

Identification of C^1 and C^2 Discontinuities for Surface Meshes in CAD

Xiangmin Jiao * Narasimha R. Bayyana

*College of Computing, Georgia Institute of Technology
266 Ferst Drive, Atlanta, GA 30332, USA*

Abstract

In computer-aided design and meshing, it is often important to identify the discontinuities (singularities) in coarse surface meshes. Due to the potential low resolution and noise, it is challenging to detect such features. We propose systematic characterizations of the edges, corners, and curves at the singularities of a mesh and develop algorithms to identify these geometric features. Our algorithms can detect C^1 discontinuities as well as typical C^2 discontinuities. We present the analysis of correctness of our method and demonstrate its effectiveness experimentally using stereolithographic (STL) meshes as well as coarse finite-element meshes of real-world objects.

Key words: Discontinuities, singularity analysis, feature identification, feature extraction, surface modeling, reverse engineering

1 Introduction

Surfaces and their discretizations play a fundamental role in computer-aided design and engineering as well as scientific simulations. In many applications, a surface may have a number of C^2 continuous patches, with a potential loss of C^1 or C^2 continuity along patch boundaries. In geometric processing, these C^1 and C^2 discontinuities constitute important features that require special attention in order to obtain highly accurate and reliable discretizations. In numerical computations, these singularities often also require special care to avoid “pollution errors” and potential associated instabilities. Nowadays, a surface geometry is often given by only

* Corresponding author. Current affiliation: Department of Applied Mathematics & Statistics, State University of New York, Stony Brook, NY 11794. Phone: (+1) 631 6511061.

Email addresses: xjiao@sunysb.edu (Xiangmin Jiao),
rao.bayyana@gmail.com (Narasimha R. Bayyana).

a surface mesh without a continuous CAD model, such as given by a stereolithographic (STL) mesh, or a mesh obtained from the solutions of dynamic numerical simulations. Identification of the singularities in these meshes is critical to remesh, segment, or reverse-engineer such surfaces [1, 2, 3, 4, 5] and has become increasingly important in CAD. Detection of similar types of features is also important for problems in computer graphics and visualization, such as mesh simplification [6], denoising [7], and image registration [8].

In this paper, we consider the problem of identifying C^1 and C^2 discontinuities (or singularities) in a surface mesh for CAD and meshing. To meet the needs of these targeted application areas, a good feature detector should satisfy a number of requirements, which we summarize as follows:

Reliability (Robustness): The detector must be reliable or robust in the sense that it can avoid *false negativeness* (i.e., without missing true discontinuities) and minimize *false positiveness* (i.e., without introducing false discontinuities) under realistic assumptions, it must be applicable to the *coarse* meshes used in CAD, and it must be numerically stable and can tolerate moderate *noise* in the meshes. In addition, it should identify not only edges but also *corners*.

Flexibility: The detector should be adjustable to meet the needs of different applications (e.g., to extract only the most salient C^1 discontinuities, all the C^1 discontinuities, or both C^1 and C^2 discontinuities).

Simplicity: The detector should be easy to implement and analyze. It should have as few parameters as possible, and the parameters should have *intuitive* geometric meanings with *default* values that rarely require tuning.

Achieving the above goals, especially the goal of reliability, is very challenging, because of the lack of precise mathematical description of the surface and the low resolution of the meshes. Due to the low mesh resolution, the noise in the mesh is often highly correlated instead of having a uniform or Gaussian distribution. In fact, it is difficult, if not impossible, even to compute the curvatures accurately for these coarse meshes or near singularities, as the curvatures are inherently sensitive to perturbation. The methods based on curvatures are therefore unreliable for detecting singularities for coarse meshes (see subsection 2.2 for more discussion). The methods based on dihedral angles alone are also sensitive to noise and often suffer from both false positiveness and false negativeness. In addition, CAD models often contain tapering singularities that gradually vanish, which are very difficult to identify. Figure 1 illustrates these difficulties using the fandisk model, for which most existing methods in the literature suffer from false positiveness and false negativeness similar to those in Figure 1(a-b). It should be intuitively clear that it is even more difficult to identify C^2 discontinuities. Furthermore, because the ridge curves may be connected in complex ways, the detection of corners poses additional challenges, which might be the reason why most previous methods omitted this important issue.

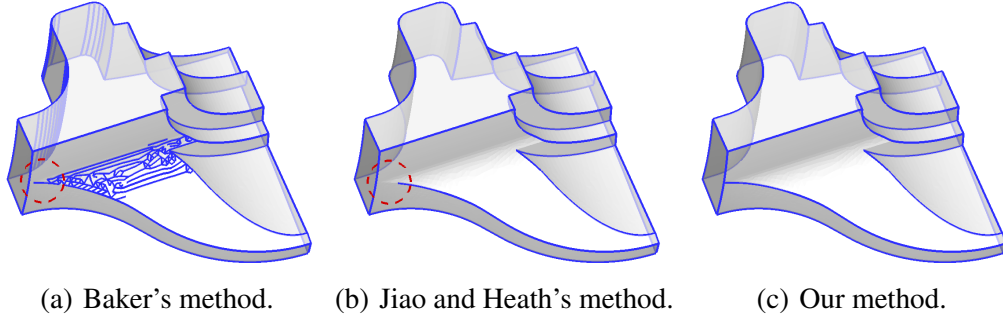


Figure 1. Example of detected C^1 curves of fan disk using previous methods in [4] (a) and in [9] (b) as well as our proposed method (c). Previous methods suffered from false negativity (as circled out) and/or false positiveness.

In this paper, we develop a method for detecting discontinuities (or singularities) and strive to meet the aforementioned requirements. The main contributions of this paper are as follows. First, we propose systematic characterizations of edges, corners, and curves at singularities. Our characterizations involve multiple measures, which are combined using a novel heuristic framework based on a probabilistic argument. Second, we construct new algorithms for detecting C^1 discontinuities and typical C^2 discontinuities that are incident on C^1 discontinuities and separate flat and non-flat regions. Our proposed method is defined strictly in a bottom-up fashion, which enables not only simpler control flow and analysis of our algorithms but also the detection of C^2 discontinuities. We provide analysis of the correctness of our algorithms and also demonstrate their effectiveness for STL meshes of CAD models as well as finite-element meshes of real-world objects.

The remainder of the paper is organized as follows. Section 2 clarifies some assumptions, reviews some background knowledge on discrete and differential geometry, and surveys some related work. Section 3 presents a set of measures that are useful for the identification of singularities. Section 4 describes our characterizations of C^1 discontinuities based on the measures and presents an algorithm for detecting them. Section 5 extends our characterizations and algorithm to detect C^2 discontinuities. Section 6 shows experimental results using our method for both C^1 and C^2 discontinuities. Section 7 concludes the paper with a discussion of applications and future research directions.

2 Background

2.1 Definitions and Assumptions

The subject of this paper is on the analysis of surface meshes. A surface mesh is a collection of topological entities of zero, one, or two dimensions along with their

incidence and *adjacency* relationships. We refer to the zero-dimensional entities as *vertices*, the one-dimensional entities *edges*, and the two-dimensional entities *faces*. Each edge in the mesh has two directed variants with opposite directions. Each directed edge is referred to as a *halfedge* and it is said to point from the *origin vertex* to the *destination vertex*. A halfedge is said to be *incident* on its origin vertex, and an edge is said to be *incident* on both of its vertices. We consider only *orientable* surfaces, and in particular two-manifold with or without boundary [10], and assume that the halfedges within each face form a counterclockwise loop with respect to the outward surface normal. Non-manifolds may be addressed by decomposing them into two-manifolds with boundary, which can then be processed separately. For ease of presentation, we will discuss only triangle surface meshes. However, the method that we develop is also applicable to quadrilateral meshes or mixed meshes with triangles and quadrilaterals.

We sometimes use terminology from differential geometry for smooth surfaces [11] and their counterpart for discrete surfaces [12]. For a curve γ in \mathbb{R}^2 , the magnitude of the *curvature* at a point $\mathbf{p} \in \gamma$ is equal to the inverse of the radius of the osculating circle at \mathbf{p} , and its sign indicates whether the curve is locally convex or concave. For a surface Γ in \mathbb{R}^3 , a *normal section curve* is the intersection of Γ with a plane orthogonal to Γ . The *principal curvatures* κ_1 and κ_2 of Γ are the maximum and minimum curvatures along the normal section curves passing through a point, and their corresponding tangent directions are the *principal directions*. The mean curvature is $(\kappa_1 + \kappa_2)/2$. The *Gaussian curvature* is $\kappa_1\kappa_2$.

For discrete surfaces modeling a smooth surface, the curvatures are defined as approximations to those of the underlying surfaces. From a numerical-analysis point of view, asymptotically convergent approximations can be computed from truncated Taylor series expansions of the height function of the surface [13, 14]. From a computational-geometry point of view, the Gaussian curvatures of smooth and discrete surfaces are connected by the Gauss-Bonnet theorem [10]. Let the *angle defect* (a.k.a. the *discrete Gaussian curvature*) at a vertex v be $2\pi - \sum_i \theta_i$, where θ_i denotes the interior angle at v in its i th incident face (cf., Figure 3(a)). The theorem states that the sum of the angle defect of a discrete surface is equal to the integral of the Gaussian curvature of a smooth surface with the same topology. Therefore, the angle defect at a vertex is an *integral measure* of the Gaussian curvature for some “control area” around the vertex. Analogously, let the *signed turning angle* at a vertex at a polygonal curve be the angle between the tangent directions of its incident edges, with a positive sign at convexity and negative sign at concavity. The turning angle of the curve is an *integral measure* of the curvature. These integral measures may not converge to the true integrals because it is difficult to define the control areas for general meshes, but they provide insights into the connection between smooth and discrete surfaces.

Most surfaces used in CAD and scientific computations are piecewise smooth. A surface patch is considered “smooth” if it has continuous second derivatives. A

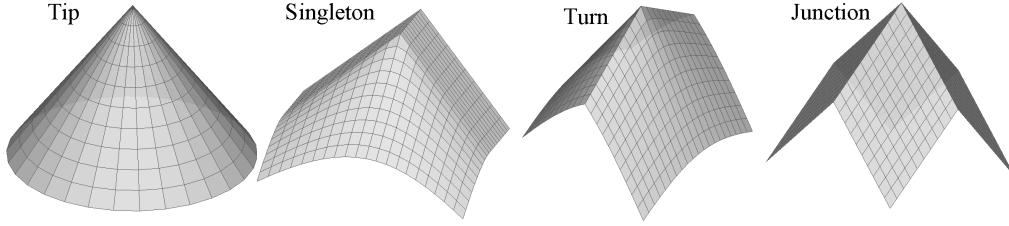


Figure 2. Different types of corners.

C^1 discontinuity corresponds to the lack of continuity in surface normal, so the curvature (especially the maximum curvature) is undefined. A C^2 discontinuity corresponds to the lack of continuity in curvatures. In this paper, the *features* of a surface refer to these C^1 and C^2 discontinuities, although this term may have very different meanings in some other literature. We assume that the discontinuities are composed of piecewise smooth curves, which we refer to as *feature curves* (and more specifically, C^1 and C^2 curves). Assuming a mesh models a piecewise smooth surface, the discontinuities or features in the mesh correspond to the discretization of the discontinuities in the underlying surface. We assume these discontinuities are composed of edges and vertices in the mesh, which we refer to as *feature edges* (or *ridge edges*) and *feature vertices*, respectively. The other edges and vertices are referred to as *non-features*. The halfedges of feature and non-feature edges are referred to as *feature* and *non-feature halfedges*, respectively. We refer to a vertex at the intersection of feature curves as a *corner*, and refer to the other feature vertices as *ridge vertices*.

A corner may be incident on zero, one, two, or more ridge curves, as illustrated in Figure 2. We refer to these cases as *tip*, *singleton*, *turn*, and *junction*, respectively. The specific definitions or characterizations of the discontinuities in a discrete surface, however, are an integrate part of the detection framework and will be addressed in later sections.

2.2 Related Work

In CAD and meshing, the characterization and detection of singularities have received insufficient attention until recently, even though the importance of these singularities is widely recognized. A common practice is to apply some thresholds to the dihedral angles of edges (see e.g., [1, 2]). In [9, 15], Jiao and Heath formalized a set of characteristics of C^1 discontinuities using dihedral angle, turning angle, angle defect, and ridge valence. They also developed an algorithm for detecting C^1 discontinuities. A key property of the algorithm in [9], which is inherited by this paper, was to improve robustness by exploring the collective properties of curves, instead of relying on isolated numerical values at vertices or edges. However, the algorithm in [9] was difficult to implement and analyze.

Inspired by [9], Baker developed a method in [4] and introduced a novel idea of comparing the tangent directions of edges against the principal directions at vertices for improved robustness. Baker also proposed to select the parameters in feature detectors based on the statistics of a given mesh. However, the statistical measures used in [4] were global properties and could not capture or adapt to the local properties near singularities. In our experimentation, Baker’s method often produced a large amount of false positiveness for complex models.

In [16], Jiao and Alexander developed a simple detector for C^1 discontinuities. That method was based on a concept called *medial quadric*, which is related to the medial axis [17], quadrics [18], tensor voting [19], and normal-vector voting [20]. More discussions of these techniques will be given in subsection 3.3. Some variants of that method were later used in dynamic surface meshes [21, 22] and feature-preserving mesh optimization [23], but they lacked effective filtering of false positiveness.

Not surprisingly, the detection of C^2 features is even less advanced. Recently, Yamakawa and Shimada [24] proposed a method, called *polygon crawling*, which can identify typical C^2 discontinuities for STL meshes generated from CAD models. However, their method is not applicable to finite-element meshes due to the potential unevenness introduced by surface mesh generators. More recently, Várady et al. developed a method for automatic extraction of surface structures based on discrete Morse theory [25], which can extract features similar to C^2 discontinuities for densely sampled meshes. As we will demonstrate later, our framework for detecting C^1 discontinuities can be easily extended to detect the C^2 discontinuities of interest to CAD and meshing.

Besides the detection of singularities for CAD and meshing, the more general problem of detecting “features” in discrete surfaces or images has been studied extensively in computer graphics, visualization, image processing, computer vision, and related fields. In these fields, the “features” are often defined differently from ours and are identified for different purposes. For example, some authors defined the features to be along the extremes of maximum curvature of smooth surfaces [26, 27, 28] or the boundaries between convex and concave patches [29, 30]. Sometimes, they are defined to be the “sharp” edges (e.g., [7, 31]), or the most “salient” singularities. These different types of features are typically identified for segmentation, simplification, denoising, or visualization of 3-D objects. Their emphasis is often on the visual effects instead of accuracy. Due to the difference in the targeted applications, the methods in those fields do not detect less salient features, such as tapering features or C^2 discontinuities.

Among the methods for visually important features, curvature-based methods are probably the most popular. They rely on estimation of curvatures [20, 28] or even the derivatives of curvatures [26, 32]. A number of methods have been proposed for computing or estimating curvatures for discrete surfaces; see [14, 33] and ref-

erences therein. A fundamental difficulty in curvature estimation lies in the fact that the curvatures are second-order derivatives and hence are inherently sensitive to perturbation. Some methods were proposed to control this instability (e.g., [32] and [34]). These stabilized estimators can deliver smoother or more visually pleasing results for noisy data, but they introduce errors and may in turn lead to less accurate feature detectors for less noisy meshes, which is evidential since broken feature curves are common for curvature-based feature detectors in the literature. Such results are hardly satisfactory for CAD and meshing.

Besides the intrinsic difficulties in curvature estimation, identification of discontinuities using curvature is a “chicken-and-egg” problem, because the computation of the curvatures assumes the smoothness of the local neighborhood. Some authors have reported issues regarding to the robustness of curvature estimations for point-set surface [35] and raised questions about their use in the detection of singularities [36]. We also observed that most existing curvature-estimators for surface meshes behaved erroneously near singularities and more robust estimators can be constructed by first filtering out the points across singularities.

3 Measures and Primitives

In this section, we describe a few measures that are useful for the identification of discontinuities in discrete surfaces. We strive to develop a comprehensive set of measures that can capture all the discontinuities and at the same time keep them as simple as possible. The variants of some of these measures have been used separately in the literature [4, 9, 16]. However, we present a more thorough analysis and describe simple and stable procedures to compute them. By themselves, none of these measures can provide a robust way for identifying feature edges or vertices, but they will serve as the building blocks for robust, collective identification of feature curves in later sections.

3.1 Angle-Based Measures at Vertices and Edges

We first present some angle-based measures, namely, angle defect, (one-sided) turning angle, and dihedral angle. Among these the turning angle and dihedral angle are unsigned. We explain their relationship with curvature-based measures and present additional indicators to distinguish convexity from concavity.

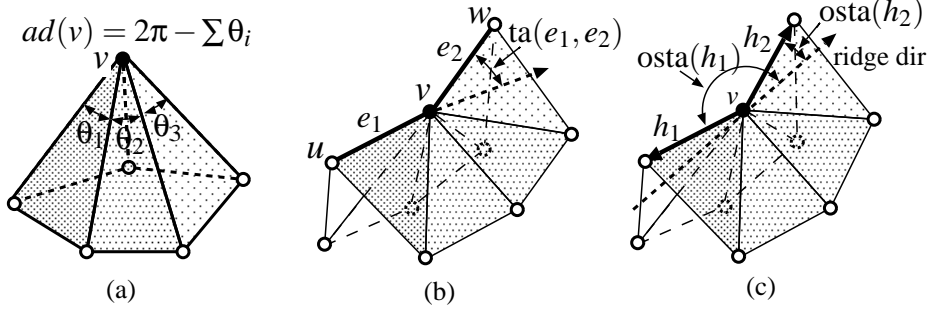


Figure 3. Illustrations of (a) angle defect, (b) turning angle, and (c) one-sided turning angle.

3.1.1 Angle Defect

As defined earlier (cf. section 2.1 and Figure 3(a)), the *angle defect* is an integral measure of the Gaussian curvature for closed smooth surfaces [10]. For two-manifolds with boundary, we extend its definition and define the angle defect at a *border vertex* to be the difference between 2π and twice of the sum of the interior angles at the vertex, i.e., $2\pi - 2\sum_i \theta_i$. The value of angle defect is no greater than 2π . It can be negative if the vertex is at a saddle point or its neighborhood is highly wrinkled.

Compared to pointwise Gaussian curvature, the angle defect has a number of advantages: It is well-defined at singularities and is scale invariant, with a clear geometric meaning. Its computation does not involve numerical differentiation and is insensitive to perturbation. Assume the vertices interpolate a given surface. Asymptotically, the angle defect vanishes at smooth regions and in the interior of ridge curves as the mesh is refined, but its magnitude remains large at corners (especially tips and singletons). Therefore, there is an inherent gap between its values at corners and those elsewhere. This inherent gap is highly desired for robust identification of tips and singletons, although the gap may be less pronounced for coarse meshes.

3.1.2 Turning Angle and One-Sided Turning Angle

We define the *turning angle* between two directed ridge edges uv and vw as the angle between their tangents, i.e., $\arccos(\hat{\mathbf{d}}_{uv}^T \hat{\mathbf{d}}_{vw})$, where $\hat{\mathbf{d}}_{uv} \equiv (\mathbf{v} - \mathbf{u}) / \|\mathbf{v} - \mathbf{u}\|$ and similarly for $\hat{\mathbf{d}}_{vw}$ (cf. Figure 3(b)). By definition, its value is between 0 and π . The turning angle is an unsigned integral measure of the curvature of a ridge curve at a vertex. This measure is scale invariant, is insensitive to perturbation (except for vertices adjacent to extremely short ridge edges), and has an inherent gap between its values at the turns of ridge curves and those within ridge curves. However, its definition requires two ridge edges and therefore is inconvenient to use for identifying feature edges.

To address this issue, we define a *one-sided turning angle (OSTA)* of a directed

edge (i.e., halfedge) to be the angle between its tangent direction and the ridge direction (or the principal direction along minimum curvature) at its origin vertex (see Figure 3(c)). A similar concept was used by Baker [4], but we use a different approach to evaluate the ridge direction as explained in subsection 3.3. Given a ridge vertex s , let \hat{e} denote its ridge direction. The OSTA of a halfedge vw is then $\arccos(\hat{e}^T \hat{d}_{vw})$. Let θ be the turning angle between the two ridge edges incident on v . The OSTAs of their corresponding halfedges originated from v are then $\theta_1 \approx \theta/2$ and $\theta_2 \approx \pi - \theta/2$, respectively.

3.1.3 Dihedral Angle

At edges, the simplest measure is the *dihedral angle*. Let \hat{n}_1 and \hat{n}_2 be the unit outward normals of the two incident faces of an edge. The dihedral angle between the two faces is then $\arccos(\hat{n}_1^T \hat{n}_2)$. We define the dihedral angle for a border edge to be π . The dihedral angle is widely used in the identification of discontinuities for its simplicity and computational efficiency. In addition, it shares some similar properties as angle defect and turning angle, namely, well-definedness at singularities, scale invariance, and an inherent gap between its values at singularities and elsewhere. It is also insensitive to perturbation except for edges incident on extremely poorly shaped triangles, for which face normals can be sensitive to perturbation.

Similar to angle defect and turning angle, the dihedral angle is connected to the curvature of a smooth surface, although less intuitively. Consider cutting the surface by a plane through a point on the edge (e.g., the midpoint) locally orthogonal to the edge and the surface. The plane intersects the discrete surface at a polygonal curve, and the dihedral angle is therefore an integral curvature measure (namely, the turning angle) of this polygonal curve. In other words, the dihedral angle is an approximate integral curvature measure of the underlying surface along the direction orthogonal to the edge.

3.1.4 Indicator of Convexity

Compared to the signed curvatures, the angle measures (in particular dihedral angle and turning angle) have some limitations, namely, their inability to distinguish convexity from concavity of the surface or a feature curve. This issue is not important for the identification of discontinuities, but it is critical for some other problems such as the identification of inflection points of a smooth surface. We therefore address this issue for completeness.

At an edge, the convexity of the surface is indicated by the *signed volume* bounded by its two incident faces. Given an edge vw , let $vw\mathbf{x}$ and $wv\mathbf{y}$ be its two incident faces, whose vertices are in counterclockwise order. The signed volume is one sixth of $\mathbf{d}_{vw}^T (\mathbf{d}_{vx} \times \mathbf{d}_{vy})$, where $\mathbf{d}_{vw} = \mathbf{w} - \mathbf{v}$ and similarly for \mathbf{d}_{vx} and \mathbf{d}_{vy} . A negative volume indicates convexity of the surface along the direction orthogonal to the

edge, and a positive volume indicates concavity. This signed volume recovers the sign of the dihedral angle. It is also an integral measure and hence is numerically stable.

To check the convexity of a feature curve locally at a ridge vertex, one may use the *signed area* bounded by its two incident ridge edges, which recovers the sign of the turning angle. For the OSTA, we must determine the convexity of a halfedge at its origin. Given an edge \boldsymbol{vw} , and let $\hat{\boldsymbol{e}}$ be the ridge direction and $\hat{\boldsymbol{n}}$ be the outward normal at its origin. The edge is convex at its origin if $\boldsymbol{d}_{vw}^T(\hat{\boldsymbol{e}} \times \hat{\boldsymbol{n}})$ is negative. We will address the estimation of $\hat{\boldsymbol{e}}$ and $\hat{\boldsymbol{n}}$ in subsection 3.3.

Unlike dihedral angle and turning angle, the angle defect is signed. Like Gaussian curvature, it can identify elliptic, parabolic, and hyperbolic points but cannot distinguish convexity from concavity. At a vertex the convexity of the surface must be classified with respect to a specific tangent direction due to the potential presence of saddle points. Given a vertex \boldsymbol{v} , let $\hat{\boldsymbol{n}}$ denote the outward normal at the vertex. Let \boldsymbol{w} be a point on the surface near \boldsymbol{v} (in particular, a point in a face incident on \boldsymbol{v}). The convexity of the surface along the direction \boldsymbol{d}_{vw} (or more specifically along its tangent component $\boldsymbol{d}_{vw} - \boldsymbol{d}_{vw}^T \hat{\boldsymbol{n}} \hat{\boldsymbol{n}}$) is indicated by $\boldsymbol{d}_{vw}^T \hat{\boldsymbol{n}} < 0$.

3.2 Relative Strengths and Ridge Valence

Because discontinuity is a local property, we define some relative measures of halfedges incident on a vertex. We say a halfedge h is *locally strongest* (or *l-strong*) *in OSTA* if (1) its OSTA is smaller than some threshold θ_t and is the *minimum* among the halfedges that share the same origin vertex with h and whose DA is greater than some small threshold θ_f , or (2) its OSTA is greater than $\pi - \theta_t$ and is the *maximum* among these halfedges. Let \boldsymbol{t}_h denote the tangent of a halfedge h , ϑ_h its dihedral angle, and φ_h its OSTA. In general, there are at most two halfedges that are l-strong in OSTA among the halfedges incident on a ridge vertex, which are $\arg \min_{\{h|\varphi_h < \theta_t \wedge \vartheta_h > \theta_f\}} \varphi_h$ and $\arg \max_{\{h|\varphi_h > \pi - \theta_t \wedge \vartheta_h > \theta_f\}} \varphi_h$, respectively.

Similarly, we say a halfedge h is *l-strong in DA* if (1) its DA is greater than θ_f and is the *maximum* among the halfedges that share the same origin with h , whose OSTAs are *smaller* than $\pi/2$, and whose DAs are greater than θ_f , or (2) its DA is greater than θ_f and is the *maximum* among the halfedges that share the same origin with h , whose OSTAs are *greater* than $\pi/2$, and whose DAs are greater than θ_f . In general, there are at most two halfedges that are l-strong in DA among the halfedges incident on a ridge vertex, which are $\arg \max_{\{h|\boldsymbol{t}_h^T \hat{\boldsymbol{e}} > 0 \wedge \vartheta_h > \theta_f\}} \vartheta_h$ and $\arg \max_{\{h|\boldsymbol{t}_h^T \hat{\boldsymbol{e}} < 0 \wedge \vartheta_h > \theta_f\}} \vartheta_h$, respectively, where $\hat{\boldsymbol{e}}$ is the ridge direction at the vertex. To support two-manifolds with boundary, a half-edge of a border edge is considered to be l-strong both in DA and OSTA.

To protect some feature edges and corners with very large angles, we say an edge is *unconditionally strong* (or *u-strong*) in DA if its dihedral angle is larger than some large threshold θ_F . We say an edge is *stronger in DA* than another if it has a larger dihedral angle. For additional protection at the end-edges of feature curves, we introduce another threshold θ_e (where, $\theta_f \leq \theta_e \leq \theta_F$) and say an edge is *e-strong* if its dihedral angle is greater than θ_e . We say a vertex is *u-strong in AD* if the magnitude of its angle defect is greater than some threshold θ_D , which is useful in identifying sharp tips or singletons (cf. Figure 2(a) and (b)). Similarly, we say a vertex is *u-strong in TA* if its turning angle is greater than some threshold θ_T (where $\theta_T \geq 2\theta_t$), which is useful in identifying the sharp turns between ridge curves (cf. Figure 2(c)). At junctions (cf. Figure 2(d)), AD and TA may be arbitrarily small. To identify them robustly, we must consider the *ridge valence* at a vertex, i.e., the number of its incident ridge curves.

The computations of the DA, TA, and OSTA involve arccos of some inner product of unit vectors. For efficiency, the comparison of these measures can be implemented as direct comparison of the inner products, although our discussions always refer to the angles for clarity.

3.3 Ridge Direction and Vertex Normal

In some of the above measures, it is critical to compute the ridge directions and vertex normals reliably. Some previous work computed them using polynomial fitting [4], which is not meaningful near singularities and hence is unreliable. We present a method using the *medial quadric* [16], which uses an eigenvalue analysis to solve a least-squares problem locally at each vertex.

The medial quadric is formulated to approximate the direction from a vertex toward the points that minimize the weighted sum of squared distances to its incident faces. This direction points toward the medial axis [17] at well-defined ridges and corners, and its least-squares formulation leads to a quadric [18], so we call this technique the *medial quadric*. Given a vertex, let $\widehat{\mathbf{m}}_j$ be the face normal of its j th incident face. The medial quadric is given as a 3×3 linear system $\mathbf{M}\mathbf{d} = \mathbf{b}$, where

$$\mathbf{M} = \sum_j \omega_j \widehat{\mathbf{m}}_j \widehat{\mathbf{m}}_j^T, \quad \mathbf{b} = \sum_j \omega_j \widehat{\mathbf{m}}_j, \quad (1)$$

and ω_j is the face area associated with the j th face. We refer to \mathbf{M} as the *normal tensor*.

The normal tensor is symmetric and positive semi-definite (i.e., $\mathbf{M} = \mathbf{M}^T$ and $\mathbf{x}^T \mathbf{M} \mathbf{x} \geq 0$ for $\mathbf{x} \in \mathbb{R}^3$). Consider the eigenvalue decomposition $\mathbf{M} = \sum_{i=1}^3 \lambda_i \widehat{\mathbf{e}}_i \widehat{\mathbf{e}}_i^T$, where the λ_i are the eigenvalues with $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$, and $\widehat{\mathbf{e}}_i$ are the corresponding eigenvectors. Figure 4 illustrates the eigenvalues and eigenvectors of the

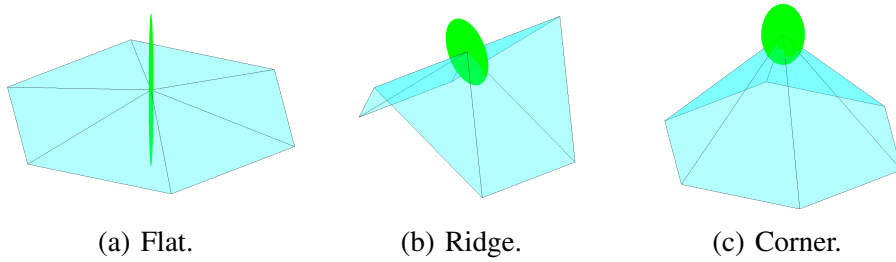


Figure 4. Correlation of eigenvalues of normal tensor and local structures at singularities. Normal tensor has one large eigenvalue at flat regions (a), two large eigenvalues at ridges (b), and three large eigenvalues at sharp corners (c).

normal tensor for three different cases, where the axes of the ellipsoids are aligned with the directions of eigenvectors and the semiaxes are proportional to eigenvalues. The eigenvalues provide an indication of singularity at the vertex. If the surface is smooth and well resolved at v , then the face normals are nearly identical around a vertex. Therefore $\lambda_2 \ll \lambda_1$ and $\lambda_3 \ll \lambda_1$, and the matrix \mathbf{M} has rank one. However, at a ridge vertex with two distinct normal directions, $\lambda_3 \ll \lambda_2$ and \mathbf{M} has rank two. At a corner, all three eigenvalues have similar sizes and \mathbf{M} has a full rank. The above observation is essentially the foundation of normal-vector voting [20]. At non-acute ridges where the dihedral angle θ between its two sides is smaller than $\pi/2$, the relative ratio of the eigenvalues provides a reliable estimation of the dihedral angle (in particular, $\lambda_2/\lambda_1 \approx \tan^2(\theta/2)$), so it is a fairly reliable indicator for non-acute ridges [16]. However, the normal tensor cannot distinguish a near cusp from a flat surface because of its squares nature, so the techniques based on it alone is unreliable in the presence of acute ridges, as pointed out in [16].

Instead of using it as an indicator of ridges, we use the normal tensor to estimate the ridge directions: At a ridge vertex, the null space of \mathbf{M} , i.e., $\hat{\mathbf{e}}_3$ in this case, is aligned with the ridge direction. The sign of the eigenvector $\hat{\mathbf{e}}_3$ is immaterial, so is the sign of the ridge direction, as long as we always use the same signed direction at a vertex. The eigenvector $\hat{\mathbf{e}}_3$ is well-conditioned at ridge vertices but is ill-conditioned if $\lambda_2 \approx \lambda_3$. Therefore, we use the ridge direction only if the surface is not flat at the vertex (in particular, if $\lambda_2/\lambda_1 \geq \varepsilon$ for some small $\varepsilon \approx \tan^2(\theta_f/2)$ and $\lambda_3/\lambda_2 \leq 0.7$).

To approximate the vertex normal, we use the direction \mathbf{d} within the domain space of \mathbf{M} , i.e.,

$$\mathbf{d} = \sum_{\{i|\lambda_i \geq \varepsilon\lambda_1\}} \mathbf{b}^T \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i / \lambda_i. \quad (2)$$

The filtering of small eigenvalues ensures numerical stability. The vector \mathbf{d} approximates the true normal to first-order accuracy at smooth regions, and the sign of \mathbf{d} is guaranteed in the sense that $\mathbf{d}^T \mathbf{b} > 0$. If the weights do not vary by orders of

magnitude, the direction \mathbf{d} is insensitive to the weights and produces well-aligned normals along singularities, as demonstrate in [16].

4 Identification of C^1 Discontinuities

We now present our method for identifying discontinuities using the aforementioned measures, first focusing on C^1 discontinuities and then extending to C^2 discontinuities in the next section. Unlike their counterparts for smooth surfaces, these discontinuities cannot be easily characterized using simple mathematical terms. We will first select *candidate edges* and *vertices* by combining the aforementioned measures to avoid false negativeness and to minimize false positiveness, and then connect these edges and vertices into curves to filter out false positiveness based on empirical rules.

4.1 Candidate Vertices and Edges

Observe that the feature edges tend to have large dihedral angles and small turning angles within a feature curve. Therefore, the feature halfedges are mostly likely to be u-strong in DA or l-strong in DA or OSTA at their origin vertices. However, some feature halfedges may not be u-strong or l-strong especially at the intersections of different curves. To avoid missing feature edges (i.e., false negativeness), we thus must pay special attention to the potential corners, especially the junctions. On the other hand, some non-feature halfedges may be u-strong in DA due to noise or be l-strong in DA or OSTA especially along the directions of minimum curvatures. These false feature edges may cause interference to the feature curves and too many of them can limit the effectiveness of the later filtration.

For effective selection of candidate edges and vertices, we explore the connectivities of the halfedges at their vertices:

Attachment: A halfedge h is *attached* to its origin vertex v if the dihedral angle of h is greater than θ_f and (1) h is l-strong in *either* DA or OSTA, or (2) v is incident on a *sharp edge* or is a *sharp corner* or an *ambiguous vertex*.

Strong attachment: h is *strongly attached* to v if the dihedral angle of h is greater than θ_f and (1) h is l-strong in *both* DA and OSTA, or (2) h is a *sharp edge*, or v is a *sharp corner* or an *ambiguous vertex*.

Strong attachment implies attachment. Part one of the above definitions addresses the more general cases and is relatively intuitive. An edge is said to be a *sharp edge* if it is u-strong in DA. A vertex is said to be a *sharp corner* if it is u-strong in AD. A vertex is said to be an *ambiguous vertex* if its ridge direction (and in turn OSTA)

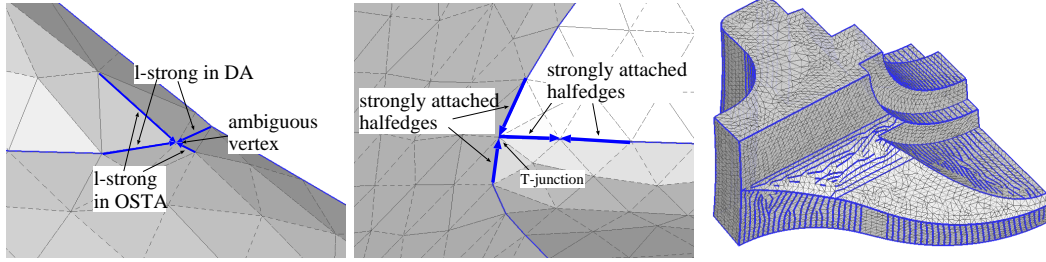


Figure 5. Examples of ambiguous vertex (left) and T-junction (right). Figure 6. Quasi-strong edges of fan disk model.

is undefined due to a full-rank normal tensor (i.e., $\lambda_2/\lambda_1 \geq \varepsilon$ and $\lambda_3/\lambda_2 \approx 1$, cf. subsection 3.3). Figure 5(left) shows an example of an ambiguous vertex. Near these edges and vertices the feature edges are likely to be not l-strong in both DA and OSTA and hence we include them in the criteria to avoid false negativity. Based on attachedness, we then characterize vertices and edges as follows:

Strongly attached vertex: A vertex is *strongly attached* if at least one halfedge is *strongly attached* to it.

Quasi-strong halfedge: A halfedge is *quasi-strong* if (1) both of its vertices are *strongly attached* and the halfedge itself is *attached* to its origin, or (2) each of its vertices has two strongly attached halfedges and its opposite halfedge is l-strong in both DA and OSTA.

Quasi-strong edge: An edge is *quasi-strong* if both of its halfedges are *quasi-strong*.

The above characterizations are fairly intuitive except for the second condition of quasi-strong halfedges. This condition protects the T-shaped junctions as illustrated in Figure 5(right), where the “side” candidate edge may not be attached and hence requires this special treatment. The quasi-strong edges constitute a core concept of our method. We select the quasi-strong edges as the *candidate edges*, and select the vertices of the quasi-strong edges along with the sharp corners as the *candidate vertices*. Figure 6 shows the quasi-strong edges in a mesh of the fan disk model with $\theta_f \approx 1^\circ$ and the other default parameters given in subsection 4.3.

In the definition of quasi-strong edges, the use of strongly-attached vertices may appear to be somewhat complex. To show why this definition is desirable, we compare it with three simpler alternatives: First, one may require both halfedges of a candidate edge to be l-strong in both DA and OSTA. Second, one may require both of the halfedges to be l-strong in either DA or OSTA. Third, one may require one halfedge to be l-strong in both and the other l-strong in either.

We analyze these alternatives using a simple probabilistic argument, focusing on only the more general cases and omitting the special treatments of sharp edges, sharp corners, ambiguous vertices, and T-junctions. Let p_1 and p_2 denote the probabilities for a feature halfedge being not l-strong in DA and OSTA, and q_1 and q_2

be the probabilities for a non-feature halfedge being l-strong in DA and OSTA, respectively. In general, p_1 and p_2 are close to zero, but q_1 and q_2 can be fairly large (as large as $2/d$, where d is the valence of a vertex). For simplicity, assume these probabilities are independent of each other. The first alternative is unacceptable because the probability of false negativeness (i.e., missing a candidate edge) is $2(p_1 + p_2 - p_1p_2)$, which is too large. For the second alternative, the probability of false positiveness (i.e., having a false candidate edge) is $2(q_1 + q_2 - q_1q_2)$, which is close to 1. For the third alternative, the probability of false negativeness is quadratic in p_1 and p_2 , which is small, but the probability of false positiveness is cubic in q_1 and q_2 , which may still be too large because q_1 and q_2 are far larger than p_1 and p_2 .

In comparison, for the quasi-strong definition, the probability for one of the incident ridge halfedge of an interior vertex of a ridge curve being not strongly attached is $p_v = p_1 + p_2 - p_1p_2$, so the probability for the vertex being not strongly attached is p_v^2 , which is quadratic in p_1 and p_2 . The probability for a ridge halfedge being not attached is p_1p_2 . Since a candidate edge is not quasi-strong only if one of its vertices is not strongly attached or one of its halfedge is not attached, its probability is quadratic in p_1 and p_2 , which is very small. In terms of false-positiveness, first consider a non-feature edge not incident on a feature vertex. The probability for one of its vertices being strongly attached is quadratic in q_1 and q_2 . For the non-feature edge to be quasi-strong, both of its vertices must be strongly attached, so its probability is therefore at least quartic in q_1 and q_2 . Next, consider a non-feature edge incident on a ridge vertex v away from sharp edges and corners. Its corresponding halfedge h is attached to the vertex only if the ridge halfedges incident on v are not strongly attached, so the probability for h being attached is no greater than $\min\{2(q_1 + q_2), p_1, p_2\}$, and the probability for its corresponding edge to be quasi-strong is quadratic in p_1 and p_2 , which is again very small. Therefore, the quasi-strong criteria deliver a good balance between false negativeness and false positiveness in general.

As evidential in Figure 6, when using the quasi-strong criteria alone false positiveness can still occur quite often along the directions of minimum curvatures and near sharp edges and sharp corners. Therefore, additional filtration is needed to eliminate these false candidate edges. Fortunately, because of the low probability of false positiveness for the general cases, the non-feature candidate edges tend to be connected in relatively simple patterns, so we can devise some simple and effective filtration procedures as we discuss in the next subsection.

Before concluding this subsection, we summarize the algorithm for identifying quasi-strong edges.

- 1: for each vertex, compute AD, compute normal tensor to evaluate ridge directions, and identify sharp corners and ambiguous vertices;
- 2: for each halfedge, compute its DA and OSTA; determine their local extreme values at vertices;
- 3: for each halfedge, determine whether it is u-strong, l-strong, e-strong, attached,

- or strongly attached;
- 4: for each vertex, determine whether it is strongly attached;
- 5: for each edge, determine whether it is quasi-strong.

The output of this procedure includes a list of candidate halfedges, the flags indicating their strongness and attachedness, and the flags indicating whether each vertex is a sharp corner or ambiguous vertex. The e-strong halfedges are not needed during this procedure but will be used in the later filtration step. To assess the efficiency of our algorithm, assume the valences of vertices are bounded by a small constant, and it takes constant time to evaluate each angle measure or to retrieve the adjacent faces of a given face, which can be achieved with the aid of a halfedge data structure. The cost of this step is proportional to the total number of edges of the mesh (denoted by n). Let m denote the number of candidate edges. If a binary tree (such as the map in C++ STL) is used for ease of implementation in the bookkeeping of candidate edges, the time complexity would then be $O(n + m \log m)$, which would still be $O(n)$ if $n = \Omega(m \log m)$. In terms of memory requirement, our algorithm requires only storing the angle measures at vertices and some flags at vertices and halfedges, besides the halfedge data structure.

4.2 Classification and Filtration of Candidate Curves

After selecting the candidate edges and vertices, we then connect them into *candidate curves* for filtration. Similar to feature curves, the candidate curves may be open or closed and be connected at junctions or turns. To characterize these candidate curves, we must pay special attention to these junctions and turns. We start by defining four different types of connections of a candidate halfedge h at its origin vertex. The first two types correspond to halfedges whose origin vertices have only one or two incident candidate halfedges.

Singleton and dangling: h is said to be a *singleton* halfedge if it is the only candidate halfedge incident on its origin. A singleton halfedge is *dangling* if it is not u-strong in DA or its origin is not a sharp corner.

Semi-joint: h is *semi-joint* if its origin v has two incident candidate halfedges, v is a sharp corner or the turning angle between the two candidate halfedges is greater than θ_T , and at least one candidate edge incident on v is not u-strong.

The dangling and semi-joint halfedges are very common; see Figure 7 for some examples. An example of a singleton but non-dangling halfedge is a halfedge incident on a singleton corner vertex (cf. Figure 2). For semi-joint end-edges, the additional condition of u-strong edges protects against false negativeness of under-resolved sharp ridge curves.

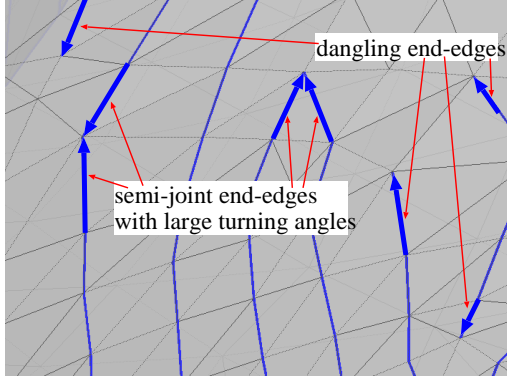


Figure 7. Examples of dangling end-edges and semi-joint end-edges.

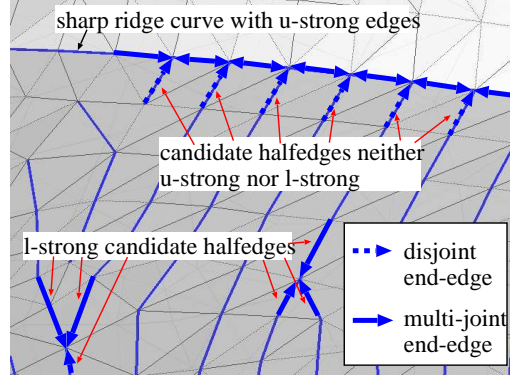


Figure 8. Examples of disjoint end-edges and multi-joint end-edges.

The last two types correspond to the halfedges whose origins have three or more incident candidate halfedges, and we classify them based on whether a halfedge is l-strong in DA or OSTA with additional safeguards near the edges that are u-strong or e-strong in DA:

Disjoint: h is said to be *disjoint* if its origin has three or more incident candidate halfedges, and one of the following is true: (1) h is l-strong in neither DA nor OSTA, it is not u-strong in DA, its origin vertex is neither a sharp corner nor an ambiguous vertex, (2) h is not l-strong in both DA and OSTA, it is not e-strong in DA, and its origin vertex is neither a sharp corner nor an ambiguous vertex, (3) its origin is incident on an edge u-strong in DA and h is not e-strong in DA, or (4) its origin is incident on an acute edge (i.e., a non-border edge whose dihedral angle is greater than 90°) and h is not u-strong in DA.

Multi-joint: h is said to be a *multi-joint* if its origin has three or more incident candidate halfedges and h is not disjoint.

The first two conditions of disjoint end-edges occur more often. Figure 8 shows some examples of such end-edges. The other two conditions counteract the frequent false positive quasi-strongness near u-strong and acute edges, respectively.

Formally, a *candidate curve* is a sequence of candidate edges connected at candidate vertices, forming either a closed loop or an open curve whose end-edges are singleton, semi-joint, disjoint, or multi-joint. We say the dangling, semi-joint, or disjoint end-edges are *obscure end-edges*. We say a candidate curve is a *feature candidate curve* if it is a sub-curve of a C^1 curve and say it is a *non-feature candidate curve* otherwise.

Observe that the end-edges of non-feature curves tend to be obscure. Furthermore, although non-feature curves may be long, they tend to have few edges with large dihedral angles (specifically, with no more than k edges with DAs greater than θ_k for some k and θ_k , and the probability of false positiveness decreases exponentially in k). Based on these observations, we then classify candidate curve as follows:

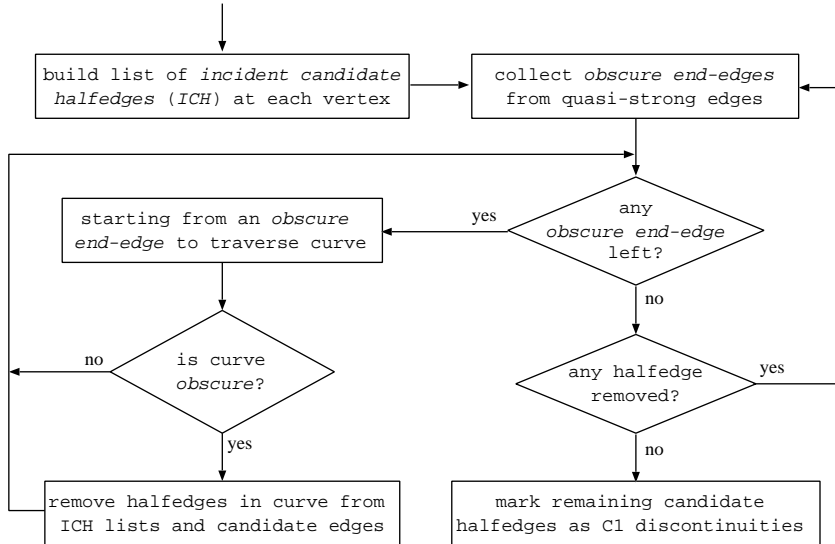


Figure 9. Control flow of filtration of quasi-obscure curves.

Salient curve: A candidate curve is *salient* if it is a closed curve or its end-edges are non-obscure.

Obscure curve: A non-salient candidate curve is said to be *obscure* if (1) both of its end-edges are obscure or one of its end-edges is dangling, and it has fewer than k edges with DAs greater than θ_k , or (2) one of its end-edges is obscure, both of its end-vertices are incident on edges that are u-strong in DA, and the curve does not have any edge with DA greater than θ_k .

Semi-salient curve: A curve is *semi-salient* if it is neither salient nor obscure.

Most non- C^1 curves, as those in Figure 6(a), satisfy the first condition of obscure curves. However, because the quasi-strong criteria (in particular, the attachment criteria of halfedges) are relatively loose for halfedges next to sharp edges, a non- C^1 curve sometimes may have a non-obscure end-edge, which are guarded against by the second condition of obscure curves.

In addition, a non- C^1 curve may be classified as salient or semi-salient curves in the presence of other non-feature curves; for example, if both end-edges of a non-feature curve are multi-joint. These non-feature curves are topologically unstable in that they would become obscure after removing other obscure curves. We say a candidate curve is *level-1 quasi-obscure* if it is obscure, and it is *level- i quasi-obscure* for $i > 1$ if it would become obscure after removing the edges in the lower-level quasi-obscure curves.

We now describe the procedure for filtering the quasi-obscure curves (cf. Figure 9). We compute the connectivity of candidate edges and then classify the resulting candidate curves. For ease of implementation and efficiency, we do not explicitly construct a list of the candidate curves but instead maintain a list of incident candidate halfedges (ICH) for each vertex to facilitate the traversal of the curves.

- 1: for each vertex, create a list of ICH;
- 2: collect obscure end-edges and determine their types;
- 3: starting from each obscure end-edge, traverse its corresponding candidate curve using ICH lists as connectivity table, determine whether the curve is obscure, and remove halfedges from ICH and candidate edge lists if obscure;
- 4: if any obscure curve was found, go back to step 2; otherwise, return remaining candidate halfedges as C^1 feature edges and finalize the classification of candidate vertices.

The identification of end-edges and obscure curves are based on the criteria described earlier. Step 3 is the core of the above procedure; during the k th iteration this step removes the obscure edges that constitute the level- k quasi-obscure curves. When the procedure terminates, all the quasi-obscure curves are filtered out (note that these quasi-obscure curves will be used again later for identifying C^2 curves). In the procedure, we traverse the curves using the ICH lists. In particular, from a halfedge h whose opposite halfedge is h' , we visit a halfedge g if g and h' are the only two halfedges in the ICH list of the destination vertex of h and they are not semi-joint.

In terms of efficiency, the computational cost of this procedure is linear time in the number of candidate edges, given that deleting a candidate edge takes constant time. In terms of memory requirement, the only additional memory is the ICH lists, whose total size is linear in the number of nodes plus the number of candidate edges.

Analysis of Correctness

An effective filter should avoid false negativeness under realistic conditions. To help the understanding of applicability and limitations of our filtration procedure, we formulate some sufficient conditions for feature curves under which no feature edge is contained in quasi-obscure curves:

- (1) All the feature edge are quasi-strong, and in the interior of feature curves all feature halfedges are u-strong in DA, l-strong in DA or OSTA, or incident on ambiguous vertices.
- (2) All the end-edges incident on *junction* vertices are multi-joint.
- (3) All the end-edges incident on *turn* vertices are u-strong in TA.
- (4) All the end-edges incident on *singleton* vertices are sharp edges or incident on sharp corners.
- (5) Each feature curve with semi-joint end-edges has more than k edges with DAs greater than θ_k .

A feature curve is said to be *quasi-salient* if it satisfies all of the above conditions. In the appendix, we show that quasi-saliency is sufficient to avoid false negativeness.

However, quasi-saliency is not a necessary condition. In particular, our method can also identify C^1 curves with disjoint end-edges or tapering ends if the curve contains more than k edges with DAs greater than θ_k and there is no non-feature candidate halfedge near the ends. Observe that most C^1 curves are quasi-salient or well-resolved near the end-edges, and non-feature curves are typically quasi-obscure, so after filtration only C^1 curves tend to remain.

4.3 Selection of Parameters

Our algorithm uses a few parameters. We categorize these parameters into five groups, with an abbreviation FDTEK:

- θ_f, θ_F : thresholds for l-strong and u-strong dihedral angles (in general, $\theta_F \gg \theta_f$)
- θ_D : threshold for u-strong angle defect
- θ_t, θ_T : thresholds for u-strong OSTA or turning angle (in general $\theta_T \geq 2\theta_t$)
- θ_e : threshold for e-strong DA of end-edges (in general $\theta_f \leq \theta_e \leq \theta_F$)
- k, θ_k : for long feature curves with obscure ends

The first four parameters ($\theta_f, \theta_F, \theta_D$, and θ_t) are used in the identification of quasi-strong edges and strong corners, and the latter four are used in the filtration of quasi-obscure curves. We emphasize that our algorithm primarily relies on the relative strengths of the measures (i.e., l-strongness). The above parameters are primarily for guarding against extreme cases (in particular, u-strongness, e-strongness, and semi-obscure curves), and therefore most of these parameters can typically use a set of default values without much tuning and hence can be made transparent to users. An exception is θ_f , which may be adjusted based on whether fine features (such as C^2 curves) are desired. From our experiments, we obtained the following default values: $\theta_f \approx 10^\circ$ (or 1° if C^2 curves or tapering features are desired), $\theta_F \approx 65^\circ$, $\theta_D \approx 60^\circ$, $\theta_T \approx 40^\circ$, $\theta_t \approx 20^\circ$, $\theta_e \approx 25^\circ$, $\theta_k \approx 50^\circ$, and $k \approx 5$, which we used in all of our tests unless otherwise noted.

5 Identification of C^2 Discontinuities

We now extend our framework to identify C^2 discontinuities. A piecewise smooth surface is discontinuous in curvatures along the tangent direction orthogonal to the C^2 curves. These curves are important for remeshing or reverse engineering of CAD models from STL meshes, so we focus on such meshes. Because of highly sparse and irregular mesh connectivities, it is impractical to compute the principal curvatures accurately for these coarse meshes. By using the angle measures (which are integral curvature measures), mesh density, along with our priori knowledge of quasi-strong edges and C^1 discontinuities, it becomes relatively straightforward to

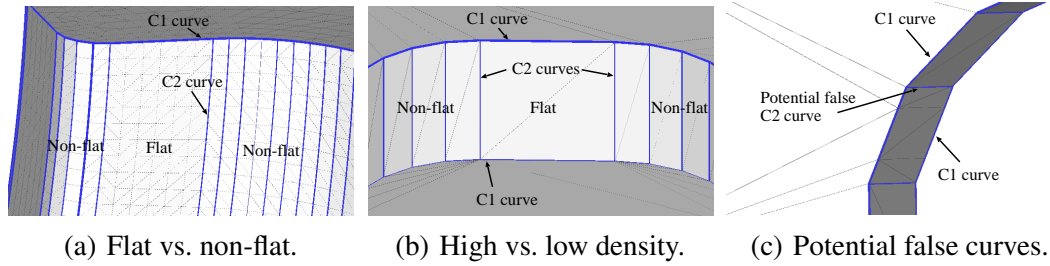


Figure 10. Examples of C^2 curves (a and b) and potential pitfalls (c).

extend our method to detect typical C^2 curves as we define shortly.

5.1 Characterization of C^2 Discontinuities

Similar to the identification of C^1 discontinuities, we extract some basic properties of C^2 discontinuities based on experimental observations. Figure 10(a) and (b) show two examples of C^2 discontinuities. These and other typical C^2 discontinuities in CAD models have the following properties:

- (1) C^2 curves separate nearly flat and non-flat (convex or concave) regions;
- (2) edges in C^2 curves are quasi-strong for some small θ_f (e.g., $\approx 1^\circ$);
- (3) end-edges in C^2 curves are obscure and attached to vertices in C^1 curves.

The first property above is the key, and it has also been observed by Yamakawa and Shimada in [24]. The second property indicates that we can extract C^2 curves from the quasi-obscure curves. In particular, there are likely quasi-strong edges on the non-flat side of C^2 curve but not on the flat side. The third property allows us to take advantage of C^1 feature curves for additional safeguards. We refer to the quasi-obscure curves that satisfy the above conditions as *semi-obscure*. These semi-obscure curves include the C^2 curves of practical interests but may also contain some false positiveness near inflection points of a surface. Such curves may be filtered out with further post-processing by determining the convexity or concavity of the curves, but we omit their treatment in this paper. In addition, the characterization does not capture C^2 curves that are closed loops, because they are very rare and would have been classified as salient curves in the previous steps.

5.2 Identification of Semi-Obscure Curves

The basic idea of our algorithm is to first identify the C^2 discontinuities in C^1 curves, traverse quasi-obscure curves from those C^2 discontinuities to classify the flatness of each side of its edges, and then classify the two sides of the curve based on the edge classifications.

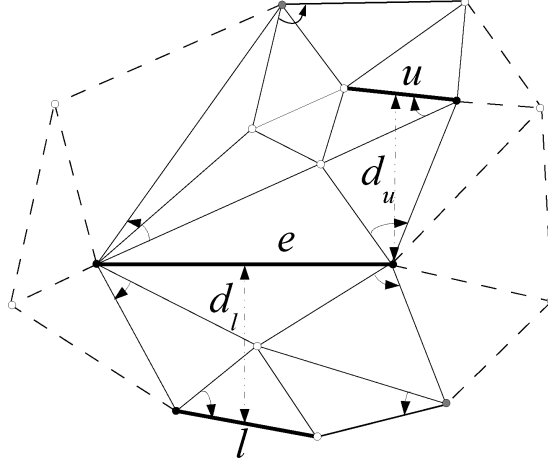


Figure 11. Identification of nearby quasi-strong edges on each side of quasi-strong edge e .

Given a non- C^1 quasi-strong edge e , intuitively we may determine the flatness of one side of e by finding a “parallel” edge of e and classify based on the flatness of that edge. Because of mesh irregularity, however, finding such parallel edges is difficult and unreliable. We instead first locate potential end-vertices of C^2 curves by identifying the C^2 discontinuities in C^1 curves, which simplifies the problem for its lower dimensionality. A vertex is at a C^2 discontinuity of a C^1 curve if the curve is nearly straight on one side but curved on the other side. The curvedness at a vertex is indicated by a turning angle greater than θ_f or an incident non- C^1 quasi-strong edge. This indicator alone may lead to false negativity if the straight side is a single long edge (cf. Figure 10(b)), and may lead to false positivity if an edge on a curved side is subdivided by the mesh generator (cf. Figure 10(c)). To avoid false negativity, if both sides are classified as curved, we count the number of curved vertices on the side with the shorter edge within the distance equal to a fraction (e.g., 0.8 times) of the longer edge. If the number is greater than one, then the curvature varies substantially, so the vertex is considered to be a C^2 discontinuity. To address the false positivity, if the side with the shorter edge is straight but the other side is curved, we further check whether there is any curved vertex within a distance slightly larger than the longer side (e.g., ≥ 1.5 times).

After identifying the potential end-vertices, we traverse their corresponding quasi-obscure curves to classify the edges. In particular, for each edge e in a curve, we determine whether there is a quasi-strong edge that does not fall onto a different side of a C^1 curve than e , forms a small angle ($\leq \theta_t$) with e , and is connected with e by another edge. Such an edge is found by visiting each edge l incident on the vertices of e and then visiting the edges incident on the other vertex of l without crossing over C^1 curves, as illustrated in Figure 11. We also compute the average of the orthogonal distances from the mid-points of the quasi-strong edges to e on each side. If there is no non- C^1 quasi-strong edge found on one side, then this side is flat. If both sides have one or more non- C^1 quasi-strong edges, but the distance on one side is substantially larger (e.g., ≥ 2.5 times larger) than the other side,

then the former side is classified as flat. Otherwise, a side is classified as non-flat. Finally, we classify a side of a curve to be flat if the side is flat for the majority of its edges, similar for non-flatness. A curve is considered semi-obscure if its two sides are classified differently.

6 Experimental Results

In this section, we present some experimental results for detecting C^1 and C^2 discontinuities using our method and compare it against some other methods. Verification of feature detection is in general a difficult process, because the “correctness” is often undefined. For some models, we may rely on the inspection of the experts. For some others, we have to verify indirectly by checking the consistency of the results, such as cross-checking the output for different meshes of the same model or the symmetric regions of a surface. The “correctness” for graphics models cannot be easily verified even indirectly, and also those models are less relevant to our problem, so we do not use graphics models in our tests but use finite-element and STL meshes of man-made objects, as commonly used in CAD. Our method was applied directly without any modification or pre-processing of the input meshes. All of our tests used the default parameters unless otherwise noted.

6.1 Results for C^1 Discontinuities

We first present results on C^1 discontinuities using finite-element meshes. Additional results for STL meshes will be given in the next subsection. Finite-element meshes are often very rough and contain noise and unevenness due to the low resolution of the mesh or the numerical/geometric errors during mesh generation. The roughness of these meshes can hinder the detection of singularities. Figure 12 shows the results for a mesh discretizing half of a Boeing aircraft. This mesh contains 5,023 vertices and 10,326 triangles. The detected features (including the border curves) are shown in blue. The airplane is in translucency to visualize the feature curves obstructed by the airplane body. Due to the low resolution of this mesh, it caused significant difficulties to the other methods, including the method in [9] and our own implementation of Baker’s method in [4]. As shown in Figure 12(a), the method in [9] caused false negativeness at the cockpit and false positiveness at the nozzle, as circled out in the figure. In Figure 12(b), Baker’s method introduced a large amount of false positiveness when using the default parameters in [4]. In comparison, Figure 12(c) shows the result using our method, and the correctness of the result was confirmed by our collaborators at Boeing.

To show why the above problem is challenging, Figure 12(d) and (e) show the zoomed-in views of the cockpit and the nozzle. Near the cockpit, the feature curve

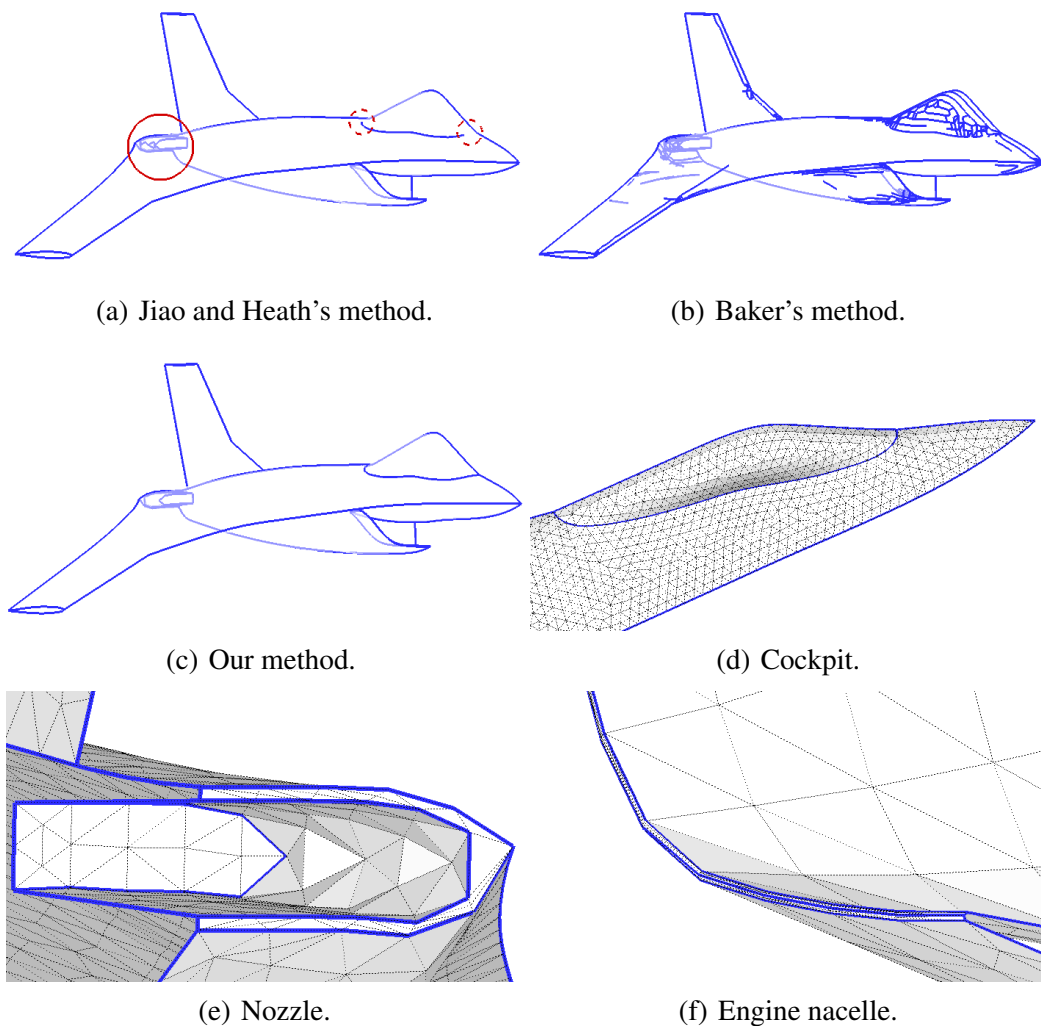


Figure 12. Comparison of detected C^1 curves for Boeing airplane. (a) Previous method in [9] led to false positiveness at nozzle and missed edges at cockpit. (b) Method in [4] introduced a large amount of false positiveness. (c) Our method delivered correct results as verified by Boeing’s expert, with zoomed-in view of the detected C^1 curves (d-f).

tapers as the dihedral angles gradually decrease from about 90° to about 10° , causing difficulties to existing methods. Near the nozzle, the mesh is very coarse, making it inappropriate to apply curvature-based methods. In addition, there are many edges that have large dihedral angles and are interconnected in a complex fashion, making it difficult for methods based on dihedral angles. Our method is robust for these difficult cases. In addition, Figure 12(f) shows a thin-wall structure under the engine nacelle of the airplane, where the mesh is under-resolved, and it is difficult to identify the singularities even manually. Our method delivered clean feature curves that are well suited for further manipulation of the mesh.

To assess the effectiveness of our method for different mesh resolutions, we used two meshes modeling half of a Falcon aircraft. The coarse mesh contains 2,289 vertices and 4,431 triangles, and the fine mesh contains 18,831 vertices and 37,165

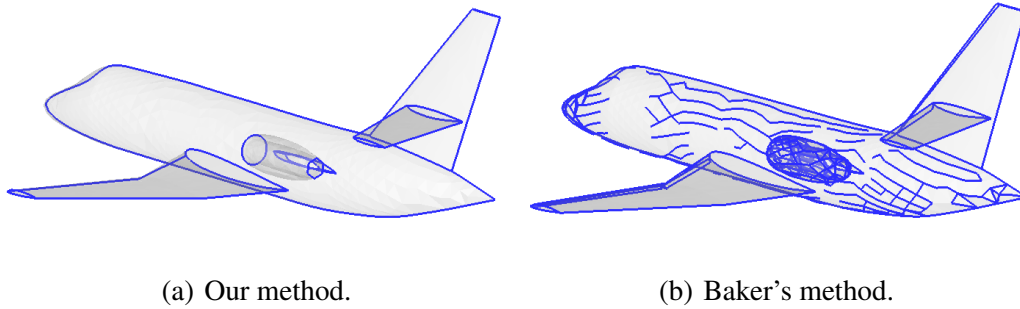


Figure 13. Comparison of C^1 curves for Falcon aircraft. (a) Our method delivered the same set of feature curves for coarse or fine meshes. (b) Method in [4] introduced much false positiveness.

triangles. Both meshes are under-resolved at the leading edge of the wings and near the tail. Our method detected the same curves for both meshes, as shown in Figure 13(a). Baker's method [4] again introduced much false positiveness as shown in Figure 13(b). The previous method in [9] delivered the same results for this model, but our new algorithm is simpler while being more robust for complex models.

As it is impractical to compare all the existing methods in the literature, Figure 1 shows the result of a widely used model, namely fandisk, as an indirect reference of comparison of our method against the others. Baker's method [4] led to false negativeness at the end of the tapering feature and at the same time introduced false positiveness (cf. Figure 1(a)). The method in [9] had no false positiveness but had false negativeness (cf. Figure 1(b)). Most results for the fandisk in the literature often behaved qualitatively similarly to these two (see e.g., [33]). Our proposed method (cf. Figure 1(c), with $\theta_f = 1^\circ$), on the other hand, detected singularities that cleanly decomposed the surface into a number of C^1 continuous patches.

To evaluate the effect of the parameters on the detected singularities, we focus on the parameter θ_f , which may require adjustment for detecting fine features. The other parameters, on the other hand, rarely require tuning. We used two very different values (1° and 10°) for θ_f , and the two plots in Figure 14 show the percentage of the "candidate" edges with different criteria for the test models using these two thresholds, respectively. A large number of edges have dihedral angles greater than these thresholds. The numbers of quasi-strong edges are much smaller but differed substantially for different θ_f . After filtration our method produced identical results for different θ_f for the airplanes. For the fandisk model, the result with $\theta_f = 1^\circ$ gave the curves in Figure 1(c) while the result with $\theta_f = 10^\circ$ missed two edges at the tapering end, where the DAs are smaller than 10° . These results demonstrate the stability of our method against the changes in θ_f , due to the facts that our criteria rely primarily on relative measures and the collective properties of feature curves. Finally, because of the nearly linear time complexity, our method is efficient in practice. The computations of the above tests each took a fraction of a second on a Linux desktop computer with a 3 GHz Intel Pentium IV processor.

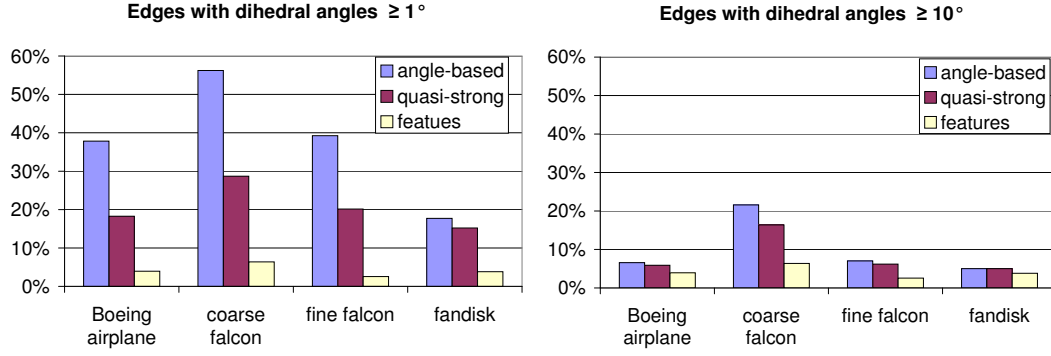


Figure 14. Percentage of edges with different selection criteria.

6.2 Results for C^2 Discontinuities

The detection of C^2 discontinuities is often important for mesh generation or adaptation from a coarse STL mesh obtained from a CAD model, and therefore we tested our method primarily using this type of meshes. We used 1° for θ_f and used the default values for the other parameters. Figures 15(a-e) show the results for five STL meshes using our method. The detected C^2 discontinuities are shown in dashed red curves and the C^1 discontinuities are shown in solid blue. The mesh edges are shown in light gray to give a hint about the mesh density. These STL meshes have little noise in the nodal positions but have irregular mesh connectivities and poor-shaped triangles. Some of these surfaces have symmetric patterns, but the mesh connectivities do not inherit such symmetry and hence may become a source of interference. In these examples, our method delivered symmetric results for all the symmetric geometry, indicating that our method is insensitive to the interference of mesh connectivity.

Figure 15(f) shows the detected C^1 and C^2 curves of the fandisk model. Different from the others, this mesh is not STL, but our method still extracted C^2 curves and the result exhibited clean, symmetric structures. However, as circled out in the figure, the fandisk model contains inflection points. It is difficult to identify those inflection points directly. By design, our method extracted one curve on either side of the inflection points. These curves are beneficial for remeshing to protect the inflection points, but if one would like to exclude them or to identify the inflection points, it is not difficult to develop a post-processing step using the convexity indicators in subsection 3.1.4.

In Figure 15, meshes (a) and (b) were courtesy of Drs. Yamakawa and Shimada, the developers of polygon crawling [24]. For these two models, the C^2 curves we identified are the same as their “type-2 features.” However, polygon crawling is specifically designed for STL meshes generated directly from CAD models, so it is less general than ours and is not applicable to the fandisk model or the airplanes that we have shown earlier.

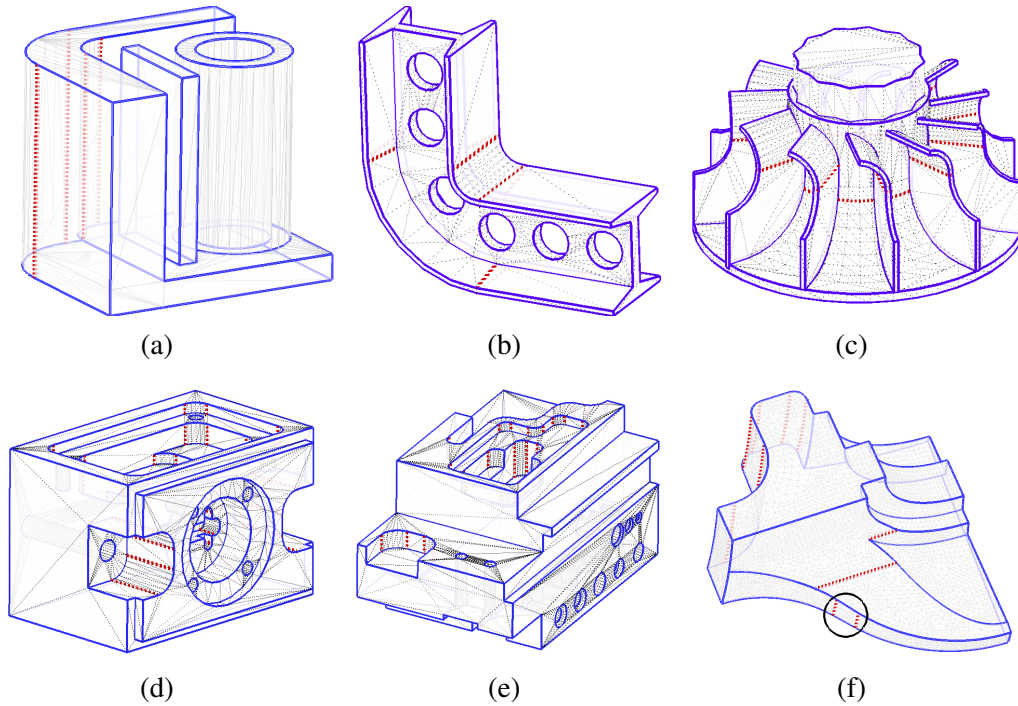


Figure 15. Detected C^1 (solid blue) and C^2 (dashed red) curves of test models.

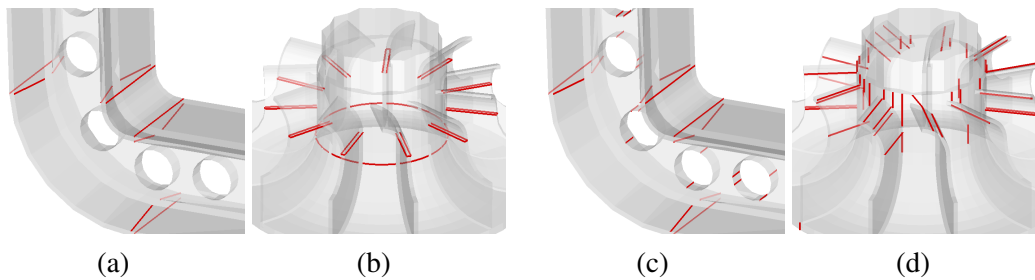


Figure 16. Comparison of detected C^2 curves. Our method delivered symmetric results consistent despite asymmetry of mesh connectivity (a-b). Alternative method was sensitive to mesh connectivity and delivered asymmetric results (c-d).

Our method worked nearly perfectly for the meshes above. However, these models are by no means trivial, because of their irregularities of mesh connectivity and edge lengths. To demonstrate the potential adverse impact of these irregularities, Figure 16 shows the detected C^2 curves for the models in Figure 15(b-c) using our method and an alternative method based on parallel edges as we mentioned earlier. We refer to these two models as “bent beam” and “fan,” respectively. The alternative method introduced some false positiveness in the cylindrical patches of the bent beam and false negativeness in the fan. Our method is simpler and more reliable primarily because of the effectiveness of the quasi-strong criteria.

7 Discussions

In this paper, we presented a framework for extracting discontinuities in surface meshes, focusing on C^1 discontinuities as well as typical C^2 discontinuities that are incident on C^1 discontinuities and separate flat and non-flat regions. A core concept of our framework is *quasi-strong edges*, whose criteria avoid false negativeness and reduce false positiveness simultaneously by combining a number of geometric and topological measures. We presented numerically stable procedures for computing the geometric measures and efficient algorithms for extracting the discontinuities. Compared to the previous sophisticated methods in [4, 9], our method is simpler, more general, and at the same time more reliable. We analyzed the correctness of our method and demonstrated its effectiveness using a variety of finite-element and STL meshes. Note that no feature-detection method (including ours) is absolutely robust. In particular, the filtration rules of our method may lead to false negativeness if the input mesh violates the quasi-saliency assumption.

Our framework can be used in a number of applications, such as numerical simulations with moving boundaries, remeshing of finite-element meshes, and reverse engineering of CAD models. These methods have a spectrum of requirements on the detected singularities: In numerical simulations or remeshing of non-CAD models, it may suffice only to identify the quasi-strong edges with somewhat restrictive thresholds (in particular, with a larger θ_f such as 20°) and without additional filtering, whereas remeshing or reverse engineering of CAD models may require full-fledged filtration. In addition, segmentation of surfaces may also take advantage of the detected C^1 and C^2 discontinuities, but some customization or extension may be needed because the curves in such applications cannot have singleton end-edges.

In terms of future research directions, our framework may be extended in a number of ways. First, our framework focuses on discontinuities, but it may be combined with curvature-based techniques to further identify the “features” within the smooth surface patches after identifying the discontinuities. Second, our framework assumes that the discontinuities are composed of the edges and vertices in the mesh. It would be useful to relax this requirement by integrating the framework with geometric flows and remeshing to denoise surfaces while sharpening the smeared discontinuities as in [22], and to extend the framework to resolve the noise in the topology of the mesh (such as gaps or duplicated triangles in the mesh). Finally, although the parameters of our framework, especially those used in the selection of quasi-strong edges, typically do not require tuning, it may be useful to investigate the selection of the parameters based on local statistics of quasi-strong edges.

Acknowledgments We thank Dr. Todd Michal of Boeing for providing the mesh of the Boeing airplane. We thank Dr. Eric Shaffer and Mr. John Norris of Uni-

versity of Illinois for their help with I/O utilities. We thank anonymous reviewers whose suggestions have significantly improved the readability of this paper. This work was supported by a subcontract from the Center for Simulation of Advanced Rockets of the University of Illinois at Urbana-Champaign, which is funded by the U.S. Department of Energy through the University of California under subcontract B523819.

References

- [1] S. J. Owen, D. R. White, Mesh-based geometry: a systematic approach to constructing geometry from a finite element mesh, in: Proc. 10th Int. Meshing Roundtable, 2001, pp. 83–96.
- [2] P. J. Frey et al, Yams: an efficient surface remeshing tool, <http://www.ann.jussieu.fr/frey/logiciels/yams.html>.
- [3] A. Razdan, M. Bae, A hybrid approach to feature segmentation of triangle meshes, *Comput. Aid. Des.* 35 (2003) 783–789.
- [4] T. Baker, Identification and preservation of surface features, in: Proc. 13th Int. Meshing Roundtable, 2004, pp. 299–310.
- [5] D. Bespalov, W. C. Regli, A. Shokoufandeh, Local feature extraction and matching partial objects, *Comput. Aid. Des.* 38 (2006) 1020–1037.
- [6] M. Garland, P. S. Heckbert, Surface simplification using quadratic error metrics, in: Proc. SIGGRAPH, 1997, pp. 209–216.
- [7] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, in: Proc. SIGGRAPH, 2003, pp. 950–953.
- [8] X. Pennec, N. Ayache, J. P. Thirion, Landmark-based registration using features identified through differential geometry, in: I. N. Bankman (Ed.), *Handbook of Medical Imaging*, Academic Press, 2000, pp. 499–513.
- [9] X. Jiao, M. T. Heath, Feature detection for surface meshes, in: Proc. 8th Int. Conf. on Numerical Grid Generation in Computational Field Simulations, 2002, pp. 705–714.
- [10] E. D. Bloch, *A First Course in Geometric Topology and Differential Geometry*, Birkhäuser, 1997.
- [11] M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.
- [12] E. Grinspun, M. Desbrun, K. Polthier, P. Schröder, A. Stern, *Discrete differential geometry: An applied introduction*, SIGGRAPH Course Notes (2006).
- [13] D. Meek, D. Walton, On surface normal and Gaussian curvature approximations of given data sampled from a smooth surface, *Comput. Aid. Geom. Des.* 17 (2000) 521–543.
- [14] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, *Comput. Aid. Geom. Des.* 22 (2005) 121–146.
- [15] X. Jiao, M. T. Heath, Overlaying surface meshes, part ii: Topology preservation and feature matching, *Int. J. Comput. Geom. Appl.* 14 (2004) 403–419.

- [16] X. Jiao, P. Alexander, Parallel feature-preserving mesh smoothing, in: Proc. Int. Conf. on Comput. Sci. and Appl., 2005, pp. 1180–1189.
- [17] J. O’Rourke, Computational Geometry in C, 2nd Edition, Cambridge, 1998.
- [18] P. S. Heckbert, M. Garland, Optimal triangulation and quadric-based surface simplification, *Comput. Geom.* (1999) 49–65.
- [19] G. Medioni, M.-S. Lee, C.-K. Tang, *A Computational Framework for Segmentation and Grouping*, Elsevier, 2000.
- [20] D. L. Page, Y. Sun, A. Koschan, J. Paik, M. Abidi, Normal vector voting: crease detection and curvature estimation on large, noisy meshes, *J. Graphical Models* 64 (2002) 1–31.
- [21] X. Jiao, Face offsetting: a unified framework for explicit moving interfaces, *J. Comput. Phys.* 220 (2007) 612–625.
- [22] X. Jiao, A. Colombi, X. Ni, J. Hart, Anisotropic mesh adaptation for evolving triangulated surfaces, in: Proc. 15th Int. Meshing Roundtable, 2006, pp. 173–190.
- [23] X. Jiao, Volume and feature preservation in surface mesh optimization, in: Proc. 15th Int. Meshing Roundtable, 2006, pp. 62–69.
- [24] S. Yamakawa, K. Shimada, Polygon crawling: feature-edge extraction from a general polygonal surface for mesh generation, in: Proc. 14th Int. Meshing Roundtable, 2005, pp. 257–274.
- [25] T. Várady, M. A. Facello, Z. Terék, Automatic extraction of surface structures in digital shape reconstruction, in: M.-S. Kim, K. Shimada (Eds.), *Geometric Modeling and Processing – GMP 2006*, Springer, 2006, pp. 1–16.
- [26] A. G. Belyaev, E. V. Anoshkina, T. L. Kunii, Ridges, ravines, and singularities, in: A. T. Fomenko, T. L. Kunii (Eds.), *Topological Modeling for Visualization*, 1997, pp. 375–383, chapter 18.
- [27] G. Stylianou, G. Farin, Crest lines extraction from 3d triangulated meshes, in: *Hierarchical and Geometrical Methods in Scientific Visualization*, 2003, pp. 269–281.
- [28] K. Hildebrandt, K. Polthier, M. Wardetzky, Smooth feature lines on surface meshes, in: *Eurographics Symposium on Geometry Processing*, 2005, pp. 85–90.
- [29] M. Nomura, N. Hamada, Feature edge extraction from 3d triangular meshes using a thinning algorithm, in: Proc. SPIE - Vision Geometry X, 2001, pp. 34–41.
- [30] U. Clarenz, M. Rumpf, A. Telea, Robust feature detection and local classification for surfaces based on moment analysis, *IEEE Trans. Visual. Comput. Graphics* 10 (2005) 516–524.
- [31] A. Hubeli, M. Gross, Multiresolution feature extraction from unstructured meshes, in: Proc. IEEE Vis., 2001, pp. 287–294.
- [32] Y. Ohtake, A. Belyaev, H.-P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, in: Proc. SIGGRAPH, 2004, pp. 609–612.
- [33] M. Meyer, M. Desbrun, P. Schroeder, A. Barr, Discrete differential geometry operators for triangulated 2-manifolds, *Visualization and Mathematics III* (2003) 35–58.

- [34] Y.-L. Yang, Y.-K. Lai, S.-M. Hu, H. Pottmann, Robust principal curvatures on multiple scales, in: Eurographics Symposium on Geometry Processing, 2006.
- [35] N. Amenta, Y. Kil, Defining point-set surfaces, in: Proc. SIGGRAPH, 2004, pp. 264–270.
- [36] S. Fleishman, D. Cohen-Or, C. Silva, Robust moving least-squares fitting with sharp features, in: Proc. SIGGRAPH, 2005, pp. 544–552.

A Sufficiency of Quasi-saliency

We show that under the quasi-saliency condition, no feature edges can be contained in quasi-obscure curves. First, consider feature edges in a closed feature curve γ , which is salient by itself. If a non-feature candidate edge is incident on a vertex in γ , then its incident feature halfedges become multi-joint, so γ remains salient. If two or more vertices of γ have incident non-feature candidate edges, then γ is broken into sub-curves, which are all salient by the same argument. Therefore, it is an invariant that the edges in γ are always in salient sub-curves, so they would never be filtered out.

Second, consider an open feature curve γ whose end-vertices are junctions. Because of condition 2 of quasi-saliency, γ is salient. If a non-feature candidate edge is incident on at a vertex in γ , then γ is broken into sub-curves, which remain salient following the same argument as above.

Finally, consider a feature curve γ of which one end-vertex is a turn or singleton. Because of conditions 3 and 4 of quasi-saliency, γ must be semi-salient. If any non-feature candidate edge is incident on a vertex of γ , the sub-curves would be either salient or semi-salient. We therefore conclude that under quasi-saliency condition the filtration of quasi-obscure curves does not introduce any false negativeness.