Incremental Reconstruction of Sharp Edges on Mesh Surfaces

Charlie C. L. Wang
Department of Automation and Computer-Aided Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
E-mail: cwang@acae.cuhk.edu.hk

Abstract

Limited by the regular grids in computing, many modelling approaches (e.g., field-based methods) sample 3D shape insensitive to sharp features therefore exhibit aliasing errors, by which a lot of sharp edges and corners are lost on the reconstructed surface. An incremental approach for recovering sharp edges on an insensitive sampled triangular mesh is presented in this paper, so that shape approximation errors are greatly reduced. Either chamfered or blended sharp edges on an input triangular mesh could be successfully reconstructed by the signals inherent in the mesh. As a non-iterative method, our approach could be finished in a very short time comparing to those diffusion-based sharp-feature reproducers. The region embedding sharp features is first identified through normal variations. The positions of vertices in the sharp-feature embedded region are then predicted progressively from outer to the inner of sharp regions so that sharp edges could be recovered in the sense of region shrinking.

Keywords: sharp edges, reconstruction, insensitive sampled, triangular mesh, geometry processing

1. Introduction

At present, a lot of geometric modelling approaches sample signals on uniform grids (also called volumetric representation) to reconstruct and modify the shape of three-dimensional objects, where the transformation between boundary and volumetric representations occurs in the successive way of: surface-volume conversion, shape processing on grid data, and volume-surface conversion. For example, the mesh simplification algorithm represented in [16] converts a given mesh into the voxel representation, then adopts less triangular elements to approximate the isosurface defined by voxels. The volumetric representation is also conducted to help fix topological errors on a reconstructed surface in acquisition applications ([20] and [26]). Also, in computer-aided engineering, there are many approaches [3, 4, 12, 38] using this implicit representation on grids to evolve the shape and topology of a structure so that provide optimal mechanical properties.

In above applications, sharp edges and corners are degraded on the resultant surface of evolution. Over-sampling could reduce the aliasing error by taking the cost of increasing storage memory; however as be observed in [21], even if an over-sampling is applied, the associated aliasing error will not be absolutely eliminated as the surface normals on the reconstructed model usually do not converge to the normal field of the original model. To solve this problem, the authors in [21] encoded the normals on each grid node (i.e., recording a Hermite data set) and recovered sharp features based on the encoded normals. However, in many applications (e.g., the field-based shape modelling and

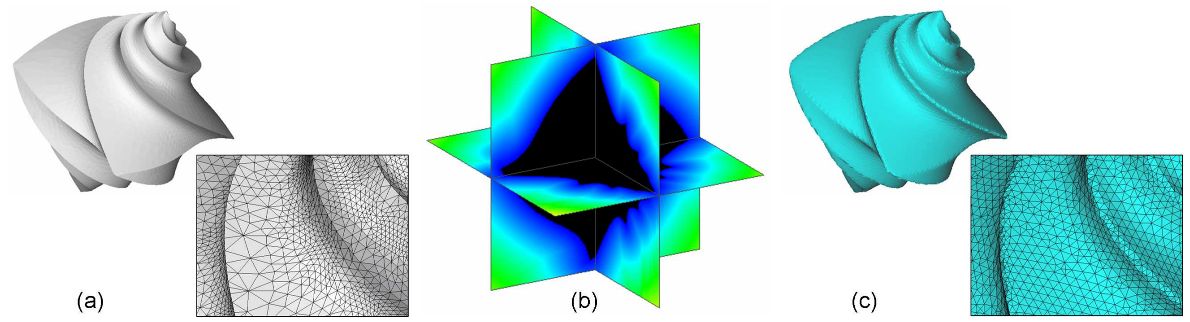

Figure 1: Using the Shrinkwrap algorithm [30] to contouring the isosurface of a uniformly sampled distance field produces blended aliasing regions on the original sharp edges: (a) original flower mesh, (b) the distance-field generated by the closest point transform (CPT) [25] - where black denotes a point inside the given surface, and (c) the isosurface generated by [30].

evolution [3, 4, 12, 38]), normals cannot be accurately provided - i.e., can only be approximated from the shape signals. Therefore, the reconstruction technique of sharp edges on feature-insensitive sampled models is needed. An ideal recovering technique for sharp edges is expected to be fast, and takes mesh surfaces as the only input. An approach for the similar purpose was presented in [2]. However, in [2] the method assumes that *chamfered* edges do not contain sample points, as in the case of the Marching cubes algorithm [23], surface reconstruction approaches, and some remeshing algorithms. Our approach does not rely on such an assumption. This is motivated by the need to filter meshes produced by dynamic remeshing strategies (e.g., [30, 35]). Fig.1 gives an illustration about how the smooth degeneration is produced.

This paper proposes an incremental approach to recover sharp edges on a triangulated mesh M through geometry prediction. Limited by the nature of an incremental approach, here we assume that: *the noises or errors are not exhibited on the non-sharp-feature region.* The region Ω embedding sharp features is firstly recognized by a *Uniformly Supported Second-Order Difference* (USSOD), which measures the maximal normal variation around a vertex on M. Then the sharp region is shrunk into its skeleton Ψ. During region shrinking, the positions of newly removed vertices from Ω are predicted by the triangles existed in the non-sharp region $\overline{\Omega} = M \setminus \Omega$. After that, vertices on the skeleton Ψ are repositioned to generate the geometry of single-line-wide sharp edges.

Comparing to other techniques to reconstruct sharp edges, our method yields several advantages as follows:

- A robust algorithm is developed which does not rely on the identification of chamfered edges - i.e., randomly destroyed and smoothly blended sharp edges can also be recovered. The reconstruction approach can be applied on a non-manifold object being represented as an assembly of open mesh patches.

- The sharp edges are reconstructed from the signals inherent in the given mesh surface so that our method requests no normal been recorded on volumetric data.

- The approach is a non-iterative method (i.e., no diffusion process), which gives the result that thousands of triangles can be processed in an interactive speed on a PC with standard configuration.

- A new sharp-fold detector is developed based on the *uniformly supported second-order difference* - since it is uniformly supported, it has less influence by the irregularity of given meshes.

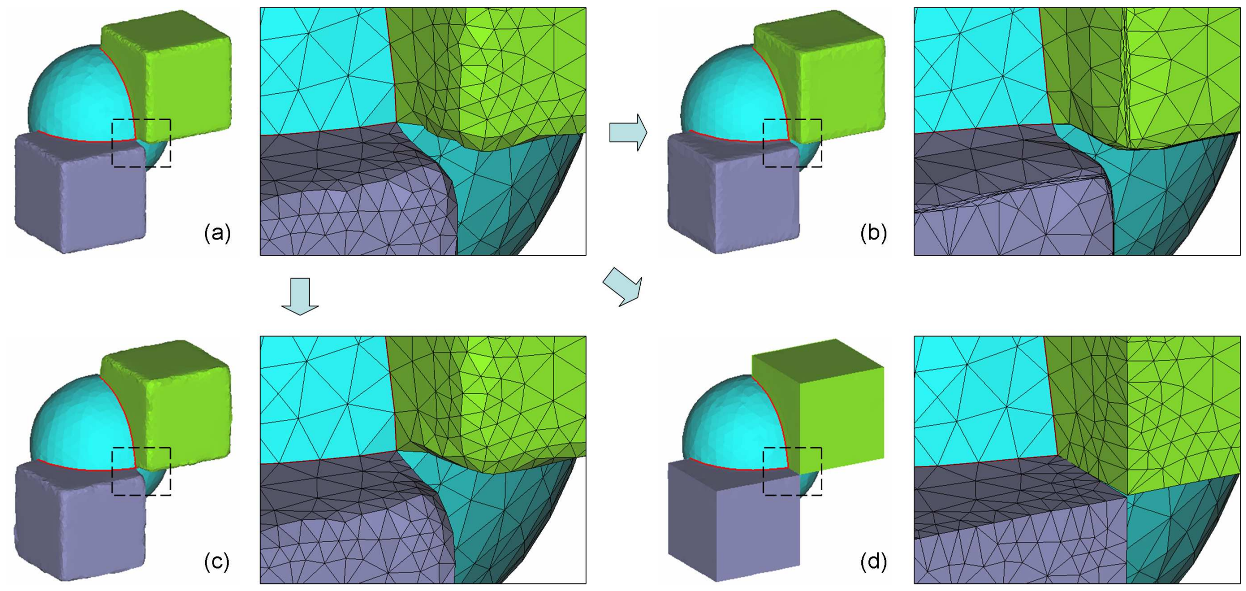

Figure 2: Recovering sharp edges on a non-manifold structure: (a) the given insensitive-sampled meshes; (b) the result after 30 runs of Bilateral filtering [18] - vertices are significantly drifted; (c) the result after 5 runs of Bilateral denoising [11] - sharp edges can hardly recovered; (d) the result of our method - sharp edges are well reconstructed.

- The connectivity on given meshes are retained during sharpening, which is important to some applications (e.g., metamorphosis).
- The sharp curves on resulting surfaces can be easily detected through the dihedral angles on edges.

The rest of the paper is organized as follows. After reviewing the related work in literature, the methodology of our incremental reconstruction approach is presented in section 3. Section 4 details the implementation techniques. The experimental results shown in section 5 prove that our method can successfully recover sharp edges on insensitive sampled triangular meshes in a very short time. Finally, our paper ends by the conclusion section.

2. Related Work

There are many attempts to encode the original surface normals during sampling (e.g., [19, 21, 27, 28]), where a Hermite data set is generated to reconstruct sharp features. But if the volumetric data is not sampled from an explicit mesh representation or the Hermite data cannot be maintained during the evolution of implicit data, e.g., in the heterogeneous object modelling of [3, 4, 38] no normal could be supplied, we therefore need to reconstruct sharp features only from the given mesh surfaces.

In recent years, a wide variety of mesh smoothing algorithms have been proposed, which are classified into isotropic and anisotropic. The isotropic approaches [10, 22, 34] indiscriminately smooth noise and small features, so the sharp edges on a noisy model will become extremely rounded before the model becoming smooth. To solve the problem of feature preservation, techniques for anisotropic mesh smoothing have been developed (ref. [5, 7, 9, 17, 24]). The idea behind these approaches is to modify the diffusion equation to make it non-linear so that become sensitive to curvature tensors. The results are with much high quality. However, as mentioned in [18], the non-linear modification affects the numerical conditions of the diffusion equations, which leads to significant computational time. Therefore, a faster approach is requested. The authors in [11, 18] applied the Bilateral filter to smooth and denoise 3D meshes while preserving sharp features. However, we find that sharp edges cannot be well reconstructed by only one run of Bilateral filtering. Several runs of the Bilateral filtering introduced in [18] (which has already become an iterative approach) will drift the vertices so that a lot of vertices are accumulated near sharp edges. This makes the sharpness greatly reduced (i.e., the sharp edges cannot be easily detected through dihedral angles - see Fig.2b). To prevent vertex drifting, the Bilateral denoising approach in [11] limits the movement of vertices only along a specified direction (i.e., vertex normal), which makes the recovery of sharp edges more difficult (e.g., the vertex whose current normal pointing to the wrong direction cannot be correctly repositioned - the result based on our implementation of [11] is shown in Fig.2c).

Edges are the most important element on polygonal mesh to represent sharp features. Sharp edges are usually recognized and enhanced from its neighboring connectivity graph. In [14], a method was presented for extracting sharp edges on a multi-resolution mesh. Their method is based on the measurement of dihedral angles. When the sharp features are rounded by small radius, a lot of "sharp" edges are detected - so that a thinning process is applied to form patches of the surface. Watanabe and Belyaev in [37] used the identification of perceptually salient curvature extremes to detect curvature features. Similar ideas have also been presented in [15, 24, 29, 32, 40] to detect sharp features, where curvature features are defined as a portion of the model which has an extreme value of curvature in some direction. However, the sharpness geometry is not really reproduced on edges in all these approaches. In contrast, the proposed algorithm here modifies the geometry around sharp edges so that they can be easily recognized by dihedral angles.

The purpose of the technique presented in this paper is akin to [2], to recover sharp edges with triangular meshes as the only input. For the sharp edges that are smoothly blended, the algorithm of [2] fails - but our method works well. Besides, the reconstruction of sharp edge on non-manifold structures, that is not referred in [2], will be also addressed in our paper.

3. Methodology

This section presents the methodology of our sharpening algorithm. A preliminary description of this part was first presented in a conference version [36].

3.1. Signals indicating sharpness

The normal vector at a surface point can considered as the *First-Order Difference* of shape. The surface smoothness is usually measured by the variation of surface normals. Therefore, normal vectors on a given surface can be treated as signals indicating the smoothness, where signals in the lower frequency range relate to smooth surface parts while higher frequency signals correspond to the region embedding sharp features. The simplest method to detect the frequency of smoothness signal is through the dihedral angle on a triangular edge e - the dihedral angle actually measures the difference of normal vectors on the two adjacent faces of e. This so called *Second-Order Difference* (SOD) detector is used in [14] for the identification of sharp features, and adopted in [2] to measure smoothness. However, SOD is not robust to the local normal variation as its support size is non-uniform and depends on the size of triangles adjacent to an edge. For example, after blending a sharp edge into a smooth surface as shown in Fig.3a, the SOD can hardly classify edges on the blended surface part (the red ones) into the region embedding sharp features.

To robustly recognize the region degenerated from sharp edges on a given surface M, a new detector measuring the frequency of smoothness signal is studied. A window function $W(v, f)$ is first

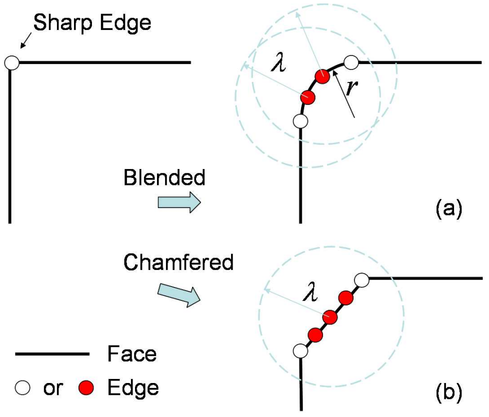

Figure 3: Region degenerated from sharp edges can be detected by USSOD on the blended region (a) where the dihedral angle based detector fails, and on the chamfered region (b) where the curvature tensor based detector fails.

defined on a surface point v for any face $f \in M$:

$$W(v, f) = \{ \begin{array}{ll} 1 & (d(v, f) \leq \lambda) \\ 0 & (d(v, f) > \lambda) \end{array} \tag{1}$$

where $d(v, f)$ returns the Euclidean distance from v to the point-set of f (i.e., not the plane holding f). By this window function, the uniformly supported second-order difference (USSOD) is given as

$$\tau(v) = \frac{1}{\lambda} \sup(W(v, f_i)W(v, f_j)(1 - P(f_i, f_j))) \tag{2}$$

with $P(f_i, f_j)$ representing the inner product of unit normal vectors on two faces $\forall f_i, f_j \in M$. The function $\tau(v)$ measures the maximum variation of smoothness signal on the surface around v with a uniform support size λ. The smaller value $\tau(v)$ returns, the smoother surface is exhibited on v - in the extreme case $\tau(v) = 0$, the surface around v is flat. In our approach, the USSOD function $\tau(v)$ is conducted to detect the sharp feature region Ω in the way that: $\forall v \in M$, we define $v \in \Omega$ if $\tau(v) \geq \kappa$ with κ as a threshold for the classification.

As been reviewed in above section, a lot of approaches employ curvature tensor to identify sharp features. The reason why we do not use the curvature tensor in [24] is for its local support. Let's consider the case as shown in Fig.3b - zero curvatures are shown on the middle point of the chamfered edge since it is flat. However, the region around this point is degraded from a sharp edge. Even if 2-ring neighbors are employed to computer curvature tensor on this point, it is still not easy to classify this point into the sharp region. Of course, we can increase the support of computing from 1-ring to k-rings neighbors, but how to determine the number k of rings is by no means a simple job. Looking back to our USSOD, the usage of a window function W makes our sharp-feature detector more robust since 3D neighbors of each input vertex are considered instead of k-ring neighbors which are very sensitive to the local mesh connectivity. Another reason of adopting W instead of local connected neighbors is to make our USSOD workable on non-manifold structures. For this reason, the robust 3D curvature tensor estimator in [1] is not adopted either. The reason why we conduct the distance from vertices to the point-sets of triangles instead of the distance between a vertex and the center of a triangle is that, the later strategy will skip over the triangles who have points fall in the support range λ but centers not. Here, the meaning and choice of the threshold parameter λ is more or less the same as the threshold for identifying sharp features in the curvature-based approaches [9, 24, 32].

3.2. Geometry predictor

The positions of vertices in the region Ω embedding sharp feature need to be transferred to produce the geometry of sharpness. A geometry predictor is defined here for this purpose in the sense of least-square fitting, which is in spirit to quadric errors in the mesh simplification approach of Garland and Heckbert [13]. The predictor repositions a vertex $v \in M$ by the smoothness signals neighboring v. Before describing the geometry predictor, the following terminology is given:

- All vertices in the non-sharp region $\overline{\Omega}$ are *static vertices*, and a vertex in Ω is named as a *sharp vertex* - after being positioned by the geometry predictor, a *sharp vertex* will become a *static vertex.*
- If all vertices in a triangle are static vertices, this triangle will not be deformed during later geometry predictions, so it is called a *static triangle*; the triangles with sharp vertices are *dynamic triangles.*

An ideal position for a vertex v minimizes the difference between its position and the smoothness signals - tangent planes. In order words, the position should minimize its distances to the tangent planes near v. Thus, the objective function for minimization is defined as

$$E_s = \sum_{f \in \Gamma(v)} [A_f (v - q_f) \cdot n_f]^2 \tag{3}$$

where the collection $\Gamma(v)$ contains the static triangles *near* the vertex v, q_f is a point on the static triangle f, n_f is the unit normal vector of f, and A_f is the area of triangle f serving as a weight to account for the variations on irregular meshes. The definition of *near* will be addressed in detail in the later section.

The position of v is requested to make E_s minimal, so we need to determine the position of v by the linear equation system

$$\frac{\partial E_s}{\partial v} = 0 \tag{4}$$

However, the coefficient matrix of Eq.(4) is not always full-rank (e.g., when there are only one or two triangles in $\Gamma(v)$), so the following reformulation is conducted. As a relaxed mesh is usually expected (ref. [28]), where every vertex v on a mesh is close to the center p_c of its 1-ring neighbors, we would like to determine a solution of Eq.(4) where the distance from v to p_c is shortest. Note that p_c is determined by all the neighbors of v including both static and sharp vertices. We replace v in E_s by $(p_c + p)$ with p as the position translation vector to be determined. Then, using the *singular value decomposition* (SVD - which can determine a solution vector with minimal norm on an ill-conditioned linear equation system) [31] to solve

$$\frac{\partial E_s}{\partial p} = 0 \tag{5}$$

we obtain a solution with minimal $\|p\|$ - i.e., the closest solution to p_c.

3.3. Progressive surface prediction

The success of sharp edge recovering relies on the order of geometry predictors been applied to the vertices in the sharp region Ω. Here, we expect that the order can reflect an effect of propagating smoothness signals from outer to the inner of Ω. A thinning scheme is adopted to shrink Ω into a skeleton Ψ which preserves the topology of sharp edges. The thinning algorithm for skeletonisation

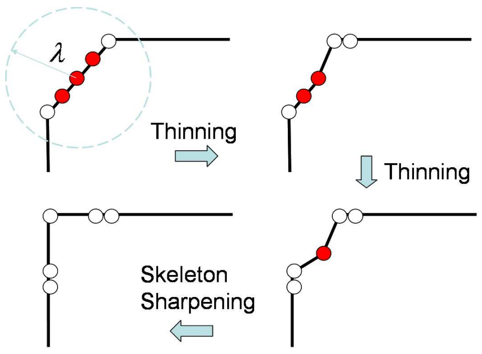

Figure 4: Illustration of the progressive surface prediction steps, where red circles represent sharp vertices and white circles denote static vertices.

is similar to the thinning algorithm in digital image processing [6], but our implementation is based on the connectivity of a mesh surface.

The conventional thinning algorithm [6] reduces a curvilinear objective to a single-pixel-wide line. Here, we exploit the thinning algorithm to shrink the sharp region Ω into a skeleton that is single-edge-wide. The thinning algorithm on Ω is in fact a process of conditional erosion, where the vertices on the boundary $\partial\Omega$ of Ω are progressively removed in two steps without destroying connectivity. Firstly, the vertices in Ω linked to a vertex $\hat{v} \in \overline{\Omega}$ are marked as candidates for removal. Then the candidates whose removal will not destroy the connectivity of Ω are removed, while others are retained.

During the thinning of Ω, the geometry predictor repositions every removed vertex, which becomes a static vertex after the geometry prediction. In this way, the surface normals which serve as the smoothness signals will be propagated from the static triangles in $\overline{\Omega}$ to the triangles adjacent to the skeleton Ψ. After that, the positions of vertices on the skeleton are repositioned so we can reconstruct sharp edges. As being illustrated in Fig.4, the positions of sharp vertices are progressively predicted to produce sharp features.

4. Implementation Details

4.1. Surface representation

The data structure conducted in this paper follows [39]: a non-manifold object M is stored as a collection of two-manifold mesh patches

$$M = \bigcup_{i=0}^{n} M_i \tag{6}$$

where each mesh surface patch M_i is defined as a pair (K, V), with K as a simplicial complex specifying the connectivity of vertices, edges, and faces (i.e., the topological graph of M_i), and $V = \{v_1, ..., v_m\}$ as the set of vertices defining the shape of a polyhedral patch in $\mathbb{R}^3$. From K, it is straightforward for our algorithm to fetch the adjacent nodes, edges, and faces of a triangular node in constant time. The continuities between patches are preserved by storing linkers on boundary vertices. The vertices belonging to different patches sharing the same position are called *common vertices*, whose positions should be kept consistent during the processing of meshes (see Fig.5 - the small yellow cubes denote common vertices). Two-manifold mesh surface (either closed or open) can also be represented in this data structure as a single patch.

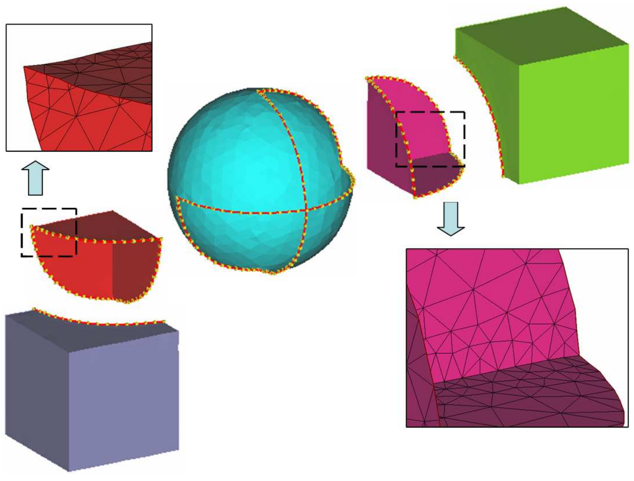

Figure 5: The reconstruction of sharp edges on a non-manifold structure which is a collection of mesh patches previously given in Figure 1.

4.2. Sharp region classification

For a given mesh M, the uniformly supported second-order difference $\tau(v)$ is computed on every vertex $v \in M$. For an inner vertex $v \in M_i \subset M$, the USSOD is only evaluated on the mesh patch M_i; but for a boundary vertex v_b, USSOD is evaluated on all the meshes containing v_b and its common vertices. To accelerate the evaluation of $\tau(v)$, we quickly filter out those faces with $W(v,f)=0$ by the following solution. The bounding box of M, $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$, is subdivided into $I \times J \times K$ sub-regions $\mathbb{R}(i,j,k) \subset \mathbb{R}^3$ with an uniform width which is equivalent to the window size λ of $\tau(v)$. The pointer of a triangle face f is stored in a sub-region $\mathbb{R}(i,j,k)$ if its bounding box $B(f)$ satisfies

$$\mathbb{R}(i,j,k) \bigcap B(f) \neq \phi. \tag{7}$$

By this space subdivision, for the vertex $v \in \mathbb{R}(l,m,n)$, $\tau(v)$ is only computed on the triangles whose pointers are held in $\mathbb{R}(i,j,k)$ with $i \in [l-1, l+1]$, $j \in [m-1, m+1]$, and $k \in [n-1, n+1]$.

The choice of window size λ is not free. In order to obtain an anti-aliasing of degenerated sharp edges, we usually choose λ between 0.5 to 1.5 times of the sampling size used in the surface-volume conversion. Too small λ will let the window function $W(v,f)$ fail on locally smooth regions, while too large λ will classify a smooth vertex v into sharp-region in mistake if the sharp-region is close to v in Euclidean distance but far from v in the Geodesic distance. From our experience, when adopting the above range of λ on the models reconstructed from the uniform volumetric sampling, the misclassification seldom occurs. If the sampling size in the surface-volume conversion is unknown, λ can only be determined in a trial-and-error manner, which usually starts from $\lambda = \overline{L}$ for regular meshes or $\lambda = 1.5\overline{L}$ for irregular meshes. The value of threshold κ indicates the user's definition of sharpness, we leave it as a parameter to be interactively assigned in the manner similar to [11]: the user selects several points p_i of the mesh where the surface is expected to be sharp, then the values of USSOD, $\tau(p_i)$, at these points are computed - among which $\min\{\tau(p_i)\}$ is assigned to κ. In fact, the value of window size λ can also be suggested by the user selected sharp points p_i - we can incrementally increase λ (starting from $0.5\overline{L}$) until all p_i have been classified into the sharp region.

Figure 6 shows a comparison of sharp region detection results on a MechPart model from the dihedral angle based method [14] (with the dihedral angle threshold $\cos^{-1} 0.75$), the curvature tensor based method [24] (with the curvature threshold $1/\overline{L}$ and $0.25/\overline{L}$), and our USSOD-based method

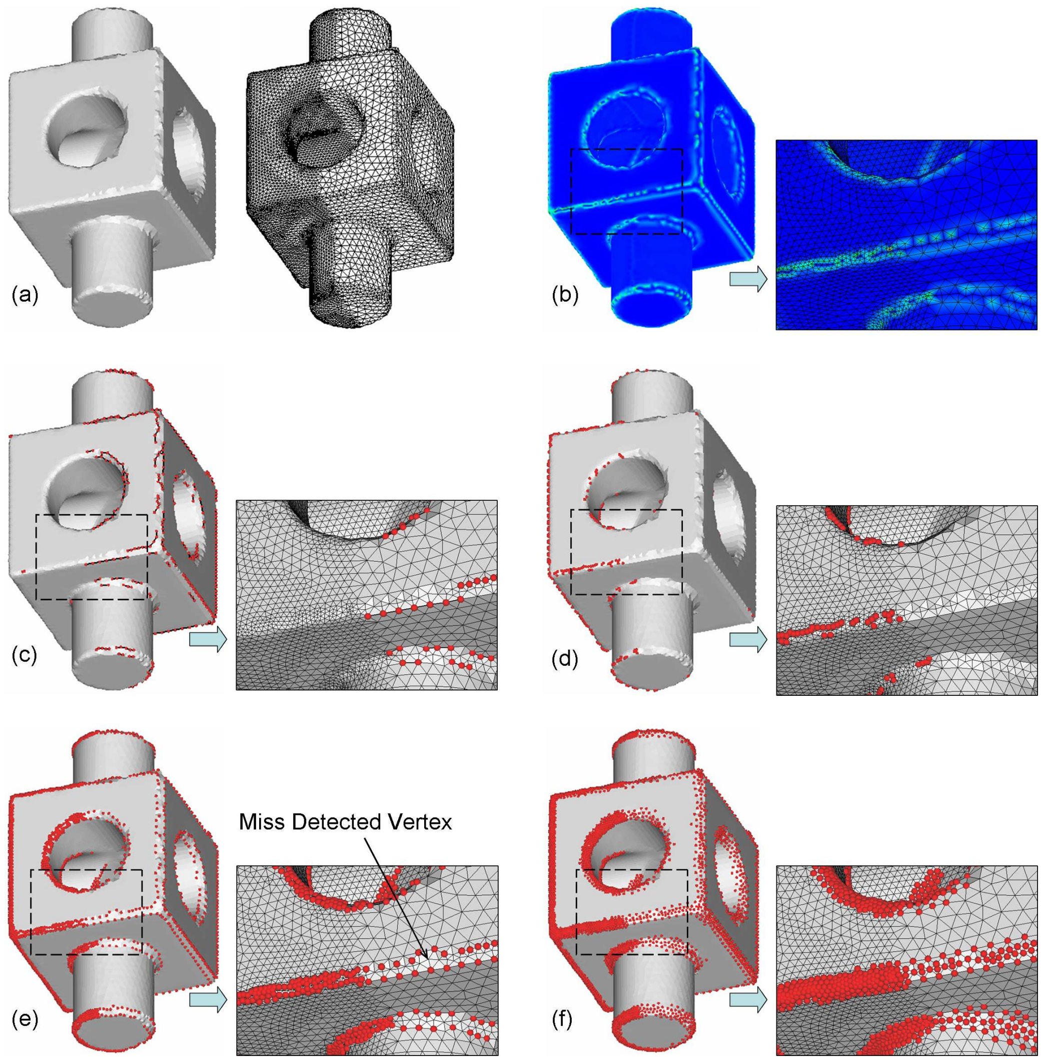

Figure 6: For a given MechPart model with irregular mesh (a), (b) shows the color map of its mean curvature. The classification results from (c) the dihedral angle based method [14], (d, e) the curvature tensor based method [24], and (f) our USSOD-based methods are compared, where vertices denoted by red spheres are classified to be sharp.

(with $\lambda/\overline{L} = 1.5$ and $(1 - \kappa/\lambda) = 0.75$). $\overline{L}$ is the average edge length on M. Note that we only compare the vertex classification but not the sharpness recovering result here. The mesh on the given MeshPart model is generated from a distance-field by the method of [35] in the dynamic remeshing sense. To demonstrate the advantages of USSOD, we adopt different edge-length criteria on the left and the right parts of the model and also add some noises on the aliasing region - so that the irregularity is introduced (see Fig.6a). Fig.6b gives the color map of mean curvatures on the surface - red represents the maximal value while blue denotes the minimum. On the result shown in Fig.6c, the edges with its dihedral angle greater than $\cos^{-1} 0.75$ are found and presented by bold lines - the scheme fails on the smoothly blended regions (i.e., the left part of given model). Fig.6d gives the sharp region classification result with threshold $1/\overline{L}$. The vertex classification scheme conducted here follows [24]: with a threshold T, a vertex v is considered to be smooth only if 1) $|\kappa_{\min}| \leq T$ and $|\kappa_{\max}| \leq T$, or 2) $|\kappa_H| = \min(|\kappa_{\min}|, |\kappa_{\max}|, |\kappa_H|)$ (with κ_H the mean curvature at v); otherwise, v is a sharp-region vertex. A lot of sharp regions are incorrectly classified to be smooth in Fig.6d. After relaxing the lower bound of curvature for sharp features from $1/\overline{L}$ to $0.25/\overline{L}$, more vertices are recognized to be sharp (see Fig.6e). However, the ones locally flat (e.g., the vertex been pointed out in Fig.6e) are unavoidable missed. Although the anisotropic diffusion introduced in [24] may fix an initial misclassification during the mesh evolution, we cannot adopt this technique in our approach as we need to have a correct classification to serve for our non-iterative sharpening approach. The result from our USSOD-based scheme in Fig.6f can elegantly find all regions degenerated from sharp edges.

4.3. Geometry predictor

In the geometry predictor (Eq.(3)), $\Gamma(v)$ contains the static triangles f_i *near* the vertex v. In detail, the definition of *near* is

$$d(v, f_i) \leq \alpha\lambda \tag{8}$$

where $d(v, f)$ returns the Euclidean distance from v to the point-set of f_i and α is an coefficient for the support size of *near*. In our implementation, the value of α is iteratively increased starting from 0.5 with step 0.1 until the number of static triangles satisfying the above condition is non-zero. All static triangles falling in this range are inserted into $\Gamma(v)$. With the static triangles in $\Gamma(v)$, the equation (5) can then be derived into

$$CX = B \tag{9}$$

with

$$C = \begin{pmatrix} \sum A_i n_i^x n_i^x & \sum A_i n_i^x n_i^y & \sum A_i n_i^x n_i^z \\ \sum A_i n_i^x n_i^y & \sum A_i n_i^y n_i^y & \sum A_i n_i^y n_i^z \\ \sum A_i n_i^z n_i^x & \sum A_i n_i^z n_i^y & \sum A_i n_i^z n_i^z \end{pmatrix} \quad B = \begin{pmatrix} \sum A_i n_i^x ((q_i - p_c) \cdot n_i) \\ \sum A_i n_i^y ((q_i - p_c) \cdot n_i) \\ \sum A_i n_i^z ((q_i - p_c) \cdot n_i) \end{pmatrix},$$

where n_i represents the unit normal of a static triangle f_i in $\Gamma(v)$, q_i is the center of f_i, A_i is the area of f_i, and p_c is the center of 1-ring neighbors of the vertex v. When the vertex v is on the boundary of a surface patch in a non-manifold model, since the position of v_b is commonly determined by all surface patches meeting at this vertex, p_c is determined by the average of the vertices neighboring to both v and its common vertices.

The SVD solution of X is the translation vector p of v from p_c (i.e., the new position of v given by the geometry predictor is $(p_c + p)$). As the geometry predictor is defined by tangent planes, nearly parallel tangent planes will lead to a degeneration case with the norm of translation vector p extremely long. This is usually generated by the normal vectors with noises embedded, which should be prevented. Thus, if the translation vector p have its magnitude greater than $4\overline{L}$ ($\overline{L}$ is the average edge length on M), we just truncate p by $4\overline{L}$.

4.4. Skeletonisation

The thinning algorithm progressively moves vertices from the boundary of Ω into $\overline{\Omega}$. Our implementation is based on the pseudo-code listed in Appendix. To efficiently detect vertices on the boundary of Ω, a list $\partial\Omega_{next}$ is introduced to maintain the new boundary vertices formed by removing vertices from Ω. It is easy to find that in general cases the numbers of 1-ring neighbors to any vertex is less than a small constant number, so the computational time of ***Algorithm*** Thinning(Ω) is in linear complexity.

We need a method to efficiently detect whether removing a vertex from Ω will break its connectivity. The local connectivity of Ω after removing v is detected by computing the local transitive closure. The sharp vertices in 1-ring neighbors of v are denoted by $\Theta(v)$. An adjacency matrix A is first constructed: for two vertices v_i and v_j in $\Theta(v)$ (where i and j are local indexes of vertices in $\Theta(v)$), if there is a triangle edge in M linking them, the corresponding coefficients a_{ij} and a_{ji} in A are set to one; otherwise, zeros are given. After that, the transitive closure T of the local graph defined by A is computed by the Warshall's algorithm [33]. Then, the following proposition can be derived on T for detecting the local connectivity of Ω after removing v.

Proposition For the transitive closure T defined on $\Theta(v)$, if $t_{ij} = 0$ $(\forall t_{ij} \in T)$, the local connectivity of Ω is destroyed by removing v from Ω.

Proof: For the definition of a transitive closure, we have known that if $t_{ij} = 0$ $(\forall t_{ij} \in T)$, it means that there is no path between v_i and v_j on the original graph defined by A. However, if the vertex v is kept as a sharp vertex, all pairs of vertices in $\Theta(v)$ could have a path reaching each other through v (i.e., they are locally connected). Thus, if there is any zero element found in T, the removal of v from Ω will break the local connectivity of Ω.

Q.E.D.

In order to remaining the linkage between Ω and the boundary of a surface patch, the removal of sharp vertices on the boundaries of any $M_i \subset M$ disallowed; also to prevent the shrinkage of skeleton Ψ on its tail vertices, the vertex v with only one sharp vertex in its 1-ring neighbors $\Theta(v)$ must be retained in Ω.

Algorithm Thinning(Ω) returns a shrunken Ω containing only the vertices on its skeleton Ψ. However, as shown in Fig.7a, local closures may be exhibited, we then conduct the following filter to open local closures:

- Any edge $e \in M$ has both its vertices on Ψ is defined as a sharp-edge candidate;
- If all candidate are defined as sharp edges, some local closures will be formed near joints of Ψ (e.g., see Fig.7a); thus, for any triangle has three sharp-edge candidates, only the two adjacent to the vertex with greatest inner angle are set to sharp edge - the local closures are opened (see Fig.7b).

After applying this filter on the returned Ω from ***Algorithm*** Thinning(Ω), we obtain the skeleton Ψ that is single-edge-wide.

4.5. Final contouring

The vertices on the skeleton Ψ are classified into *branch vertices* and *joint vertices.* For a vertex $v \in \Psi$, if the number of its adjacent sharp edges is greater than two, v is a *joint vertex*; otherwise, it is a *branch vertex.* The sharpening of surface on the skeleton vertices is finished in two steps:

- All *branch vertices* are first repositioned by the geometry predictor, after which all branch vertices are converted into static vertices;

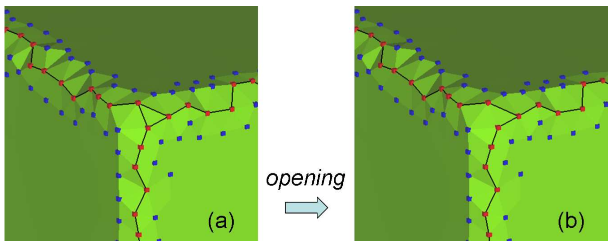

Figure 7: Local closures are formed in (a) after thinning Ω, and need to be opened into a final skeleton (b). The red small cubes represent the sharp-vertices on skeleton, the blue cubes denote the sharp-vertices been removed during thinning, and the bolded black edges present the sharp edges.

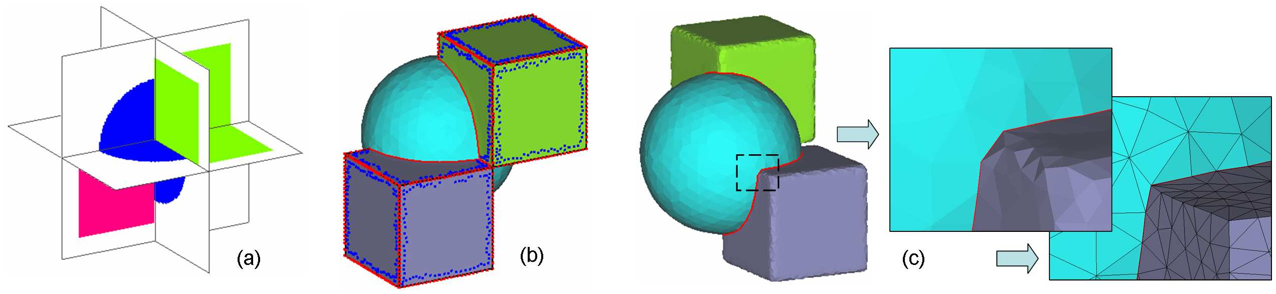

Figure 8: Example I - a heterogeneous model in implicit representation (a), where different colors denote different materials are filled at the point - the method presented in [35] is conducted to convert the field representation into a non-manifold mesh surface (the left part of (c)); after detecting sharp regions, the sharp edges are reconstructed, where the red cubes in (b) are the vertices on the skeleton of thinning and the blue cubes are the ones removed from Ω during thinning. Sharp corners on the boundaries are well recovered (c).

- All *joint vertices* are finally positioned by the geometry predictor.

By isolating the sharpening of branch vertices and joint vertices, when computing the position of a joint vertex, the triangles adjacent to earlier branch vertices also contribute to the position of the joint vertex - this makes the sharpness at joint vertices better.

During the final contouring, the geometry predictor given in Eq.(5) and Eq.(9) is still exploited to reposition vertices on the skeleton consecutively but with little modification. First of all, the relaxation requirement for a vertex v on the final skeleton Ψ is different from the one used in thinning: here, v is expected to be located at the average position of its 1-ring neighboring skeleton-vertices instead of all its 1-ring neighbors, so p_c conducted in Eq.(9) is different. Secondly, the set of static triangle, $\Gamma(v)$, is adjusted. For a *branch vertex* v_b, since its final position needs to consider the surfaces from both sides of the skeleton, we set the longest distance, $d_{\max}$, between v and its 1-ring neighbors as the support size of $\Gamma(v_b)$, so $\Gamma(v_b)$ holds all the static triangles with its distance to v not greater than $d_{\max}$. Also, by the similar reason, the support size of $\Gamma(v_c)$ for a *joint vertex* v_c is changed in the same manner.

5. Results and Discussion

Our incremental restoration algorithm has been tested on several models, which are constructed either directly from a volume model or through a surface-volume-remeshing conversation routine. To demonstrate the functionality of our algorithm, we measure the Hausdorff distance (E_{max}) and

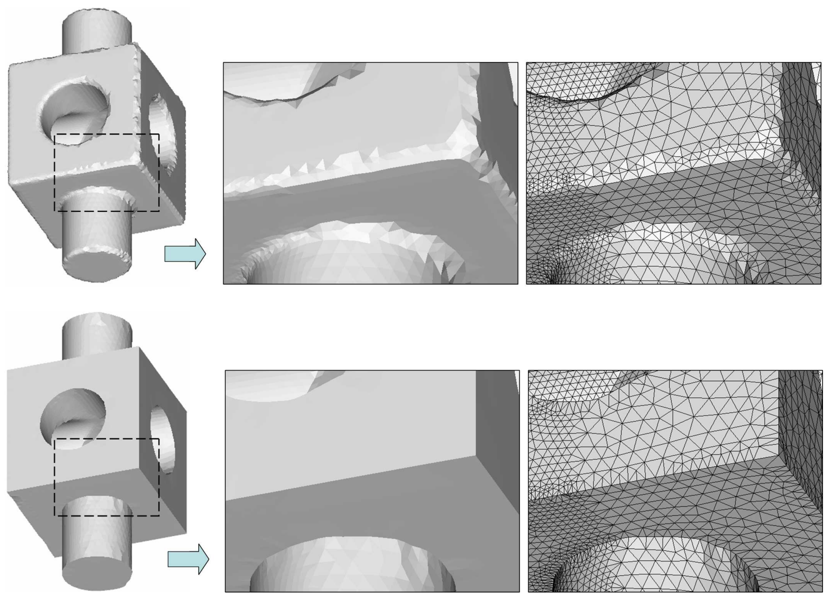

Figure 9: Example II - MechPart: the sharp edges are randomly destoried on the given model with irregular meshes, which are successfully recovered.

the mean distortion (E_{mean}) between the original model and the remeshed model before and after the restoration of sharp edges. All errors are computed by the publicly available tool [8]. We have observed that our algorithm can significantly reduces the errors (both E_{max} and E_{mean}) in several seconds when running on a low-end PC. The computational statistics are listed in Table 1, where all errors are reported as percents of the bounding-box diagonal. For the first and the last examples, errors are missing because we did not have an original surface to compare with.

Our first model is a non-manifold object, which has previously been shown in Fig.2. The surface patches are directly constructed from an implicitly represented heterogeneous model (i.e., field-based modelling - see Fig.8a), where different colors represent that different material stuffs are filled. The surface of a heterogeneous model is usually non-manifold. On the resultant model from our sharp edge restoration, the sharpness is well reconstructed (see Fig.2c, 5, and 8c). The second example (previously given in Fig.6) is a mechanical part which is usually adopted as a benchmark for testing the surface reconstruction results. In order to demonstrate the functionality of processing irregular meshes, the left and right parts of the mesh model exhibit different resolutions. Our recovering result is given in Fig.9. The third and fourth examples are conducted to demonstrate the ability of our approach for recovering curved sharp edges - the results are shown in Fig.10. All above examples are CSG-like. Therefore, in Example V and VI, the restoration of sharp edges on free-from surfaces are tested. As shown in our results (Fig.11 and 12), the sharp features are successfully recovered on the free-form surface. More testing examples on freeform objects are given in Fig.15 at the end of this paper.

Another interesting investigation concerns the error reduction on a given model with different sampling rates. Theoretically, our sharp feature restoration algorithm should in some degree increase the accuracy of an feature-insensitive sampled model no matter in which frequency it is sampled; also, with the increase of sampling density, the model produced by our approach is expected to give less error. The testing results from our approach follow these presumptions. Four different sampling densities are chosen to convert the original model into a volume representation, which is then converted into a mesh model by the method in [35]. After that, the mesh models are processed

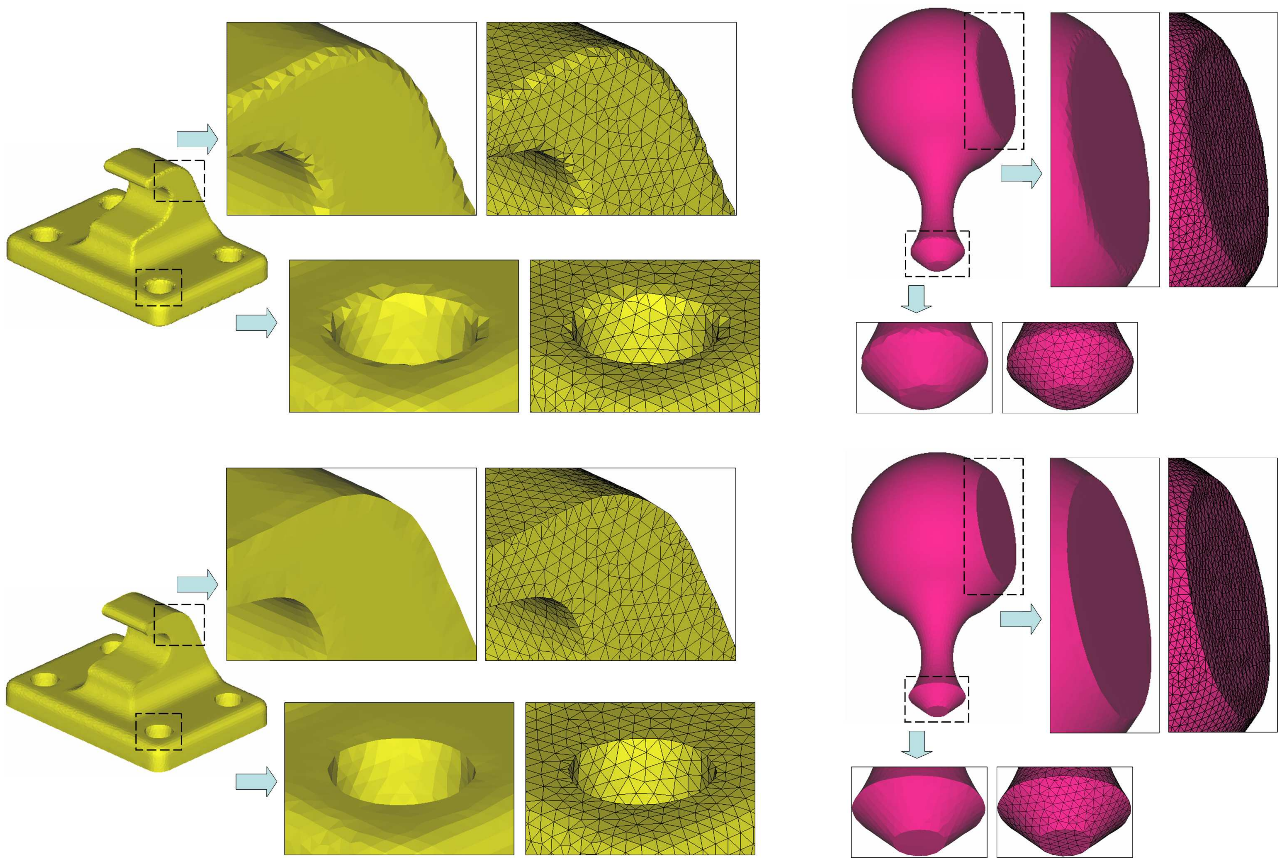

Figure 10: Example III & IV - Anchor Plate (left) and Knob (right): the given model (top row), and the resultant model (bottom row) with curved sharp edges restored.

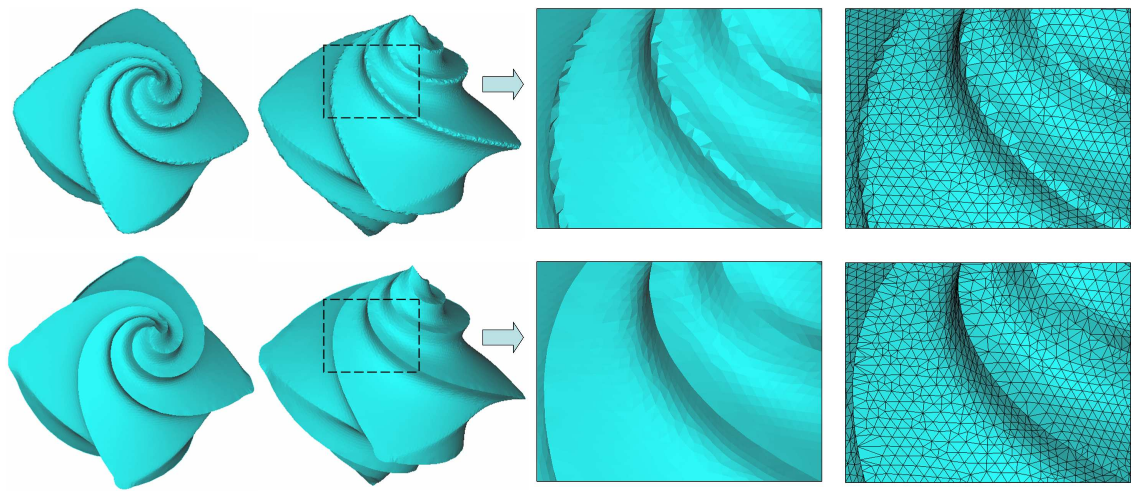

Figure 11: Example V - Flower (a freeform model): the given model (top row) and the model with sharp edges recovered (bottom row).

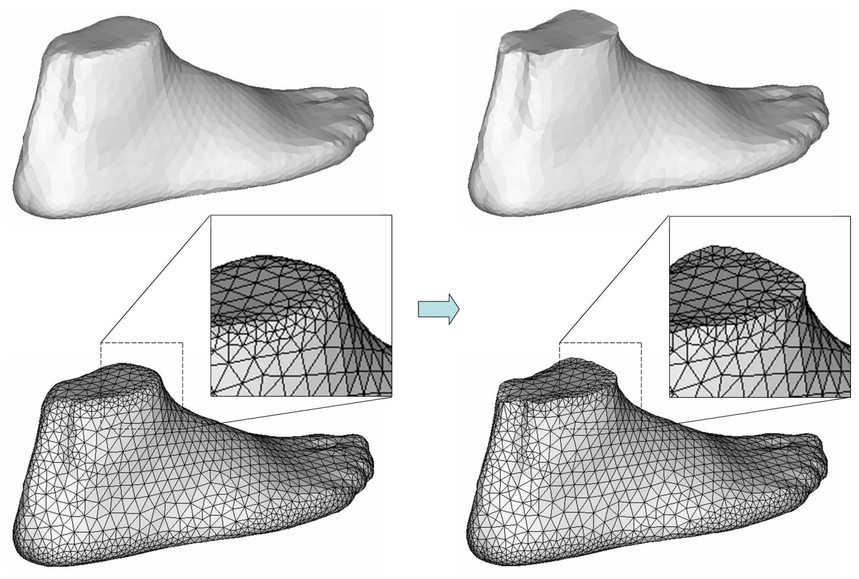
Figure 12: Example VI - Foot (a freeform model): the given model (top row) and the model with sharp edges recovered (bottom row).

by our algorithm to reconstruct sharp edges. Figure 13 shows the results of our tests on the MechPart example (a relative regular model) and the Flower example (a free-form object).

We can easily detect the sharp curves on the resultant meshes through the dihedral angles since the sharpness geometry has been recovered. For a triangular edge, if the inner product of the unit normal vectors on its adjacent two faces is less than $(1 - \kappa/\lambda)$, the edge is identified as a sharp line segment. The resultant sharp curves on the example I-V are shown in Fig.14.

5.1. Limitations

Similar to other sharpening algorithms, the feature that blends smoothly into a flat area may be miss-sharpened by our approach if the sampling rate is not able to generate enough great normal variations around aliasing regions. Also, some unwanted sharpening will be given on small radius blends (e.g., in Fig.15b and 15c). These small features will effect the sharp region classification - so that unsatisfactory results may be generated. In fact, during our tests, determining a best value for $\lambda/\overline{L}$ and κ/λ is by no means a trivial work. We are planning to develop some more robust and adaptive sharpness identification techniques that are based on higher order differences on given surfaces (e.g., the gradient of normal variation). The robust 3D curvature tensor estimation method presented in [1] will also be considered.

Our approach will have difficulty to distinguish the sharp-regions when they are close to each other. More specifically, if two sharp regions are directly linked by some triangle edge, undesired sharp corners will be generated on the sharp curves by our approach. A better sharp-region recognition method needs to be investigated to solve this problem.

At last, for the edges that are correctly placed at sharp features showing no aliasing error, we cannot distinguish them from the aliased features. Thus, they will be relocated even if there is no aliasing error on them. The development of a procedure to avoid processing the areas with no aliasing errors is also considered as one of our future research.

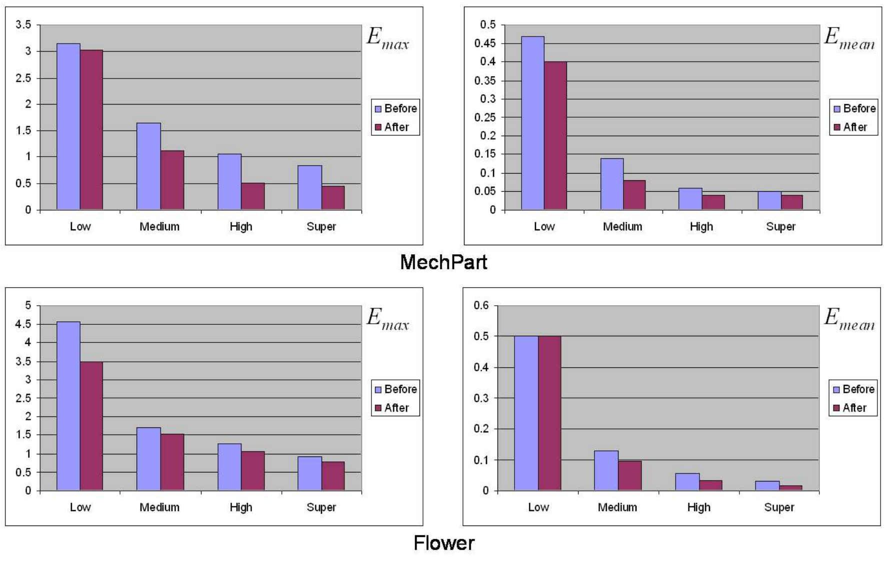

Figure 13: The analysis of accuracy effected by different sampling rates on the "MechPart" and "Flower" examples - both are tested in four different sampling densities: low - sampled at 64 x 64 x 64; medium - the model is sampled by a 128 x 128 x 128 grid; high - 192 x 192 x 192; and in super resolution - 256 x 256 x 256 samples are conducted.

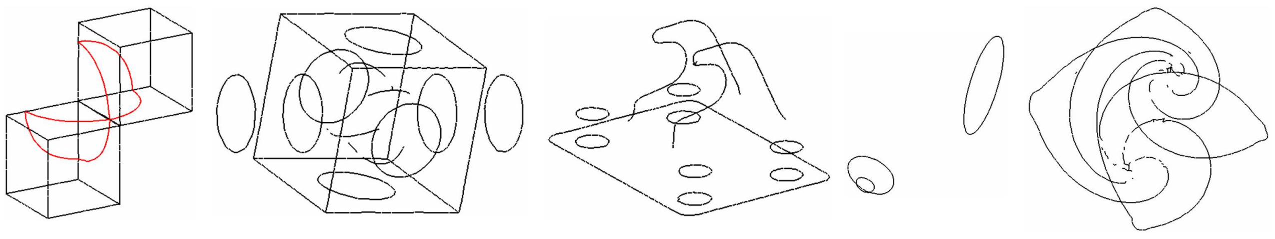

Figure 14: Sharp curves extracted on the result models by using dihedral angle, where the red curves in example I are boundaries of assembled patches.

Table 1 Computational Statistics

Model	Fig.	Triangle number	Before recovering		After recovering		$\lambda / \overline{L}$	κ / λ	Time of sharp region classification	Time of recovering
			E_{max}	E_{mean}	E_{max}	E_{mean}				
2Cube+Sphere	2, 5, 8	7,231	-	-	-	-	0.75	0.25	5.1s	0.36s
MechPart	6, 9	31,572	2.13	0.07	0.63	0.05	1.5	0.25	25.2s	4.3s
Anchor Plate	10	21,280	0.66	0.04	0.43	0.03	0.5	0.25	6.9s	1.1s
Knob	10	14,064	0.53	0.03	0.14	0.02	0.5	0.06	5.5s	0.42s
Flower	11	25,104	1.45	0.07	0.68	0.02	0.5	0.25	11.9s	1.2s
Foot	12	3,628	-	-	-	-	0.75	0.25	3.1s	0.2s
Ball-joint	15a	36,956	-	-	-	-	1.0	0.25	22.1s	3.7s
Venus	15b	57,788	-	-	-	-	0.75	0.05	13.1s	3.1s
Isis	15c	93,820	-	-	-	-	1.5	0.25	49.1s	10.7s

* Note: the running time is tested on a PC with AMD Althon XP-M 2400+ CPU (1.6GHz) + 512MB RAM.

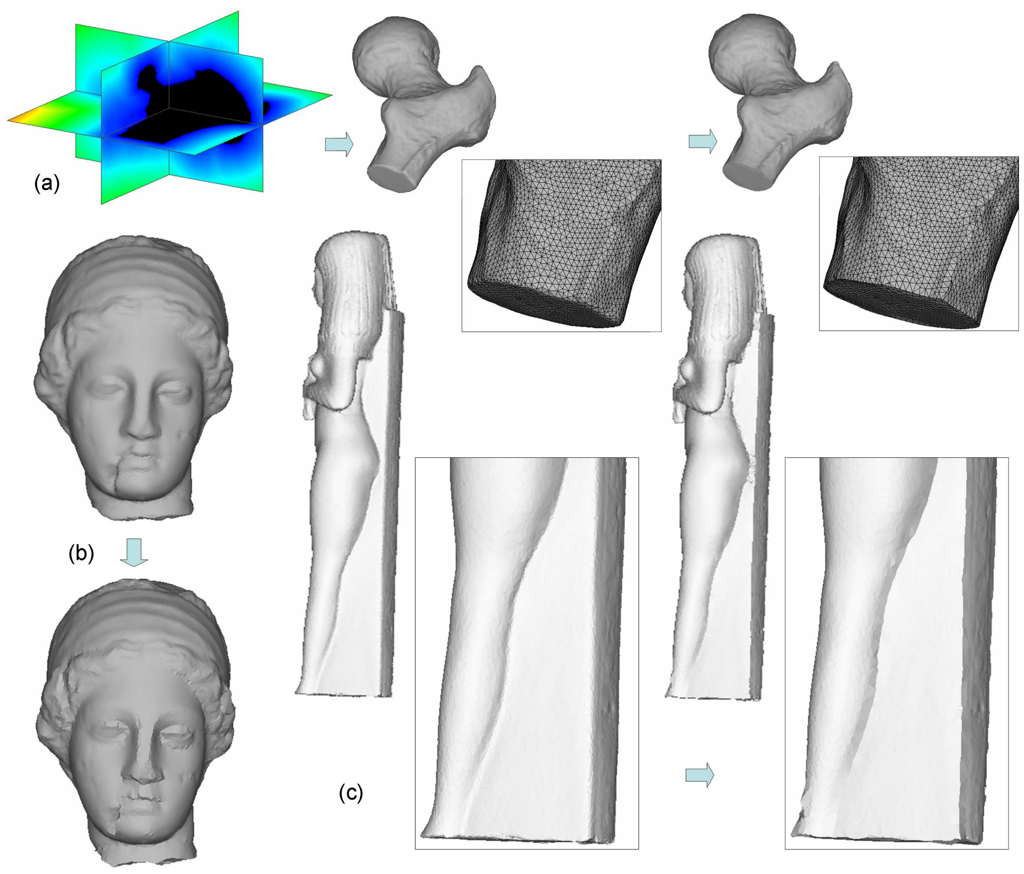

Figure 15: More testing results on freeform models: a) the ball-joint reconstructed from a signed distance-field, (b) the Venus model, and (c) the Isis model.

6. Conclusion

In this paper, we present an incremental approach for recovering sharp edges on a triangular mesh from feature-insensitive sampling, which embeds noises and shape approximation errors around the sharp features. With our method, either chamfered or blended sharp edges on an input triangular mesh could be successfully reconstructed. As a non-iterative method, our approach could be finished in a very short time comparing to those diffusion-based sharp-feature reproducers. The region embedding sharp features is first identified through normal variations. The positions of vertices in the sharp-feature embedded region are then progressively predicted from outer to the inner of sharp regions so that sharp edges could be recovered in the sense of region shrinking. Examples have been tested by our implementation to demonstrate the functionality of the approach, and the limitations of this method have been discussed.

Acknowledgments

The author would like to thank the helpful comments given by the anonymous reviewers.

Appendix

The pseudo-code of thinning algorithm is given as below.

```
***Algorithm*** Thinning($\Omega$) {
    Construct the boundary $\partial\Omega$ of $\Omega$;
    *do* {
        $\partial\Omega_{next} \Leftarrow \phi$;
        *for* (each vertex $v$ in $\partial\Omega$) {
            *if* (the removal of $v$ will not locally open $\Omega$) {
                Position $v$ by *geometry predictor*;
                Remove $v$ from $\partial\Omega$ and $\Omega$;
                    *for* (each vertex $v_L$ linked to $v$)
                        *if* (($v_L$ in $\Omega$) AND ($v_L$ *NOT* in $\partial\Omega$))
                            Insert $v_L$ into $\partial\Omega_{next}$;
            }
        }
        $\partial\Omega \Leftarrow \partial\Omega + \partial\Omega_{next}$;
    }*while*($\partial\Omega_{next} \neq \phi$);
    return $\Omega$;
}
```

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic polygonal remeshing", *ACM Transactions Graphics*, vol.22, no.3, pp.485-493, 2003.

[2] M. Attene, B. Falcidino, M. Spagnuolo, and J. Rossignac, "Sharpen & Bend: Recovering curved edges in triangle meshes produced by feature-insensitive sampling", *IEEE Transactions on Visualization and Computer Graphics*, vol.11, no.2, pp.181-192, 2005.

[3] V. Adzhiev, E. Kartasheva, T. Kunii, A. Pasko, and B. Schmitt, "Cellular-functional modeling of heterogeneous objects", *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pp.192-203, 2002.

[4] A. Biswas, V. Shapiro, and I. Tsukanov, "Heterogeneous material modeling with distance fields", *Computer Aided Geometric Design*, vol.21, pp. 215-242, 2004.

[5] C. L. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces", *ACM Transactions on Graphics*, vol.22, no.1, pp.4-32, 2003.

[6] K. R. Castleman, *Digital image processing*, Englewood Cliffs, N.J.: Prentice Hall, 1996.

[7] A. U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing", *Proceedings of IEEE Visualization 2000*, pp.397-405.

[8] P. Cignoni, C. Rocchini, R. Scopigno, "Metro: measuring error on simpli-fied surfaces", *Computer Graphics Forum*, vol.17, no.2, June 1998, pp.167-174.

[9] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Anisotropic feature-preserving denoising of height fields and bivariate data", *Proceedings of Graphics Interface*, pp.145-152, 2000.

[10] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow", *Proceedings of SIGGRAPH 99*, pp.317-324, 1999.

[11] S. Fleishman, I. Drori, D. Cohen-Or, "Bilateral mesh denoising", *ACM Transactions on Graphics*, vol.22, no.3, pp. 950-953, 2003.

[12] M. Freytag, V. Shapiro, and I. Tsukanov, "Field modeling with sampled distances", *Computer-Aided Design*, in press.

[13] M. Garland and P.S. Heckbert, "Surface simplification using quadric error metrics", *Proceedings of SIGGRAPH 97*, pp.209-216, 1997.

[14] A. Hubeli and M. Gross, "Multiresolution feature extraction for unstructured meshes", *Proceedings of Visualization'01*, pp.16-25, 2001.

[15] A. Hubeli and M. Gross, "Multiresolution methods for nonmanifold models", *IEEE Transactions on Visualization and Computer Graphics*, vol.7, no.3, pp.207-221, 2001.

[16] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang, "Voxel based object simplification", *IEEE Visualization'95 Proceedings*, pp.296-303.

[17] K. Hildebrandt and K. Polthier, "Anisotropic filtering of non-linear surface features," *Computer Graphics Forum*, vol.23, no.3, 2004.

[18] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing", *ACM Transactions on Graphics*, vol.22, no.3, pp. 943-949, 2003.

[19] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of Hermite data", *ACM Transactions on Graphics*, vol.21, no.3, July 2002, pp.339-46.

[20] T. Ju, "Robust repair of polygonal models", *ACM Transactions on Graphics*, vol.23, no.3, 2004, pp.888-895.

[21] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data", *Proceedings of SIGGRAPH 2001*, pp.57-66, 2001. New York, NY, USA.

[22] L. P. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive multi-resolution modeling on arbitrary meshes", *Proceedings of SIGGRAPH 1998*, pp.105-114, 1998.

[23] W. Lorensen and H. Cline, "Marching cubes: a high resolution 3D surface construction algorithm", *Computer Graphics*, vol.21, pp.163-169, 1987.

[24] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds", *Proceedings of Visualization and Mathematics*, 2002.

[25] S. Mauch, "A fast algorithm for computing the closest point and distance transform", Technical Report, http://www.acm.caltech.edu/ seanm/projects/cpt/cpt.html, 2000.

[26] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques", *IEEE Transactions on Visualization and Computer Graphics*, vol.9, no.2, 2003, pp.191-205

[27] Y. Ohtake and A. Belyaev, "Dual-prime mesh optimization for polygonized implicit surfaces with sharp features", *Proceedings of the eighth ACM Solid Modeling Symposium*, K. Lee and N. Patrikalakis (Eds.), 2003, pp. 171-178.

[28] Y. Ohtake, A. Belyaev, and A. Pasko, "Dynamic mesh optimization for polygonized implicit surfaces with sharp features", *The Visual Computer*, vol.19, pp.115-126, 2003.

[29] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Ridge-valley lines on meshes via implicit surface fitting", *ACM Transactions on Graphics*, vol.23, no.3, pp.609-612, 2004.

[30] K. van Overveld and B. Wyvill, "Shrinkwrap: An efficient adaptive algorithm for triangulating an iso-surface", *The Visual Computer*, vol.20, no.6, pp.362-379, 2004.

[31] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes in C : the Art of Scientific Computing* (2nd ed.), Cambridge: Cambridge University Press, 1995.

[32] C. Rössl, L. Kobbelt, and H.-P. Seidel, "Extraction of feature lines on triangulated surfaces using morphological operators", *Proceedings of the 2000 AAAI Symposium on Smart Graphics*, pp.71-75, 2000.

[33] K. A. Ross, and C. R. B. Wright, *Discrete mathematics* (4th ed.), Upper Saddle River, N.J.: Prentice Hall, 1999.

[34] G. Taubin, "A signal processing approach to fair surface design", *Proceedings of SIGGRAPH 95*, pp.351-358, 1995.

[35] C. C. L. Wang, "Direct extraction of surface meshes from implicitly represented heterogeneous volumes", *submitted.*

[36] C. C. L. Wang, "Non-iterative reconstruction of sharp edges", *Proceedings of Pacific Graphics 2005*, pp.127-129, Macao, October 12-14, 2005.

[37] K. Watanabe and A. Belyaev, "Detection of salient curvature features on polygonal surfaces", *Computer Graphics Forum*, vol.20, no.3, pp.385-392, 2001.

[38] M. Y. Wang and X. Wang, "A level-set based variational method for design and optimization of heterogeneous objects", *Computer-Aided Design*, vol.37, pp.321-337, 2004.

[39] C. C. L.Wang, Y. Wang, and M. M. F. Yuen, "Feature-based 3D non-manifold freeform object construction", *Engineering with Computers*, vol.19, no.2-3, pp.174-190, 2003.

[40] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "Fast and robust detection of crest lines on meshes", *Proceedings of ACM Symposium on Solid and Physical Modeling*, 2005.