Title:
Edge Restoration for Triangle Mesh Models Derived from Grid-Based Machining Simulation

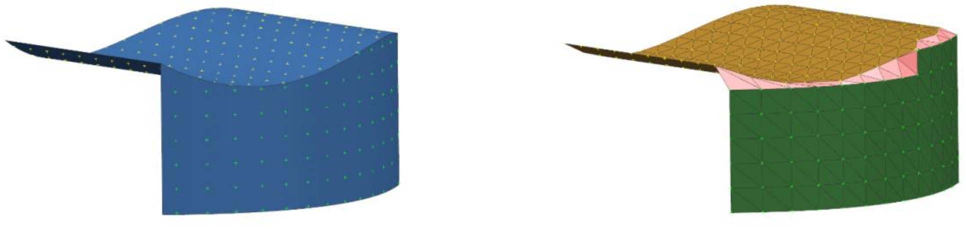
Authors:
Ziqi Wang, qiqi007@mail.ustc.edu.cn, The University of British Columbia
Jack Szu-Shen Chen, jsschen38@gmail.com, The University of British Columbia
Jimin Joy, jiminjoy@mail.ubc.ca, The University of British Columbia
Hsi-Yung Feng, feng@mech.ubc.ca, The University of British Columbia

Keywords:
Machining Simulation, Workpiece Model, Machined Edge Restoration

DOI: 10.14733/cadconfP.2017.384-388

Introduction:

Chamfered edges occur in triangle mesh models generated from discrete machining simulation methods such as vector approximation methods [3],[11],[13] and voxel and space partitioning methods [6-9]. The 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 appears as a set of triangles that runs between the triangles representing the surfaces on either side of the original edge (Fig. 1). Chamfered edges happen for triangle meshes generated from the vector and voxel based simulations because of the grid structure of the underlying simulation methods. Sampled surface points are only available at grid crossings. It is often the case that edges do not cross the grid exactly. These edges are not captured, resulting in chamfered edges. Presence of chamfered edges reduces the accuracy of the generated mesh model and deteriorates the visual quality. Thus, chamfered edges need to be restored as they happen unwantedly. Machining simulation using vector and voxel based simulation methods is quite popular due to its efficiency but the issue of chamfered edges needs to be properly resolved. To address this issue, this paper presents a fast edge restoration scheme for triangle mesh models generated from grid-based machine simulation.



(a) (b)

Fig. 1: Chamfered edges: (a) Two ideal machined surfaces with an intersecting edge and sampled 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.

In order to restore the edges of a triangle mesh from grid-based machining simulation, a new method has been developed after considering existing methods applied in similar situations. Among the existing methods, feature conserving triangle mesh generation for tri-dexels by Ren et al. [13] 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. Generic edge feature restoration techniques are available such as bilateral denoising by Fleishman et al. [4] and sharp feature recovery using an energy optimization technique by Liu et al. [12]. However, these methods target noisy meshes generated from physical part scanning. The need to deal with noise derived from 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.

Proposed Method:

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 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 developed in this work to detect chamfered edges efficiently. 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 method, the primary advantage is its simplicity and speed. It is only necessary to distinguish between edges and non-edges, making the segmentation problem simple. The edge-based method can, thus, quickly extract edges with tolerable precision. In this work, *K*-mean clustering [5] is adapted and used for the edge segmentation. Per-vertex principal curvature values are used as the edge segmentation measure. Nonetheless, edge-based segmentation lacks extraction precision. Manual editing is often needed to achieve good extraction precision [1]. Without manual post-processing, broken, incomplete and incorrect edge segments are typically obtained. To overcome this issue, a refinement process is applied and the result from an inexpensive feature-based segmentation method is then used to filter the edge-based segmentation results. The employed refinement process checks for broken edge segments. If a particular edge segment is not closed and is within a specified length, the edge segment is removed. After the edge segment check, it is necessary to verify each edge vertex for correctness by examining the normal variance of the edge vertex's one-ring neighbors. If the variance suggests that the edge vertex lies on a flat region, the edge vertex is moved to the non-edge group.

To complete the edge extraction task, feature-based segmentation is used to improve the results of edge-based segmentation. Since the subject of feature segmentation has been studied in depth, many methods are available. Comprehensive surveys of mesh segmentation by Shamir [14] and Agathos et al. [1] include many of these methods. For the purpose of this work, the basic region growing method suffices. The method is simple and effective. In fact, due to the special properties of the grid-based machining simulation model meshes, the basic region growing method performs very well. Since the meshes are derived through a grid system, the difference in vertex normals within a grid is minimal. This enables the basic region growing method, utilizing the vertex normal difference as the segmentation measure, to divide the input meshes into individual feature patches efficiently and reliably. Still, feature-based segmentation via region growing does not perform precise edge extraction completely. However, feature-based segmentation focuses on features rather than edges; therefore, it produces similar but often different restored edges than the edge-based method. This difference is used to filter the results from the edge-based method. Any broken and/or incomplete edge segment that falls on a feature is removed. Complete edge segments determined from the edge-based segmentation are kept. Further, the feature-based segmentation result is used to identify the corner triangles which are a subset of the edge triangles. The parallel combination of the two types of methods gives high computational speeds 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 estimate of normals is complete, the edges and corners are restored by splitting the edge and corner triangles. This is done based on the method of Attene et al. [2]. As shown in Fig. 2, new vertices representing edges and corners (M' in Fig. 2) are added and their locations are determined from the adjacent per-vertex normals. It should be noted that the method presented here has used both curvature and normal estimates at each mesh vertex to restore the machined edges and corners. This is conceptually different than many existing methods such as the method of Kobbelt et al. [10] that evaluates only the variation of normals at mesh vertices.

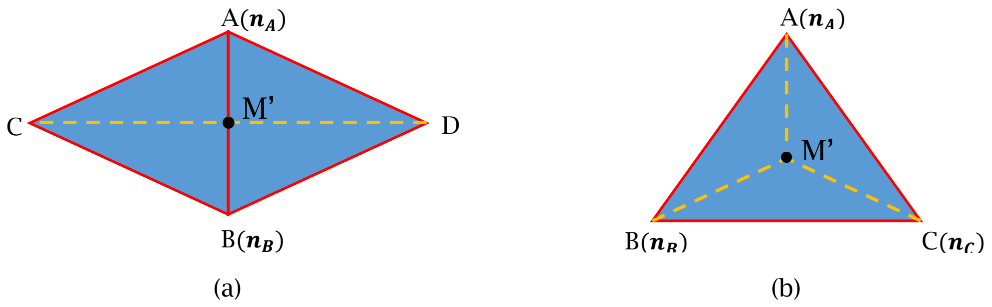
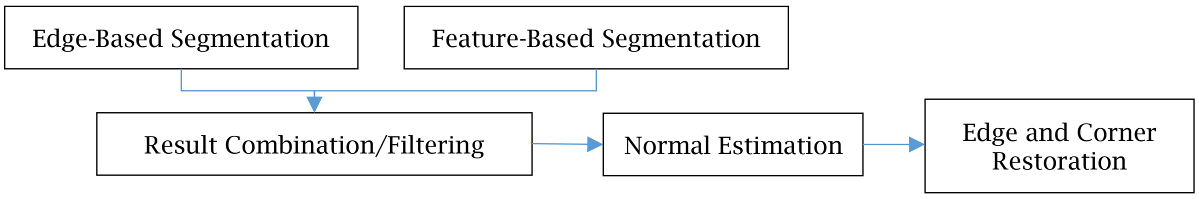


Fig. 2: Triangle split for edge restoration: (a) Edge triangles, and (b) Corner triangle.

It should be noted that the quality of the resulting triangles due to the above point insertion is not always good. The most critical problem is flipped triangles for edge triangles and spikes for corner triangles. A flipped triangle can be generated when the projection of the new edge vertex M' to the triangle's plane it is derived from lies outside of the triangle's projected boundary. This does not happen for edges that span features with very different normals. For edges with adjacent features that have similar normals, flipped triangles become a notable issue. A slight variance in the approximated normal from the actual normal can cause M' to be displaced far from M (the actual edge point location). Furthermore, since a triangle mesh is only a piecewise approximation of the actual machined geometry, mesh resolution can likewise 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. If the resolution used is not sufficient, M' can be displaced far from M. To avoid these flipped edge triangles, projection of M' must fall within the projected boundary of the triangles which M' is derived from. It has been decided that given the cause of flipped edge triangles, it is best to avoid adding any new M' for the triangles that do not satisfy the above condition. These flipped triangles result from the limit of 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 condition is not satisfied, the edge triangle is not split and M' is not added.

When the same issue discussed above occurs for corners, spikes are generated. The mechanism that causes spikes is the same as that for flipped triangles. Hence, if adding M' results in a spike, M' is not added. To identify a spike, 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 condition and the one to avoid flipped triangles, the proposed edge restoration method in this work, shown in Fig. 3, produces very good results.



Edge-Based Segmentation
Feature-Based Segmentation
Result Combination/Filtering
Normal Estimation
Edge and Corner Restoration

Fig. 3: Main steps in the presented edge restoration method.

Summary of Implementation Results:

By effectively combining simple and yet efficient edge-based and feature-based segmentation and edge restoration methods, a computationally fast edge restoration method for triangle mesh models derived from grid-based machining simulation is developed. Fig. 4 shows some edge restoration results. The model edges for the three cases shown can be seen to be restored correctly. For the first case (the Mech model), as this is a model machined by 2½-D flat-end milling with no machined scallops, the edge restoration is achieved with a 100% rate. For the second case (the FrogFace model), as this is a model machined by 3D ball-end milling, machined scallops are present on free-form surface patches that are adjacent to the flat top surface, the developed edge restoration method restores about 96% of the edge and corner points. For the third case (the Cavity model), as all the machined surfaces have machined scallops, the edge and corner restoration rate is at 79%. The lower restoration rate is expected since intersections of machined surfaces all characterized with scallops produce a very complex region of intersecting edges (many scallop edges intersecting with the model edges), which makes it very difficult to correctly restore the model edges. Even so, the developed method is found to be able to restore edges and corners better than the existing methods. Tab. 1 lists the total processing time for the three cases along with the breakdown for the three main processing steps.

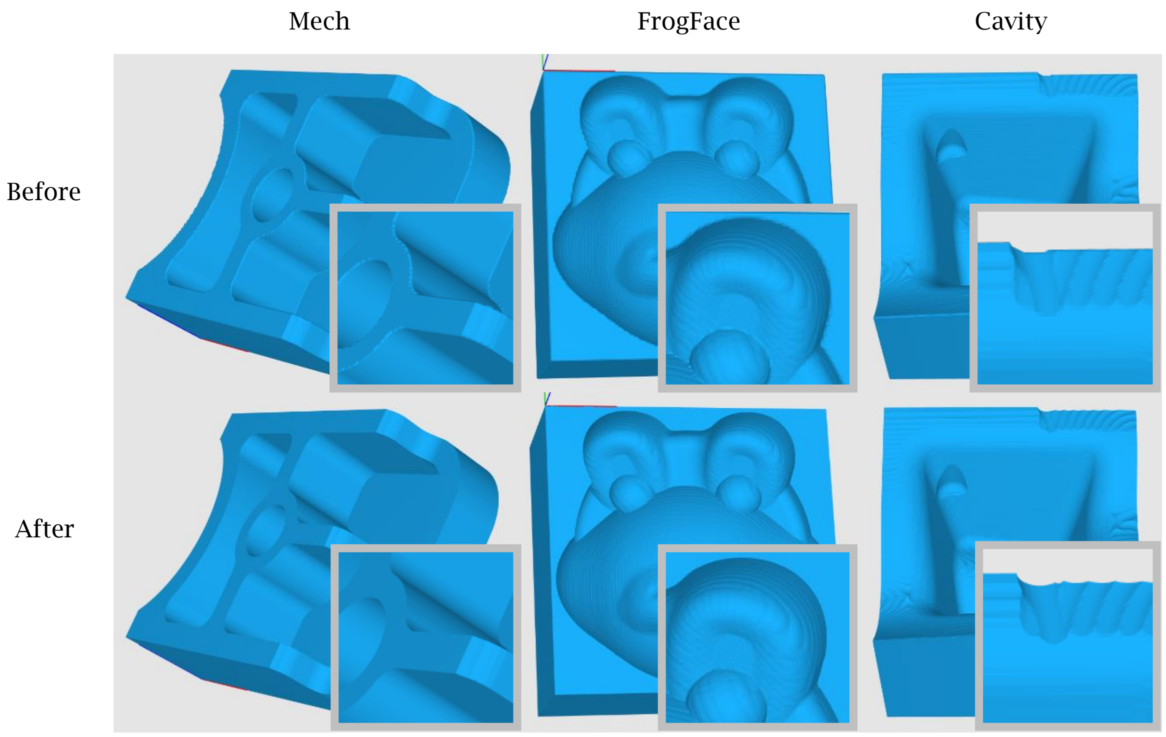


Fig. 4: Edge restoration cases for triangle mesh models from grid-based machining simulation.

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

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

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 is also gratefully acknowledged. The FrogFace model was created by the third author of this paper 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 was retrieved from the Siemens NX CAST Library of part models.

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] 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.

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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