COMPUTER-AIDED DESIGN & APPLICATIONS
2018, VOL. 15, NO. 6, 905–915
https://doi.org/10.1080/16864360.2018.1462571

Machined sharp edge restoration for triangle mesh workpiece models derived from grid-based machining simulation

Ziqi Wang, Jack Szu-Shen Chen, Jimin Joy and Hsi-Yung Feng

The University of British Columbia, Canada

ABSTRACT

Grid-based machining simulation methods have become very popular due to their advantages in computational efficiency. However, these methods are prone to generating machined workpiece models with chamfer edges, which reduce model accuracy and visual quality. An effective method is presented in this paper targeting the restoration of machined edges from these chamfer edges for triangle mesh models derived from grid-based machining simulation. The method starts by detecting the chamfer edge triangles via utilizing both edge-based mesh segmentation and feature-based mesh segmentation. The outcomes of the two segmentation methods are then compared and combined to achieve improved detection accuracy. To restore the machined edges, the detected chamfer edge triangles are split by adding new points based on the neighbors of the chamfer edge triangles. Triangle quality checks are imposed during the point addition process in order to ensure that the restored edges do not negatively impact the triangle mesh quality. Once all the applicable new points are added for all the chamfer edge triangles, the edge restoration task is complete. The presented method has been implemented and executed on sample triangle mesh models derived from grid-based machining simulation. Improved edge restoration compared to that of existing edge restoration methods has been observed. Sub-second computational performance is attained for the majority of the test cases.

KEYWORDS

Machining simulation; workpiece model; machined edge restoration

1. Introduction

Chamfered edges occur for triangle mesh workpiece models generated from discrete machining simulation methods such as the vector grid based tri-dexel method [23] and voxel grid based FSV-rep method [15]. The ideal sharp edge of intersection between two machined surfaces is replaced by a thin chamfer surface. In the case of representing the machined workpiece as a triangle mesh model, the thin chamfer surface shows as a set of connected triangles that run between the triangle patches representing the surfaces on either side of the ideal edge. Chamfered edges happen for triangle meshes generated from the vector or voxel based simulation because of the grid structure of the underlying geometric modeling methods. Presence of the chamfered edges reduces the accuracy of the generated workpiece model and deteriorates its visual quality. Hence, these undesirable chamfered edges need to be restored to their ideal sharp form. Machining simulation using vector or voxel based modeling methods is quite popular due to their computational efficiency. However, the issue of chamfered edges needs to be resolved. This paper presents an effective machined edge restoration method for triangle mesh workpiece models derived from grid-based machine simulation.

Triangle mesh is a versatile geometric representation format for machining simulation as it provides simple visualization and surface analysis functionalities. Geometric machining simulation is an essential part of the emerging virtual machining technology [2]. Compared to other representation formats, triangle mesh can represent the workpiece shape at reasonable accuracy with relatively low model size. This is achievable since triangle mesh uses simple triangular elements to approximate the workpiece surface geometry and a large number of smaller triangles can be used at areas of high curvature and a lesser number of large triangles at flatter areas. It makes triangle mesh a simple and efficient format for visualization and surface analysis in machining simulation.

Though triangle mesh has the aforementioned benefits, it is not suitable for efficient machining simulation involving repeated update of the workpiece model. Other modeling methods are, thus, often used for this purpose. In fact, in the early years of machining simulation

CONTACT Hsi-Yung Feng feng@mech.ubc.ca; Ziqi Wang qiqi007@mail.ustc.edu.cn; Jack Szu-Shen Chen jsschen38@gmail.com; Jimin Joy jjoy@alumni.ubc.ca

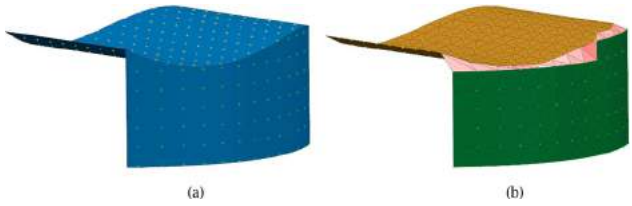

Figure 1. Chamfered edges: (a) Two ideal machined surfaces with an intersecting edge and sàmpled points on each surface obtained from the fixed spatial grid, and (b) Triangle mesh approximation from the sampled points resulting in an additional chamfer face (pale red) in place of the original edge.

research, boundary representation (B-rep) using NURBS [25] was used. Again, due to computational efficiency issues, NURBS based B-rep is not the preferred method being used for machining simulation. Triangle mesh based machining simulation [4],[11] was attempted in place of NURBS based B-rep but could not address the computational efficiency issue well, either. Vector modeling using *Z*-maps [3] and dexels [26] were the class of methods developed to achieve the best computational efficiency. Unfortunately, the computational efficiency was achieved at the price of model accuracy. Tri-dexel method [6],[19],[23] is an improved vector modeling method that can handle general milling simulation with good model accuracy. Space partitioning methods such as voxels [14–17],[28] are another active class of discrete modeling methods. To reduce the large memory demand of an accurate model for conventional voxel modeling methods, an improved voxel modeling method referred to as frame-sliced voxel representation (FSV-rep) has been developed [15]. All these discrete geometric modeling methods typically need to produce a triangle mesh based model representation to facilitate visualization and further downstream applications. Consequently, the quality of the generated triangle mesh from the discrete modeling methods is of great significance.

The triangle mesh workpiece model generated from the discrete grid-based machining simulation methods such as tri-dexels and FSV-rep suffers from the chamfered edge issue due to the following reason: because of the grid structural nature of tri-dexels and FSV-rep, only sampled points at the grid crossings on the workpiece surface are captured. This, however, does not ensure that such sampled points are available to reconstruct the edges of intersection between machined surfaces as depicted in Fig. 1(a). In fact, most, if not all, of the edges of intersection lack any sampled point for the reconstruction and end up being missed. Instead, a thin chamfer face is created between the sets of sampled points from the faces on the sides of the edge as depicted in Fig. 1(b). Such false chamfered edges can be classified according to the type of surfaces that intersect to produce the edges (Fig. 2). Three types of chamfered edges can be defined: (1) Type 1: edges between two smooth surfaces (case a in Fig. 2), (2) Type 2: edges between a smooth surface and a surface with machined scallops (case b in Fig. 2), and (3) Type 3: edges between two surfaces with machined scallops (case c in Fig. 2). Among the three types of chamfered edges, Type 1 is the simplest to detect and rectify whereas Types 2 and 3 are increasingly more complex. Regardless of the complexity, these chamfered edges need to be restored in order for the grid-based machining simulation methods to have comparable simulation output quality as NURBS and triangle mesh based machining simulation.

The possible approaches to avoid the false chamfered edges on a triangle mesh generated from a vector or space partitioning method can be conceptually categorized as in-process and post-process approaches. The in-process approaches are those being part of the workpiece update process for the underlying simulation

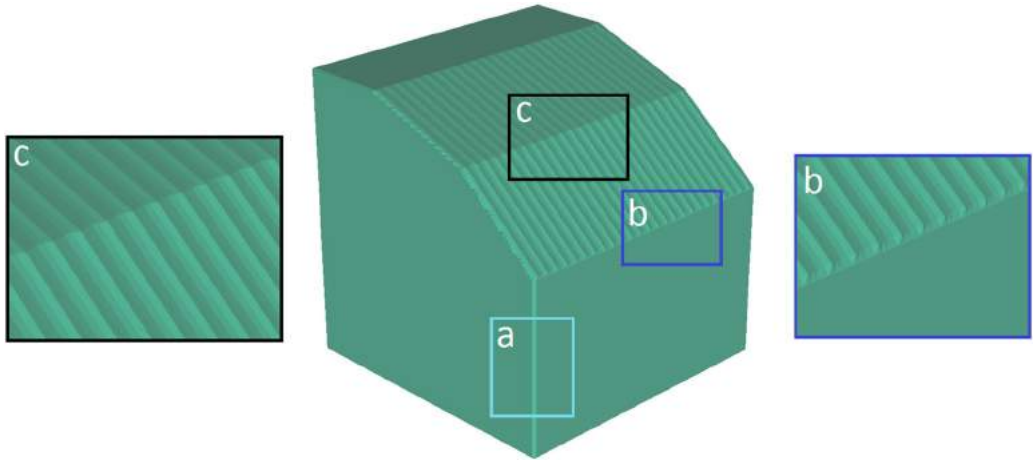

Figure 2. A typical machining case showing the three types of chamfered edges.

model itself. This is, however, challenging to pursue as the existing discrete methods are not structured to store the edge data. In order to achieve an in-process edge creation for the model, fundamental changes will be needed for the underlying geometric modeling method. The post-processing approaches, on the other hand, can be applied without being part of the workpiece update process. The essential requirement is to develop an edge restoration method which is applicable and effective for the associated machining simulation method. The method developed and presented in this paper is a post-processing method utilizing the characteristic properties of the triangle mesh generated by grid-based machining simulation.

To restore the machined edges of a triangle mesh model derived from grid-based machining simulation, a new method has been developed after considering the existing methods as applied in similar situations. Among the existing methods, feature conserving triangle mesh generation for tri-dexels by Ren et al. [23] is a notable method as it works using a space partitioning intermediate model called regularized tri-dexels. The method stores the surface normal vector at each dexel end point and updates it from the cutting tool envelope surface when the dexel is trimmed during the simulation. The triangle mesh generation later uses these surface normal vectors to restore the edge features. Using the dexel end points coincident with edges of each grid cell, the boundary loop for the triangle patch(es) within the grid cell is identified. The surface normal vector of each dexel end point in the loop is then used to identify the appropriate additional sampled points to be added to restore portions of the edge feature within the grid cell. To adapt this method for generic grid-based machining simulation is not possible since internal modification of the simulation method is required, in particular to store the surface normal vector data. As for generic edge feature restoration techniques, notable ones include bilateral denoising by Fleishman et al. [8] and sharp feature recovery using an energy optimization technique by Liu et al. [20]. These methods, however, target noisy meshes generated from physical part scanning. The need to deal with noise due to scanning causes unnecessary overheads which push these algorithms' processing time above the acceptable limit for machining simulation. Edge restoration in machining simulation is meant to restore the chamfered edges without noticeable alteration to the overall simulation time. Since machining simulation time is typically in the order of seconds, edge restoration needs to be a sub-second processing task. Therefore, a fast edge restoration method for triangle mesh models derived from grid-based machining simulation is needed and presented in this paper.

2. Methodology

The initial step in edge restoration is the proper identification of chamfered edges. Identification of these entities not only significantly narrows the processing regions for the subsequent edge restoration algorithm, it also segments the triangle mesh model into geometrically similar patches. This facilitates faster and more accurate information extraction necessary for the realization of a computationally fast edge restoration scheme.

A parallel two-component edge extraction approach is adopted in this work to detect chamfered edges efficiently and reliably. One component is an edge-based segmentation method and the other is a feature-based segmentation method. This work takes advantage of the benefits of the two dissimilar segmentation methods and attempts to mitigate their individual issues via the strength of the other method. For the edge-based segmentation method, the primary advantage is its simplicity and computational speed. It is only necessary to distinguish between edges and features (non-edges), making the segmentation problem simple. The edge-base method can, thus, quickly extract edges with satisfactory precision. However, the edge-based method often needs manual editing in order to achieve good extraction results [1]. Without manual post-processing, broken, incomplete and incorrect edge segments are typically obtained. To overcome this issue, the result from an efficient feature-based segmentation method is utilized to filter the edge-based segmentation results. Although also unable to perform accurate edge extraction completely, feature-based segmentation focuses on geometric features rather than edges. It will, therefore, produce similar but often different edge extraction results. The difference is used to filter the edge-based segmentation results. Specifically, any broken and/or incomplete edge segment that falls on a feature is removed with complete edge segments kept. Furthermore, the segmentation result is used to identify the corner triangles which are a subset of the edge triangles. The combination of the two edge extraction methods gives fast computational speed and good edge extraction results.

With the edges and corners extracted, per-vertex normals are estimated via averaging the one-ring non-edge face normals to facilitate edge and corner restoration. Once the normal estimate is complete, the edges and corners are restored through splitting the edge and corner triangles. New vertices representing edges and corners are added and their locations are determined based on the adjacent per-vertex normals. Fig. 3 outlines the presented edge restoration method. Further technical details are given in the following sections.

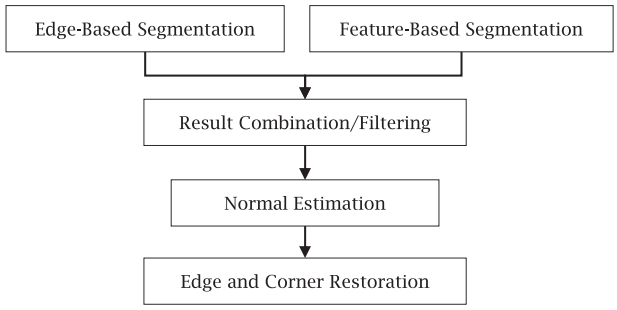

Figure 3. Main steps in the presented edge restoration method.

3. Edge-based segmentation

In general, edge-based segmentation used in computer graphics focuses on the extraction of specific edge lines [9],[22]. Most often, edge lines are defined via the first and second order curvature derivatives on shapes approximated by dense triangle meshes. However, in triangle mesh workpiece models produced by a grid-based machining simulation system, finding an edge line is not useful. In fact, no specific edge lines exist since the edges are chamfered. Hence, the applicable approach for such mesh models with chamfered edges is to detect the edge regions rather than the edge lines. The edge region detection essentially divides the mesh vertices into two groups: edge vertices and non-edge vertices. To perform edge region segmentation, the method developed by Huang and Ascher [12] is used with some simplifications. Although the method indeed promotes the extraction of edge regions rather than edge lines, it is still built around the assumption that the input models possess very sharp edges. For mesh models with chamfered edges, this method would fail. Nevertheless, the method can be adapted to work for mesh models with chamfered edges. The adapted method is simple and robust. It uses K-means clustering to group the mesh vertices with further processing to refine the segmentation result.

3.1. K-means clustering

The core of the edge region segmentation method of Huang and Ascher [12] is K-means clustering. K-means is an unsupervised clustering algorithm that allocates every data point to one of the K clusters to minimize the within-cluster sum of squares of a specific measure. For this work, per-vertex curvature is used as the measure. The per-vertex curvature is widely used in segmentation studies [1],[24], especially the per-vertex principal curvatures which provide a suitable classification measure for edge and non-edge points. By utilizing K-means clustering with principal curvatures as the measure, the edge and non-edge portion of the mesh model can be distinguished efficiently.

To setup the objective function for edge detection only (no corner detection), suppose all the principal curvatures of every vertex $[k_1(\boldsymbol{x}_i), k_2(\boldsymbol{x}_i)]$have been computed and that $k_1(\boldsymbol{x}_i) < k_2(\boldsymbol{x}_i)$. The clustering objective function is then formulated as [12]:

$$\sum_{k=1}^{2} \sum_{v \in S_k} ||\boldsymbol{v} - \boldsymbol{c}_k{}^2|| \tag{3.1}$$

where S_k represents the set of points which belong to group k and $\boldsymbol{c}_k$ is the center of set S_k.

To solve this optimization problem, two initial values of $\boldsymbol{c}_1^0$ and $\boldsymbol{c}_2^0$ are specified based on the principal curvatures. The terminal condition is:

$$||\boldsymbol{c}_1^j - \boldsymbol{c}_1^{j+1}|| < \epsilon_K, ||\boldsymbol{c}_2^j - \boldsymbol{c}_2^{j+1}|| < \epsilon_K \tag{3.2}$$

where $\epsilon_K = 10^{-5}$. K-means clustering then partitions the vertices into the desired two sets, edge and non-edge. Two questions still remain. First is how to calculate the principal curvatures $[k_1(\boldsymbol{x}_i), k_2(\boldsymbol{x}_i)]$ and the second is how to choose good initial values of $\boldsymbol{c}_1^0$ and $\boldsymbol{c}_2^0$.

3.2. Principal curvatures and initial values

Calculating principal curvatures accurately can be a computationally intensive and expensive task. Algorithms with higher precisions typically take an order of magnitude longer to calculate the principal curvatures than the high computational speed requirement of this work [21]. As stated previously, the objective of this work is to perform machined edge restoration for machining simulation, which has been targeted as a sub-second task. Hence, some trade-offs need to be made between computational efficiency and accuracy. To ensure efficient and satisfactorily accurate calculation of principal curvatures, local height intensity [12] has been employed to estimate the principal curvatures $[k_1(\boldsymbol{x}_i), k_2(\boldsymbol{x}_i)]$ as the initial values of $\boldsymbol{c}_1^0$ and $\boldsymbol{c}_2^0$. Since the local height intensity only depends on the first order information, it possesses good numerical robustness.

3.3. Refinement

To refine the quality of the segmentation result, some further adjustment is required. Specifically, extra and/or missing edge points may occur after the segmentation. This creates isolated and/or broken edges, respectively. To improve the edge-based segmentation result, a refinement process proposed by Huang and Ascher [12] is used. Essentially, the refinement process attempts to remove

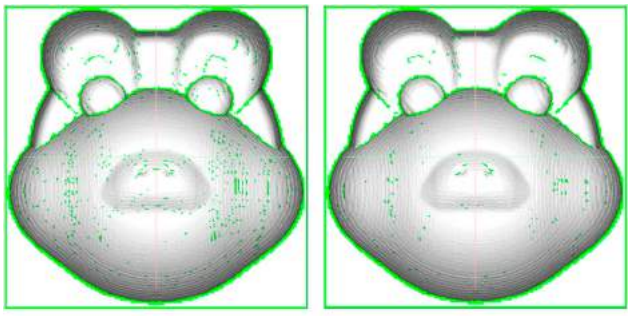

Figure 4. Edge-based segmentation result: (a) Before refinement, and (b) After refinement.

isolated edges and to extend possible broken edges. An integer P is used to control the number of vertices the refinement process can traverse on the mesh in an attempt to connect a possible broken edge. A value of 25 is found to work well for P. After the refinement, it is necessary to check each edge vertex for correctness by examining the normal variance of the edge vertex's one-ring neighbors. If the normal variance suggests that the edge vertex in effect lies on a flat region, the edge vertex is moved to the non-edge cluster. The normal variance threshold that dictates flatness is set as 10°. Fig. 4 shows the edge-based segmentation result on a machined cavity of a frog face before and after the refinement process. It can be seen that many of the isolated edge points have been removed while the valid edge points are retained.

4. Feature-based segmentation

With the edge-based segmentation method outlined above, chamfered edges on a triangle mesh model can be detected. However, the result still needs improvement prior to edge restoration. As shown in Fig. 4 (b), a good number of incorrect edge vertices still exist. This result would lead to incorrect edge restoration if used directly. Therefore, a feature-based segmentation method is used to improve the edge-based segmentation result prior to edge restoration.

Since feature segmentation has been an active area of research, many methods are available. Good surveys on existing mesh segmentation methods are available in the literature [1],[23]. For the purpose of this work, the basic region growing method will suffice. The method is simple and effective. In fact, due to the special geometric properties of the machined workpiece model mesh, the basic region growing method performs very well. Specifically, the workpiece model meshes are derived from a spatial grid, which minimizes the vertex normal difference within the specific grid. This enables the basic region growing method, utilizing the vertex normal difference as the segmentation measure, to appropriately divide the workpiece model mesh into individual features. The basic region growing method adopted in this work is outlined as follows:

1. Choose a seed triangle from the mesh which has not been visited, assign a new feature-group index to it, and then add it to the queue Q.
2. Take a triangle **u** from the queue Q and check the distance between **u** and one of its neighbor triangles **v**. If the difference $d(\boldsymbol{u}, \boldsymbol{v})$ between the normals of the triangles **u** and **v** is less than ε_θ (set as 8°) and **v** has not been assigned to a feature group, **v** is then set to be in the same feature group as **u** and placed into the queue Q.
3. Continue Step 2 until Q is empty (all vertices satisfying $d(\boldsymbol{u}, \boldsymbol{v}) < \varepsilon_\theta$ have been visited) and go back to Step 1.
4. For each vertex, choose to become a member of the feature group which has the most triangles in its one-ring neighborhood.

With the above region growing segmentation method, the geometry is broken up into feature patches. The interfaces of these feature patches are the second set of edge result. Combining the edge-based segmentation and feature-based segmentation results, the edge triangles are those triangles that connect with chamfered edges that form non-broken edges or are broken but situated on the interfaces of the segmented features. Feature-based segmentation, thus, helps reducing the number of incorrectly identified edge vertices from edge-based segmentation. Furthermore, the feature-based segmentation result aids in distinguishing edge triangles from corner triangles. The edge-based segmentation divides the mesh into only two regions, edge and non-edge. Corners are not extracted directly: the corner triangles are embedded in the edge triangles but not extracted. Feature-based segmentation can reliably distinguish these corner triangles from the plain edge triangles. With the feature-based segmentation result, corner triangles can be extracted from the edge regions through simply examining the adjacent triangles of an edge triangle. If the triangle lies between two features, it is a plain edge triangle. If the triangle lies among three features, it is a corner triangle.

It should be noted that region growing segmentation is not 100% accurate, either. The region growing segmentation method can falsely create small feature areas. To address this issue, any segmented feature patches composed of less than a specified small number of triangles are labeled "tiny". The specific number of triangles in a tiny feature patch is evidently variable but it has been found that the number 25 works well for many meshes.

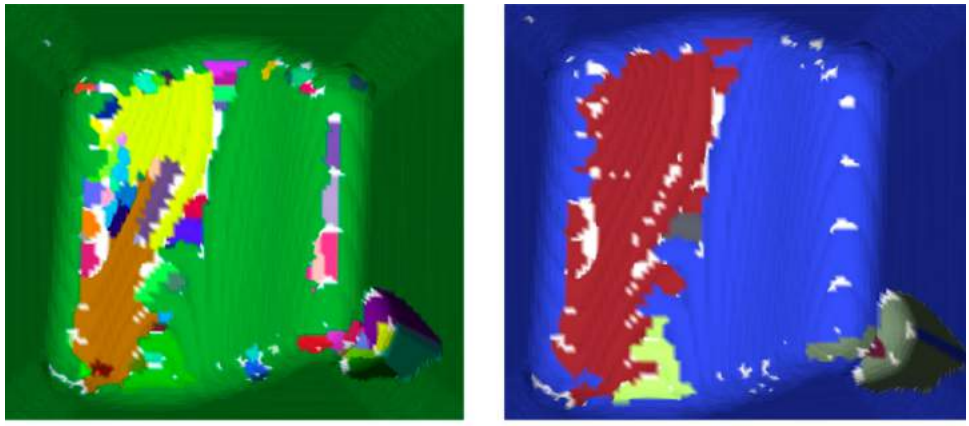

Figure 5. Merging of over-segmented feature patches: (a) Before merging, and (b) After merging.

Once a feature patch has been classified as tiny, its feature-group index is deleted and the feature patch is moved into its adjacent non-edge group. Furthermore, the potential over-segmentation issue is a bit more complex. A greedy algorithm [10] has been used to merge the over-segmented areas. It can be seen in Fig. 5a that the relatively flat bottom machined surface of the Cavity model (to be shown in Fig. 8) has been over-segmented into many false small feature patches (shown in individual colors). The employed merging process is able to significantly reduce the large number of small feature patches to only a handful of larger patches as shown in Fig. 5b.

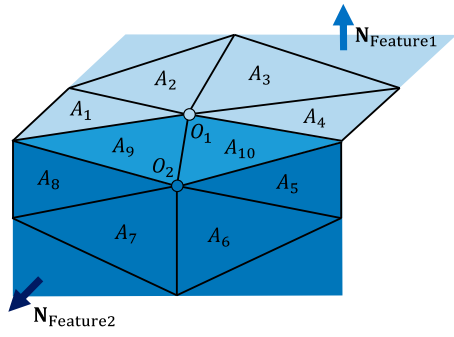

Figure 6. Estimated normal calculation.

5. Edge restoration

With the chamfered edge and corner triangle extraction completed, the edges and corners can be recovered. Recovering of such sharp features is often done based on some mesh de-noising methods [8],[27],[29]. However, most of the de-noising filters assume that the input data contains some noise, which obviously is not the case for machining simulation since every mesh vertex in the workpiece model is an exact tool and workpiece intersection point. An efficient numerical optimization method by Attene et al. [5] is employed in this work to split the chamfered edge and corner triangles to sharpen the edges/corners using the normals of the involved vertices. It is, thus, necessary to compute a reliable normal estimate at each involved vertex. The correct vertex normal is the normal that represents the surface normal of the adjacent non-edge feature and estimated from the corresponding adjacent triangles of the vertex.

5.1. Normal estimation

Normal estimation around sharp edges is generally a very challenging task due to the presence of noise in the data points [7],[13]. In machining simulation, normal estimation is not as challenging and can be done efficiently since all vertices are computed without noise. To determine correct normals at vertices in a chamfered region, the detected edge and corner result is to be used first. For each chamfered edge or corner vertex, only the adjacent non-edge feature patch is used for normal estimation. The non-edge patches represent features in the machined workpiece model. Their intersections are to form the sharp edges and corners. Therefore, their normals are required to restore the correct edges and corners. To estimate the normal at O_1 in Fig. 6, the following expression is used:

$$\mathbf{N}_{O_1} = \sum_{A_1,A_2,A_3,A_4} \theta_i \mathbf{N}_i / \sum \theta_i \qquad (5.1)$$

where θ_i is the angle of the corner in triangle A_i that is incident to O_1 and $\mathbf{N}_i$ is the face normal of the triangle A_i. The triangles A_1, A_2, A_3, A_4 are used to estimate the normal at vertex O_1 and the triangles A_5, A_6, A_7, A_8 are used to estimate the normal at O_2.

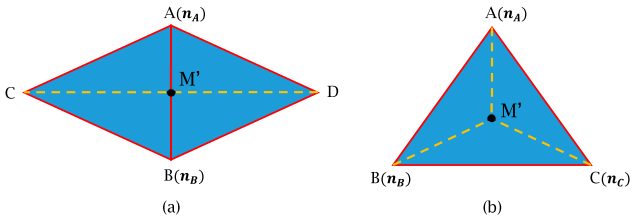

Figure 7. Triangle split for edge restoration: (a) Edge triangles, and (b) Corner triangle.

5.2. Restoration

After the chamfered edge and corner triangle detection, the next step for machined edge restoration is to apply a splitting operation based on the method of Attene et al. [5] to restore the edges and corners, as shown in Fig. 7. Let M represent the newly inserted point which will lie on the ideal edge or corner of the mesh. Since only an approximation to the ideal location of M is attainable in this work, the newly inserted point M is designated as M' instead in the figure. With the approximated normal at each vertex ($\boldsymbol{n_A}$ and $\boldsymbol{n_B}$), the new point M' is determined for the edge triangles, as illustrated in Fig. 7(a), by:

$$\begin{cases} (\boldsymbol{M'} - \boldsymbol{A}) \cdot \boldsymbol{n_A} = 0 \\ (\boldsymbol{M'} - \boldsymbol{B}) \cdot \boldsymbol{n_B} = 0 \\ (\boldsymbol{M'} - (\boldsymbol{A} + \boldsymbol{B})/2) \cdot (\boldsymbol{n_A} \times \boldsymbol{n_B}) = 0 \end{cases} \tag{5.2}$$

For the corner triangle, as illustrated in Fig. 7(b), the new point M' is determined by:

$$\begin{cases} (\boldsymbol{M'} - \boldsymbol{A}) \cdot \boldsymbol{n_A} = 0 \\ (\boldsymbol{M'} - \boldsymbol{B}) \cdot \boldsymbol{n_B} = 0 \\ (\boldsymbol{M'} - \boldsymbol{C}) \cdot \boldsymbol{n_C} = 0 \end{cases} \tag{5.3}$$

With the new points inserted as above, machined edges and corners can be restored. It should be noted here that the method presented in this paper has used both curvature and normal estimates at each mesh vertex to reliably restore the machined edges and corners. This is conceptually different than many existing methods based on evaluating only the variation of normals at mesh vertices such as the method of Kobbelt et al. [18]. Not only superior restored edge quality is expected due to the consideration of both curvature and normal variations but the restoration can be achieved as a sub-second computational task using the method in this work.

5.3. Remove flipped and spike triangles

The quality of the resulting triangles from the above point insertion process is not always good. The most critical problem is the occurrence of flipped triangles for edge triangles and spike triangles for corner triangles. A flipped triangle can occur when the projection of the new edge vertex M' onto the triangle plane it is derived from lies outside of the triangle's boundary. This does not happen for edges that interface feature faces with very different normals. For edges with adjacent feature faces that have very similar normals, flipped triangles can become a problem due to the normals being approximated. A slight variance in the approximated normal from the ideal normal can cause M' to be displaced very far from M (the ideal edge point location). Furthermore, since a triangle mesh is only a piecewise approximation of the actual machined surface geometry, mesh resolution can also cause flipped triangles. Due to the lack of resolution, small features can be lost within a triangle. This implies that the normals of the features adjacent to an edge triangle can disagree. It can than lead to a false M' location. If higher resolution is used, better approximation of M' can be found. However, if the resolution used is insufficient, M' can be displaced far from the ideal location M.

In order to avoid these flipped edge triangles, the following constraint has to be satisfied when adding M' for an edge triangle:

$$[(\boldsymbol{N}_{20} \times \boldsymbol{D}) \cdot \boldsymbol{N}_2][(\boldsymbol{N}_{21} \times \boldsymbol{D}) \cdot \boldsymbol{N}_2] < 0 \tag{5.4}$$

where $\boldsymbol{N}_2$,$\boldsymbol{N}_{20}$, $\boldsymbol{N}_{21}$, $\boldsymbol{D}$ are:

$$\begin{aligned} \boldsymbol{N}_2 &= \boldsymbol{n}_a \times \boldsymbol{n}_b \\ \boldsymbol{D} &= \boldsymbol{B} - \boldsymbol{A} + [(\boldsymbol{B} - \boldsymbol{A}) \cdot \boldsymbol{N}_2]\boldsymbol{N}_2 \\ \boldsymbol{N}_{20} &= \boldsymbol{N}_2 \times \boldsymbol{n}_a \\ \boldsymbol{N}_{21} &= \boldsymbol{N}_2 \times \boldsymbol{n}_b \end{aligned} \tag{5.5}$$

It has been decided that given the cause of the flipped edge triangles, it is best to avoid adding any new M' for the triangles that do not satisfy the constraint above. These flipped triangles are the result of an invalid triangle mesh approximation. It is not meaningful to find a solution when the approximation accuracy or resolution is inadequate to gauge where M should be located. Therefore, if the above constraint is not satisfied, the edge triangle is not split and M' is not added.

When the same issue discussed above occurs for corners, spike triangles are typically generated. The mechanism that causes spike triangles is the same as that for flipped triangles, except that it occurs for a corner triangle. For instance, if the low mesh resolution leads to three normals in a corner triangle to be nearly parallel, the resulting M' can be far displaced from its ideal location M. There is no telling if M is actually at the tip of the spike or anywhere else because there is inadequate information regarding the "should-be" shape of this particular corner. Hence, if adding M' will result in a spike triangle, M' is not added. To identify a spike

triangle, the differences between the normals of the triangles that surround a corner triangle are compared. If an angle difference of these normals is above 150°, M' is not added. With this added constraint and the one to avoid flipped triangles, the edge restoration method presented in this paper produces quality and well-structured meshes.

6. Implementation results

Fig. 8 shows three typical results of the presented edge restoration method for machining simulation. The "before" workpiece mesh models was produced by FSV-rep [16]. Three-axis and five-axis milling tool paths were used to shape the workpiece geometry, resulting in machined triangle mesh models with chamfered edges. The FSV-rep resolution of all these test cases was set as 0.5 mm. All simulation was performed on a Windows-based PC with 3.3 GHz processor and 12GB of memory.

As can be seen in the figure above (especially in the zoom in views), the edges and corners for the three cases shown are restored correctly. For the first case (the Mech model), as this was a model machined by 2½-D flat-end milling with no machined scallops, the edge restoration was achieved with a 100% rate. This means that all edge and corner triangles were identified correctly and all new edge and corner points were inserted at the correct locations. For the second case (the FrogFace model), as this was a model machined by 3D ball-end milling, machined scallops were present on free-form surface patches that were adjacent to the flat top surface, the presented edge restoration method restored about 96% of the edge and corner points. 4% of the edge and corner triangles were either not detected correctly or were not moved to the correct locations. For the third case (the Cavity model), as all the machined surfaces had machined scallops, the edge and corner restoration rate was at 79%. The lower restoration rate was expected since intersections of machined surfaces all characterized with scallops would produce a very complex region of intersecting edges (many scallop edges intersecting with the model edges), which made it very difficult to correctly restore the model edges. Even so, the developed method was found to be able to restore edges and corners in a visually satisfactory manner.

The method presented in this paper is seen to be able to restore edges and corners better than the leading grid-based machining simulation software, ModuleWorks. Fig. 9 shows a comparison among the machined geometry of a partial impeller produced by Siemens NX (served as the reference and run at maximum resolution), the presented method, and ModuleWorks. The workpiece model resolution for both the presented method and ModuleWorks was set as 0.5 mm. It can be seen in the figure that ModuleWorks has produced some questionable edge restoration results (seen in the call-out enlarged views). For the presented method, the questionable areas were much better processed.

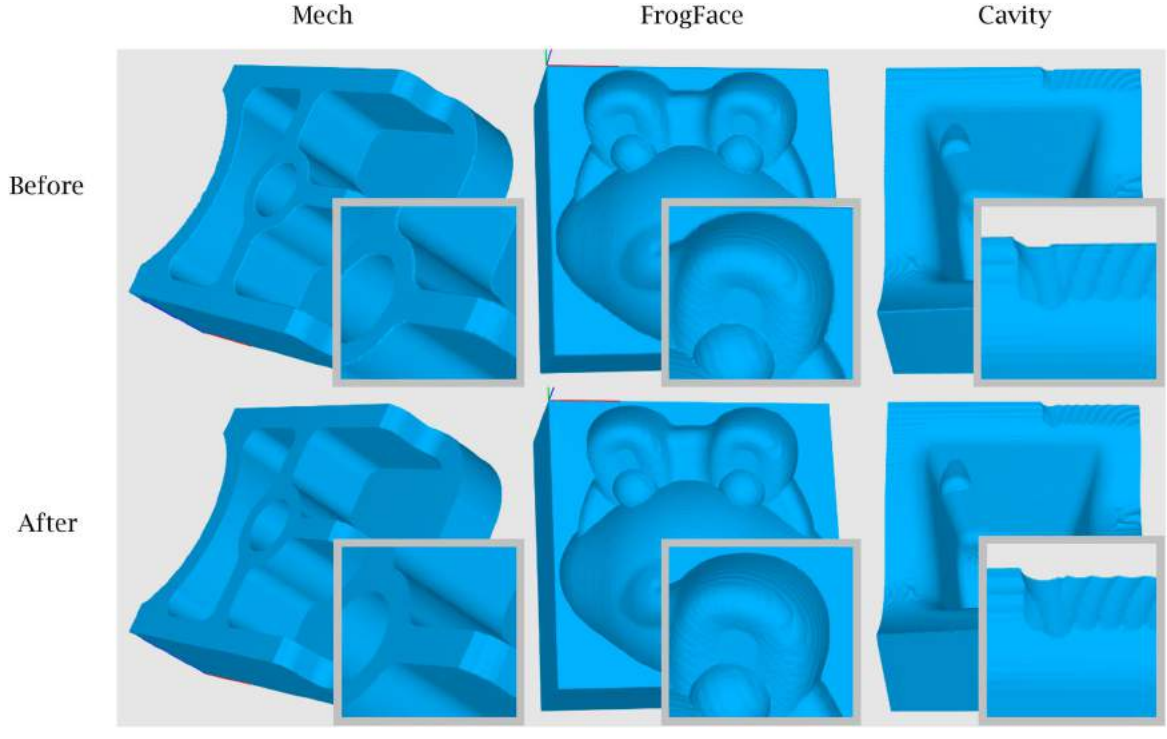

Figure 8. Edge restoration cases for triangle mesh models from grid-based machining simulation.

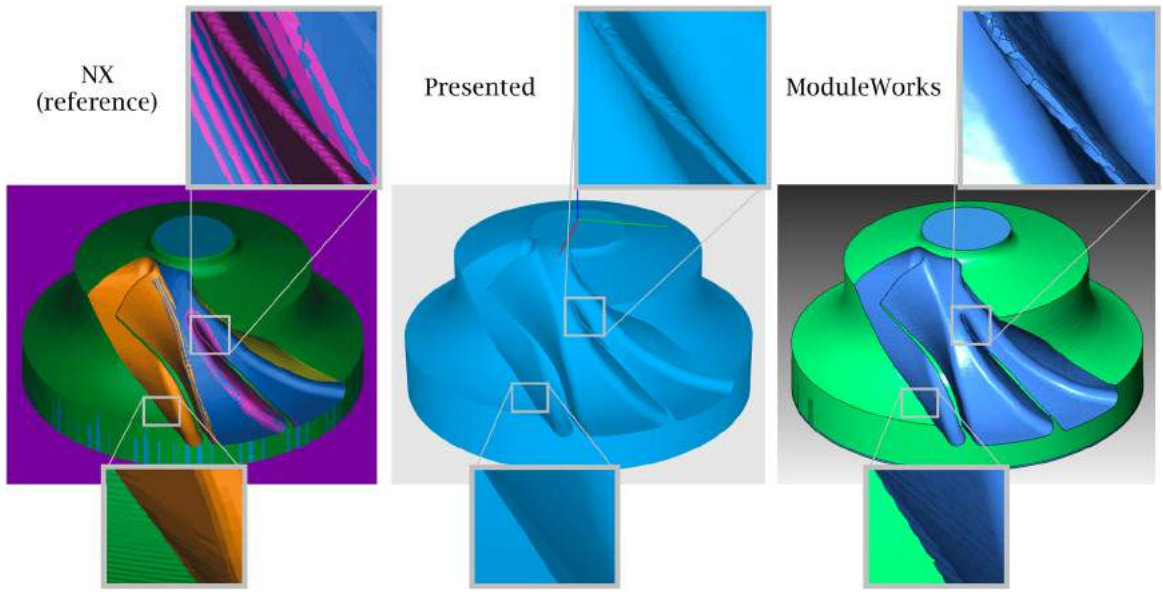

Figure 9. Comparison of edge restoration results.

Table 1. Total processing time and its breakdown of the presented edge restoration method.

Number of Triangles	Mech 277,524	FrogFace 255,792	Cavity 272,152	Impeller 382,254
Edge-Based Segmentation (sec.)	0.037	0.047	0.053	0.069
Feature-Based Segmentation (sec.)	0.116	0.100	0.147	0.168
Edge and Corner Restoration (sec.)	0.084	0.069	0.084	0.116
Total (sec.)	**0.237**	**0.216**	**0.284**	**0.353**

The presented method is also seen to be able to complete the edge restoration task efficiently. Tab. 1 lists the total processing time for the four cases shown in Figs. 8 and 9 along with the breakdown for the three main processing steps. The presented method can complete the edge restoration task in all these cases under 0.5 seconds with each processing step taking a small amount of time. As the number of faces in the model increases, the processing time increases as expected.

7. Conclusions

This paper presented a method to restore machined edges for triangle mesh workpiece models derived from grid-based machining simulation. The method is computationally fast with high restoration rate. By combining existing simple but efficient edge-based and feature-based segmentation and edge restoration algorithms, an effective edge restoration method is developed. Edges and corners of complex machined geometry can be restored with sub-second computational time. With the presented method, the chamfer edges inherit in grid-based machining simulation can be addressed satisfactorily. Comparable simulation output quality to that of much less efficient NURBS and triangle mesh approaches is achieved. Still, the presented method cannot guarantee 100% edge and corner restoration for all the machined geometry due to challenges in correctly identifying all the chamfer edge and corner triangles. Feature detection and segmentation of triangle meshes is a challenging and active research subject. Many factors, such as triangle size, aspect ratio and distribution, affect the detection and segmentation results. The need for edge restoration to be a sub-second computing task and the vast variety of machined geometry further adds to the difficulty. Continuing work is evidently required in feature detection and segmentation in order to improve the restoration rate of machined edges for grid-based machining simulation.

Acknowledgements

The machining simulation research work being conducted in our group has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The Mitacs Globalink Research Internship awarded to the first author of this paper is also gratefully acknowledged. The FrogFace model was created by the paper's third author along with his teammates Ted Angus, Alicia Figueira and Josh Harrington as part of a graduate course project at UBC Mechanical Engineering. The Cavity model and the Impeller model were retrieved from the Siemens NX CAST Library of part models.

ORCID

Ziqi Wang http://orcid.org/0000-0002-3817-3922
Jack Szu-Shen Chen http://orcid.org/0000-0002-1225-6340
Jimin Joy http://orcid.org/0000-0003-4499-2366
Hsi-Yung Feng http://orcid.org/0000-0001-6189-6910

References

[1] Agathos, A.; Pratikakis, I.; Perantonis, S.; Sapidis, N.; Azariadis, P.: 3D mesh segmentation methodologies for CAD applications, Computer-Aided Design and Applications, 4(6), 2007, 827-841. https://doi.org/10.1080/16864360.2007.10738515

[2] Altintas, Y.; Kersting, P.; Biermann, D.; Budak, E.; Denkena, B.; Lazoglu, I.: Virtual process systems for part machining operations, CIRP Annals - Manufacturing Technology, 63(2), 2014, 585-605. https://doi.org/10.1016/j.cirp.2014.05.007

[3] Aras, E.; Feng, H. Y.: Vector model-based workpiece update in multi-axis milling by moving surface of revolution, International Journal of Advanced Manufacturing Technology, 52(9-12), 2011, 913-927. https://doi.org/10.1007/s00170-010-2799-8

[4] Aras, E.; Yip-Hoi, D.: Geometric modeling of cutter/workpiece engagements in three-axis milling using polyhedral representations, ASME Journal of Computing and Information Science in Engineering, 8(3), 2008, 031007. https://doi.org/10.1115/1.2960490

[5] Attene, M.; Falcidieno, B.; Rossignac, J.; Spagnuolo, M.: Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces, Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2003, 62-69.

[6] Benouamer, M. O.; Michelucci, D.: Bridging the gap between CSG and Brep via a triple ray representation, Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications, 1997, 68-79. https://doi.org/10.1145/267734.267755

[7] Fleishman, S.; Cohen-Or, D.; Silva, C. T.: Robust moving least-squares fitting with sharp features, ACM Transactions on Graphics, 24(3), 2005, 544-552. https://doi.org/10.1145/1073204.1073227

[8] Fleishman, S.; Drori, I.; Cohen-Or, D.: Bilateral mesh denoising, ACM Transactions on Graphics, 22(3), 2003, 950-953. https://doi.org/10.1145/882262.882368

[9] Gal, R.; Sorkine, O.; Mitra, N. J.; Cohen-Or, D.: iWIRES: An analyze-and-edit approach to shape manipulation, ACM Transactions on Graphics, 28(3), 2009, Article No. 33. https://doi.org/10.1145/1531326.1531339

[10] Garland, M.; Heckbert, P. S.: Surface simplification using quadric error metrics, Proceedings of SIGGRAPH '97, 1997, 209-216. https://doi.org/10.1145/258734.258849

[11] Gong, X.; Feng, H. Y.: Cutter-workpiece engagement determination for general milling using triangle mesh modeling, Journal of Computational Design and Engineering, 3(2), 2016, 151-160. https://doi.org/10.1016/j.jcde.2015.12.001

[12] Huang, H.; Ascher, U.: Surface mesh smoothing, regularization, and feature detection, SIAM Journal on Scientific Computing, 31(1), 2008, 74-93. https://doi.org/10.1137/060676684

[13] Huang, H.; Wu, S.; Gong, M.; Cohen-Or, D.; Ascher, U.; Zhang, H.: Edge-aware point set resampling, ACM Transactions on Graphics, 32(1), 2013, Article No. 9. https://doi.org/10.1145/2421636.2421645

[14] Jang, D.; Kim, K.; Jung, J.: Voxel-based virtual multi-axis machining, International Journal of Advanced Manufacturing Technology, 16(10), 2000, 709-713. https://doi.org/10.1007/s001700070022

[15] Joy, J.; Feng H. Y.: Frame-sliced voxel representation: An accurate and memory-efficient modeling method for workpiece geometry in machining simulation, Computer-Aided Design, 88, 2017, 1-13. https://doi.org/10.1016/j.cad.2017.03.006

[16] Joy, J.; Feng H. Y.: Efficient milling part geometry computation via three-step update of frame-sliced voxel representation workpiece model, International Journal of Advanced Manufacturing Technology, 2017, in press. https://doi.org/10.1007/s00170-017-0168-6

[17] Karunakaran, K. P.; Shringi, R.; Ramamurthi, D.; Hariharan, C.: Octree-based NC simulation system for optimization of feed rate in milling using instantaneous force model, International Journal of Advanced Manufacturing Technology, 46(5-8), 2010, 465-490. https://doi.org/10.1007/s00170-009-2107-7

[18] Kobbelt, L. P.; Botsch, M.; Schwanecke, U.; Seidel, H.-P.: Feature sensitive surface extraction from volume data, Proceedings of SIGGRAPH '01, 2001, 57–66. https://doi.org/10.1145/383259.383265

[19] Lee, S. W.; Nestler, A.: Virtual workpiece: Workpiece representation for material removal process, International Journal of Advanced Manufacturing Technology, 58(5-8), 2012, 443-463. https://doi.org/10.1007/s00170-011-3431-2

[20] Liu, Z.; Pan, M.; Yang, Z.; Deng, J.: Recovery of sharp features in mesh models, Communications in Mathematics and Statistics, 3(2), 2015, 263-283. https://doi.org/10.1007/s40304-015-0059-9

[21] Magid, E.; Soldea, O.; Rivlin, E.: A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data, Computer Vision and Image Understanding, 107(3), 2007, 139-159. https://doi.org/10.1016/j.cviu.2006.09.007

[22] Quan, W.; Meng, W.; Zhang, X.: The extraction of feature lines on 3D models: A survey, Proceedings of the 2014 International Conference on Virtual Reality and Visualization, 2014, 220–225. https://doi.org/10.1109/ICVRV.2014.11

[23] Ren, Y.; Zhu, W.; Lee, Y. S.: Feature conservation and conversion of tri-dexel volumetric models to polyhedral surface models for product prototyping, Computer-Aided Design and Applications, 5(6), 2008, 932-941. https://doi.org/10.3722/cadaps.2008.932-941

[24] Shamir, A.: A survey on mesh segmentation techniques, Computer Graphics Forum, 27(6), 2008, 1539-1556. https://doi.org/10.1111/j.1467-8659.2007.01103.x

[25] Spence, A. D.; Abrari, F.; Elbestawi, M. A.: Integrated solid modeller based solutions for machining, Computer-Aided Design, 32(8-9), 2000, 553-568. https://doi.org/10.1016/S0010-4485(00)00042-7

[26] Stifter, S.: Simulation of NC machining based on the dexel model: A critical analysis, International Journal of Advanced Manufacturing Technology, 10(3), 1995, 149-157. https://doi.org/10.1007/BF01179343

[27] Sun, X.; Rosin, P.; Martin, R.; Langbein, F.: Fast and effective feature-preserving mesh denoising, IEEE Transactions on Visualization and Computer Graphics, 13(5), 2007, 925-938. https://doi.org/10.1109/TVCG.2007.1065

[28] Wou, S. J.; Shin, Y. C.; El-Mounayri, H.: Ball end milling mechanistic model based on a voxel-based geometric representation and a ray casting technique, Journal of Manufacturing Processes, 15, 2013, 338-347. https://doi.org/10.1016/j.jmapro.2012.12.003

[29] Zheng, Y.; Fu, H.; Au, O. K. C.; Tai, C. L.: Bilateral normal filtering for mesh denoising, IEEE Transactions on Visualization and Computer Graphics, 17(10), 2011, 1521-1530. https://doi.org/10.1109/TVCG.2010.264