n_t\| = 2|\sin(\frac{a_{sta}}{2})| \quad (4)$$

where ds^* is the arc element on the Gaussian image G^* .

Thus, our overall feature distance metric is defined according to both the procedure angle a_{pro} and the state angle a_{sta} :

$$\begin{aligned} f_d^V(s, t) &= (1 + \cos(a_{pro}))(1 + |\sin(a_{sta})|) \\ &= (1 + \frac{n_s \cdot e_{st}}{\|e_{st}\|})(1 + \|n_s \times n_t\|) \end{aligned} \quad (5)$$

This metric is a dimensionless feature sensitive metric, i.e., it is independent of the model and part size. In above definition, there are two additional terms of 1: the first one is used to avoid negative metric, the second

one prevents the cancelation of one item as the other approaches 0. We complete the initial vertex-based diffusion with this metric, as shown in Fig. 3a. Note that additional enhanced functions also can be applied to different angle regions for various filtering effects.

In the following, we introduce our diffusion algorithm. For each vertex, we define its property with a tetrad $\{v_i, if_known, id, f_d^V\}$, where v_i is a vertex with the index i , $i = 1, 2, \dots, |V|$; if_known is a Boolean variable to decide whether this vertex's part id is determined (known or unknown); id is the vertex's part id ($id=1, 2, \dots, n$; 0 is reserved for the initial unknown state); f_d^V is this vertex's feature distance to some known vertex in 1-ring neighborhood. Initially, the vertices that have a known vertex in 1-ring neighborhood are inserted to a candidate list. At each step, we find a vertex with the shortest feature distance, change its state from unknown to known, delete it from the candidate list, and insert the unknown vertices in its 1-ring neighborhood to the candidate list. Our vertex-based diffusion procedure is shown in Algorithm 1.

Algorithm 1: Vertex-based part diffusion algorithm

Data: Mesh M , and n sketches by user's strokes. The initial vertices V_{init} are computed out by projecting these sketches to the image plane.
Result: n parts on Mesh M .

```

1 begin
2   /* initiate vertices' if_known value */
3   if  $v_i \in V_{init}$  then
4      $v_i.if\_known \leftarrow true$ ;
5   else
6      $v_i.if\_known \leftarrow false$ ;
7   /* initiate candidate_list */
8    $candidate\_list \leftarrow \{v_i, if\_known, id, f_d^V\}, v_i \in$ 
9     1-ring unknown vertices of  $V_{init}$ ;
10  while  $candidate\_list \neq \emptyset$  do
11    find  $v_{min}$  with the minimal  $f_d^V$  value in
12     $candidate\_list$ ;
13     $v_{min}.if\_known \leftarrow true$ ;
14    /* update candidate_list */
15    for  $v_i \in$  1-ring of  $v_{min}$  do
16      if  $v_i.if\_known = true$  then
17        continue ;
18      else if  $v_i \in candidate\_list$  then
19        if  $v_i.f_d^V > f_d^V(v_i, v_{min})$  then
20           $\{v_i, if\_known, id, distance\} \leftarrow$ 
21             $\{v_i, if\_known, v_{min}.id, f_d^V(v_i, v_{min})\}$ ;
22        else
23           $candidate\_list \leftarrow$ 
24             $\{v_i, if\_known, v_{min}.id, f_d^V(v_i, v_{min})\}$ ;
25    delete  $v_{min}$  from  $candidate\_list$ ;
26 end
  
```

4 Generating facet parts with angle-based feature sensitive metric

After the initial vertex-based diffusion, we have marked each vertex with a part id. Because the usual outputs are facet parts, we need to convert these vertex parts into facet ones. In order to tag each facet with a part id, we adopt the following strategies:

◊ For the facet f_s with three vertices of the same id, the facet id is equal to the vertices' id.

◊ Otherwise, the facet f_s 's id is equal to the facet f_t ' id with the closest feature distance.

We define the facet-based feature distance $f_d^F(s, t)$ similar to the vertex-based feature distance. Here, we only need to consider the *state* value. As illustrated in Fig. 4b, $f_d^F(s, t)$ decreases with the angle a_{sta} from -180° to 0° (and also from 180° to 0°), which acts like the curve trend of $(-\cos(|x|))$.

$$f_d^F(s, t) = -\cos(|a_{sta}|) = -n_s^F \cdot n_t^F \quad (6)$$

where n_s^F, n_t^F are the normals of the source and target facet, respectively.

According to above strategies, vertex parts are turned into facet parts, as shown in Fig. 3b.

5 Refining boundaries

After obtaining facet parts, we refine the boundaries of parts. Existing refinement methods can be classified into three categories: topology-variant and geometry-variant, topology-invariant and geometry-variant, topology-invariant and geometry-invariant. Most algorithms focus on the first one for the flexibility and good effect. However, some important applications of mesh segmentation, such as cross-parameterization and metamorphosis [4], require the outputs to be compatible with the inputs, i.e. with the same topology. So we opt for topology-invariant algorithms. Specifically, our refinement methods based on convex property and visual angle feature take topology-invariant and geometry-invariant strategies.

5.1 Adjusting facet parts by convex property

As mentioned above, human perception usually divides an object into parts along negative minima of principle curvatures. This implies that the meaningful parts are in some sense convex. Here, we adopt convex hull fitting scheme, which constructs more compact convex-hulls in boundary regions, to refine the boundaries. In [6, 15], the convex property is computed within a whole patch. However, iteratively constructing convex hull for the whole region is much time-consuming for all adjustment steps together, thus it is not suitable for real-time applications. In our scheme, we only consider the neighborhood

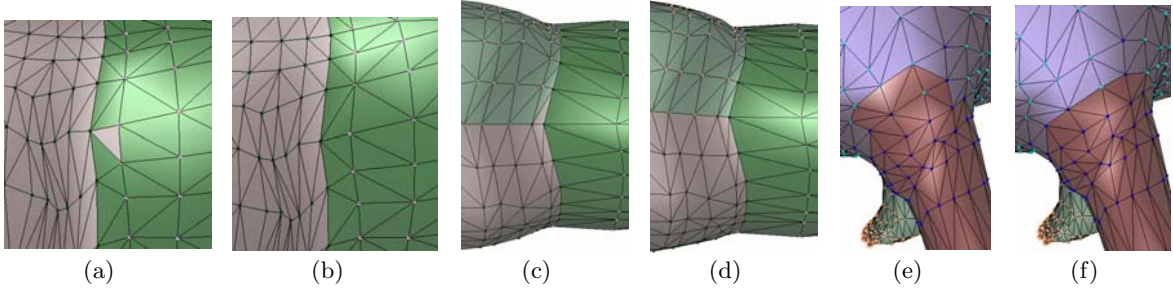


Fig. 5 The effects of our boundary refining strategies. (a) and (b) are the effects before and after applying one-side incremental convex hull adjustment strategy. (c) and (d) are the effects before and after applying two-sides dynamic convex hull adjustment strategy. (e) and (f) are the effects before and after applying angle property adjustment strategy.

around a facet to be adjusted. Thus it effectively overcomes the above drawbacks. The neighborhood facets can be 1-ring, 2-ring, or 3-ring, etc. In our experiments, we found 2-ring achieves an optimal balance in speed and effect. We adopt two convex hull adjustment strategies as described below.

One-side incremental convex hull adjustment strategy. Given an edge e_i lying on the boundary, we find its 1-ring facets F_i^{a-ring} belonging to part P_a and compute the volume $Vol(CH(F_i^{a-ring}))$ of F_i^{a-ring} 's convex hull. Then we get e_i 's incident facet f_i^b belonging to part P_b . We also compute the volume $Vol(CH(F_i^{a-ring} \cup f_i^b))$. If Inequation 7 is satisfied, the incident facet f_i^b is added to part P_a .

$$\frac{Vol(CH(F_i^{a-ring} \cup f_i^b)) - Vol(CH(F_i^{a-ring}))}{Vol(CH(F_i^{a-ring}))} < T_d \quad (7)$$

In this paper, we take the threshold $T_d = 0.1$. Likewise, the same operation is applied to part P_b . Due to the incremental convex hull construction, the computation even for overall boundary edges is still very fast. The adjustment effect is shown in Fig. 5(a)(b).

Two-sides dynamic convex hull adjustment strategy. Given an edge e_i lying on the boundary, we find its 1-ring facets F_i^{a-ring} belonging to part P_a and compute the volume $Vol(CH(F_i^{a-ring}))$ of F_i^{a-ring} 's convex hull. F_i^{b-ring} and $Vol(CH(F_i^{b-ring}))$ are obtained in the same way. Then we get e_i 's incident facet f_i^b belonging to part P_b . We also compute the volume $Vol(CH(F_i^{a-ring} \cup f_i^b))$ and $Vol(CH(F_i^{b-ring} - f_i^b))$. If Inequation 8 is satisfied, the incident facet f_i^b is added to part P_a .

$$\begin{aligned} & Vol(CH(F_i^{a-ring})) + Vol(CH(F_i^{b-ring})) > \\ & Vol(CH(F_i^{a-ring} \cup f_i^b)) + Vol(CH(F_i^{b-ring} - f_i^b)) \end{aligned} \quad (8)$$

Likewise, the same operation is applied to part P_b . The adjustment effect is shown in Fig. 5(c)(d).

5.2 Adjusting facet parts by visual angle property

In this step, we refine the boundaries directly according to human visual perception for a regular boundary, i.e., minimizing the difference of adjacent boundary angles. Given an edge e_i lying on the boundary of part P_a and its two vertices v_{prev}, v_{curr} , we get e_i 's incident facet f_i^b belonging to part P_b and e_i 's incident facet f_i^a belonging to part P_a . We compute the boundary angles a_{prev} and a_{curr} covering part P_a for v_{prev} and v_{curr} , respectively. Then we apply the following adjustment strategy:

- ◊ If $a_{curr} > 180^\circ$ and $|a_{curr} - a_{prev}| > A_{td}$, add the facet f_i^b to part P_a .
- ◊ If $a_{curr} < 180^\circ$ and $|a_{curr} - a_{prev}| > A_{td}$, add the facet f_i^a to part P_b .

Here, we take the threshold $A_{td} = 120^\circ$. Because this step is very fast, we perform it $N = 3$ times for better effect. The adjustment effect is shown in Fig. 5(e)(f).

6 Experimental results

Fig. 6 shows the GUI of our sketch-based interactive system. The user simply draws freehand 2D sketches on the 3D model with the mouse, without parameter tuning. Then our system gives the desired segmentation results. In Fig. 7, the user completes the stroke drawing just in one viewpoint within a few seconds. The tested models not only include human and animal models, but also the CAD models (Fig. 8). More results are shown in Fig. 1, 2 and 9. The whole interactive procedure is very convenient both for the experienced user and the novice.

Thanks to our dimensionless feature sensitive metrics, the system works well for meshes with various sizes. A significant advantage of our sketch-based system over automatic methods is that it can easily segment multi-level parts just in one session, such as the tiny right thumb vs. the huge body in Fig. 1, the little right horn vs. the big limbs in Fig. 7(a)(b). Moreover, our refinement algorithms generate natural boundaries for the segmentation.

The experiments were carried out on a 1.8Ghz notebook with 1Gb memory. From the overall running time

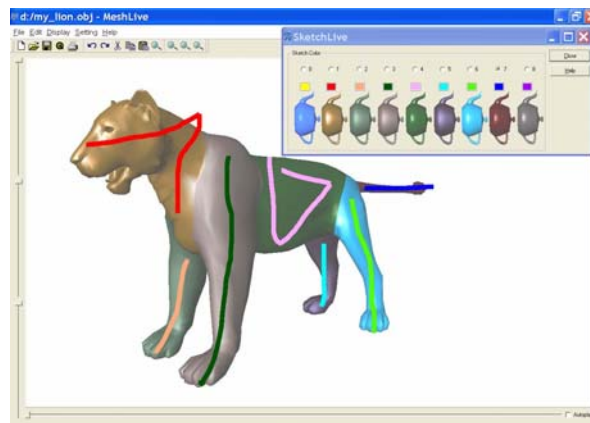


Fig. 6 The GUI of our sketch-based interactive system.

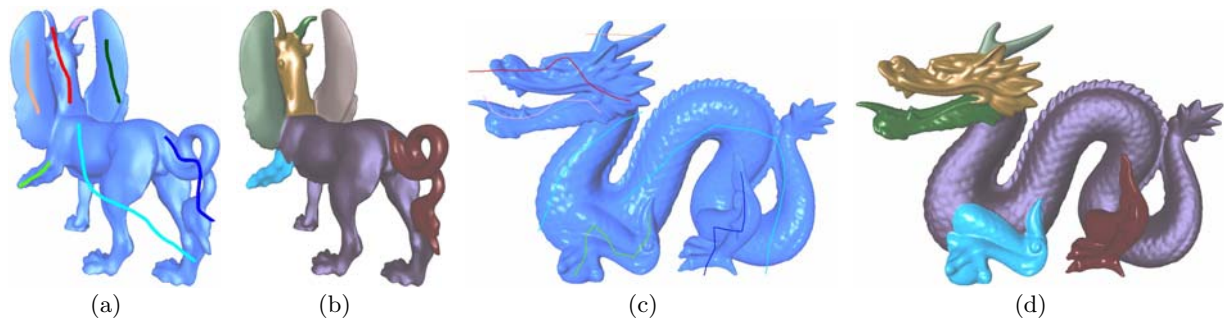


Fig. 7 The segmentation results of animals. The segmentation result of the Feline model is shown in (a),(b). The segmentation result of the Dragon model is shown in (c),(d).

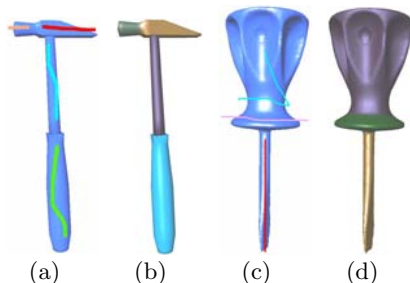


Fig. 8 The segmentation results of CAD models. The segmentation result of the Hammer model is shown in (a),(b). The segmentation result of the Screwdriver model is shown in (c),(d).

statistics (including the boundary refinement time) shown in Table 1, it can be seen that our method is fast enough for the real-time applications. We also test the very huge model such as the Dragon model (871414 facets, 1300K edges, 66.8MB, shown in Fig. 7c), which was computed in 72.562 seconds. Moreover, for large models, running the algorithm first on a simplified model and then using the hierarchical correspondence between the simplified mesh and the original one can significantly reduce the times.

7 Conclusions and future work

To meaningfully segment a model is a challenging task, as human visual perception is extremely complicated and highly subjective. This paper presents a sketch-based interactive framework which segments meshes according to the user's intention, thus gives an effective solution to this problem. Our feature sensitive metrics are independent of the model and part size. We show that they give the clear physical illustration to discrete differential geometric features. Furthermore, the boundary refinement algorithms based on convex hull features and angle features achieve natural segmentation effects. Experiments indicate that our method not only produces meaningful segmentation the user desires, but also improves the user's experience.

As we discussed the value of mesh segmentation, has been recently pronounced in the applications like mesh parameterization, texture mapping, and high-quality metamorphosis. As for cross-parameterization between meshes, it currently suffers from two main problems: the user burden induced by manually specifying many marker points, the slow speed caused by the non-linear procedure. We believe the sketch-based segmentation framework possesses a lot potential in overcoming these drawbacks. We plan to extend this work in the future.

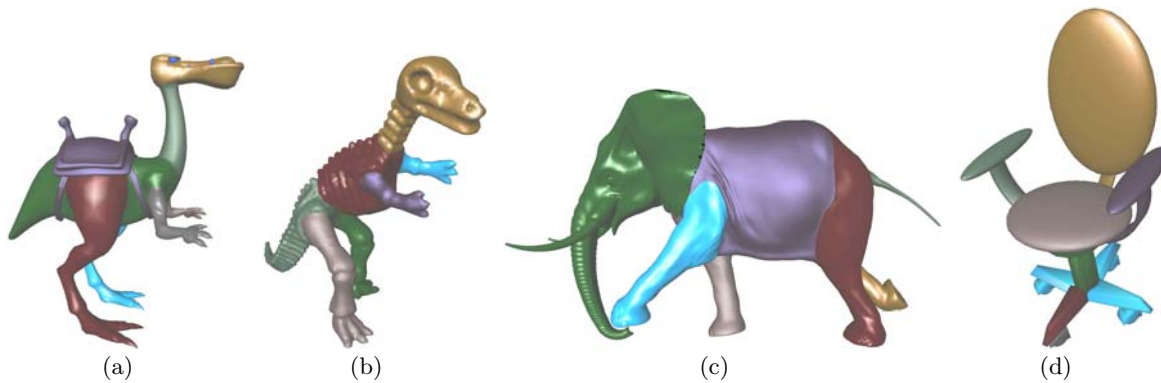


Fig. 9 More segmentation results produced by our method. (a),(b),(c) and (d) are the segmentation results of the Dinopet model, the Dinosaur model, the Elephant model and the Officechair model, respectively.

Mesh	Facet Number	Running time (seconds)
Lion	9996	0.2
Dinopet	15945	0.313
Camel	19536	0.5
Feline	19790	0.562
Dinosaur	28096	0.825
Elephant	84638	2.713

Table 1 Typical timing statistics (in seconds) of our method

References

- Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. In: ACM Transactions on Graphics, **23(3)**, 652–663 (2004)
- Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: ACM Transactions on Graphics, **21(3)**, 362–371 (2002)
- Kraevoy, V., Sheffer, A.: Cross-parameterization compatible remeshing of 3D models. In: ACM SIGGRAPH, **23(3)**, 861–869 (2004)
- Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. In: Computer Graphics Forum, **21(3)**, 219–228 (2002)
- Zuckerberger, E., Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications. In: Computers Graphics, **26(5)**, 733–743 (2002)
- Lien, J.-M., Amato, N. M.: Approximate convex decomposition of polygons. In: Computational Geometry: Theory Applications, **35(1-2)**, 100–123 (2006)
- Li, Y., Sun, J., Tang, C.-K., Shum, H.-Y.: Lazy snapping. In: ACM Transactions on Graphics, **23(3)**, 303–308 (2004)
- Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering cuts. In: ACM Transactions on Graphics, **22(3)**, 954–961 (2003)
- Mangan, A. P., Whitaker, R. T.: Partitioning 3D surface meshes using watershed segmentation. In: IEEE Transactions on Visualization and Computer Graphics, 308–321 (1999)
- Mortara, M., Patan, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives bodies. In: ACM Symposium Solid Modeling Applications, 339–344 (2004)
- Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.-P.: Mesh scissoring with minima rule part salience. In: Computer Aided Geometric Design, **22(5)**, 444–465 (2005)
- Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. In: The Visual Computer, **21(8-10)**, 649–658 (2005)
- Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. In: The Visual Computer, **22(3)**, 181–193 (2006)
- Ji, Z., Liu, L., Chen, Z., Wang, G.: Easy mesh cutting. In: Eurographics, **25(3)**, 283–291 (2006)
- Kraevoy, V., Julius, D., Sheffer, A.: Shuffler: Modeling with interchangeable parts. In: Technical Report, TR-2006-09, University of British Columbia (2006)
- Biederman, I.: Recognition by components. In: Psychological Review, **94**, 115–147 (1987)
- Hoffman, D. D., Richards, W. A.: Parts of recognition. In: Cognition, **18**, 65–96 (1984)
- Hoffman, D. D., Signh, M.: Saliency of visual parts. In: Cognition, **63**, 29–78 (1997)
- Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for multi-scale analysis decomposition of triangle meshes. In: Algorithmica, **38(1)**, 227–248 (2003)
- Shamir, A.: A formulation of boundary mesh segmentation. In: International Symposium on 3D Data Processing, Visualization and Transmission, 82–89 (2004)
- Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh segmentation - a comparative study. In: Shape Modeling International, (2006)
- Pottmann, H., Steiner, T., Hofer, M., Haider, C., Hainbury, A.: The isophotic metric its application to feature sensitive morphology on surfaces. In: ECCV, 560–572 (2004)
- Zelevnik, R., Herndon, K., Hughes, J.: Sketch: An interface for sketching 3d scenes. In: ACM SIGGRAPH, 163–170 (1996)
- Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: ACM SIGGRAPH, 409–416 (1999)
- Kho, Y., Garland, M.: Sketching mesh deformations. In: ACM Symposium Interactive 3D Games and Graphics, 147–154 (2005)
- Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D.: A sketch-based interface for detail-preserving mesh editing. In: ACM SIGGRAPH, **24(3)**, 1142–1147 (2005)
- Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In: ICCV, 902–907 (1995)
- Meyer, M., Desbrun, M., Schr, P., Barr, A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and Mathematics III, 35–57 (2003)