Pacific Graphics (2019)
C. Theobalt, J. Lee, and G. Wetzstein (Editors)

Feature Curve Network Extraction via Quadric Surface Fitting

Zhengda Lu[1,2], Jianwei Guo[2,1 †], Jun Xiao[1 †], Ying Wang[1], Xiaopeng Zhang[2,1] and Dong-Ming Yan[2,1]

[1]University of Chinese Academy of Sciences, China
[2]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China

Abstract

Feature curves on 3D shapes provide a high dimensional representation of the geometry and reveal their underlying structure. In this paper, we present an automatic approach for extracting complete feature curve networks from 3D models, as well as generating a high-quality patch layout. Starting from an initial collection of noisy and fragmented feature curves, we first filter non-salient or noisy feature curves by utilizing a quadric surface fitting technique. We then handle the curve intersections and curve missing by conducting a feature extension step to form a closed feature curve network. Finally, we generate a patch layout to reveal a highly structured representation of the input surfaces. Experimental results demonstrate that our algorithm is robust for extracting complete feature curve networks from complex input meshes and achieves superior quality patch layouts compared with the state-of-the-art approaches.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Shape analysis;*** *Mesh models;*

1. Introduction

Obtaining real-world 3D models has become easy due to the improved acquisition techniques for 3D geometry and easy access to 3D modeling tools. In recent years, a lot of efforts have been devoted to extracting high-level representation from raw geometric data, especially man-made objects. Among them, extracting meaningful *feature curve networks* (FCNs) have proven useful in a variety of downstream applications, such as reverse engineering [NSP10], shape abstraction [MZL*09, DGGDV11], mesh segmentation [ZDCJ17], remesh [LHJ*14, KYD*18], shape modeling [PLS*15, GSV*17], and functional maps computation [GBKS18], to name a few.

Feature curves are composed of crest lines on surfaces that convey the most essential characteristics of 3D shapes. Techniques detecting such feature curves usually use local shape properties (e.g., curvature, normal voting tensor) to extract creases [OBS04, YBS05], However, the detected feature curves are scattered on the surfaces, thus they are difficult to be directly used for further analysis and applications. To form a closed and valid feature curve network, user interactive is needed to connect the curves by using several stokes, but it is a challenging task for ordinary users. Many patch-based methods are proposed to automatically extract feature curve networks as patch boundaries from a mesh segmentation [NSP10, WHL*13]. This kind of method works well for extracting sharp feature edges but easily misses smooth features, thus resulting in non-detailed feature curve networks [CIE*16].

In this paper, we propose an automatic method for feature curve network extraction from 3D models. We present an efficient approach to filter out unimportant feature curves by combining the curve curvature and surface fitting error. Thus we could avoid identifying high-frequency noise (weak features) as false positives. Second, we handle the curve intersections and curve missing by applying feature extensions to create a connected 3D curve network. As a result, we can easily find the surface patches by the complete FCN. Our approach is capable of achieving superior-quality patch layouts comparable with state-of-the-art approaches. In summary, the main contributions of this work include the following:

- we present a simple yet effective framework that can extract valid feature curve networks from scanned surface meshes.
- we generate high-quality patch layouts to obtain a compact high-level representation of 3D models.

2. Related work

There is a large amount of research on feature detection and feature curve network extraction. A detailed survey is out of the scope of this paper, and we focus on those techniques most related to our approach.

Feature detection. Feature detection on discrete surfaces is usually converted to trace ridges and valleys, which are the loci of points where the positive (negative) variation of the surface normal in the direction of its maximal change attains a local maxi-

[†] Corresponding authors: jianwei.guo@nlpr.ia.ac.cn, xiaojun@ucas.ac.cn

DOI: 10.2312/pg.20191338

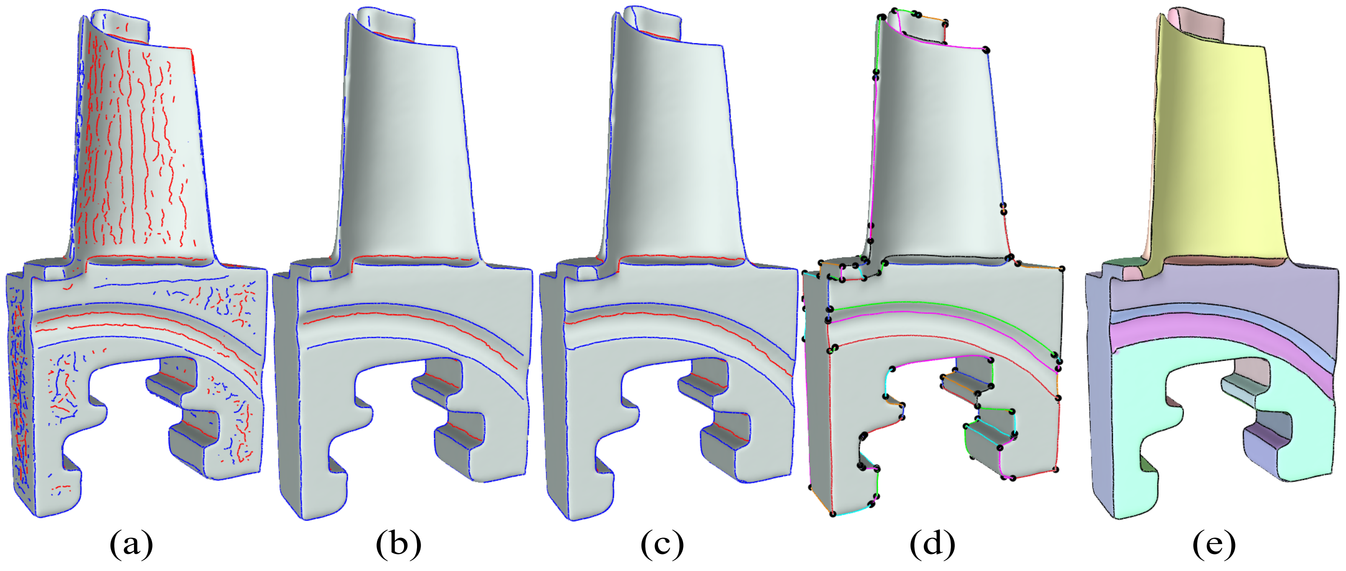

Figure 1: *The pipeline of our framework. (a) input mesh and feature curves. (b) salient feature curves after filtering. (c) feature curve extension result. (d) final feature curve network. (e) patch layout after segmentation and boundary smoothing, where different patches are shown in different colors.*

mum (minimum). Ridges and valleys extraction relies on the computation of differential properties such as normals and curvatures. Ohtake et al. [OBS04] combine multi-level implicit surface fitting and finite difference approximations to estimate curvature and curvature derivatives, then extract high-quality ridge and valley lines from these estimates. Yoshizawa et al. [YBS05] use the local polynomial fitting to calculate the curvature tensor and curvature derivatives to detect crest lines. Kim and Kim [KK06] utilize a modified MLS (moving-least-squares) approximation to estimate the local differential information and detect ridge and valley vertices by zero-crossings. Due to the limitation of local surface properties, the generated feature curves usually contain many non-salient curves and are not connected in the regions where multiple surfaces intersect.

Curve network extraction. A set of methods exist which extract curve networks to create abstractions of 3D shapes. Nieser et al. [NSP10] introduce the topology consistent feature graph to separate the surface along feature lines and form new regions with low varying curvature by thickening the edges of feature graph. de Goes et al. [DGGDV11] define the concept of an Exoskeleton where they generate a segmentation of the surface, revealing meaningful perceptual parts, which are refined by employing VSA. Cao et al. [CYW15] take the curve length and curvature into account by filtering a dense set of features for salient and long curves to generate a surface patch layout for 3D models. Gehre et al. [GLK16] extract a scale conforming feature curve network, which preserves the maximum set of prominent feature curves within a prescribed scale from an initial set of feature lines. Gehre et al. [GLK18] find the global co-occurrence information of feature curves to merge them into feature curve templates and complete missing data in noisy feature curve networks. In contrast to existing approaches, our goal is to generate a feature curve network for various kinds of shapes (not only man-made).

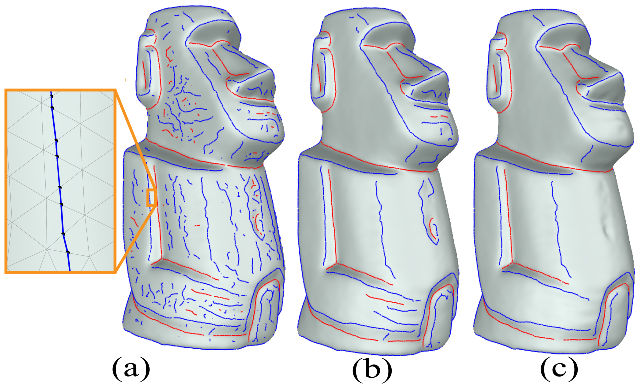

Figure 2: *Illustration of our filtering process using R and E. (a) input mesh and feature curves. (b) use parameter $R = 0.005$ to filter feature curves. (c) use the parameter $E = 3e^{-6}$ to filter other feature curves.*

3. Methodology

The input to our algorithm is a two-manifold triangular mesh $\mathcal{M}$, which consists of a set of triangle facets $\{t_i\}_{i=1}^{m}$ and vertices $\{\mathbf{v}_i\}_{i=1}^{q}$. We assume that an initial set of fragmented feature curves has been computed by using the method of Yoshizawa et al. [YBS05] due to its robustness. Our algorithm consists of three main steps, as illustrated in Fig. 1. We explain the details of each step in the next section.

3.1. Feature Curve Filtering

Given a curve $c = \{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$, the feature points are not the vertices of the mesh in general. To simplify our calculation, we map these feature points to the closet mesh vertices $\{\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_n\}$ and compute the curve curvature R as:

$$R = \frac{\sum_{i=1}^{n} C_{RMS}(\mathbf{p}_i)}{n}, \tag{1}$$

$$C_{RMS}(\mathbf{p}_i) = \sqrt{k_{max}^2 + k_{min}^2} \tag{2}$$

where $C_{RMS}(\mathbf{p}_i)$ is the root mean square curvature of $\mathbf{p}_i$, k_{max} and k_{min} are the maximal and minimal principal curvatures of the vertex $\mathbf{v}_i$ which is nearest with $\mathbf{p}_i$.

We use R to filter out non-salient curves lying on flat surface.

We propose the surface fitting error, E, to measure whether one curve is the intersecting curve of adjacent surfaces. We adopt the concept of quadric surface fitting [YWLY12]. Since each feature curve cuts across the triangles of $\mathcal{M}$, we collect these triangles as initial seeding set $\{t_i\}$. Then we visit each triangle in $\{t_i\}$, and add their adjacent triangles to $\{t_i\}$ with 3-iterations. Then we fit a general quadric surface (including the plane, sphere,cylinder,etc.) for $\{t_i\}$.

Firstly, we compute the measurements R for each curve to filter mostly non-salient curves locating on smooth areas. Then we compute E for the other curves. If the average error E of a curve is smaller than a pre-specified threshold, we regard the curve is located inside a quadric surface and filter it out. The result of this step is shown in Fig. 2.

3.2. Feature Curve Network Completion

After filtering, we have to extend the reserved curves to get a complete curve network. To complete the feature curve network, we extend each curve outwards from its two endpoints, $\mathbf{p}_0$ and $\mathbf{p}_n$. For the extension, we compute a candidate set of mesh vertices, C_v, which is a subset of the 1-ring neighboring vertices of $\mathbf{v}_0$, if the angle between the vector $\mathbf{v}_k - \mathbf{v}_0$ and the curve direction at $\mathbf{p}_0$ is less than 90°.

After building the candidate set, we define the extension cost value $F(\mathbf{v}_k)$ to measure the possibility of $\mathbf{v}_k$ to be selected for the extension. We use both the curvature direction and curve fitting error to calculate the extension cost function $F(\mathbf{v}_k)$:

$$F(\mathbf{v}_k) = E[\mathbf{v}_k] + G(t_{min}(\mathbf{v}_k), t_{min}(\mathbf{v}_0)) \tag{3}$$

where $E[\mathbf{v}_k]$ is the fitting error as defined above and G is the function to calculate the minimal principal curvature direction angle between $\mathbf{v}_k$ and $\mathbf{v}_0$.

Besides, to accelerate the extension speed, we collect all candidate vertices of all curves and use a global priority queue to maintain them. At each time, we pop the candidate $\mathbf{v}_k$ with the least cost from the queue and connect it to the corresponding curve endpoint. Besides, if $\mathbf{v}_k$ has been also mapped to another curve, it means two feature curves are connected by $\mathbf{v}_k$. In this case, the current curve will not extend in this direction anymore. Once a candidate is popped, the priority queue is updated by updating the candidate vertices set and their cost values. Finally, we extract the complete feature curve network when the priority queue becomes empty. A FCN result is shown in Fig. 3 (a).

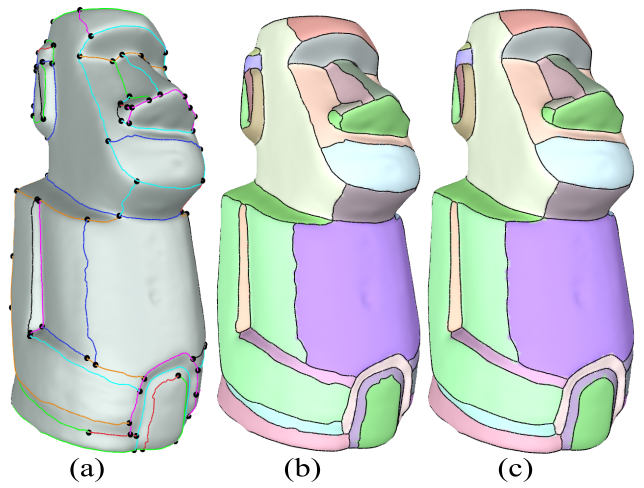

Figure 3: *Patches generation: (a) complete feature curve network. (b) primary patches with so many zigzags on boundaries. (c) final patch layout after boundary smoothing.*

3.3. Patch Generation

Since there may be some gaps in the mapped feature curve vertices. Thus we firstly check and recover the closure of each mapped feature curve.

For each mapped feature curve, we use the inward tracing scheme again. Firstly, we put the vertex $\mathbf{v}_0$ mapped by the endpoint $\mathbf{p}_0$ into a queue. Then we pop out the first vertex in the queue, and put its adjacent vertex $\mathbf{v}_k \subseteq \{\mathbf{v}_i\}$ into the queue. If all vertices in $\{\mathbf{v}_i\}$ can be visited when the queue is empty, the mapped feature curve is closed. Otherwise, we need to recover the closure of this curve. If $\mathbf{v}_j$ is not adjacent with other vertex in $\{\mathbf{v}_i\}$, we find the nearest vertex $\mathbf{v}_k \subseteq \{\mathbf{v}_i\}$ and add the vertices in the shortest path between $\mathbf{v}_j$ and $\mathbf{v}_k$ to $\{\mathbf{v}_i\}$.

After we map feature curves to mesh vertex, We generate patches by a region growing process. We put an unmarked vertex $\mathbf{v}_i$ into a queue, then we pop out the first vertex in the queue and put its adjacent unmarked vertex $\mathbf{v}_k$ into the queue. We get a patch visited by the queue until it is empty. This step is repeated until all mesh vertex is marked and we get all patches for the input mesh. One example is shown in Fig. 3 (b). Since the patch boundaries are usually not smooth due to the mesh discretization, we use the method introduced by [NSP10] to smooth the boundaries. The smoothed patch layout is shown in Fig. 3 (c).

4. Experimental Results

In this section, we conduct a number of experiments on various input models to demonstrate the efficacy of our approach. We implemented our algorithm using C++, and all shown results are obtained

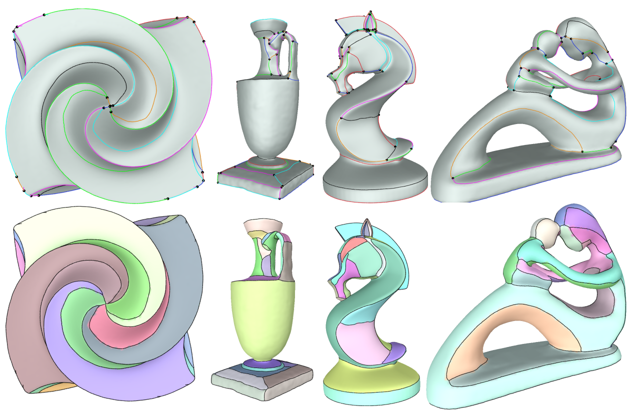

Figure 4: *A gallery of examples generated by our framework. For each model, we show the complete feature curve network and patch layout.*

Model	|f|	Alg.	|C|	|P|	|E|	|T|
Octa	59996	[CSAD04]	0	35	0.219388	1.595
		[YWLY12]	0	36	0.0238784	8.66
		[NSP10]	634	16	0.0643459	0.314
		[MS04]	634	12	0.129881	0.186
		[ZDCJ17]	634	17	0.0612	3.489
		[CYW15]	38	17	0.0485	0.369
		Our	52	21	0.0383	0.857
Moai	63994	[CSAD04]	0	27	0.0997578	1.454
		[YWLY12]	0	24	0.040191	7.42
		[NSP10]	761	41	0.0313272	0.324
		[MS04]	761	45	0.0354475	0.192
		[ZDCJ17]	761	39	0.0294	2.699
		[CYW15]	58	38	0.0242	0.475
		Our	69	46	0.0181	0.729
harddisk	193000	[CSAD04]	0	80	0.191569	10.266
		[YWLY12]	0	83	0.0332704	31.985
		[NSP10]	4558	333	0.0470268	0.719
		[MS04]	4558	40	0.216324	1.5
		[ZDCJ17]	4558	72	0.1241	22.299
		[CYW15]	359	60	0.1906	1.729
		Our	730	101	0.0969	4.297

Table 1: *Evaluation and comparison of different methods.* $|f|$ *is the number of triangles in each model,* $|C|$ *is the number of reserved curves in the final network,* $|P|$ *is the number of generated patches, and* $|E|$ *is the hybrid distance between the fitting surfaces and the input mesh, and* T *is the running time, respectively .*

on a desktop computer equipped with an Intel i7-7700k processor clocked at 4.2GHz, 16 GB of RAM.

4.1. Evaluation

We have evaluated our method on a wide range of 3D models with different complexities. Fig. 4 shows the results of extracting complete feature curve networks from these surface meshes. With our approach, we are able to identify both sharp and smooth feature curves that capture the main geometric characteristics, e.g., mesh boundaries and fading features. Then by extending and connecting the reserved curves, we obtain the complete feature curve networks, as shown in the top row of Fig. 4. The bottom row shows the patch segmentation result of the input mesh. Thus we decompose a surface into structural parts.

4.2. Comparison

We compare our method against previous approaches using different mesh segmentation strategies. Fig. 5 shows all of the comparison results on three input models. The numerical statistics about the patch result are presented in Table 1. From the Fig. 5 (a, b, c, d), we observe that [CSAD04], [YWLY12], [NSP10] and [MS04] miss many important feature curves and their patch layouts are not satisfactory. Due to curvature variation in quadric surface, they always produce additional patches. As shown in Fig. 5 (e), [ZDCJ17] fails to bridge feature curves which further apart in blending regions. Although both our method and [CYW15] directly extend and connect pruned feature curves, [CYW15] usually remove more curves by only using the curve length and curvature. In contrast, we preserve more salient curves to produce a more satisfactory curve network by combining the curve curvature and surface fitting error. Thus, we could generate more reasonable patch layouts, the fitting error of our patches always displays a smaller fitting error compared with [CYW15] (see in Table 1).

4.3. Limitations

We successfully extract complete feature curve networks from various 3D models. Nevertheless, since our feature filtering and extension operations only work on individual curves, we do not consider the distance or relationships between different curves. Thus our results miss the global information of the shape, such as symmetry, parallelism, co-planarity, etc. Another limitation is that we cannot guarantee a valid feature curve network from meshes with low sampling resolution and non-manifold inputs.

5. Conclusion

In this paper, we have proposed a new method to address the problem of extracting feature curve networks from 3D models. We successfully tackled this problem by utilizing operations of feature curve filtering, feature curve network completion, and patch generation. We can detect both strong and weak features and connect them into a valid and complete feature curve networks. The proposed algorithm is also robust to scan data. We have demonstrated our approach with a variety of examples. The extracted networks can be used to various geometry processing tasks such as feature preserving remeshing and the smoothing of the noisy geometry.

In the future, we would like to extract templates of reoccurring feature curve cycles by analyzing the local properties such as parallelism, symmetry, and hierarchy of the complete feature curve network on given surfaces. Furthermore, we intend to combine our FCNs with editing systems. The global structure of our extraction can provide significant benefits to the intuitive handling of 3D shapes.

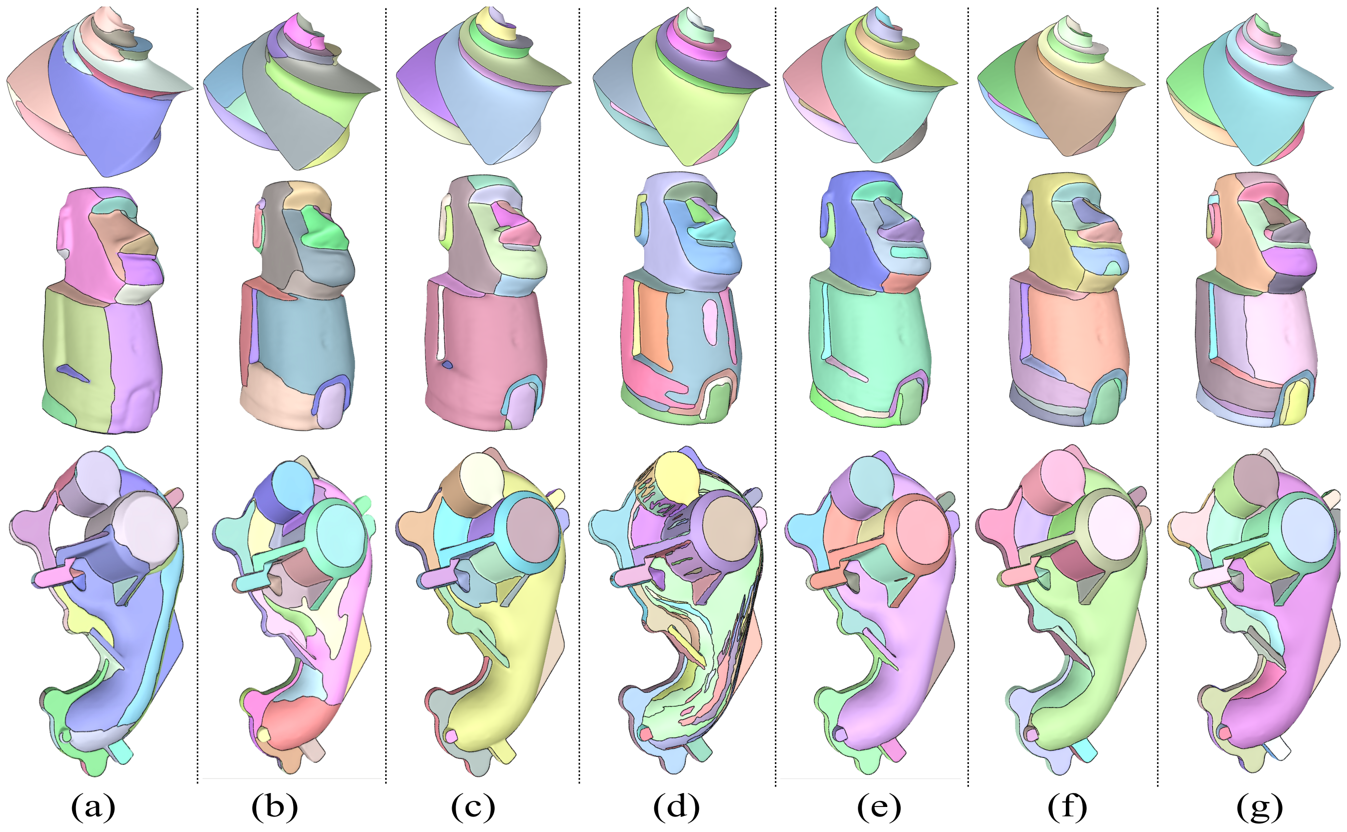

Figure 5: *Comparison with previous methods on models of "Octa", "Moai" and "harddisk". From left to right: results of Cohen-Steiner et al. [CSAD04] (a), Yan et al. [YWLY12] (b), Nieser et al. [NSP10] (c), Mitani and Suzuki [MS04] (d), Zhuang et al. [ZDCJ17] (e), Cao et al. [CYW15] (f), and our method (g), respectively.*

References

[CIE*16] Campen M., Ibing M., Ebke H.-C., Zorin D., Kobbelt L.: Scale-invariant directional alignment of surface parametrizations. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 1–10. 1

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. In *ACM Trans. on Graphics* (2004), vol. 23, ACM, pp. 905–914. 4, 5

[CYW15] Cao Y., Yan D.-M., Wonka P.: Patch layout generation by detecting feature networks. *Computers & Graphics 46* (2015), 275–282. 2, 4, 5

[DGGDV11] De Goes F., Goldenstein S., Desbrun M., Velho L.: Exoskeleton: Curve network abstraction for 3d shapes. *Computers & Graphics 35*, 1 (2011), 112–121. 1, 2

[GBKS18] Gehre A., Bronstein M., Kobbelt L., Solomon J.: Interactive curve constrained functional maps. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 1–12. 1

[GLK16] Gehre A., Lim I., Kobbelt L.: Adapting feature curve networks to a prescribed scale. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 319–330. 2

[GLK18] Gehre A., Lim I., Kobbelt L.: Feature curve co-completion in noisy data. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 1–12. 2

[GSV*17] Gori G., Sheffer A., Vining N., Rosales E., Carr N., Ju T.: Flowrep: descriptive curve networks for free-form design shapes. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 59. 1

[KK06] Kim S.-K., Kim C.-H.: Finding ridges and valleys in a discrete surface using a modified mls approximation. *Computer-Aided Design 38*, 2 (2006), 173–180. 2

[KYD*18] Khan D., Yan D.-M., Ding F., Zhuang Y., Zhang X.: Surface remeshing with robust user-guided segmentation. *Computational Visual Media 4*, 2 (2018), 113–122. 1

[LHJ*14] Ling R., Huang J., Jüttler B., Sun F., Bao H., Wang W.: Spectral quadrangulation with feature curve alignment and element size control. *ACM Transactions on Graphics (TOG) 34*, 1 (2014), 11. 1

[MS04] Mitani J., Suzuki H.: Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM transactions on graphics (TOG)* (2004), vol. 23, ACM, pp. 259–263. 4, 5

[MZL*09] Mehra R., Zhou Q., Long J., Sheffer A., Gooch A., Mitra N. J.: Abstraction of man-made shapes. In *ACM Trans. on Graphics* (2009), vol. 28, ACM, p. 137. 1

[NSP10] Nieser M., Schulz C., Polthier K.: Patch layout from feature graphs. *Computer-Aided Design 42*, 3 (2010), 213–220. 1, 2, 3, 4, 5

[OBS04] Ohtake Y., Belyaev A., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM transactions on graphics (TOG) 23*, 3 (2004), 609–612. 1, 2

[PLS*15] Pan H., Liu Y., Sheffer A., Vining N., Li C.-J., Wang

W.: Flow aligned surfacing of curve networks. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 127. 1

[WHL*13] WANG S., HOU T., LI S., SU Z., QIN H.: Anisotropic elliptic pdes for feature classification. *IEEE transactions on visualization and computer graphics 19*, 10 (2013), 1606–1618. 1

[YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling* (2005), ACM, pp. 227–232. 1, 2

[YWLY12] YAN D.-M., WANG W., LIU Y., YANG Z.: Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design 44*, 11 (2012), 1072–1082. 3, 4, 5

[ZDCJ17] ZHUANG Y., DOU H., CARR N., JU T.: Feature-aligned segmentation using correlation clustering. *Computational Visual Media 3*, 2 (2017), 147–160. 1, 4, 5