Computers & Graphics 46 (2015) 275–282

Contents lists available at ScienceDirect

Computers & Graphics

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

SMI 2014

Patch layout generation by detecting feature networks

Yuanhao Cao [a], Dong-Ming Yan [a,b,*], Peter Wonka [a,c]

[a] Visual Computing Center, King Abdullah University of Science and Technology, Saudi Arabia
[b] NLPR-LIAMA, Institute of Automation, Chinese Academy of Sciences, China
[c] Department of Computer Science and Engineering, Arizona State University, USA

ARTICLE INFO

Article history:
Received 29 June 2014
Received in revised form
23 September 2014
Accepted 27 September 2014
Available online 23 October 2014

Keywords:
Curve network
Patch layout
Segmentation
Fitting

ABSTRACT

The patch layout of 3D surfaces reveals the high-level geometric and topological structures. In this paper, we study the patch layout computation by detecting and enclosing feature loops on surfaces. We present a hybrid framework which combines several key ingredients, including feature detection, feature filtering, feature curve extension, patch subdivision and boundary smoothing. Our framework is able to compute patch layouts through concave features as previous approaches, but also able to generate nice layouts through smoothing regions. We demonstrate the effectiveness of our framework by comparing with the state-of-the-art methods.

1. Introduction

The patch layout of 3D surfaces encodes the underlying high-level geometric and topological structure. Various criteria have been used for defining specific patch layouts. For example, the geometric criteria include planarity, developability, conformality, and concavity, while the topological criteria include the suitability for quad meshing, triangle meshing, or subdivision surfaces.

Many computer graphics applications require a patch layout of a 3D model, e.g., texture mapping [1], paper-craft design [2], reverse engineering [3], model simplification and shape abstraction [4]. In many cases, multiple criteria are required for a good patch layout, e.g., in an architecture design, both planarity and suitability for quad remeshing are required to generate *Planar Quad Meshes* (PQMesh) that are useful for construction [5].

The problem of patch layout generation has been extensively studied during the last few decades. Most of the existing work focuses on defining patch boundaries along the concave/sharp features of the surfaces. However, there are many other types of features available on the surfaces that are seldom been used for patch layout computation, such as ridges and valleys [6]. The reason is that most of the ridge and valley curves are discretely distributed on the surfaces, and it is difficult to connect then to form a valid feature network.

In this paper, we propose a framework for patch layout generation from the ridges and valleys on surfaces. Our approach combines several key ingredients, including feature detection, feature curve extension, boundary smoothing and patch subdivision. Fig. 1 shows an example of our algorithm. Unlike previous methods, our approach is able to generate patch boundaries on sharp features as well as smooth regions on the surfaces. In the results section we will compare the approximation quality with existing state-of-the-art approaches by triangulating the patch networks. The main contributions of this paper include:

- A hybrid framework that can extract patch layouts that are better than the state-of-the-art.
- A novel criterion to evaluate the quality of patch layouts by measuring the reconstruction error between input and reconstructed models.

2. Related work

The problem of patch layout generation is closely related to parameterization, feature detection, mesh segmentation, multi-resolution modeling and quadrilateral mesh generation. We briefly review the most related work in the following.

Feature extraction: An exhaustive survey of the feature extraction techniques is out of the scope of this paper (see [7] and references therein), we only focus on the methods that are most related to our work. Lee et al. [8] generalize the 2D active contour model (snakes) to 3D mesh surfaces. The user sketches an initial feature curve on the input surface, and the 3D snake iteratively snaps to the features. Lee et al. [9] further apply the 3D snake method for mesh segmentation. Our work is similar to [9], but our curve network computation does not only rely on the concave features. Ohtake et al. [6] construct an implicit surface $F(x)=0$ to

* Corresponding author.
E-mail addresses: yuanhao.cao@kaust.edu.sa (Y. Cao), yandongming@gmail.com (D.-M. Yan), pwonka@gmail.com (P. Wonka).

http://dx.doi.org/10.1016/j.cag.2014.09.022
0097-8493/

approximate the set of the mesh vertices and normals, and then use the implicit surface fitting method to calculate extremal coefficients for ridges and valleys detection. Kim and Kim [10] use a modified *Moving-Least-Squares* (MLS) approximation method to generate a local fitting surface around each vertex, and then use this fitted surface to calculate the local curvatures. Hildebrandt et al. [11] propose a new computation scheme based on discrete differential geometry which avoids costly computations of higher order approximating surfaces. In the post-processing, they use a Laplacian smoothing to smooth the extremality coefficients on a surface to improve the stability of the extraction of feature lines and the smoothness of their appearance. Nomura and Hamada [12] extract the feature curves by calculating the skeleton of the feature region defined by the concavity and convexity.

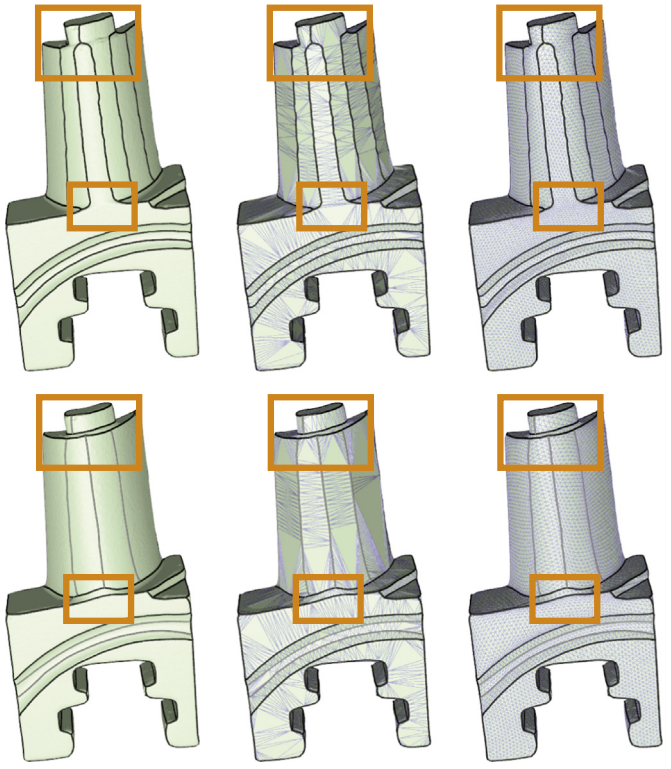


Fig. 1. Patch layouts (top) generated by VSA and our method (bottom). The black lines are the feature curves computed by the first step of our framework, and the gray lines are the inner boundaries after applying VSA to patches. The curve networks are then triangulated (middle) or filled by N-sided patches (right) for quality evaluation. Our framework generates better results when the number of patches is small. The problematic regions where VSA cannot generate feasible segmentations are highlighted, while our method computes better layouts.

Segmentation/layout generation: There are two main types of surface segmentation methods, patch-based and part-based [13]. The patch-based approach shares many similarities with patch layout generation. Lévy et al. [1] chartificate the surface into texture-atlas under the *Least-Square Conformal Mapping* constraints. Cohen-Steiner et al. [14] proposed a powerful variational approach for computing planar patch layouts of surfaces, called *Variational Shape Approximation* (VSA). Instead of minimizing the L^2 distance, they minimize the $L^{2,1}$ metric which considers the normal approximation of each patch. Wu and Kobbelt [15] and Yan et al. [16,17] extend the VSA framework to spherical/cylindrical patches and quadric patches, respectively. Varady et al. [3] propose a region growing based method for reverse engineering of CAD surfaces.

The recent work of Chen et al. [18] presents a framework to evaluate different mesh segmentation techniques, and they also build a benchmark for surface mesh segmentation, called *Princeton Segmentation Benchmark* (PSB), which is widely used by the subsequent research. de Gose et al. [19] introduce a new type of surface feature skeleton for surface abstraction. The segmented meshes of PSB are used as input, and then VSA is then applied for each part. However, this approach only works well for models with concave features, it fails to generate a patch layout for general surfaces. The reader is referred to the comprehensive survey paper by Shamir [13] for more details of mesh segmentation.

Another branch of patch layout generation emphasizes quadrilateral patches computation [20–24]. We refer to the recent survey [25] for details.

Surfaces from curves: Given the curve network of a surface, it is a challenging task to recover the geometry of the original surface. Orbay and Kara [26] propose a sketch-based modeling interface for creating smooth surfaces [26]. Abbasinejad et al. [27] introduce a new algorithm for automatic generation of piecewise-smooth surfaces from curve networks. Bessmeltsev et al. [28] present a design-driven approach for quadrangulating closed 3D curve networks. Zou et al. [29] present an algorithm for triangulating 3D spatial polygons. In this paper, we use the algorithm of [29] to evaluate the quality of the generated patch layouts of different algorithms.

3. Methodology

3.1. Overview

In this paper, we present a hybrid framework for extracting a high quality patch layout from an input triangle mesh surface on

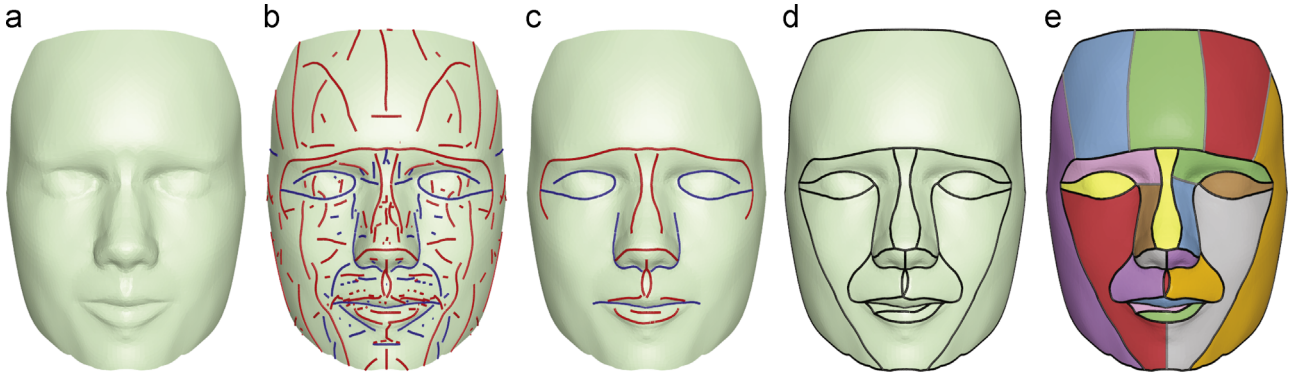


Fig. 2. Pipeline of the presented layout computation framework. (a) Input mesh. (b) Results of feature extraction. (c) Salient feature curves after filtering. (d) Initial patch layout after feature extension. (e) Final patch layout after subdivision and boundary smoothing. Different patches are shown in different colors. The new boundaries introduced by subdivision are shown in gray. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

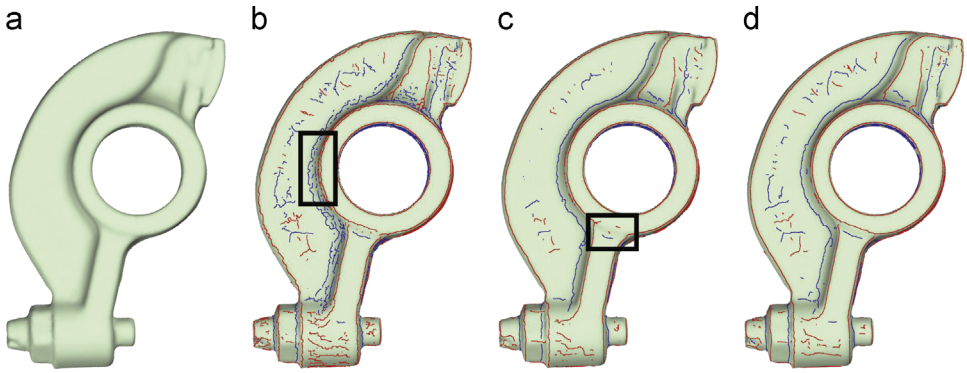


Fig. 3. Ridges and valleys on the surface with different scales. (a) Input mesh. (b) The results are noisy when the scale is small. (c) When the scale is large, some features cannot be captured. (d) All the feature curves are captured using our multi-scale method.

which the triangles are distributed uniformly. We assume that the input triangle mesh is 2-manifold, with or without boundaries, an example is shown in Fig. 2(a).

Our framework consists of five main steps, as shown in Fig. 2. Given an input surface, we first extract the ridges and valleys as the basic feature elements (Section 3.2). Next, we filter out the short and unimportant feature elements based on some intuitive heuristics (Section 3.3). We further extend the remaining feature curves to form a complete feature network (Section 3.4). Once the curve network is generated, we employ variational shape approximation to subdivide certain patches (Section 3.5). The boundary of the curve network is finally smoothed using a local geodesic smoothing operation (Section 3.6). We evaluate the quality of the patch layout by triangulating the curve networks with a recent algorithm [29] and present comparisons (Section 4).

3.2. Feature detection

We start by detecting features on mesh surfaces. In this step, we follow the method of [30] to extract the feature curves. Given a mesh M, the surface at the k-ring neighborhood of a vertex P can be locally fitted by a bivariate cubic polynomial

$$h(x,y) = 1/2(b_0x^2+2b_1xy+b_2y^2) + 1/6(c_0x^3+3c_1x^2y+3c_2xy^2+c_3y^3) \quad (1)$$

in the local coordinate system where the normal direction of vertex P is the Z-direction and the tangent plane at P is the XY-plane. The coefficients of this polynomial $(b_0, b_1, b_2, c_0, c_1, c_2, c_3)$ can be solved by the so-called adjacent-normal cubic approximation method in [31].

Then the extremity coefficients e_{max} and e_{min} defined as the derivatives of the principal curvatures k_{max}, k_{min} along their corresponding principal directions $\mathbf{t}_{max} = (t_1, t_2), \mathbf{t}_{min} = (t_3, t_4)$ can be calculated from (c_0, c_1, c_2, c_3) directly

$$e_{max} = \partial k_{max}/\partial \mathbf{t}_{max} = \begin{pmatrix} t_1^2 \\ t_2^2 \end{pmatrix}^T \begin{pmatrix} c_0 & c_1 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}, \quad (2)$$

$$e_{min} = \partial k_{min}/\partial \mathbf{t}_{min} = \begin{pmatrix} t_3^2 \\ t_4^2 \end{pmatrix}^T \begin{pmatrix} c_0 & c_1 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} t_3 \\ t_4 \end{pmatrix}. \quad (3)$$

Then we use the procedure described in [6] to check whether the mesh edge contains a ridge vertex. If there are two ridge vertices on edges of a triangle, they are connected by a straight line segment. If there are three ridge vertices on edges of a triangle, we connect these three ridge vertices to the centroid of the triangle. The result of this step is shown in Fig. 2(b).

Although the feature extraction described above works well for smooth inputs, it is still difficult to choose a proper scale k to extract

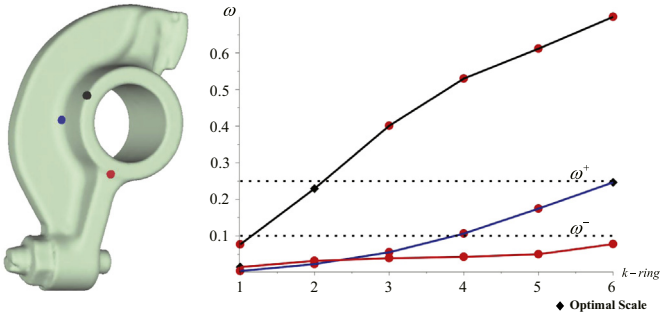


Fig. 4. Scale selection for three representative points. Left: three different vertices on the rocker-arm model; Right: the ω_k of three different vertices and the black diamond symbols indicate the optimal scale k.

all the features on the surface for complicated models. As shown in Fig. 3, using a small scale could generate many disconnected short feature curves, while using a large scale cannot capture the small details. Hence, we propose to use multi-scale feature extraction. First, we define a parameter to measure the surface variation using the normal tensor voting theory [32,33]. The normal voting tensor $T_k(P)$ of vertex P is defined as

$$T_k(P) = \sum_{f_i \in N_k(P)} \mu_{f_i} n_{f_i} n_{f_i}^T, \quad (4)$$

where $N_k(P)$ is the k-ring triangle set of vertex P, and n_{f_i} is the unit normal vector of each triangle f_i. μ_{f_i} is a weight defined by

$$\mu_{f_i} = \frac{A(f_i)}{A_{max}} \exp\left(-\frac{\| c_{f_i} - P \|}{\max \| c_f - P \|}\right), \quad (5)$$

where the $A(f_i)$ is the area of f_i, A_{max} is the maximum area of a triangle in $N_k(P)$, c_{f_i} is the barycenter of f_i and $\max \| c_f - P \|$ is the maximum distance from the barycenter to the vertex P. Then the surface variation of the k-ring neighborhood of vertex P is defined as

$$\omega_k(P) = \frac{\lambda_2 + \lambda_3}{\lambda_1}, \quad (6)$$

where the $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ are the eigenvalues of tensor $T_k(P)$. Then we design a simple way to set the scale k for each vertex by $\omega_k, k = 1 \cdots m$, and two predefined threshold $\omega_- < \omega_+$. If the number of $\omega_k \geq \omega_-$ is less than $m/2$, we consider that the neighborhood of the current vertex is approximately flat and set $k=1$ for this vertex. If the number of $\omega_k \geq \omega_-$ is larger than $m/2$, we consider the neighborhood of the current vertex as a bent area and set k to its corresponding ω_k which is most close to ω_+. For other cases, k is set by the maximum variation of ω_k which is the ratio of ω_k and ω_{k+1}. In our experiment, we set m, ω_- and ω_+ to 6, 0.1 and 0.25

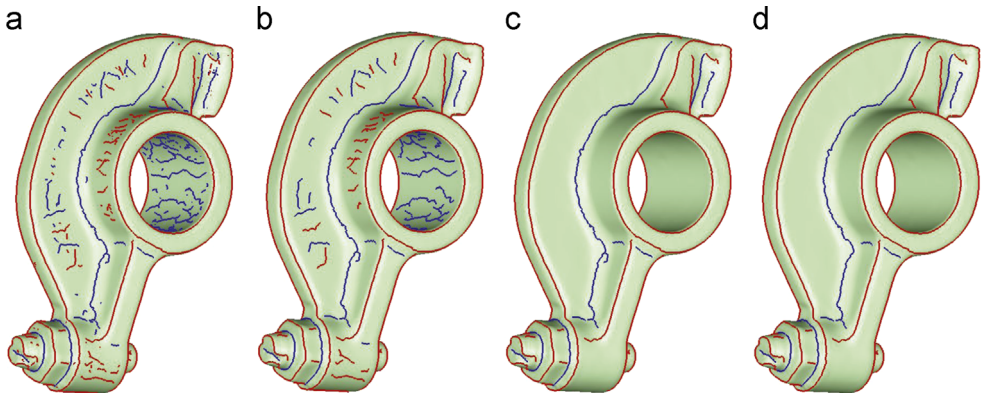


Fig. 5. Illustration of our filter process using L, R and T. (a) All the feature curves on the model. (b) First using $L=0.023$ parameter to filter the feature. (c) Then using the parameter $R=12.158$ to filter feature. (d) Finally using $T=0.252$ to get the salient feature.

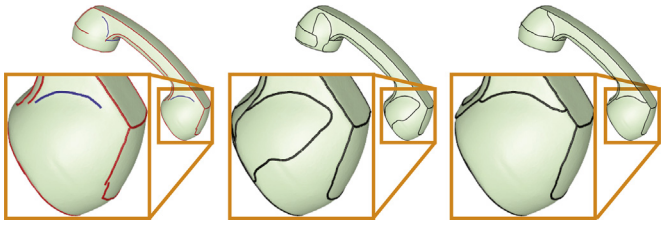


Fig. 6. Illustration of our improved curve extension algorithm. Left: the initial feature curves; middle: the extension result of [34]; and right: our result.

respectively. An example that describes the validity of the scale selection is shown in Fig. 4.

3.3. Feature filtering and connection

The feature curves extracted in previous step usually contain lots of short curves and non-salient curves. In this step, we filter out the curves that we are not interested in. We extend the filtering algorithm presented in [30], where the strength of a feature line is measured by the following scale-independent quantity:

$$T = \int ds \cdot \int \sqrt{|e_{max}|^2 + |e_{min}|^2}\, ds. \tag{7}$$

In our experiments, we found that it is not enough to filter out the non-salient curves based on Eq. (7). We propose to add two more parameters for the filtering. Given a curve represented by a set of vertices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$: the curve length L and the curve curvature R

$$L = \sum_{i=1}^{i=n-1} \|\mathbf{P}_{i+1} - \mathbf{P}_i\|, \quad R = \frac{\sum_{i=1}^{i=n} \mathbf{C}_{RMS}(\mathbf{P}_i)}{n}$$

are considered as additional measurements, where $\mathbf{C}_{RMS}(\mathbf{P}_i)$ is the root mean square curvature of $\mathbf{P}_i$.

We first use the curve length parameter to filter out very short curve segments. For the remaining curve segments, we try to connect two curve segments to generate a longer curve by checking the ending points of these two curves. If two ending points of two curve segments are in 1-ring neighborhoods and the angles $\alpha \le \pi/3, \beta \le \pi/3, \gamma \le \pi/2$, we connect these two ending points to generate a longer curve (as shown in right figure). And then we use the other two parameters T and R to filter the curve segments and only keep the salient feature curves. The result of this step is shown in Fig. 2(c). One example of using parameters L, R, T to filter features is shown in Fig. 5.

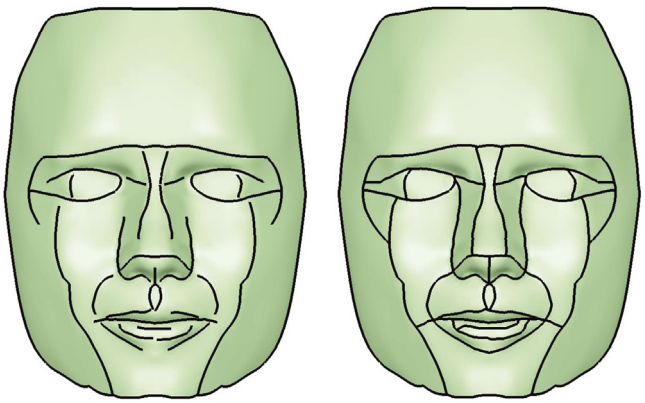
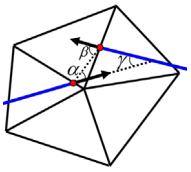


Fig. 7. One feature extension result with more finer feature curves on surface compared to Fig. 2.



3.4. Feature extension

The salient feature curves capture the key parts of the curve network of the input mesh. However, these curves are disconnected and we cannot derive a valid layout directly from them. Therefore, we have to extend these feature curves in the curvature directions to let them meet. We extend the method of Hsu et al. [34] for this purpose. We only describe our improvement in the following.

Assume that a feature curve is represented by a sequence of n ordered points $\{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n\}$. The points lie on the mesh, but they do not have to be vertices of the mesh. To simplify our presentation, we assume that the end points of feature curves to be extended are located on mesh vertices (e.g., we snap them to the nearest vertices). The new vertices to be added to the extended curves will only be selected from the mesh vertices.

We first consider how to extend a curve from its end point $\mathbf{P}_n$ (extending $\mathbf{P}_1$ works in the same way). Let $\{\mathbf{N}_n\}$ denote the 1-ring neighbors of $\mathbf{P}_n$. We first select a set of candidates $\{\mathbf{C}_n\} \subseteq \{\mathbf{N}_n\}$ for the extension. Each vertex $\mathbf{P}_c \in \{\mathbf{C}_n\}$ is selected from $\mathbf{N}_n$ by using a

simple angle filtering criterion such that $\mathbf{P}_c - \mathbf{P}_n$ does not deviate from the curve too much. Next, we calculate a cost value $F(\mathbf{P}_c)$ for each vertex $\mathbf{P}_c \in \{\mathbf{C}_n\}$ to indicate the possibility of $\mathbf{P}_c$ to be selected. The cost function F jointly considers multiple factors: curvature, continuity of principal directions, smoothness. We follow most of these terms of [34], and we have improved one of them in the following.

When computing the directional energy term, the algorithm in [34] only uses the minimal principle direction of the current end point for energy evaluation. This does not work in the regions with non-consistent principle directions, e.g., isotropic regions. We instead evaluate the directional energy by averaging the principle direction of previous m ($m=5$ in our experiments) points on the curve. And we choose the new vertex with the minimal distortion.

Next, we describe how to extend multiple curves at the same time instead of extending only one curve. We collect all the candidates of all curves and evaluate the cost for each candidate. The candidate with the minimal cost is selected to be added to the curve it is connected to. Once a new vertex is selected, the candidate set and the cost values are update. We stop the extension of the ending point of a feature curve if there is any point on the existing feature curves within a threshold (e.g., the one ring neighborhood of this ending point is used in our implementation). Then we fill the gap between the ending point with existing feature points so that the ending of curve is as straight as possible. This step is repeated until the candidate set is empty.

We compare our improved curve extension algorithm with the original approach in Fig. 6 and show another feature extension result with more finer features on the surface compared to Fig. 2 in Fig. 7.

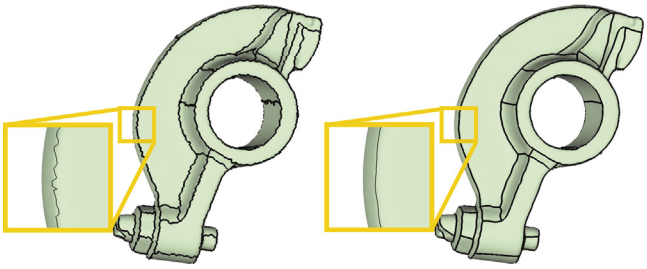


Fig. 8. The final patch layout of rocker-arm model before and after boundary smoothing. Left: the patch layout before smoothing with so many zigzags on boundary curve. Right: the smooth boundary curves of the patch layout after smoothing.

3.5. Patch subdivision

The model has actually been segmented into different patches by the feature curve networks and there is no feature curve that we need in each region. But we still further process each patches with patch subdivision so that each refined patch is approximately flat for the purpose of reconstruction. We make use of VSA algorithm to realize patch subdivision. For each patch, we started the VSA algorithm from two seeds which are selected randomly

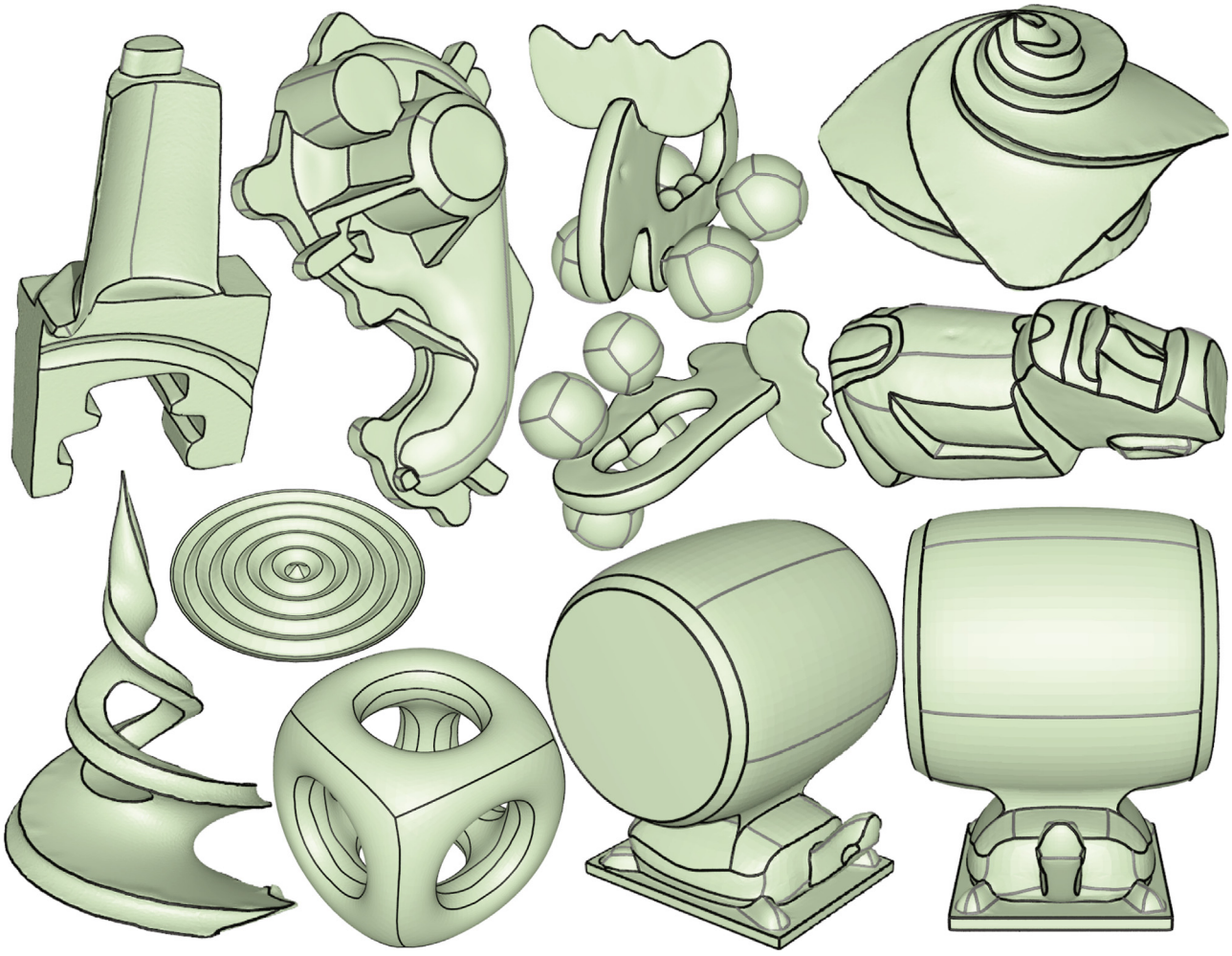


Fig. 9. Feature networks computed by our framework.

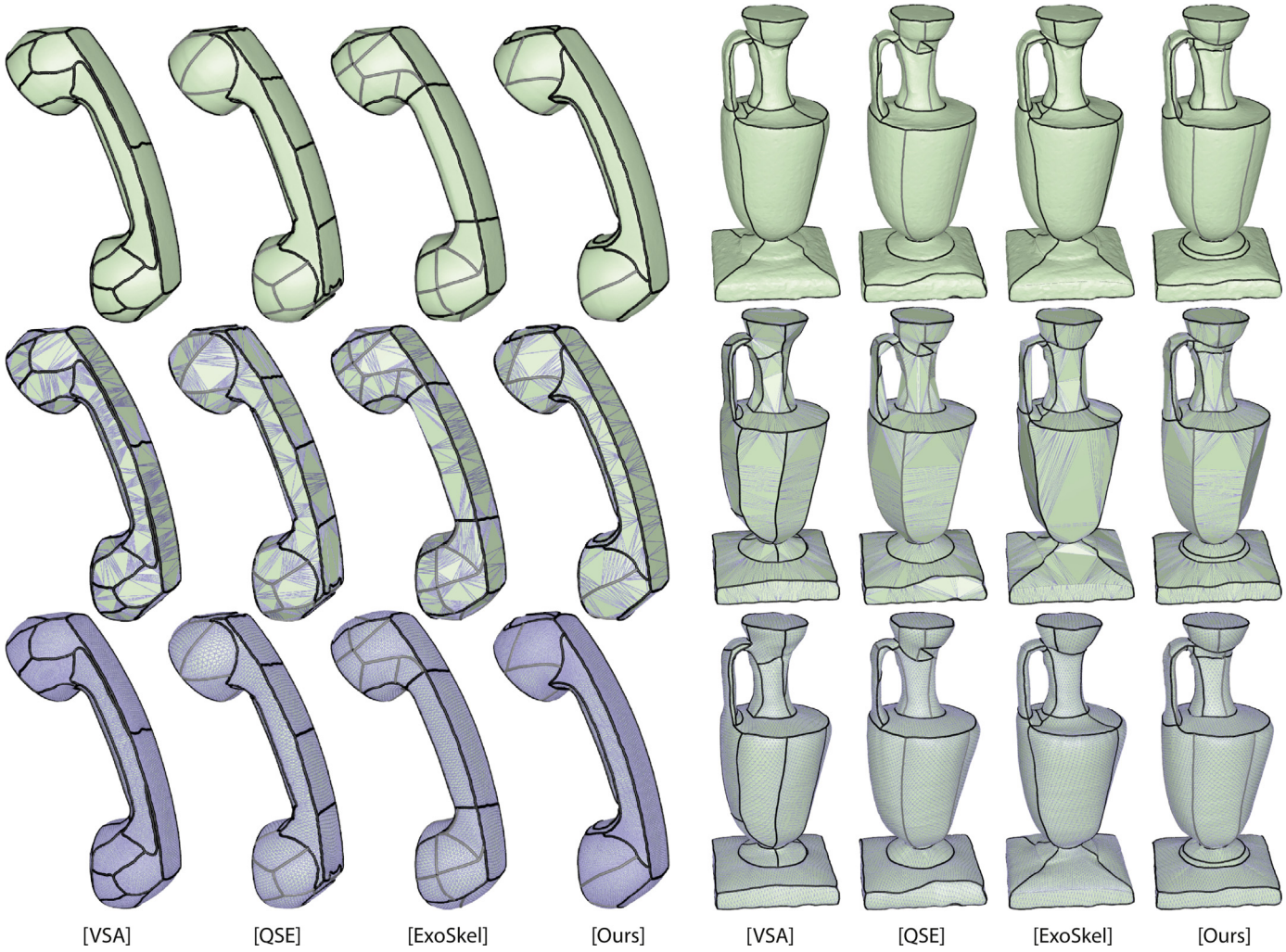


Fig. 10. Comparison with VSA, QSE, and ExoSkel methods. For each example, the top row are the patch layouts, middle row are the triangulation results and the bottom row are the results of filling N-sided patches.

and then add one more seed to this region if the maximum angle between normals of two triangles in this patch is larger than $\pi/2$.

3.6. Boundary smoothing

Our patch layout generation pipeline ends up after patch subdivision. Each patch is enclosed by some boundary edges. Since the boundary curves are usually not smooth, we need a postprocessing step to further smooth the boundaries. We use the method of [35] to smoothen the boundary curve γ by minimizing alignment energy

$$E(\gamma) = \int_{\gamma} \left(\frac{\langle \dot{\gamma}, JX_{min} \rangle}{\|\dot{\gamma}\| \|X\|} \right)^2 ds \tag{8}$$

such that boundary curve γ is aligned to the field of minimal principle curvature directions X_{min}, where J is the 90° rotation of X_{min} in the tangent planes. One smoothed patch layout is shown in Fig. 8.

3.7. Complexity analysis

Each step of our pipeline has linear complexity with regard to the size of the input mesh. We discuss the feature detection step as an example. For a triangular mesh with m points and n triangles, we calculate the extremity coefficient of each point through a locally fitted bivariate cubic polynomial to its k-ring neighborhood. The complexity is $O(m)$. Then we traverse all the triangles to check whether this triangle contains a ridge curve and the complexity is $O(n)$.

4. Experimental results

We tested our framework on various input surfaces. Fig. 9 shows a collection of generated patch layouts. Our experiments are conducted on a PC with 3.33 GHz GPU, 24 GB memory, and Windows7 operating system. Our algorithm is very efficient since we do not involve any time-consuming optimization.

Evaluation: We evaluate our layout quality by reconstructing the surfaces from the curve networks with different techniques, using a 3D polygon triangulation approach [29] and an N-sided hole filling technique [36]. We compute the Haursdoff distance and the *Root Mean Square* (RMS) distance between the reconstructed surfaces and the input surfaces to measure the reconstruction quality. We use the Metro tool [37] for this purpose.

Comparison: We compare our approach with previous methods including VSA [14], QSE [16], and ExoSkel [19], as shown in Fig. 10. Since QSE and our method always generate fewer patches than other approaches, we further insert new patches in regions with larger non-planar distortion by applying VSA subdivision. To make a fair comparison, we use the same number of patches for all the methods on the same test model. The quality evaluation of the reconstruction results are listed in Table 1. The meshes reconstructed from our

Table 1
Evaluation and comparison of the curve network generation methods. $|f|$ is the number of triangles in each model, $|P|$ is the number of patches, H is the Haursdoff distance, and R is the RMS distance between reconstructed meshes and input surfaces, respectively. The best results are shown in bold font.

Model	$\|f\|$	Alg.	[29]			N-sided		
			$\|P\|$	H	R	$\|P\|$	H	R
Phone	41512	VSA	40	0.0024	0.0008	40	**0.0014**	**0.0003**
		QSA	40	0.0027	0.0008	40	0.0017	0.0004
		EXO	40	0.0069	0.0010	40	0.0068	0.0009
		Ours	40	**0.0024**	**0.0005**	40	0.0016	0.0004
Vase	28950	VSA	39	0.0347	0.0125	39	0.0226	0.0058
		QSA	39	0.0580	0.0188	39	0.0333	0.0072
		EXO	39	0.0827	0.0245	39	0.0319	0.0087
		Ours	39	**0.0315**	**0.0118**	39	**0.0213**	**0.0055**
Face	11162	VSA	22	0.0187	**0.0055**	22	0.0211	0.0036
		Ours	22	**0.0164**	0.0065	22	**0.0101**	**0.0019**
Octa	32716	VSA	24	0.5233	0.2368	24	0.1278	0.0428
		Ours	24	**0.3171**	**0.1676**	24	**0.0589**	**0.0197**
Blade	97576	VSA	63	1.5401	0.4332	63	1.0241	0.2295
		Ours	63	**1.3287**	**0.3805**	63	**0.9742**	**0.2186**

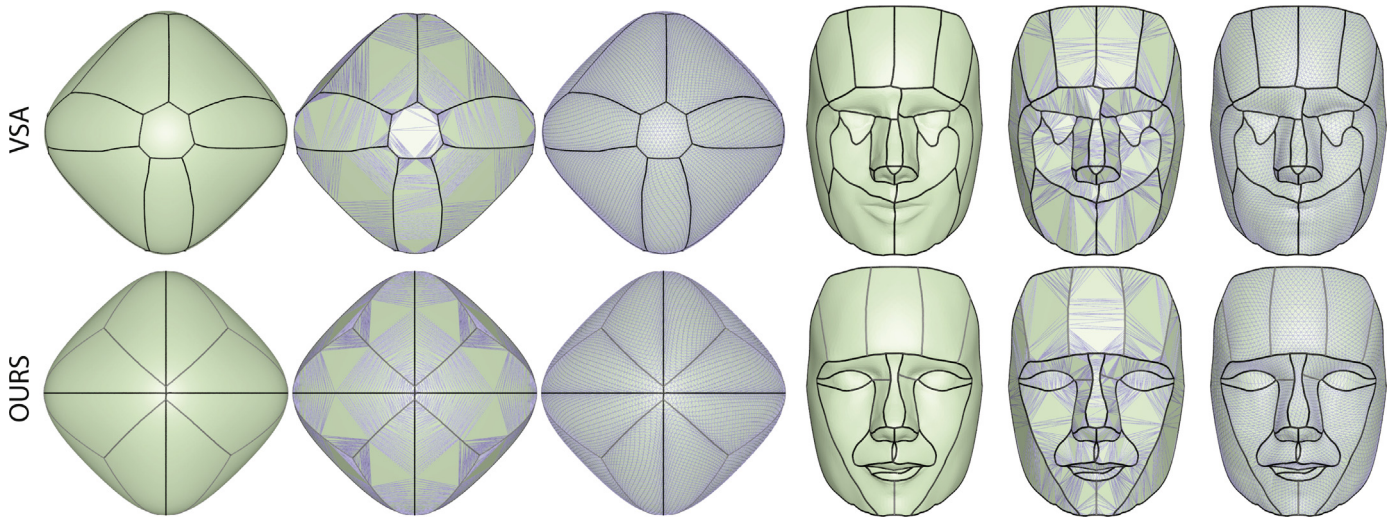


Fig. 11. Comparison with VSA. For each test, we show the patch layout, the triangulation result of [29], and the results of filling with N-sided patches [36].

patch layouts always exhibit smaller approximation error compared to others.

Unlike QSE and ExoSkel, which are only suitable for CAD models or free-form models with part components, our framework works well on more general input surfaces. E.g., both QSE and ExoSkel are failed to generate a valid patch layout for examples shown in Fig. 11. While our framework is able to compute more naturally layouts aligned to local salient features even for smooth regions. The approximation quality is also shown in Table 1. Fig. 12 shows a visual comparison with the slippage analysis method [38], where our approach generates better boundary curves.

Failure example: Our algorithm fails to generate a nice layout if the input mesh has too many small details. Fig. 13 shows such an example.

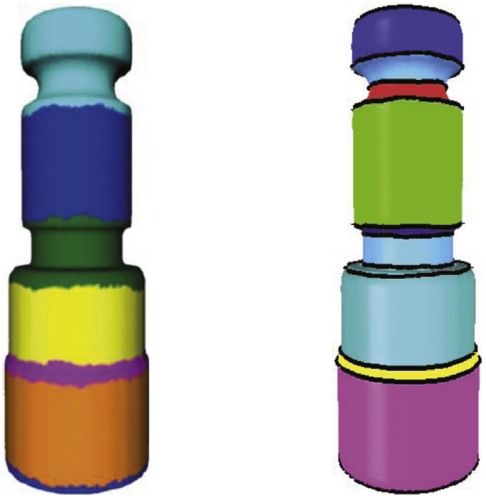


Fig. 12. Comparison with [38] (Part3 model). Left: result of [38] and right: our result.

5. Conclusion and future work

In this paper, we have presented a framework for computing a patch layout on surfaces. Our pipeline combines and improves upon several existing algorithms, and is able to generate more natural patch layouts following the salient feature curves of the surfaces.

One limitation of our work is that we only take clean 2-manifold input meshes as an input, and we cannot generate a nice layout for noisy/non-manifold input data. Our algorithm is also sensitive to the mesh quality and density. In these cases, the feature extension algorithm fails to generate a plausible patch

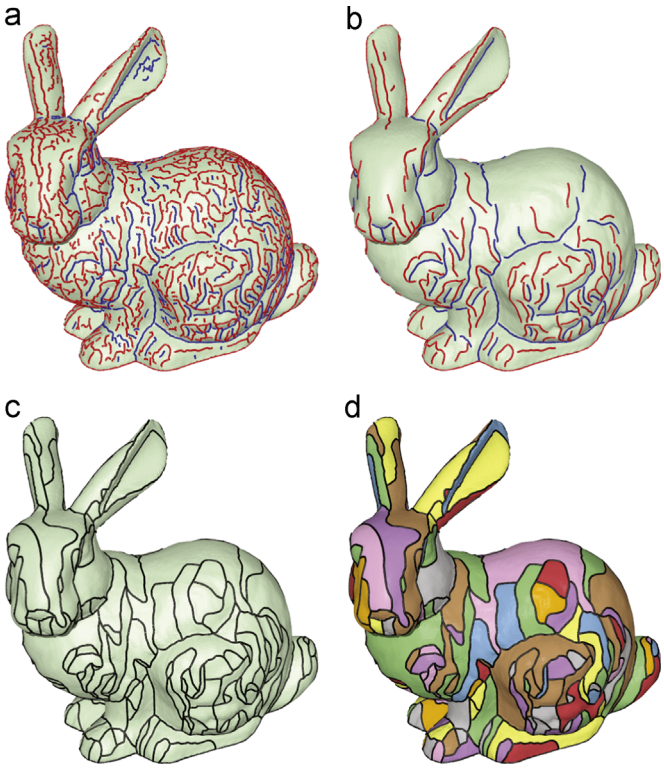


Fig. 13. A failure example. (a) The result of feature extraction. (b) The result of feature filtering. (c) The result of patch layout generation. (d) Different patches are shown in different colors. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

layout. We plan to improve our approach by considering the distance of details in the feature extension. Another limitation is that we did not consider high level structure information of the input surfaces, such as symmetry. We cannot preserve the symmetric structures in the generated layouts, e.g., the face model. We plan to address this problem as well in the future. We are also interested in computing feature aligned quad layout using our approach.

Acknowledgments

This work was funded by the KAUST Visual Computing Center, Boeing company, the National Natural Science Foundation of China (61372168, 61331018, and 61271431), and the U.S. NSF.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2014.09.022.

References

[1] Lévy B, Petitjean S, Ray N, Maillot J. Least squares conformal maps for automatic texture atlas generation. ACM Trans Graph (SIGGRAPH) 2002;21(3):362–71.
[2] Mitani J, Suzuki H. Making papercraft toys from meshes using strip-based approximate unfolding. ACM Trans Graph (SIGGRAPH) 2006;23(3):259–63.
[3] Várady T, Facello MA, Terék Z. Automatic extraction of surface structures in digital shape reconstruction. In: 4th International conference on geometric modeling and processing—GMP, 2006. p. 1–16.
[4] Mehra R, Zhou Q, Long J, Sheffer A, Gooch A, Mitra NJ. Abstraction of man-made shapes. ACM Trans Graph (SIGGRAPH Asia) 28 (5) (2009) #137, 1–10.
[5] Liu Y, Pottmann H, Wallner J, Yang Y-L, Wang W. Geometric modeling with conical meshes and developable surfaces. ACM Trans Graph (SIGGRAPH) 2006;25(3):681–9.
[6] Ohtake Y, Belyaev A, Seidel H-P. Ridge–valley lines on meshes via implicit surface fitting. ACM Trans Graph (SIGGRAPH) 2004;23(3):609–12.
[7] Lai Y-K, Zhou Q-Y, Hu S-M, Wallner J, Pottmann H. Robust feature classification and editing. IEEE Trans Vis Comput Graph 2007;13(1):34–45.
[8] Lee Y, Lee S. Geometric snakes for triangular meshes. Comput Graph Forum (EUROGRAPHICS) 2002;21(3):299–320.
[9] Lee Y, Lee S, Shamir A, Cohen-Or D, Seidel H-P. Mesh scissoring with minima rule and part salience. Comput Aided Geom Des 2005;22(5):444–65.
[10] Kim S-K, Kim C-H. Finding ridges and valleys in a discrete surface using a modified mls approximation. Comput Aided Des 2006;38(2):173–80.
[11] Hildebrandt K, Polthier K, Wardetzky M. Smooth feature lines on surface meshes. In: Proceedings of the third Eurographics symposium on geometry processing, 2005.
[12] Nomura M, Hamada N. Feature edge extraction from 3D triangular meshes using a thinning algorithm. In: Society of photo-optical instrumentation engineers (SPIE) conference series, vol. 4476, 2001. p. 34–41.
[13] Shamir A. A survey on mesh segmentation techniques. Comput Graph Forum 2008;27(6):1539–56.
[14] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Trans Graph (SIGGRAPH) 2004;23(3):905–14.
[15] Wu J, Kobbelt L. Structure recovery via hybrid variational surface approximation. Comput Graph Forum (EUROGRAPHICS) 2005;24(3):277–84.
[16] Yan D-M, Liu Y, Wang W. Quadric surface extraction by variational shape approximation. In: 4th international conference on geometric modeling and processing, 2006. p. 73–86.
[17] Yan D-M, Wang W, Liu Y, Yang Z. Variational mesh segmentation via quadric surface fitting. Comput Aided Des 2012;44(11):1072–82.
[18] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. ACM Trans Graph SIGGRAPH 2009;28(3).
[19] de Goes F, Goldenstein S, Desbrun M, Velho L. Exoskeleton: curve network abstraction for 3D shapes. Comput Graph 2011;35(1):112–21.
[20] Boier-Martin I, Rushmeier H, Jin J. Parameterization of triangle meshes over quadrilateral domains. In: Proceedings of the symposium on geometry processing, SGP '04, 2004. p. 193–3.
[21] Tarini M, Puppo E, Panozzo D, Pietroni N, Cignoni P. Simple quad domains for field aligned mesh parametrization. ACM Trans Graph (SIGGRAPH Asia) 2011;30(6):142–1–142:12.
[22] Campen M, Bommes D, Kobbelt L. Dual loops meshing: quality quad layouts on manifolds. ACM Trans Graph (SIGGRAPH) 2012;31(4):110:1–110:11.
[23] Bommes D, Campen M, Ebke H-C, Alliez P, Kobbelt L. Integer-grid maps for reliable quad meshing. ACM Trans Graph (SIGGRAPH) 2013;32(4):98:1–12.
[24] Gunpinar E, Suzuki H, Ohtake Y, Moriguchi M. Generation of bi-monotone patches from quadrilateral mesh for reverse engineering. Comput Aided Des 2013;45(2):440–50.
[25] Bommes D, Lévy B, Pietroni N, Puppo E, Silva C, Tarini M, et al. Quad-mesh generation and processing: a survey. Comput Graph Forum 2013;32(6):51–76.
[26] Orbay G, Kara LB. Sketch-based modeling of smooth surfaces using adaptive curve networks. In: Proceedings of the eighth Eurographics symposium on sketch-based interfaces and modeling, SBIM '11, 2011. p. 71–8.
[27] Abbasinejad F, Joshi P, Amenta N. Surface patches from unorganized space curves. Comput Graph Forum (Proc SGP) 2011;30(5):1379–87.
[28] Bessmeltsev M, Wang C, Sheffer A, Singh K. Design-driven quadrangulation of closed 3D curves. ACM Trans Graph (SIGGRAPH Asia) 2012;31(5).
[29] Zou M, Ju T, Carr N. An algorithm for triangulating multiple 3D polygons. Comput Graph Forum 2013;32(5):157–66.
[30] Yoshizawa S, Belyaev A, Seidel H-P. Fast and robust detection of crest lines on meshes. In: Proceedings of the ACM symposium on solid and physical modeling, 2005. p. 227–32.
[31] Goldfeather J, Interrante V. A novel cubic-order algorithm for approximating principal direction vectors. ACM Trans Graph 2004;23(1):45–63.
[32] Kim HS, Choi HK, Lee KH. Feature detection of triangular meshes based on tensor voting theory. Comput Aided Des 2009;41(1):47–58.
[33] Park MK, Lee SJ, Lee KH. Multi-scale tensor voting for feature extraction from unstructured point clouds. Graph Models 2012;74(4):197–208.
[34] Hsu S-H, Lai J-Y. Semi-automatic feature point extraction using one seed point. Int J Adv Manuf Technol 2010;51(1–4):277–95.
[35] Nieser M, Schulz C, Polthier K. Patch layout from feature graphs. Comput Aided Des 2010;42(3):213–20.
[36] Várady T, Rockwood A, Salvi P. Transfinite surface interpolation over irregular n-sided domains. Comput Aided Des 2011;43(11):1330–40.
[37] Cignoni P, Rocchini C, Scopigno R. Metro: measuring error on simplified surfaces. Comput Graph Forum 1998;17(2):167–74.
[38] Gelfand N, Guibas LJ. Shape segmentation using local slippage analysis. In: Proceedings of symposium on geometry processing, 2004. p. 214–23.