

Feature Detection for Surface Meshes [†]

Xiangmin Jiao and Michael T. Heath

Computational Science and Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{jiao,heath}@cse.uiuc.edu

Abstract

This paper addresses automatic detection of intrinsic geometric features (such as ridges and corners) in surface meshes. We propose characterizations and develop techniques for detecting such features. Our techniques have significant advantages in reliability and generality over traditional methods while being nearly optimal in performance. We integrate these techniques to match features in disparate meshes, a crucial step in correlating multiple meshes.

Introduction

Computer simulations of complex systems involve sophisticated meshing techniques, including mesh smoothing, adaptive mesh refinement, mesh motion, and data transfer between disparate meshes. Geometric features, such as ridges and corners, frequently require special treatments to achieve high accuracy and reliability. Mesh enrichment, for instance, must preserve features while striving to smooth out roughness of a mesh [4]. In surface propagation, discontinuities must be handled properly to achieve accuracy and stability [6, 9]. In transferring data (such as pressure) between meshes [6], data on each side of a ridge in one mesh must be transferred onto the corresponding side of the ridge in the other mesh to avoid unphysical solutions. Geometric features are also crucial in extracting a CAD model from a surface or volume mesh [8].

This paper addresses automatic detection of geometric features in surface meshes. Information about geometric features may sometimes be embedded in a mesh data structure, but frequently it is unavailable for various reasons. First, feature information may not have been available in the first place. For example, a mesh may have been obtained from a set of discrete points by surface reconstruction or as a solution from a numerical solver. Second, the feature information may have been omitted from the data structure even if it was available initially. Some mesh

[†]This research is supported by the Center for Simulation of Advanced Rockets funded by the U.S. Department of Energy under Subcontract B341494.

generators, for example, can export accurate information about features from a CAD model, but a user may suppress it for simplicity or compatibility. In all these cases, geometric features must be reconstructed from the mesh.

Like some other types of features, such as textures, geometric features may be apparent to human eyes, but their automatic detection is decidedly nontrivial. The difficulties are twofold. First, features are usually harder to characterize than to recognize. Human beings can recognize features even if their characterization remains fuzzy and intuitive, but computers need a definite and mathematically sound characterization before recognition can be carried out. A substantial amount of effort in this paper is devoted to a clear characterization of features. Second, artifacts of a coarse mesh can introduce “noise” into the geometry. A feature detection scheme may mistake artifacts for features or vice versa. This paper presents an efficient and reliable detection algorithm that can tolerate noise.

Characterizations

To characterize geometric features, we start by considering them in analytic surfaces defined by some abstract functions. The actual format of such functions is immaterial, but we assume that they are differentiable except along some curves and at some discrete points. Characterization of features in a surface mesh will follow after these analyses.

Features in Analytic Surfaces

In an analytic surface, features are composed of *feature points* that violate continuity or smoothness properties. We consider discontinuities only in normal directions (i.e., first derivatives) and assume the surfaces are *orientable*. By definition, normal directions are discontinuous at the boundary of a surface, and hence boundary points are feature points. Some feature points may compose a *connected smooth curve* on the surface, which we refer to as a *feature curve*. Smooth curves may in turn be connected to form more complex curves.

We classify features of a surface as follows. *One-dimensional features*, or *1-features*, of a surface are nonextensible smooth feature curves that do not intersect each other in their interiors. A *zero-dimensional feature*, or *0-feature*, is a feature point that is not contained in the interior of any 1-feature. Informally, we sometimes refer to 1-features as *ridges* and 0-features as *corners*.

In our definition, a 1-feature is a one-manifold with boundary and is required to be nonextensible (meaning two 1-features cannot be merged without violating smoothness or absence of intersections). Furthermore, two 1-features cannot intersect each other except at 0-features. One can picture a 1-feature to be a closed point set, either a *loop* or a *link* ending at 0-features; see the curves

in Figure 1. Must a 1-feature be closed? No. Consider the surface $z(x, y) = y^2 + |x| \max(0, y)^2$ (illustrated in Figure 2), which is not differentiable along the (dashed) curve $\{(x, y, z(x, y)) \mid x = 0 \text{ and } y > 0\}$ but is differentiable at the origin $(x = 0, y = 0)$. Therefore, this feature curve is open at the origin. In open-ended 1-features, the discrepancy between tangents from either side may vanish gradually, which can potentially cause difficulties for feature detection.

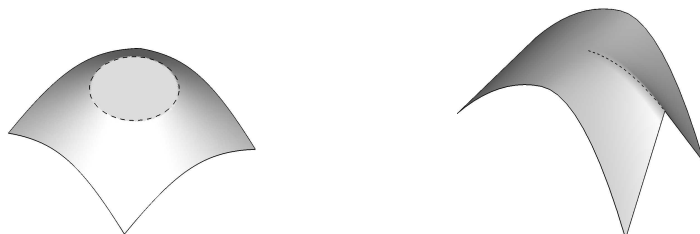


Figure 1: Sample closed 1-features: loop (dashed) and links (solid). Figure 2: Sample open-ended 1-feature (dashed).

To classify 0-features, we define the *rank* of a 0-feature to be the maximum number of its incident feature curves that do not overlap each other. The 0-features in Figure 3 have rank 0, 1, 2, and 4 (3 or more), respectively, and we refer to them as *tip*, *terminus*, *turn*, and *junction* correspondingly. Note that a 0-feature incident on only a loop-type 1-feature has rank 2 instead of 1. For completeness, we define an open-end of a 1-feature (Figure 2) to be a special 0-feature of rank -1. In general, the smaller the rank of a 0-feature, the harder the feature is to identify.

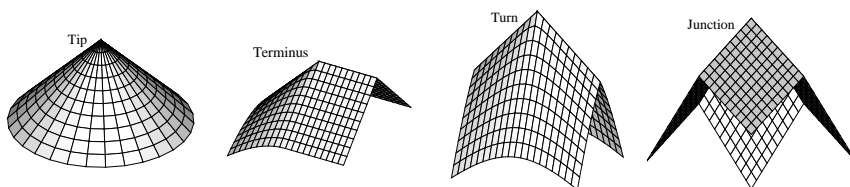


Figure 3: Examples of various types of 0-features.

Features in Surface Meshes

A *surface mesh* refers to a collection of 0-, 1-, and 2-dimensional cells (or faces), all embedded in \mathbb{R}^3 . We call the 0-dimensional cells *vertices*, the 1-dimensional cells *edges*, and the 2-dimensional cells *facets*. Vertices are points, edges are closed topological intervals, and facets are closed topological disks. A mesh is

required to be a pure complex modeling a 2-manifold with boundary; see for example [3] for formal definitions. For simplicity, we consider only triangular meshes. The ideas, however, apply to other types of meshes.

In a discrete surface, every edge looks like a ridge and every vertex looks like a corner. However, only a subset of edges and vertices are in the features of the underlying surface. We refer to the edges in the 1-features as *feature edges* and the vertices at 0-features as *feature vertices*. *Feature detection* identifies these feature cells and connects the feature edges to reconstruct feature curves. To achieve this goal, we must identify the *strong cells* that are likely to be features. We characterize strongness using numerical approximations to the curvature of a surface and its prospective feature curves. Note that curvature does not exist at features, so its approximations are meaningless there quantitatively. These approximations nevertheless are useful qualitatively because they are relatively large at features, and definitions of strongness rely on such largeness.

Strong Edges. A commonly used criterion for strong edges involves what we call the *face angle* (or *dihedral angle*), which roughly approximates the principle curvature of the surface at an edge. Let \mathbf{n}_1 and \mathbf{n}_2 be the (outward) unit normals of the incident facets of an edge e . The face angle at e , denoted by $\angle e$, is the angle between \mathbf{n}_1 and \mathbf{n}_2 , (i.e., $\angle e \equiv \arccos(\mathbf{n}_1^T \mathbf{n}_2)$) or π at boundary edges; see Figure 5. Given $\theta \in (0, \pi)$, we say e is θ -strong in face angle if $\angle e \geq \theta$.

Some traditional methods identify feature edges purely based on θ -strongness [4, 8]. These methods are efficient but unreliable. First, they are sensitive to perturbations because the geometric support of $\angle e$ is very local. Some researchers alleviate the problem by extending the geometric support and applying a least squares smoothing [5]. Unfortunately, this introduces the difficulty of choosing a proper size for the geometric support and also degrades efficiency substantially. Second, θ -strongness is sensitive to the choice of θ . Too small a θ tends to mistake artifacts for features, and too large a θ tends to exclude true features. Figure 4 shows a mesh of one half of the Falcon aircraft and its 40° -strong edges. At regions with large curvature, including the engine nacelle, the head end, and the front of the wing, many strong edges are not features. At the junction of the body and wing, some feature edges are not strong. For this mesh, there is no value for θ such that the θ -strong edges are exactly the feature edges.

We introduce a notion of relative θ -strongness (or r - θ -strongness) in the context of prospective feature curves to indicate whether two edges are likely to compose a feature curve. Let $\angle(e, g)$ denote the *edge angle* between e and g defined as the angle between their tangential directions as depicted in Figure 6(left). We observe that if $e \cup g$ forms a feature curve, then $\angle e$ and $\angle g$ must be relatively large. Furthermore, $\angle(e, g)$, which approximates the curvature of $e \cup g$, must be

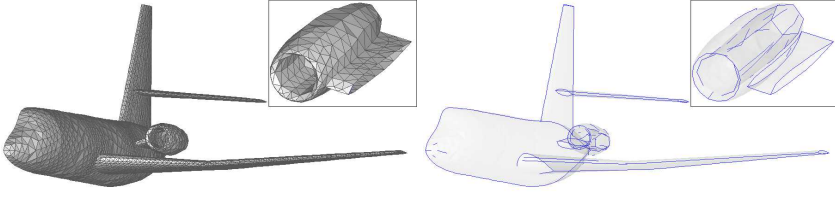


Figure 4: Mesh and 40°-strong edges of Falcon aircraft and enlarged nacelle.

relatively small. This motivates us to define a *weighted face angle* with respect to e at g to be $w(g, e) = |\cos \angle(e, g)| \angle g$. We say g is r - θ -strong w.r.t. e for some $r \geq 1$ and $\theta \in [0, \pi]$, denoted as $s(g, e, r, \theta)$, if $\angle g$ is no less than $\angle(g, e)$, and $w(g, e)$ is at least θ and is r times as great as the weighted face angles w.r.t. e at other edges incident on $\cap(g, e)$, i.e.,

$$s(g, e, r, \theta) \equiv \angle g \geq \angle(g, e) \text{ and } w(g, e) \geq \max\{\theta, \max_{g \cap e \in h \text{ and } h \neq g} \{rw(h, e)\}\}.$$

Intuitively, r - θ -strongness requires the principle curvature at g to be no less than the curvature of $g \cup e$, and $w(e, g)$ to be a strong local maximum.

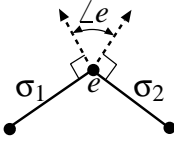


Figure 5: Face angle.

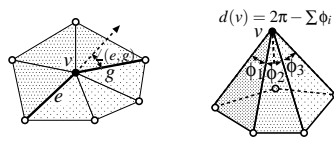


Figure 6: Edge angle and angle defect.

The notion of r - θ -strongness is more adaptive than θ -strongness because it relies more on identifying local extremes. It is also more general, as it reduces to θ -strongness if $e = g$ and $r = 0$. However, it is also more expensive to compute. We define a hybrid criterion that combines the advantages of the two. Specifically, we introduce three thresholds, θ_u , r , and θ_l , which correspond to the upper-bound of weaknesses, “signal/noise ratio”, and lower-bound of strongnesses. We say an edge g is *url-strong* w.r.t. e if g is θ_u -strong or r - θ_l -strong. An edge g is said *url-stronger* than edge h w.r.t. e if g is θ_u -strong and its face angle is greater than that at h or it is r - θ_l -strong. In the rest of the paper, we use strongness to refer to *url-strongness* unless otherwise specified.

Strong Curves. Using the *url*-notion, we define connections among the strong edges of a mesh. Specifically, an edge e is *url-connected* to g if g is *url-strongest*

w.r.t. e among edges incident on $g \cap e$. We say a curve γ is *url-strong* if it contains an edge e that is θ_u -strong and there is a *url-connected* path from e to every other edge in γ . If θ_l is no greater than the minimum face angle at all feature edges and θ_u is no greater than the minimum of the maximum face angle of every 1-feature, we observe that every feature curve is very likely to be *url-strong* for some r greater than 1 if the mesh has only moderate noise. It is even more likely that every feature curve is composed of a set of *url-strong* curves.

On the other hand, there is still a high possibility that a strong curve contains edges that are not features because of potential noise in the input data and limitations of the strongness criterion. To purify a strong curve γ , we break it into a set of subcurves between *strong vertices* and define a subcurve s (and its edges) to be *false strong* if s satisfies a set of *filtration rules*, which we define shortly.

Strong Vertices. An obvious criterion for strong vertices (junctions in particular) is to check their *ranks*, i.e., the numbers of their incident feature edges. For turns, the *feature edge angle*, defined as the angle between two incident feature edges of a vertex, is sometimes used [4]. A vertex is said θ -*strong* in feature edge angle if its value is greater than θ . Both of these criteria are based on feature edges and hence are barely useful for filtration.

We define strong vertices using a *url-thresholding* similar to that of strong edges. We define the *edge angle* at a vertex in a strong curve γ to be the angle between its two incident edges in γ ; the edge angle at end vertices of γ are defined to be π . A vertex is r - θ -*strong* in γ if its edge angle is no less than θ and is r times as great as the edge angles of its neighbor vertices in γ . A vertex is *url-strong* if it is θ - θ_u -strong (i.e. θ_u -strong) and r - θ_l -*strong* for some given θ_u , r , and θ_l .

A vertex, such as a tip or terminus, can be strong independent of strong curves. We define another criterion using *angle defect*. Let $\angle(v, \sigma)$ denote the angle between the incident edges of v in σ . The angle defect at v , denoted by $d(v)$, is the difference between 2π (or π if v is a boundary vertex) and the sum of the angles at v in its incident facets, i.e., $d(v) \equiv b\pi - \sum_{v \in \sigma} \angle(v, \sigma)$, where b is 1 for boundary vertices and 2 otherwise. This is illustrated in Figure 6(right). Angle defect is closely related to the Gaussian curvature of smooth surfaces as both of their magnitudes measure how much a surface deviates from being flat at a point. For this reason, angle defect is also called (*simplicial*) *curvature* [1]. We say vertex v is r - θ -*strong* in angle defect if $|d(v)|$ is no less than θ and is r times as great as those at its neighbor vertices, and we define *url-strongness* as before.

Filtration Rules. We characterize false strongness using two rules. A false strong curve is most likely a link containing a small number of edges due to the

strict requirements of strongness. Furthermore, both its end vertices are unlikely to be strong in angle defect. This motivates a *short-falseness rule*. For a given strong subcurve s , let b be the number of weaknesses in angle defect of its end vertices, and l be a user-specified threshold (say 6). The rule asserts s to be false strong if $b \geq 0$, one of its ends is not a known feature, and the number of edges in s is smaller than bl . In the example in Figure 4, this rule filters out the false strongness at the engine nacelle and the head end. One may fear that this rule may filter out open-ended 1-features, but we observe that these features typically contain a relatively large number of edges.

Sometimes long false strong curves are present near 1-features; see for example the one at the front of the wing in Figure 4. This phenomenon is due to boundary layers in some meshes generated using, for example, advancing-front methods. This motivates a *long-falseness rule*, which asserts a strong subcurve s to be false if none of its end vertices is strong in angle defect or is on a known feature, and every vertex is adjacent to a known feature.

Detection Algorithm

We now develop an algorithm for detecting feature curves that are composed of *url*-strong but not false strong subcurves. The basic idea is to identify strong curves in decreasing order of strongness (the motivation of processing in decreasing order is to prevent the long-falseness rule from filtering out the stronger of two parallel curves). For each strong curve γ , we identify the strong vertices in it and break it into subcurves at the strong vertices. We then filter out the false strong subcurves in γ and assert remaining ones as features. After identifying all true strong subcurves, we determine the ranks of all strong vertices, break the subcurve at vertices of rank 3 or more, and merge the subcurves at false strong vertices of rank 2. Figure 7 outlines the algorithm. We note that at line 5, the strongest edge is either *url*-strongest for given thresholds, or strongest for $r = 1$ if the other end of g is on a known feature or is strong in angle-defect. The latter allows slightly-relaxed strong edges to fill gaps between strong curves.

Our algorithm is also very efficient. The most expensive part of the algorithm is to compute face angles, which takes $O(n)$ time, where n is the size of the input mesh. The algorithm requires sorting θ_u -strong edges, which takes $O(m \log m)$ time where m is the number of strong edges. Typically, m is $O(\sqrt{n})$, although it can be $O(n)$ in the worst case for an unreasonably small θ_u . All other steps require $O(m)$ time, assuming the number of incident edges of a vertex is bounded by a constant. Therefore, the total runtime of the algorithm is $O(n + m \log m)$ and is expected to be $O(n)$. Assuming a feature detection algorithm must compute face angles, our algorithm takes a time within a factor of $1 + \varepsilon$ of optimal, where ε depends on the ratio between $m \log m$ and n and tends to 0 as a mesh is refined.

```

1: for each  $\theta_u$ -strong edge  $e$  in decreasing order of strongness do
2:   if  $e$  is already visited then Continue;
3:   Create curve  $\gamma$  at  $e$ ;
4:   repeat
5:      $g \leftarrow$  strongest edge w.r.t. front or back of  $\gamma$ ;
6:     if  $g$  is not visited then  $\gamma \leftarrow \gamma \cup \{g\}$  and mark  $g$  visited;
7:   until strongest edge w.r.t. both ends of  $\gamma$  have been visited;
8:   Break  $\gamma$  into sub-curves  $\{s_1, s_2, \dots\}$  at vertices strong in  $\gamma$ ;
9:   for each  $s_i$  do
10:    if  $s_i$  is not false strong then Assert  $s_i$  to be feature;
11:  end for
12: end for
13: Break feature curves at vertices incident on three or more feature edges;
14: Merge feature curves at rank-2 vertices that are not strong in merged curves.

```

Figure 7: Feature detection algorithm.

Experimental Study

To validate our algorithm, we carried out a series of tests using three meshes: a CAD model courtesy of Caterpillar¹ and two meshes for the Falcon aircraft. Table 1 summarizes the statistics for these cases. We first performed a parameter study. For strong edges, we found that the minimum weighted face angle at feature edges was about 11° , and the minimax face angle of 1-features was about 45° . Therefore, we chose $\theta_l = 5^\circ$ and $\theta_u = 40^\circ$. For angle defect, the upper-bound at non-features was less than 70° , and the lower-bound for features was about 30° . We set $\theta_u = 80^\circ$ and $\theta_l = 20^\circ$. We determined the bounds for edge angles in a similar fashion and set $\theta_u = 80^\circ$ and $\theta_l = 15^\circ$. The signal/noise ratios are more mesh dependent and harder to determine. In our tests, we used 4 for strong edges, 3 for angle defect, and 2 for edge angle. The maximum length of false strong subcurves not adjacent to 1-features was 6.

Case	mesh size		features				time	
	vertices	edges	1-D	0-D	edges	false	sec.	$\angle e$
CAD	7553	22749	161	145	2091	387	0.23	88%
Falcon1	2289	6721	28	18	429	115	0.06	90%
Falcon2	18831	55997	28	18	1433	290	0.46	96%

Table 1: Statistics and performance results for test cases.

¹We thank Steven J. Owen of Sandia National Laboratories for his help in obtaining the model.

Using these parameters, we tested our algorithm on a PC with a 1GHz Pentium III processor. Figure 8 shows the detected feature curves. The runtimes are listed in Table 1. For the two Falcon meshes, our algorithm reported the same set of 1- and 0-features, demonstrating its reliability. All cases take a fraction of a second to run, of which the time for computing face angles is about 90% and tends to increase as a mesh is refined. These results are consistent with our previous analyses. In summary, our experiments demonstrated that *url*-thresholding with filtration provides a flexible, reliable, and efficient approach for feature detection.

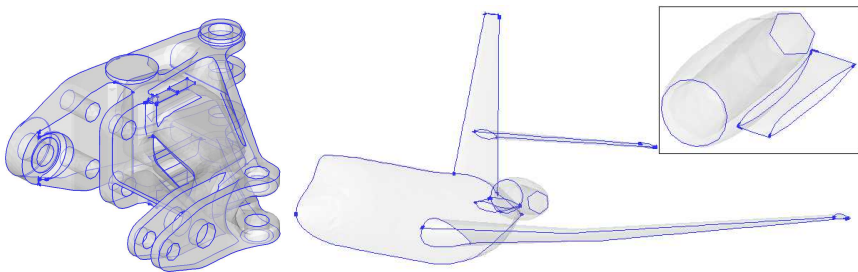


Figure 8: Features in CAD model and Falcon aircraft.

Application and Conclusion

We consider the problem of matching features in meshes modeling a common surface. This is a good test of feature detection, because it has some strong requirements on features that also apply for other applications. In [7], we reported some preliminary results on feature matching and its use in mesh overlay. We now describe how the above detection algorithm fits into feature matching.

Our matching scheme does not expect perfection from feature detection but assumes that all features are detected with few false strongnesses. We start by comparing the distance (in infinity norm) between 0-features of two meshes, and two 0-features are matched if their distance is smaller than a small fraction of the minimum edge length. Efficiency is achieved using a three-dimensional range-tree data structure [2]. The 0-features with no match are false strongnesses. We eliminate them and then reperform the merging (Line 14) of the detection algorithm.

We then match the 1-features in decreasing order of strongness. Let γ be the strongest 1-feature in one mesh K . Let e be the θ -strongest edge in γ , and p be the midpoint of e . We locate the 1-feature τ that is closest to p in the other mesh. Let q be the nearest point of p in τ . If $\|p - q\|_\infty$ is small relative to the minimum edge length, and γ is closer to q than all other features in K , then we match γ

and τ and construct a bijection between them as described in [7], with a minor change for open-ended 1-features. If a false strong curve is detected, we eliminate it and repeat the breaking and merging steps in Figure 7 before processing the next strongest feature. This provides a reliable way to match features.

In summary, this paper presented a flexible, reliable, and efficient framework for detecting geometric features. We proposed systematic characterizations for such features, developed a *url*-thresholding technique and two effective filtration rules, presented a simple and effective algorithm, and obtained very promising experimental results. We are currently applying the method to various applications. Our algorithm contains ten parameters, but the upper- and lower-bounds would not require tuning for specific problems. For the signal/noise ratios, we believe it is possible to determine optimal parameters automatically, which would enable a fully automatic framework for feature detection.

References

- [1] E. D. Bloch. *A First Course in Geometric Topology and Differential Geometry*. Birkhäuser, 1997.
- [2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2nd edition, 2000.
- [3] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.
- [4] P. J. Frey and P.-L. George. *Mesh Generation: Application to Finite Elements*. Hermes, 2002.
- [5] A. Hubeli, K. Meyer, and M. Gross. Mesh edge detection. Technical Report 351, ETH Zürich, Institute of Computer Systems, Dec. 2000.
- [6] X. Jiao. *Data Transfer and Surface Propagation in Multicomponent Simulations*. PhD thesis, University of Illinois at Urbana-Champaign, 2001.
- [7] X. Jiao and M. T. Heath. Efficient and robust algorithm for overlaying non-matching surface meshes. In *10th International Meshing Roundtable*, pages 281–292, 2001.
- [8] S. J. Owen and D. R. White. Mesh-based geometry: a systematic approach to constructing geometry from a finite element mesh. In *10th International Meshing Roundtable*, pages 83–96, 2001.
- [9] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.