Computational Visual Media
https://doi.org/10.1007/s41095-018-0107-y Vol. 4, No. 2, June 2018, 113–122
Research Article

Surface remeshing with robust user-guided segmentation

Dawar Khan[1,2], Dong-Ming Yan[1,2](✉), Fan Ding[1], Yixin Zhuang[3], and Xiaopeng Zhang[1,2]

Abstract Surface remeshing is widely required in modeling, animation, simulation, and many other computer graphics applications. Improving the elements' quality is a challenging task in surface remeshing. Existing methods often fail to efficiently remove poor-quality elements especially in regions with sharp features. In this paper, we propose and use a robust segmentation method followed by remeshing the segmented mesh. Mesh segmentation is initiated using an existing Live-wire interaction approach and is further refined using local mesh operations. The refined segmented mesh is finally sent to the remeshing pipeline, in which each mesh segment is remeshed independently. An experimental study compares our mesh segmentation method as well as remeshing results with representative existing methods. We demonstrate that the proposed segmentation method is robust and suitable for remeshing.

Keywords mesh generation; mesh segmentation; surface remeshing; triangulation

1 Introduction

In computer graphics, surface meshes are typically used for shape representation. However, these meshes are frequently generated in raw form, and as a result contain poor-quality elements. Furthermore, meshes generated, e.g., from the output of automated laser scanning, are prone to errors. Such raw meshes are difficult to use directly in downstream applications. Thus, remeshing is useful at this stage to improve mesh quality [1].

A critical target for surface remeshing is feature preservation. Feature analysis and identification remain challenging problems because a rigorous definition of features for general objects is lacking despite extensive studies on these topics. In many remeshing algorithms, a user-given feature skeleton is required as an input for use in feature preservation [2, 3]. Several approaches include efficient feature functions for implicit feature preservation [4, 5]. These approaches can efficiently handle models, such as CAD models and man-made objects with clearly defined features, or models without minimal local features. Standard approaches still cannot automatically handle models with thin and sharp features.

Input meshes are typically segmented before remeshing to address the problem caused by thin and sharp features [2, 6]. Segmentation boundaries should split such thin and sharp regions into patches. Then, each patch is remeshed independently and they are finally stitched together. Two main problems, segmentation and stitching, must be solved in this type of approach. Automatic algorithms perform segmentation by grouping triangles, and the segmentation boundaries are defined by original edges of the input. Such boundaries are irregular if the input mesh quality is low, also affecting the final stitched result.

This study aims to improve the segmentation and remeshing outputs yet with minimal user input. The proposed method operates in two phases, mesh segmentation and surface remeshing. The segmentation method is based on active user interaction. The input mesh is segmented according

1 National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. E-mail: D. Khan, dawar@ia.ac.cn; D.-M. Yan, yandongming@gmail.com (✉); F. Ding, Steven_Ding@protonmail.com; X. Zhang, xiaopeng.zhang@ia.ac.cn.
2 University of Chinese Academy of Sciences, Beijing, 100049, China.
3 National Digital Switching System Engineering & Technological Research Center, China. E-mail: yixin.zhuang@gmail.com.
Manuscript received: 2017-12-25; accepted: 2017-12-31

to user-guided segmentation boundaries or curves. The segmentation boundaries are drawn by the user using Live-wire [7], which is especially useful for sharp-featured models. Live-wire is only used for drawing boundaries, and the mesh is segmented using the proposed method. The triangles touching the user-guided segmentation curves are processed with basic operations, including vertex relocation, edge flipping, edge splitting, edge collapsing, and face labeling. These operations are iterated until a robust segmentation is reached. Constraints are applied to avoid small triangles and prevent mesh structure destruction. Thus, we achieve a segmented mesh with only minor changes to the complexity, quality, and structure of the input mesh. The main contributions of this study are as follows:

- a robust segmentation mechanism that divides an input mesh following user-guided segmentation boundaries;
- a method of producing a segmented mesh with minor (negligible) changes in complexity and structure to the input mesh, which does not introduce small angles near the segmentation curves, thus providing a meaningful and more suitable segmentation for surface remeshing;
- a segment-based surface remeshing method with additional local region operators, which can generate a high-quality mesh.

2 Related work

The literature provides numerous surface remeshing methods. For example, representative works include mesh simplification-based methods [8, 9], advancing-front-based method [10], Delaunay insertion methods [11], field-based approaches [12–14], and mesh optimization with either local operations [15–18] or global energy minimization. Global optimization approaches can be further classified as parametrization-based methods [2, 19, 20], discrete clustering methods [4], and direct 3D optimization methods [3, 21–27]. In this section, we briefly review those remeshing methods most closely related to our proposed method, focusing on feature preservation. Alliez et al. [1] present a detailed study on surface remeshing.

The simplest approach to preserving features during remeshing is to predefine feature curves, either by the user or by automatic algorithms (e.g., using dihedral angles) [3, 28]. Such a scheme functions well for models with sharp features, such as CAD models or man-made objects. However, this scheme cannot be applied naturally to free-form objects. Various solutions (e.g., feature-sensitive remeshing [29], implicit feature preservation [4, 5]) have been proposed to preserve features for general objects, especially for models with thin and sharp features. However, these solutions do not always successfully handle thin and sharp features, such as the ear of the Elk model.

A consequent remedy is to apply mesh segmentation prior to remeshing. Segmentation-based remeshing methods can be classified into two main types as follows. One type first defines a coarse mesh (or base mesh) over the input mesh, through mesh simplification. Then, the base mesh is mapped back to the original mesh and further subdivided to form a semi-regular output mesh [30]. For example, Lee et al. [31] present a unified subdivision framework to approximate an arbitrary surface by a displaced subdivision surface. This scheme is simple but efficient for evaluating surface properties. However, this method may lose sharp features and suffer from distortion at times. Mansouri and Ebrahimnezhad [32] recently present an alternative curvature-adapted subdivision method, which achieves better results with lower distortion error and higher aspect ratios (AR). However, semi-regular remeshing cannot arbitrarily modify the mesh connectivity, which constantly causes distortion in highly curved regions.

The other type of approach first segments the input mesh into patches. Then, each patch is remeshed individually, and all the patches are finally stitched together in a post-processing step. Edwards et al. [6] use variational shape approximation [33] for segmentation and centroidal Voronoi tessellation [3] for remeshing. IsoChart [34], Exoskeleton [35], Live-wire [7], and patch layout [36] can also be used to define the feature skeletons of input meshes.

However, the segmentation boundaries are not sufficiently smooth, especially for inputs with thin and poor-quality triangles, because most segmentation algorithms use triangles as primitives for clustering. Such boundaries lead to low triangle quality in the output mesh. A survey paper [37] provides additional

details concerning mesh segmentation. In the present study, Live-wire [7] is used for initial segmentation due to its anisotropic nature that automatically captures thin and sharp features. This is followed by a refinement step to straighten the segmentation boundary to improve the remeshing quality after stitching.

3 Method

3.1 Overview

The pipeline of the proposed algorithm is illustrated in Fig. 1. The input mesh is provided to Live-wire [7]; the segmentation curve is generated by user interaction. This segmentation curve along with the input mesh is further processed in the refinement phase. The refinement phase achieves an acceptably segmented mesh that is provided to the remeshing process. The remeshing method generates a high-quality mesh after applying several segment-based and global operations to the mesh. The two main steps are further described in the following subsections.

3.2 Mesh segmentation

Our segmentation method starts with Live-wire [7] initialization. Live-wire is an efficient technique for curve drawing and mesh segmentation, especially for models with thin and sharp features, such as the lion's or dog's ears or the feline's wings. However, it creates poor-quality elements (with short edges and small angles) near the segmentation curve.

We are not concerned with the segmented mesh produced by Live-wire. Instead, the segmentation curve is simply plotted over the input mesh. This curve leads to a similar segmentation to that provided by Live-wire for faces in the interiors of the segments. However, faces near the segmentation curve remain unlabeled—see the blue faces in Fig. 2, which provides an abstract view of the mesh segmentation for a simple example. The leftmost and rightmost sub-figures show the segmented meshes produced by Live-wire and our method respectively. The central sub-figure shows the Live-wire curve plotted in red over the input mesh.

Faces in the interior of the two segments are labeled in a manner similar to the result produced by Live-wire. Faces near the segmentation curve are left unlabeled (blue faces). At this stage, the curve is only a visualization and is not connected to the vertices. The unlabeled faces and their corresponding vertices and edges are processed using basic operations including vertex translation, edge flipping, edge splitting, edge collapsing, and face labeling (see Fig. 1). We now briefly describe each of these operations.

3.2.1 Vertex translation

In the first step, we move the nearest vertices of the unlabeled faces to the segmentation curve. A vertex v_i is translatable to the nearest point p on the curve if and only if satisfies the following conditions:

- Vertex v_i is nearest to p among the vertices in its one-ring neighborhood.
- The distance between vertex v_i and point p is shorter than 40% of the shortest edge length in the one-ring neighborhood edges of unlabeled faces of v_i. Edges of labeled faces are excluded

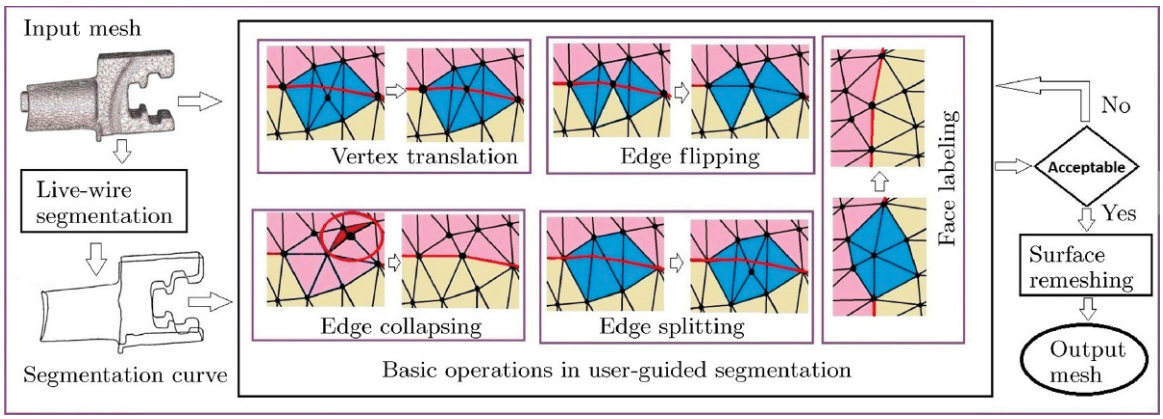


Fig. 1 Proposed user-guided segmentation with remeshing. In the basic operations (middle), red lines represent segmentation curves and the blue color shows unlabeled faces near the curve.

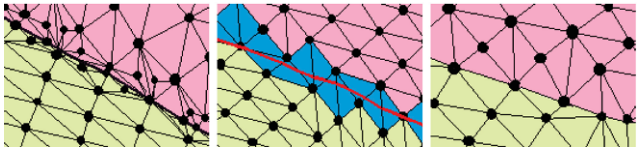


Fig. 2 Simple example of mesh segmentation. Left: Live-wire segmentation includes small triangles near the segmentation boundary. Middle: Live-wire curve (red) plotted over the input mesh; blue triangles are unlabeled. Right: Our final segmentation.

from consideration. This condition helps to avoid distortion in the mesh structure and generation of short edges.

- Vertex v_i does not result in small angles (e.g., $< 20°$) when translated to p.

Vertices that satisfy these conditions are moved toward the curve, and the affected unlabeled faces are labeled according to their neighborhoods.

3.2.2 Face labeling

Face labels determine the segment to which the face belongs. Face labeling is performed in parallel with all four other operations. Our main goal is to label the unlabeled faces in agreement with faces on either side of the curve. A face that lies completely on one side of the curve is labeled with the label of the faces on that side and the face is counted as part of that segment.

3.2.3 Edge flipping

Edges that cross the segmentation curve are flipped to join the curve. An edge is flipable if its vertices lie on opposite sides of the curve (see Fig. 3(left)). The two faces are labeled accordingly after edge flipping (see Fig. 3(right)). Flipable edges are typically found after vertex translation and edge splitting. Edge flipping is executed whenever a flipable edge is found.

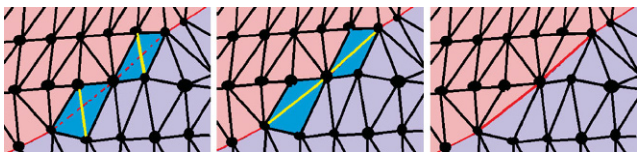


Fig. 3 Edge flipping. Red: Segmentation curve. Left: Before flipping. Yellow edges are flipable. Centre: After flipping. Right: Face labeling after edge flipping.

3.2.4 Edge splitting

The previous steps do not suffice to label all faces as the segmentation curve may cross several non-flipable edges. Edge splitting is used to address this problem. Edge splitting may either be simple, as in Fig. 4, or as complex, as in Fig. 5. In either case, every second edge is split instead of splitting all edges. In particular, only one edge split is allowed for a single triangle. The remaining edges are treated via flipping and vertex translation. In edge splitting, the edge is split (see Fig. 4(b), Fig. 5(b)) and the resultant new vertex is moved to the nearest point on the curve (see Fig. 5(c)). Edge flipping (Fig. 4(c), Fig. 5(d)) and face labeling (Fig. 4(d), Fig. 5(e)) are consequently applied.

3.2.5 Edge collapsing

Edge collapse may result in short edges near the curve. If small angles are produced ($< 20°$), the opposite edge is collapsed. Edge collapse is usually executed only once in the last steps of segmentation. If several edges are collapsed, then the previous steps are repeated (at least once) and the necessary operations are performed.

3.2.6 Curve smoothing

Finally, a curve smoothing operation, which attempts to smooth the curve wherever required, is performed after all the faces have been labeled.

3.3 Surface remeshing

The next objective is to remesh the surface with a robust segmented mesh to improve quality. Our remeshing framework functions in two phases. The first phase performs segmentation-based remeshing, while we apply global operations in the second

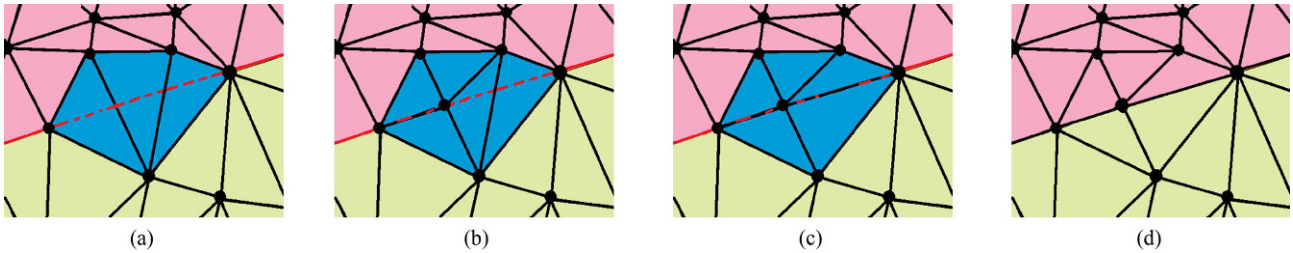


Fig. 4 Edge splitting (Case I). (a) Before splitting. (b) After splitting. (c) Edge flip. (d) Face labeling.

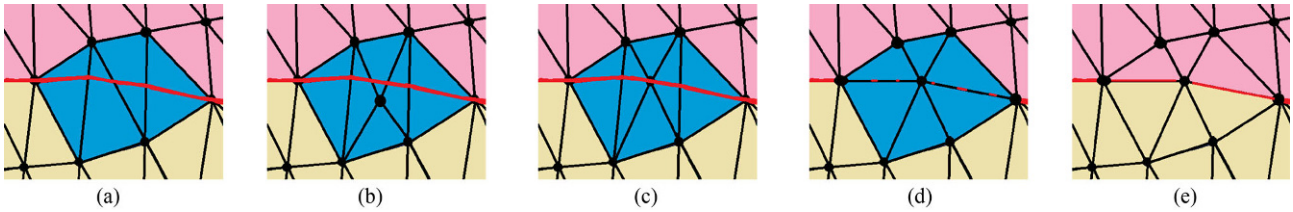


Fig. 5 Edge splitting and consecutive operations (Case II). (a) Before splitting. (b) After splitting the middle edge. (c) New vertex translated to the curve. (d) Edges flipped. (e) Face labeling.

phase and disregard the segmentation boundaries. In segmentation-based remeshing, the edges and vertices on the segmentation curve are locked, and each patch is segmented using an existing method: we use real-time adaptive remeshing (RAR) [17], selected as it is comparatively easy to control, simple to implement, and computationally efficient. RAR uses an adaptive sizing function $L(x)$ to compute the edge length L for each edge. Any edge shorter than $4L/5$ is collapsed; any edge longer than $4L/3$ is split. The two operations (edge collapsing and edge splitting) along with edge flipping for valance optimization and vertex relocation are repeated; 5–10 times are used in the original RAR method.

Initially, the mesh quality is improved without destroying sharp features when the vertices on the curve are locked. The first phase of segment-based RAR execution is repeated 10 times to generate an intermediate mesh. However, this mesh still has small angles and low-quality elements, so it is further processed in the second phase. The vertices on the segmentation boundaries are now unlocked, and further smoothing operations are applied. Each triangle with an angle $< 30°$ is flagged as a bad triangle, and vertices of this triangle and in the one-ring neighborhood of each of its vertices are flagged as bad vertices; these vertices form a local region around the bad triangle. In each local region, the bad triangle is treated with edge-based operations [5], while the vertices in the local regions are also relocated for quality improvement. We calculate the new position p_i as the Laplacian center c_i of the one ring around v_i when relocating a vertex v_i. The vertex is relocated to p_i if it does not lead to bad angles. Otherwise, the process is repeated with a new value for p_i set to $p_i = c_i + k \cdot \Delta d$, where d is a small distance calculated as $d = 1, 1/2, 1/4, 1/6, \ldots, \epsilon$ for a tiny value ϵ, while k represents the direction (left, right, up, down, etc.) of vertex movement around c_i. Thus, the optimal position for vertex translation near c_i is achieved. These operations are executed until all small angles are removed and a mesh with high-quality results.

4 Experiments

4.1 Methodology

In this section, we present experiments performed to evaluate the proposed method. We compare our results with those of the other most relevant segmentation-based methods. We performed the experiments using an Intel Core i7 at 3.60 GHz with 16 GB RAM and 64-bit Windows 7 operating system.

In the following subsections, we measured the remeshing quality in terms of $Q_{\min}$ and Q_{avg}, the minimal and average triangle quality respectively. For a triangle t, the quality $Q(t)$ is defined as

$$Q(t) = \frac{6}{\sqrt{3}} \frac{A_t}{p_t h_t}$$

where A_t is the area of triangle t, p_t is its half-perimeter, and h_t is the length of its longest edge [38].

Similarly, $\theta_{\min}$ and $\theta_{\max}$ are the minimum and maximum angles in the mesh, respectively, while $\overline{\theta}_{\min}$ represents the average of the minimum angle for each triangle. In addition, we calculated the proportion of triangles with small angle ($< 30°$).

4.2 Segmentation results

We compared our segmentation results with those of Live-wire segmentation [7] by segmenting five mesh models with both methods. Figure 6 illustrates the input and segmented meshes. It shows that no small angles exist near the segmentation curve in our results, while Live-wire results do not have this property. Table 1 summarizes quantitative results. We first give the number of vertices and other mesh quality parameters for the input mesh, and then the same parameters are recorded for the Live-wire mesh and our output mesh. Our segmentation has a comparatively minor change in the number of vertices and other mesh quality parameters.

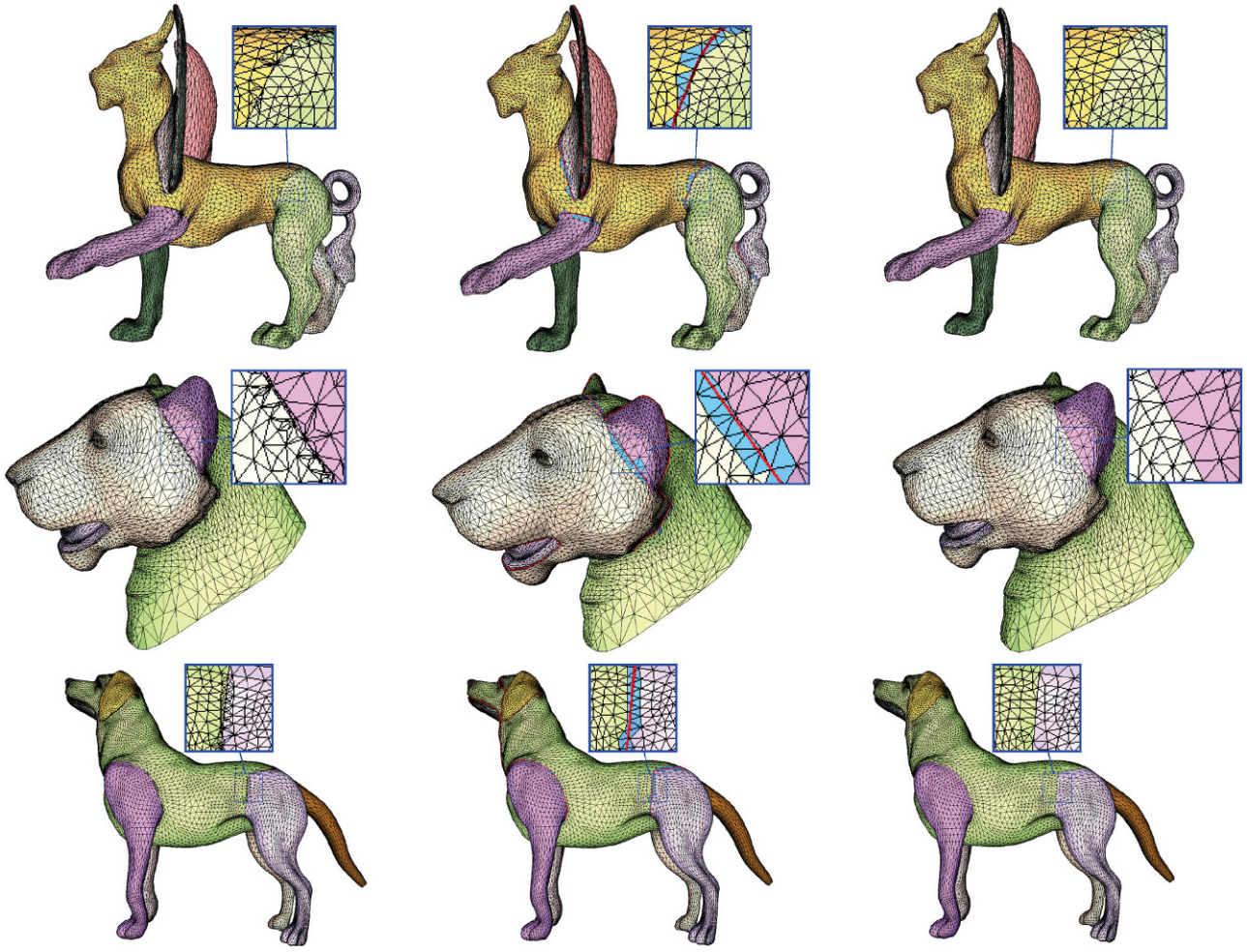


Fig. 6 Segmentation results. Left: Live-wire segmentation. A number of small triangles are created near the curve. Middle: Initialization for our method: Live-wire curve plotted over the input mesh. Blue faces are unlabeled. Right: Our segmentation. These faces become labeled with a minor change in mesh structure.

Table 1 Quantitative results for mesh segmentation and comparison with Live-wire [7]. Our method creates fewer additional faces with minimal change to the input mesh quality. #v represents total number of vertices and regular v's shows percentage of regular vertices

Model	Method	#v	$Q_{\min}$	Q_{avg}	$\theta_{\min}$	$\overline{\theta}_{\min}$	$\theta_{\max}$	$\theta < 30°$	Regular v's
Feline	Input	9998	0.0849	0.6952	3.32	37.0	168.6	26.1%	95.4%
	Live-wire	11045	0.0010	0.6634	0.04	35.2	179.7	31.4%	88.0%
	Ours	10007	0.0348	0.6930	1.20	37.0	172.0	26.4%	95.2%
Lion head	Input	8356	0.1118	0.5985	4.46	30.4	164.7	48.6%	90.6%
	Live-wire	9322	0.0005	0.5698	0.02	28.9	179.6	52.1%	82.8%
	Ours	8365	0.0041	0.5969	0.14	30.3	172.5	48.6%	90.5%
Dog	Input	18114	0.0078	0.6896	0.28	35.3	178.7	34.2%	98.8%
	Live-wire	20287	0.0022	0.6540	0.07	35.9	179.0	39.2%	89.4%
	Ours	18123	0.0078	0.6850	0.28	35.1	178.7	34.6%	98.6%

4.3 Remeshing results

We conducted further experiments by remeshing several models with our own method and the RAR method [17]. Both methods were used for uniform and adaptive remeshing. Mesh quality values were recorded for each experiment. The output meshes along with the histograms of the frequency distributions of the aspect ratios (AR) of the triangles are depicted in Figs. 7 and 8 for uniform and adaptive meshing respectively. Aspect ratio is widely used in the literature as a parameter for measuring triangle quality [39], and is the ratio of the circumradius of a

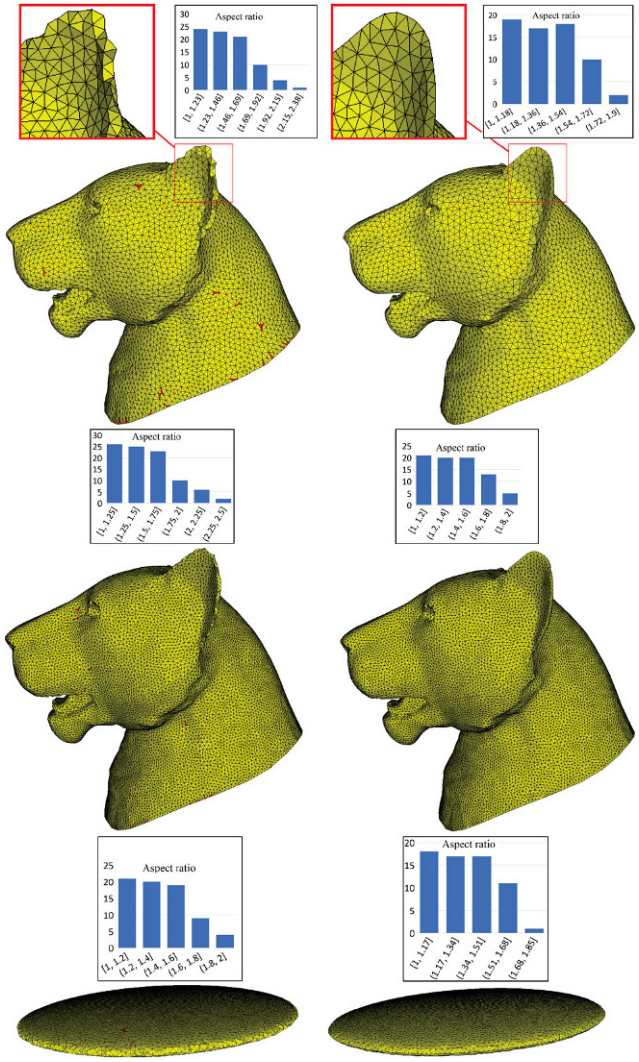


Fig. 7 Remeshing results, uniform density. Triangles with small angle ($< 30°$) are shown in red. Left: RAR results. Right: Our results.

triangle to twice its inradius:

$$AR = \frac{abc}{8(S-a)(S-b)(S-c)}$$

where a, b, and c are the lengths of the triangle's edges and $S = (a+b+c)/2$. The AR for an equilateral triangle is equal to one; a higher AR suggests lower triangle quality. Figures 7 and 8 show that our method significantly improves the ARs of triangles and mesh structure. Models with sharp features such as the lion's ears, in Fig. 7, undergo considerable improvement.

Quantitative results for uniform remeshing are given in Table 2; those for adaptive remeshing are in Table 3. In both cases our results show a significant improvement in mesh quality.

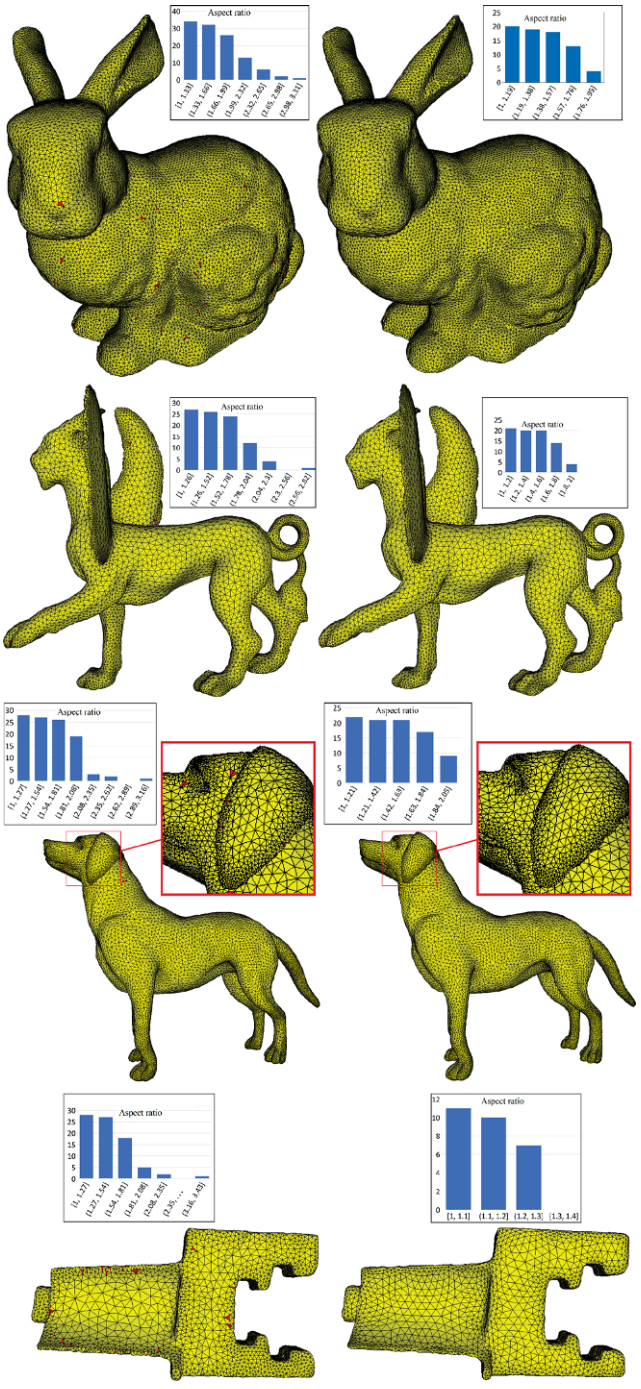


Fig. 8 Remeshing results, adaptive mesh. Triangles with small angle ($< 30°$) are shown in red. Left: RAR results. Right: Our results.

5 Conclusions and future work

We have proposed a segmentation-based remeshing framework that obtains a high-quality mesh with good aspect ratios. Our method works in two steps, mesh segmentation and remeshing. The segmentation

Table 2 Comparative remeshing results, uniform density

Model	Method	#v	$Q_{\min}$	Q_{avg}	$\theta_{\min}$	$\overline{\theta}_{\min}$	$\theta_{\max}$	$\theta < 30°$	$AR_{\max}$	AR_{avg}
Lion head 1	RAR	8356	0.4390	0.8667	21.61	49.17	119.87	0.29%	2.27	1.07
	Ours	8356	0.5246	0.8703	30.20	49.48	112.46	0%	1.79	1.06
Lion head 2	RAR	20703	0.4070	0.8677	20.31	49.20	122.44	0.15%	2.50	1.06
	Ours	20703	0.4936	0.8744	30.00	49.77	116.35	0%	1.96	1.05
Ell	RAR	19077	0.4858	0.8907	27.26	51.19	116.82	0.03%	1.20	1.05
	Ours	19077	0.5440	0.9001	30.01	51.84	109.70	0%	1.69	1.02

Table 3 Comparative remeshing results, adaptive density

Model	Method	#v	$Q_{\min}$	Q_{avg}	$\theta_{\min}$	$\overline{\theta}_{\min}$	$\theta_{\max}$	$\theta < 30°$	$AR_{\max}$	AR_{avg}
Bunny	RAR	34835	0.3396	0.8675	15.46	48.95	127.46	0.51%	3.17	1.06
	Ours	34835	0.5075	0.8754	30.03	49.60	114.53	0%	1.88	1.05
Feline	RAR	9998	0.3834	0.8595	18.95	48.30	124.68	0.64%	2.75	1.07
	Ours	9998	0.4918	0.8672	30.06	48.93	116.57	0%	1.96	1.06
Dog	RAR	18114	0.3755	0.8596	18.33	48.37	130.99	0.40%	3.06	1.07
	Ours	18114	0.4825	0.8731	30.00	49.45	117.73	0%	2.03	1.06
Blade	RAR	5002	0.3630	0.8453	22.39	47.21	132.50	1.01%	3.24	1.09
	Ours	5002	0.6695	0.8841	30.93	50.14	89.96	0%	1.30	1.05

method considers the segmentation curve as a user input. It generates a segmented mesh with no bad elements near the segmentation curve and only minor changes in mesh structure. In future, we will consider parallelizing of the method. We also hope to improve mesh quality for non-obtuse remeshing.

Acknowledgements

This work was partially funded by the National Natural Science Foundation of China (Nos. 61772523, 61372168, 61620106003, and 61331018). The first author was supported by a Chinese Government Scholarship.

References

[1] Alliez, P.; Ucelli, G.; Gotsman, C.; Attene, M. Recent advances in remeshing of surfaces. In: *Shape Analysis and Structuring. Mathematics and Visualization.* De Floriani, L.; Spagnuolo, M. Eds. Springer, Berlin, Heidelberg, 53–82, 2008.

[2] Alliez, P.; Meyer, M.; Desbrun, M. Interactive geometry remeshing. *ACM Transactions on Graphics* Vol. 21, No. 3, 347–354, 2002.

[3] Yan, D.-M.; Lévy, B.; Liu, Y.; Sun, F.; Wang, W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum* Vol. 28, No. 5, 1445–1454, 2009.

[4] Valette, S.; Chassery, J.-M.; Prost, R. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics* Vol. 14, No. 2, 369–381, 2008.

[5] Hu, K.; Yan, D. M.; Bommes, D.; Alliez, P.; Benes, B. Error-bounded and feature preserving surface remeshing with minimal angle improvement. *IEEE Transactions on Visualization and Computer Graphics* Vol. 23, No. 12, 2560–2573, 2017.

[6] Edwards, J.; Wang, W.; Bajaj, C. L. Surface segmentation for improved remeshing. In: Proceedings of the 21st International Meshing Roundtable, 403–418, 2013.

[7] Zhuang, Y.; Zou, M.; Carr, N.; Ju, T. Anisotropic geodesics for live-wire mesh segmentation. *Computer Graphics Forum* Vol. 33, No. 7, 111–120, 2014.

[8] Heckbert, P. S.; Garland, M. Survey of polygonal surface simplification algorithms. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1997.

[9] Liu, Y.-J.; Xu, C.-X.; Fan, D.; He, Y. Efficient construction and simplification of Delaunay meshes. *ACM Transactions on Graphics* Vol. 34, No. 6, Article No. 174, 2015.

[10] Schreiner, J.; Scheidegger, C. E.; Fleishman, S.; Silva, C. T. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum* Vol. 25, No. 3, 527–536, 2006.

[11] Cheng, S.-W.; Dey, T. K.; Shewchuk, J. R. *Delaunay Mesh Generation.* CRC Press, 2012.

[12] Lai, Y.-K.; Jin, M.; Xie, X.; He, Y.; Palacios, J.; Zhang, E.; Hu, S.-M.; Gu, X. Metric-driven RoSy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics* Vol. 16, No. 1, 95–108, 2010.

[13] Nieser, M.; Palacios, J.; Polthier, K.; Zhang, E. Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics* Vol. 18, No. 6, 865–878, 2012.

[14] Jakob, W.; Tarini, M.; Panozzo, D.; Sorkine-Hornung, O. Instant field-aligned meshes. *ACM Transactions on Graphics* Vol. 34, No. 6, Article No. 189, 2015.

[15] Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J.; Stuetzle, W. Mesh optimization. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 19–26, 1993.

[16] Botsch, M.; Kobbelt, L. A remeshing approach to multiresolution modeling. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 185–192, 2004.

[17] Dunyach, M.; Vanderhaeghe, D.; Barthe, L.; Botsch, M. Adaptive remeshing for real-time mesh deformation. In: Proceedings of the Eurographics, 29–32, 2013.

[18] Wang, Y.; Yan, D.-M.; Tang, C.; Liu, X. Obtuse triangle elimination for isotropic remeshing. In: Proceedings of the ACM SIGGRAPH 2017 Posters, Article No. 81, 2017.

[19] Surazhsky, V.; Alliez, P.; Gotsman, C. Isotropic remeshing of surfaces: A local parameterization approach. In: Proceedings of the 12th International Meshing Roundtable, 204–231, 2003.

[20] Marchandise, E.; Remacle, J.-F.; Geuzaine, C. Optimal parametrizations for surface remeshing. *Engineering with Computers* Vol. 30, No. 3, 383–402, 2014.

[21] Fu, Y.; Zhou, B. Direct sampling on surfaces for high quality remeshing. In: Proceedings of the ACM Symposium on Solid and Physical Modeling, 115–124, 2008.

[22] Chen, Z.; Cao, J.; Wang, W. Isotropic surface remeshing using constrained centroidal delaunay mesh. *Computer Graphics Forum* Vol. 31, No. 7, 2077–2085, 2012.

[23] Yan, D.-M.; Bao, G.; Zhang, X.; Wonka, P. Low-resolution remeshing using the localized restricted Voronoi diagram. *IEEE Transactions on Visualization and Computer Graphics* Vol. 20, No. 10, 1418–1427, 2014.

[24] Wang, X.; Ying, X.; Liu, Y.-J.; Xin, S.-Q.; Wang, W.; Gu, X.; Mueller-Wittig, W.; He, Y. Intrinsic computation of centroidal Voronoi tessellation (CVT) on meshes. *Computer-Aided Design* Vol. 58, 51–61, 2015.

[25] Liu, Y.-J.; Xu, C.-X.; Yi, R.; Fan, D.; He, Y. Manifold differential evolution (MDE): A global optimization method for geodesic centroidal Voronoi tessellations on meshes. *ACM Transactions on Graphics* Vol. 35, No. 6, Article No, 243, 2016.

[26] Yan, D.-M.; Wonka, P. Non-obtuse remeshing with centroidal Voronoi tessellation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 22, No. 9, 2136–2144, 2016.

[27] Ahmed, A. G. M.; Guo, J.; Yan, D.-M.; Franceschi, J.-Y.; Zhang, X.; Deussen, O. A simple push–pull algorithm for blue-noise sampling. *IEEE Transactions on Visualization and Computer Graphics* Vol. 23, No. 12, 2496–2508, 2017.

[28] Fuhrmann, S.; Ackermann, J.; Kalbe, T.; Goesele, M. Direct resampling for isotropic surface remeshing. In: *Vision, Modeling, and Visualization.* Koch, R.; Kolb, A.; Rezk-Salama, C. Eds. The Eurographics Association, 9–16, 2010.

[29] Lévy, B.; Liu, Y. L_p centroidal Voronoi tessellation and its applications. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 119, 2010.

[30] Payan, F.; Roudet, C.; Sauvage, B. Semi-regular triangle remeshing: A comprehensive study. *Computer Graphics Forum* Vol. 34, No. 1, 86–102, 2015.

[31] Lee, A.; Moreton, H.; Hoppe, H. Displaced subdivision surfaces. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 85–94, 2000.

[32] Mansouri, S.; Ebrahimnezhad, H. Segmentation-based semi-regular remeshing of 3D models using curvature-adapted subdivision surface fitting. *Journal of Visualization* Vol. 19, No. 1, 141–155, 2016.

[33] Cohen-Steiner, D. ; Alliez, P.; Desbrun, M. Variational shape approximation. *ACM Transactions on Graphics* Vol. 23, No. 3, 905–914, 2004.

[34] Zhou, K.; Snyder, J.; Guo, B.; Shum, H.-Y. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In: Proceeding of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 45–54, 2004.

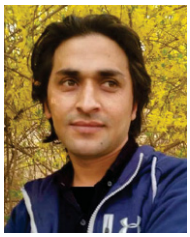
[35] De Goes, F.; Goldenstein, S.; Desbrun, M.; Velho, L. EXOSKELETON: Curve network abstraction for 3D shapes. *Computers & Graphics* Vol. 35, No. 1, 112–121, 2011.

[36] Cao, Y.; Yan, D.-M.; Wonka, P. Patch layout generation by detecting feature networks. *Computers & Graphics* Vol. 46, 275–282, 2015.

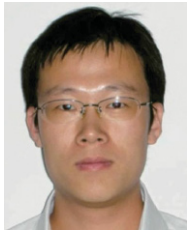
[37] Shamir, A. A survey on mesh segmentation techniques. *Computer Graphics Forum* Vol. 27, No. 6, 1539–1556, 2008.

[38] Frey, P. J.; Borouchaki, H. Surface mesh evaluation. In: Proceedings of the 6th International Meshing Roundtable, 363–374, 1997.

[39] Farin, G. E. Shape measures for triangles. *IEEE Transactions on Visualization and Computer Graphics* Vol. 18, No. 1, 43–46, 2012.



Dawar Khan is a Ph.D. student at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China. He was awarded with a Chinese Government Scholarship for his Ph.D. study. Prior to that, he received his bachelor and master degrees from the Department of Computer Science and IT, University of Malakand, Pakistan, in 2011 and 2014 respectively. His research interests include computer graphics, computational geometry, mesh processing, virtual and augmented reality, and pattern recognition.



Dong-Ming Yan is an associate professor in the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. He received his Ph.D. degree in computer science from Hong Kong University in 2010, and his master and bachelor degrees in computer science and technology from Tsinghua University in 2005 and 2002 respectively. His research interests include computer graphics, geometric processing, and visualization.

Fan Ding received his bachelor degree in software engineering from Hebei University of Technology in 2017. He is currently working as an intern at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. His research interests include computer graphics, mesh processing, and artificial intelligence.

Yixin Zhuang is an assistant researcher in the National Digital Switching System Engineering & Technological Research Center, China. He obtained his B.S. degree from Nanjing University of Aeronautics and Astronautics in 2008, and both M.S. and Ph.D. degrees from the National University of Defense Technology in 2011 and 2015 respectively. His research interests include computer graphics, and geometric modeling and processing.

Xiaopeng Zhang is a professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. He received his Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences, in 1999. He received a National Scientific and Technological Progress Prize (second class) in 2004. His main research interests include computer graphics and image processing.