EFFICIENT AND ACCURATE GEOMETRIC SIMULATION OF MULTI-AXIS MILLING OPERATIONS

by

Jimin Joy

B.Tech and M.Tech Dual Degree, Indian Institute of Technology Madras, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2017

Abstract

Geometric modeling is an essential part of process planning and verification step in the modern manufacturing practice that employs complex operations such as multi-axis milling. Geometric modeling by itself is used for tool path generation and verification. It is also essential to create important input for mechanistic simulation. Due to this great relevance, many geometric modeling methods have been employed for machining simulation. However it is still a challenge to obtain acceptable combination of accuracy, efficiency and robustness from most of the existing methods. The best known modeling methods also appear to have reached a saturation point. Yet the industrial machining cases are ever increasing in complexity and it demands for a faster method maintaining the acceptable level of accuracy.

This thesis presents an enhanced voxel representation format for modeling the machined workpiece geometry in general milling operations. The modeling format is named as Frame-Sliced Voxel representation (FSV-rep) as it uses a novel concept of frame-sliced voxels to represent the boundary of the workpiece volume in a multi-level surface voxel representation for memory-efficient implementation. Frame-sliced voxels enables approximation of the workpiece surface to achieve sub-voxel details. This thesis further identifies an efficient three-step update process that can be followed to compute machined part geometry from an initial FSV-rep workpiece model and set of tool paths. To be computationally feasible and yet robustly handling all tool path types, suitable swept volume representations are identified for various tool path categories. The three-step update process is then used in customized ways for the different categories to utilize the salient features of each. A robust and efficient approach to generate standard surface representation of the machined part geometry from the updated FSV-rep model

is also developed.

Results show that the FSV-rep model is able to provide acceptable accuracy levels while being significantly faster than popular modeling methods for machined part geometry computation in general multi-axis machining. The specialized swept volume representation identified for planar and 3-axis straight cut operations is further improving the FSV-rep update performance to be up to an order of magnitude faster than possible with general sampled swept volume representations.

Lay Summary

Computational geometry is widely used for modeling and simulation of machining processes with multi-axis milling operations. Geometric simulation is today essential to predict the shape of the part that will be created by a machining program and also to assist in predicting the mechanical load the machine tool will undergo. Because of these great relevance, many geometric modeling techniques have been attempted for the same.

This thesis presents a new geometric modeling method and an efficient way to use it in machining simulation. The work aims to outperform the existing methods in providing a better combination of accuracy, efficiency and robustness. In order to use the new modeling method, different ways of representing the moving cutting tool is explored and a suitable way is identified for each type of cutter motion. A fast and robust technique to extract popular mesh-based shape representation from the new modeling method is also developed.

Preface

This thesis presents details on the research work done by me for my PhD program under the supervision of Dr. Hsi-Yung Feng. Dr. Feng suggested the research topic of geometric machining simulation and discussed with me the need to conduct research in this area which formed the original motivation for the work. Various parts of the thesis have been or will be published in peer-reviewed research journals as listed below.

List of publications:

1. Joy J, Feng HY. Frame-sliced voxel representation: An accurate and memory-efficient modeling method for workpiece geometry in machining simulation. Computer-Aided Design. 2017;88:1-13.
2. Joy J, Feng HY. Efficient milling part geometry computation via three-step update of frame-sliced voxel representation workpiece model. International Journal of Advanced Manufacturing Technology. 2017;92(5-8):2365-78.
3. Joy J, Feng HY. Fast update of sliced voxel workpiece models using partitioned swept volumes of three-axis linear tool paths. Submitted.
4. Joy J, Chen JSS, Feng HY. Fast generation of 2-manifold triangle meshes for machined workpieces using a lookup table strategy. To be submitted.
5. Wang Z, Chen JSS, Joy J, Feng HY. Machined sharp edge restoration for triangle mesh workpiece models derived from grid-based machining simulation. Computer-Aided Design and Applications. Accepted.

Articles 1-4 were or will be written by me. For Article 5, I contributed to the parts that relate to grid-based machining simulation. All articles were or will be edited by my supervisor prior to submission. Detailed descriptions of my contributions to each article and how they correspond to the content to the thesis are provided below.

Article 1 corresponds to Chapter 3 of the thesis. I developed the new workpiece modeling format of frame-sliced voxel representation, implemented the associated data structures and performed the various case studies for comparative analysis.

Article 2 consists of the basic model update idea laid out in Chapter 4, tool instances model of Chapter 5, Section 5.3, the specific update techniques of Chapter 6, Section 6.5 and the results and discussion of Chapter 7, Section 7.2. I developed the ideas, wrote the computational programs and performed all the case studies entirely by myself except for the industrial case study on IBR machining for which the input files and original tool model implementation were provided by Dr. Jack Chen in our research group.

Article 3 will be based on the basic model update idea laid out in Chapter 4, the swept volume regions model of Chapter 5, Sections 5.4-5.10, the tool path categorization and the specific update techniques of Chapter 6, Sections 6.3 and 6.4 respectively and the results and discussion of Chapter 7, Section 7.3. I developed the ideas, wrote the computational programs and performed all the involved case studies.

Article 4 will be based on Chapter 8. I developed the idea of 22 bases lookup table of FS-voxel

shapes (Section 8.4) for FSV-rep surface generation after considering all the possible assumptions and the input features (Sections 8.2 and 8.3) and formulated the proof of applicability of the developed method (Section 8.5). Dr. Jack Chen implemented the idea into a computational program and performed the initial validation. I later developed the case studies and comparative analysis presented in the article and provided in Section 8.7 of the thesis.

Article 5 is primarily based on the research work of Mr. Ziqi Wang to restore sharp features on a surface mesh from the FSV-rep workpiece model. I provided the core program required for the work to generate the input workpiece surface mesh for the case studies and wrote the relevant sections in this article.

Table of Contents

List of Tables

List of Figures

List of Symbols

Symbol	Definition
V	voxel
T	tool
t	curve parameter
O	origin point
p, P	general point
L	voxel edge length
N	grid resolution
f	subdivision factor
u	parameter along voxel edge
M	model memory size in bytes
d	distance
R	tool radius
D	tool diameter
α, β	taper angle
h	height
ϵ	scallop size
w	voxel through gap width

List of Abbreviations

Abbreviation	Definition
IPW	In-process workpiece
CWE	Cutter-Workpiece Engagement
CNC	Computer Numerical Control
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
B-rep	Boundary representation
NURBS	Non-uniform Rational B-spline
CSG	Constructive Solid Geometry
FSV-rep	Frame-sliced Voxel Representation
SVR	Swept Volume Region
MCS	Model Coordinate System

Acknowledgements

I offer my enduring gratitude to the faculty, staff and my fellow students at UBC, who have inspired me to continue my research work. I owe particular thanks to Professor Hsi-Yung Feng, whose penetrating questions taught me to think more deeply and for enlarging my vision of science and providing coherent answers to my endless questions.

I would like to acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) under the CANRIMT Strategic Network grant as well as the Discovery grant and the International Tuition Award from UBC Faculty of Graduate and Postdoctoral Studies. I also wish to gratefully mention the various awards I received: the Altintas Manufacturing Graduate Scholarship, Faculty of Applied Science Graduate Excellence Award and MECH Continuing Graduate Student Award. All of these awards kept inspiring me to further advance in my research path.

I would also like to thank the developers, of the MeshLab software for making the mesh generation and processing program available for open source use and, of the Qt software development platform for making it available for academic use. I also acknowledge the GrabCAD community (http://grabcad.com) for the part models used in Chapter 8 of the thesis.

Special thanks are owed to my parents, siblings, numerous friends and colleagues who supported me, been by my side at difficult situations and cheered me up throughout my years of education.

In search of wisdom

Chapter 1: Introduction

Geometric modeling and simulation is a powerful tool for process verification and optimization widely used in many fields of engineering. In particular, it is applicable for machining process planning which is a major area of mechanical engineering which deals with creation of mechanical parts employing different material removal processes [1]. Machining is a very popular manufacturing process for its ability to create parts of wide range of shapes using standard cutting tools and apparatus. It can also cater to a wide range of metals and other engineering and structural materials. The following sections of this chapter will explain the background and motivation for this thesis work, review the existing techniques used thereby identifying the current requirements, establish the objectives based on the identified requirements and finally identify the research scope and describe the methodology developed in this work. The structure of this thesis is also provided in the end of this chapter.

1.1 Background and motivation

Modern day machining has many typical stages as shown in Figure 1-1 for a typical Computer Numerical Control (CNC) milling task. Milling is a major machining method that uses rotary cutting tools with one or more cutting edges along the axis on the tool periphery usually winding up as helix. The CNC milling machine comprises of as many as five feed drives that can be controlled using a set of instructions provided as NC-program to a digital controller. The feed drives advance the rotating spindle holding the milling cutter along a resultant tool path. The rotating milling cutter remove material from the workpiece on its trajectory by shearing action thereby creating new machined part surface geometry.

In order to provide the appropriate NC program to the CNC machine, the manufacturing

process planning stage has to generate the tool paths accurately. For present day complex part machining operations, it usually involves use of Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) tools to model the workpiece geometry, the reference CAD model for the target part geometry, and to generate the tool paths digitally. This process involves many manual input and parameter settings for the tool path generation. Also the tool paths generated often have cutting and operating parameters decided by the CAM software.

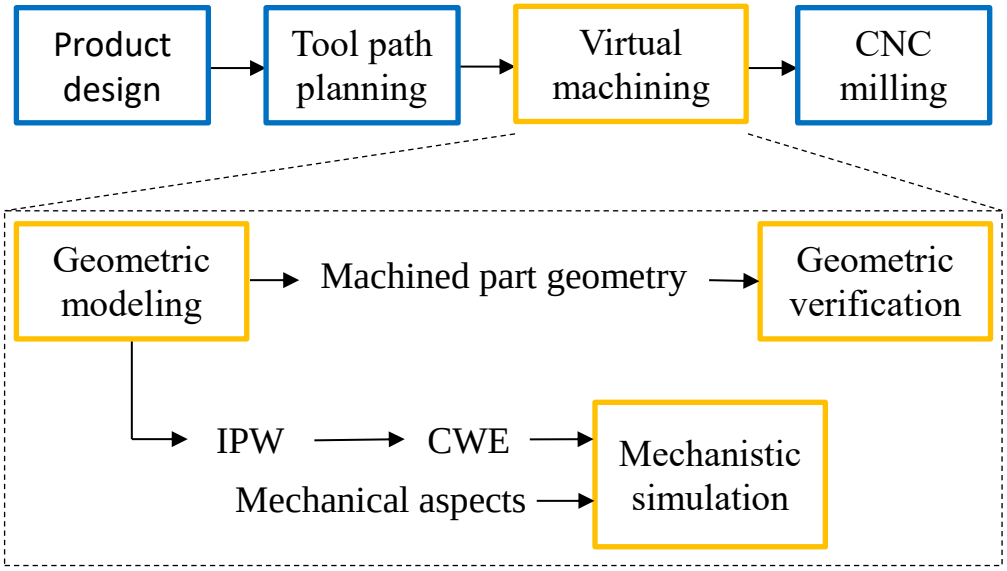


Figure 1-1 Schematic diagram showing various stages of a modern manufacturing process.

Even though automated systems are less error prone in general, the various parameter settings the user has to setup can incur inadvertent errors in the tool paths generated. Further the cutting and operating parameters decided by the CAM software can also be based on generic machine tool definitions. Both of these creates chances for potential error-prone or sub-optimal tool paths. Thus, it is imperative to have a verification system for the tool paths generated as NC programs before actually run it on the machine. This is where the role of virtual machining arises

in modern manufacturing practices.

Virtual machining essentially employs various modeling and simulation techniques to virtually perform the machining process of a workpiece model [2]. The tool path and the cutting conditions can be verified by inspection of the simulation results such as the predicted machined part geometry. Virtual machining can be divided into two major class of approaches – geometric and mechanistic.

Geometric virtual machining performs the machining simulation purely using geometric aspects of the tool path and the workpiece. This in fact provides a powerful way to readily verify the NC program without considering the mechanical aspects of the machine tool and the workpiece material. Major errors in NC programs such as dimensional and tolerance errors, collision and interference, gouging and under-cuts can be detected by geometric virtual machining [3, 4]. Advanced techniques could be needed to avoid issues such as gouging and warrants a good verification system [5, 6]. Geometric methods of interactive manufacturing planning are also reported [7].

Mechanistic virtual machining further incorporates the mechanical aspects such as tool and workpiece material properties and the structural characteristics of the machine to simulate the forces and vibrations occurring on the tool and the workpiece [8, 9, 10]. Simulation of force and vibration dynamics helps in prediction of excessive load on the machine tool, chatter issues and other issues possible such as overheating of the workpiece and cutting tool.

Even for mechanistic virtual machining, apart from the mechanical aspects, the cutter-workpiece engagement (CWE) maps is needed which is a geometric information derived from the workpiece and NC program. It is essentially the information about the instantaneous contact surface between the cutting tool and the in-process workpiece. In case of multi-axis machining,

freeform machining and even for machining of complex prismatic parts, obtaining the in-process workpiece and CWE maps analytically is difficult and often impossible. Geometric virtual machining is indeed needed as a prerequisite for generating these information for mechanistic virtual machining.

With the above-mentioned relevance for geometric virtual machining, many geometric modeling and simulation techniques have been attempted for the purpose over the years. Even though these techniques are good in terms of some of the three primary performance aspects of accuracy, efficiency and robustness, none of them seems to provide a suitable combination of the three together. Thus, there exist need to consider these geometric modeling and simulation techniques to understand their qualities and issues thereby, developing a method that can deliver the best or a better combination of the performance aspects expected from a geometric simulation method.

In the following section, the various geometric methods currently employed for virtual machining are considered and evaluated in terms of their level of accuracy, efficiency and robustness. With the observations from the evaluation, the objective, research scope and methodology for this thesis work are defined in the following sections towards the next level of geometric modeling and simulation technique for virtual machining.

1.2 Existing geometric machining simulation technology

A variety of geometric modeling approaches have been applied to virtual machining implementations aiming to achieve accuracy and efficiency in modeling the machining geometry while being robust enough to handle multi-axis tool motions. These approaches can be broadly classified into three categories:

1. solid modeling,
2. vector modeling, and
3. space partitioning.

Solid modeling can achieve an exact representation of the machined workpiece geometry. Both vector modeling and space partitioning are approximate modeling approaches [11]. They aim to achieve faster computing results at the expense of modeling accuracy.

Evaluation criteria

An effective geometric modeling method used in machining simulation should strive to possess all the three attributes: accuracy, efficiency and robustness. Accuracy measures how close the model represents the actual machined workpiece. Efficiency refers to the order of computational complexity of the algorithms involved and the resultant computational speed obtained in creating the machined workpiece model. Robustness indicates the ability of a modeling method to handle the various cases of machining scenarios that can arise. It would be ideal to have a method providing the best levels of all the three attributes together. Practically however, an effective method should aim to achieve the best possible combination of the three attributes. This is because attempts to achieve these attributes, especially accuracy and efficiency, often conflict with each other. Reasonable levels of accuracy and robustness are clearly desirable and achievable. Improving the involved computational time then becomes the critical issue for practical applications. Thus, a modeling technique toward this goal is in need.

Evaluation of existing technology

Solid modeling methods are known to be best in modeling accuracy whereas vector

modeling and space partitioning are good in efficiency. However, if it is attempted to increase the accuracy of vector modeling and space partitioning by simply increasing the approximation resolution, then the efficiency deteriorates, both memory wise and computationally.

Solid modeling based machining simulation typically uses the boundary representation (B-rep) modeling scheme [12, 13, 14]. B-rep is a way of modeling an object by using surfaces and curves to represent the boundary of the object. It uses parametric representations of the surfaces and curves as well as topological information to specify the adjacency and connectivity relationships between the geometric elements. B-rep, thus, yields an exact and easily traversable model. Another widely used simpler B-rep scheme is the 2-manifold triangle mesh model. It has also been used for representing the machining geometry [15,16]. Under this scheme, the object surface is approximated by a connected set of triangles. The solid modeling schemes are good at providing accurate machined workpiece models.

However, the intersection calculations needed for updating a NURBS based B-rep model are computationally demanding. Identification of the relevant triangles when updating a triangle mesh model also takes time. These issues impede their use in machining simulation which involves intensive and repeated removal of volume from the model. The number of geometric elements in the model also increases with each update, further decelerating the process. Machining simulation with repeated model updates is also affected adversely by the need to maintain the connectivity information among the boundary elements in the B-rep class of models. To facilitate the process and also to identify the intersecting triangles faster, a localization technique has been reported [17]. In practice, still a large number of triangles are often needed for acceptable representations of freeform surfaces present in machining simulation. This increases the number of triangle-triangle intersections and it is still a concern.

A recent research study reported by Li et al. [18] has improved the applicability of B-rep solid modeling in machining simulation. In particular, the cutter-workpiece engagement maps, which are essential to mechanical virtual machining, can be computed without the need to constantly update the machined workpiece model but doing so only once after each tool path. This much enhances the applicability of B-rep workpiece model in machining simulation. However, B-rep solid modeling is still not a competitive method in computational efficiency for performing distance queries and intersection calculations in complicated curved surface machining. This advocates for a workpiece representation method that is more straightforward to update while providing modeling results with sufficient accuracy.

Vector modeling provides a discrete representation of an object with one or more sets of axis-aligned parallel line segments called as vectors. The *Z*-map representation which uses an array of vectors in the direction of the Cartesian Z axis with one end at a fixed base plane and the other end on the top surface of the modeled shape is a typical example [19,20]. Since the vectors can be directly accessed via an indexed array, the machining update operation is not affected by the model size as much as for the solid modeling approach. Hence, the *Z*-map method is quite useful for simulating three-axis milling operations for shapes which do not have overhangs. Except for the tool motions close to the Z vector direction, the *Z*-map method offers good simulation accuracy. However, for general multi-axis milling with changing tool orientations involved, overhangs in the machined workpiece model are common and the *Z*-map method becomes insufficient.

The dexel representation is an extension of the *Z*-map method and uses a list of line segments at each vector location to model the workpiece overhangs [21]. The tri-dexels or triple-ray representation is essentially three orthogonal dexel models used together. It is the most

advanced vector modeling method that resolves the issue of poor sensitivity along the vector direction of any single vector set [22, 23]. Vector modeling with tri-dexels is thus able to perform simulation of all milling operations with efficient update of the machined workpiece model through direct access to the dexels corresponding to the removal volume. However, it lacks the ability to reduce intersection calculations as the workpiece update always happens at the same finest level grid resolution. More importantly, the intersection calculations are done for all the dexels crossing each instantaneous tool location even if they may not be eventually part of the final machined workpiece surface. These issues cause significant overhead in simulating the general multi-axis milling processes.

Space partitioning is the third geometric modeling approach employed in machining simulation. Simple uniform space decomposition methods such as a voxel grid model [24,25] are the easiest to implement but requires a high grid resolution to achieve accuracy approaching that of B-rep. Space partitioning has an evident advantage over vector modeling as the fundamental computation needed to update the machined workpiece model is to classify points as inside or outside of the cutting tool. Vector modeling, on the other hand, needs to repeatedly compute line-surface intersections. Also, with the use of hierarchical space decomposition methods such as a multi-level voxel grid or octree representation [26], the involved computations can be localized much better in the case of space partitioning. These methods also provide the potential for implementations with lower memory consumption. In fact, space partitioning using an octree voxel grid representation and augmented with composite adaptively sampled distance fields to improve the modeling accuracy without further increasing the voxel grid resolution has been recently used in machining simulation [27, 28]. The method would display slower simulation performance due to the numerical computations sometimes needed to create the

distance field in multi-axis milling and the complex procedure involved in extracting the final machined part surface from the composite distance fields. Nonetheless, it shows the potential of an improved space partitioning method for efficient machining simulation with sufficient accuracy.

Table 1-1 summarizes the relative ratings of the modeling methods outlined above with respect to the modeling accuracy, robustness, and computational efficiency.

Table 1-1 Relative ratings of the existing modeling methods for machining simulation.

Modeling Approach	Specific Method	Accuracy	Robustness	Efficiency
Solid modeling	NURBS	✓✓✓	✓✓✓	✓
	Triangle Mesh	✓✓	✓✓✓	✓
Vector modeling	Z-map/Dexels	✓✓	✓	✓✓
	Tri-dexels	✓✓	✓✓✓	✓✓
Space Partitioning	Voxels	✓	✓✓✓	✓✓
	Multi-level/Octree	✓	✓✓✓	✓✓✓

Solid modeling with NURBS based B-rep is the most accurate but lacks computational efficiency when dealing with a large number of tool path segments. Triangle mesh modeling provides limited improvement in efficiency at the expense of accuracy. Vector modeling in general gives a better efficiency. With the use of tri-dexels, vector modeling becomes robust enough to handle multi-axis milling operations. Its efficiency, however, can still be improved as discussed earlier. Space partitioning via voxels is robust to handle all milling operations. In principle, it is also computationally more efficient than solid modeling and vector modeling as it does not involve intersection calculations in updating the workpiece model and can effectively

use multi-level subdivision strategies such as octree. However, the comparatively higher model size to guarantee modeling accuracy reduces the practical efficiency of the voxel space partitioning method. Hence, the challenge is to keep the model size down by achieving a sub-voxel modeling resolution (accuracy). A method that gives the best combination of modeling accuracy, robustness and efficiency can then be attained.

The above evaluation is done focusing on simulation for machined part geometry computation for the possible process verification with it. Apart from computation of the machined part geometry, an efficient modeling method is also beneficial for other applications of machining simulation and verification such as machine/tool collision detection and avoidance, mechanistic simulation and online process control. Collision detection and avoidance is a crucial part of machining simulation especially for multi-axis operations. The tool and tool-holder motions need to be verified in order to avoid collisions. Interference checks using geometric models of the machine structure and simulation of the entire machine tool kinematic motion have been utilized for this purpose [29, 30, 31]. Mechanistic simulation of the machining process to predict cutting forces and the onset of chatter have been used to verify and optimize the process plan [32, 33, 34]. As mentioned earlier, geometric modeling is used to obtain the essential inputs of in-process workpiece and CWE maps required for the mechanistic simulation. The online process monitoring and control is another machining technology that calls for effective geometric modeling. Research studies towards assistance with augmented reality to enhance observation of an ongoing machining process have been reported [35, 36]. A CAD/CAM system fully integrated into a machining center is also an emerging reality. All of the technological areas stated above will benefit from an efficient method to simulate the machining operation, apart from process verification which is the primary focus of this work.

1.3 Thesis objectives

Based on the study of existing geometric modeling and simulation methods, it is apparent that virtual machining will greatly benefit from a new geometric modeling method for workpiece modeling and update. Hence the objectives for this thesis work are set as follows:

[O1] Achieve an accurate as well as memory efficient workpiece geometry representation.

[O2] Enable efficient and accurate update of the workpiece geometry for simulation of machined part geometry in general milling simulation.

The above objectives thus focus on the way of representing the workpiece geometry, updating it during simulation using the tool paths and generating a standard format representation for the simulated part geometry from the simulation specific model. It also demands that the new methods should be broadly applicable to all categories for milling operations involving up to 5 degrees of freedom for the tool axis.

It should also be noted that the objective of this thesis work is defined such that the developed geometric modeling methods itself can provide valuable end results. In other words, the use of the newly developed geometric methods for pure geometric virtual machining is given focus. This is justified from the fact that the simulated machined part geometry obtained from geometric virtual machining itself provide valuable support for process verification as detailed in previous section.

1.4 Research scope

The set objectives open a number of items with research scope as follows:

- ❖ New geometry modeling format – Geometry modeling is a widely applicable mathematical tool with utilization in many areas including machining simulation. The current

geometry modeling formats are unable to provide the expected level of accuracy and efficiency together at least in machining simulation. Research to achieve the best levels of these performance aspects have led to maximum attainable performance for existing approaches by now. Thus, a new generation of geometric representation and modeling method is in need and indeed demands a significant research effort as the potential new method is not trivial. Based on the relative rating of existing methods derived in previous Section 1.2, this research work will aim at developing the new geometric modeling method based on the voxel space partitioning. This is because it has the inherent quality to support efficient update for material removal. Thus, the research for this thesis work will attempt to develop a voxel based model which is accurate as well for the purpose of machining simulation while maintaining the efficiency.

- New algorithms for model update and machined part geometry simulation – In order to update a geometry model during the simulation of machining operations, a model update logic has to be identified. Every geometry modeling approach has a particular way of model update for material removal that is most suited for it as we will see in Chapter 2 Section 2.3. Thus, to efficiently use the new modeling method we shall develop, a suitable update logic should be identified as well. There is always a brute force way to update a model but to be efficient, a superior algorithm with least order of complexity is expected. This in fact require in-depth understanding of the model representation format and identification of the salient features that can lead to an efficient update logic.

- New process for surface geometry extraction – As seen from the evaluation of existing modeling approaches in use for machining simulation, it is apparent that a new and better

modeling method should be based on discrete space partitioning as it provides the best efficiency in model update with possible leverage from use of multi-level representations. The space partitioning methods however does not have an inherent surface information available to use for visualization or analysis using popular and widely standard technology available to handle surface representations such as triangle mesh models. It will be shown that currently used techniques to generate a surface representation from discrete grid based models are either not robust enough or are not the most efficient approaches possible. Thus a technique to efficiently and consistently generate valid surface representation from the new simulation model is the last but not the least research aspect considered as part of this thesis work.

1.5 Methodology

To develop and demonstrate the new geometry modeling method and its application to machining simulation, a research workflow with four major modules as depicted in Figure 1-2 below is followed.

The four modules are essentially dealing with the four major parts of a simulation system for geometric virtual machining. The geometric model of the workpiece has to be defined in the preferred modeling format using the shape of the initial blank workpiece. The tool and tool path information should be used to model the tool swept volumes that act as the Boolean material removal tools in updating the workpiece model. To update the workpiece model, a simulation procedure logic that can accept the workpiece model and tool swept volume representations as input is needed. Finally, the updated workpiece model that represents the machined part should be used to generate a standard representation of the machined part surface geometry.

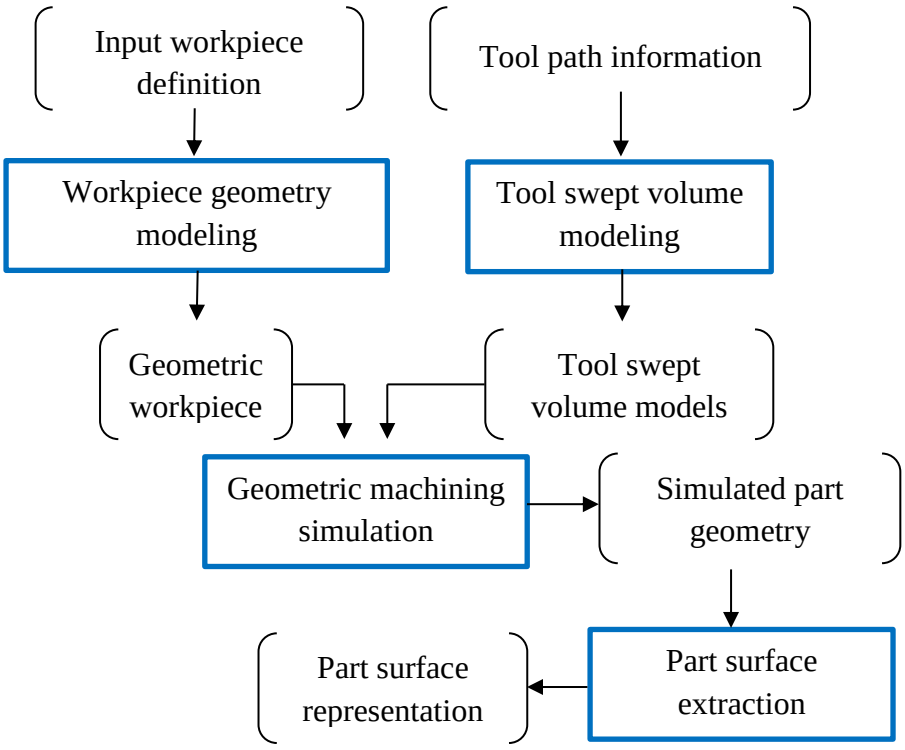


Figure 1-2 Schematic of the research and development workflow.

1.5.1 Workpiece geometry modeling

Based on the blank workpiece, the geometric model that can be used as in-process workpiece has to be created to start with. Thus, a module is identified to develop the concepts and implementation of a suitable representation format for the workpiece geometry. This module will satisfy the Objective [O1] defined in Section 1.3. Specifically, a workpiece definition in a standard format will be accepted as input. The module will then generate the suitable simulation model for the in-process workpiece. Thus there are two components for this module: (1) Concept and data-structure development for the newly identified simulation workpiece model and, (2) Pre-processing step to convert a standard definition input blank workpiece into the identified simulation workpiece model.

1.5.2 Tool swept volume modeling

Apart from the simulation workpiece model, the geometric representations of the tool swept volumes are also required to perform the geometric machining simulation. Thus a module is defined that will consider different tool swept volume representation approaches and will make an appropriate selection for the type of workpiece representation format developed in the "Workpiece geometry modeling" module. Another component of this module will be in fact to implement the required pre-processing steps to generate the selected swept volume representations from standard tool and tool path definitions after designing the suitable data structure for the selected swept volume models.

1.5.3 Geometric machining simulation

This module will identify and develop the simulation steps for using the created swept volume models to update the workpiece geometry model. Thus, this module involves analysis of the favorable characteristics of the workpiece geometry model and the swept volume models to identify the best approach for model update. Objective [O2] defined in Section 1.3 will be achieved by the end of this module. First a generic update approach for the workpiece geometry model shall be devised. Then this generic approach can be specialized to various swept volume representation techniques suitable for different machining categories. This way of forming the generic and specialized method will help in identifying a universal method for all the future use in machining simulation and then to enable the full utilization of the specific swept volume model characteristics.

1.5.4 Machined part surface generation

A final module is dedicated to study various ways of geometry representations suitable for visualization and inspection. From the discussion in Section 1.2, it is apparent the representation suitable for simulation and the one suitable for visualization and inspection are not necessarily the same. Some type of surface representation is widely used for visualization. Thus, this module will aim at considering existing techniques for surface visualization of the simulated part geometry focusing on the applicability and efficiency of each technique. Then, developing an efficient way to generate such a representation from the simulation model we develop is also part of this module. Efficiency of the surface generation technique will be given high importance after the simulation efficiency as visualization of the simulated part geometry is expected to be readily possible after the simulation steps.

1.6 Thesis structure

The rest of the thesis is constructed as outlined below with description of the contents in each chapter and their contribution to satisfying the afore set objectives, scopes and modular methodology design.

❖ Chapter 2, “Relevant methods” will review the various geometry modeling and simulation approaches that have been successfully applied for machining simulation. This will be a more in depth study than the brief review done in Section 1.2. After Section 1.2, we have identified accuracy, efficiency and robustness as the three basic performance aspects to consider. In Chapter 2, different workpiece representation methods, tool swept volume representation methods and the workpiece model update methods are studied in detail to identify the strength and weakness of each and the qualities that provide each of them its specific strengths. Review

of the current methods used for surface generation for different simulation models will also be done in this chapter. Chapter 2 will form the background knowledge for the rest of the thesis.

The complete methodology developed and described in this thesis is shown in Figure 1-3 and the various chapters presenting the different components is described below.

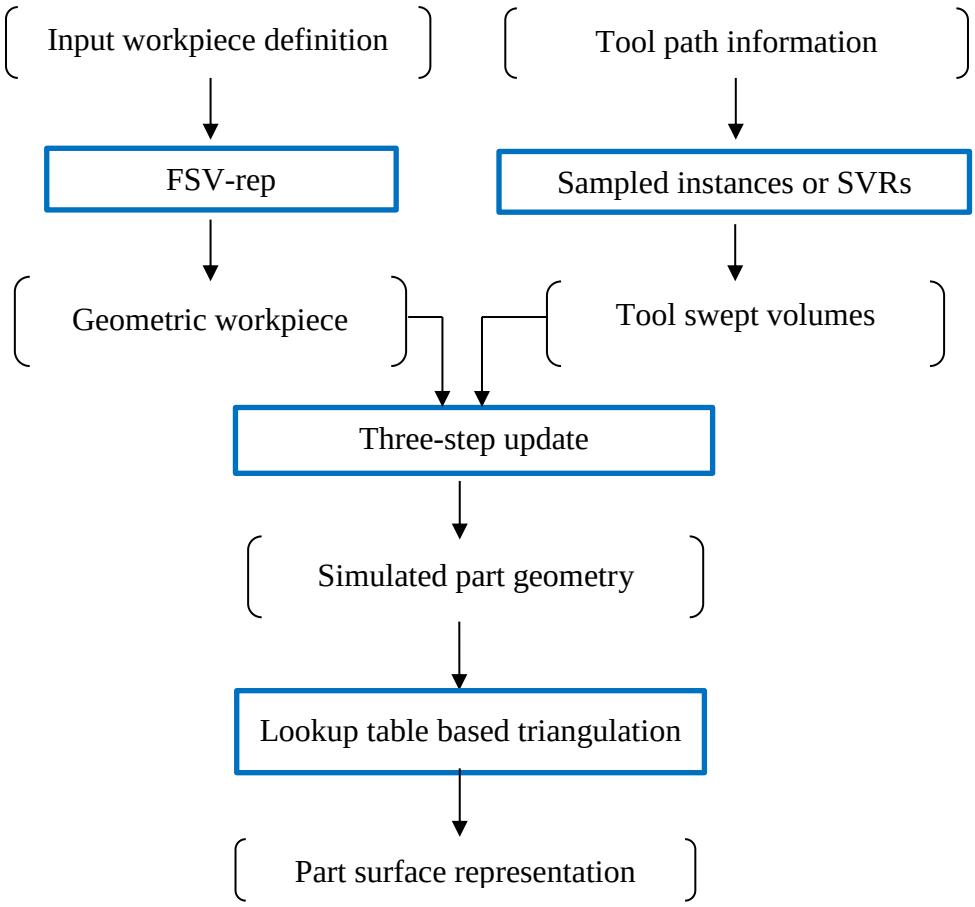


Figure 1-3 Schematic of the developed methodology.

❖ Chapter 3, "Frame-sliced voxel representation" introduces FSV-rep, the new workpiece geometry representation model developed in this thesis work. The definition and salient features of the model will be laid out first in Sections 3.1 to 3.7. The algorithms to generate FSV-rep model from standard STL geometry representation and the data structure to hold the model for simulation is provided after in Section 3.8. Section 3.9 and 3.10 will evaluate the modeling

method in terms of accuracy and memory efficiency using basic and practical machined part geometries. Chapter 3 is essentially on the module “workpiece geometry modeling” to achieve Objective [O1].

❖ Chapter 4, “Three step FSV-rep model update” constructs a generic logic for updating an FSV-rep model using a set of volume removal tools in order to obtain the final machined part FSV-rep. This logic will be developed focusing on efficiency and for utilizing all the qualities of FSV-rep models. This chapter will thus, lay down the initial part of the module “Geometric machining simulation” as an important first step towards achieving Objective [O2].

❖ Chapter 5, “Tool swept volume representation” will establish the suitable tool swept volume models for various categories of milling operation from 1-axis to 5-axis. The optimum swept volume representation for each case is then developed in the chapter for generic as well as more common special milling cutter types. Specifically, a sampled tool instances approach is used for general multi-axis tool paths whereas a customized “Swept Volume Regions (SVRs)” is used for linear three-axis tool paths. Chapter 5 will thus complete the development of concepts and definitions for the module “Tool swept volume modeling”.

❖ Chapter 6, “FSV-rep machining with tool swept volumes” develops the geometric simulation methodology with the new FSV-rep workpiece model and the suitable swept volume models developed in Chapter 5. The generic update logic constructed in Chapter 4 will be appropriately adapted for the specific swept volume representations selected. With this, the module “Geometric machining simulation” will be complete and the Objective [O2] will be achieved as well.

❖ Chapter 7 provides the details of the simulation system implemented with all the modules and discuss the results obtained for various simulation case studies for machined part geometry

computation. The discussions in Chapter 7 will identify the various factors contributing to the resultant accurate and faster computational performance of FSV-rep based simulation.

❖ Chapter 8, “FSV-rep surface generation” provides the improved surface generation technique based on efficient marching cube like look-up table that is most suited for surface generation for an FSV-rep model. The chapter also contains the proof of applicability of the new look-up table and various cases studies to show its applicability and better performance compared to other existing methods.

Chapter 2: Relevant methods

This chapter will further discuss the previous techniques used for the various components of the workflow followed by the thesis work. First it explains the major approaches used for workpiece geometry representation. Then it describes the typical ways for modeling the tool swept volume. The appropriate ways of update the workpiece geometry model followed for various combination of workpiece and tool swept volume models are analyzed next. Finally it discusses the suitable methods for the in-process workpiece and machined part surface generation and visualization.

2.1 Workpiece representation

Workpiece geometry representation can be broadly classified into solid modeling, vector modeling and space partitioning class of approaches. The following sub-sections discuss the different modeling methods falling to these classes.

2.1.1 Solid modeling

Solid modeling is a traditional way of representing shape of 3D objects. Constructive solid geometry (CSG) uses primitive shapes such as cubes, cylinders, spheres combined with different Boolean operators to represent the shape. Since the object is only implicitly represented by a tree of primitives, the model definition is easy. However, interaction with the model is often compute intensive due to the required tree traversal. Solid modeling using boundary representation (B-rep) is another way of workpiece representation widely used since initially.

In B-rep solid modeling, the surface of the object is modeled (Figure 2-1). Each surface element is oriented with a definite normal direction at every surface location. The surface is

oriented such that the normal is pointing away from the object volume locally. Further the surfaces are connected together along their intersection curves. The intersection curves terminate at the various intersection points. The trimmed surfaces, curves and the intersection points act as the face, edge and vertex boundary elements respectively forming the topology for the B-rep.

The orientation of the surfaces and the topology defining the limits of those surfaces as well as the connectivity between each other is important for B-rep solid modeling. These aspects enable B-rep solid models to maintain a valid geometry. A solid is considered valid if it is closed and 2-manifold. 2-manifoldness essentially requires the surface to be similar to a disc within the infinitesimal neighborhood of any point on it. Thus, when using a B-rep solid model, it is essential to maintain the orientation and connectivity of the boundary elements to ensure 2-manifoldness of the model.

A B-rep solid model can have any type of surfaces and curves defining its faces and edges. Thus, it has the capability to accurately and often exactly represent the surface geometry of objects. Often for mechanical parts, most of the faces can be modeled with simple planar, cylindrical, spherical or conical surfaces. Edges are also often limited to line segments, circles or conic sections. For objects with organic shapes, freeform curves and surfaces such as splines, NURBS, and two-parameter interpolation surfaces based on such curves are used.

The wide variety of surfaces and curves that can be used, and the large number of such elements that can be present in the topology of a B-rep model causes a difficulty as well to represent the machined part geometry. Often with large number of tool paths, the number of faces and edges created for the B-rep solid model can be excessively large. Large number of boundary elements slows down interaction with a B-rep model. Further, with arbitrary shape of swept volume and at times with self-intersections in case of multi-axis machining, it becomes

challenging and practically difficult to define the appropriate surfaces for the machined part surface geometry.

Triangle mesh is another B-rep solid modeling method which can provide a good approximation of the machined part geometry without using higher order surfaces and curves. It uses a set of triangles to replace the surfaces. The entire surface geometry is thus defined by the set of triangles approximating the different surfaces. As it always uses set of planar triangles to approximate the surfaces, triangle mesh does not face the issues of inability or increased surface complexity in representing the objects even in case of parts created by multi-axis machining. With its reduced complexity and yet being capable of providing acceptable accuracy, triangle mesh has become a very favorable format for visualization, rapid prototyping and machining simulation as well.

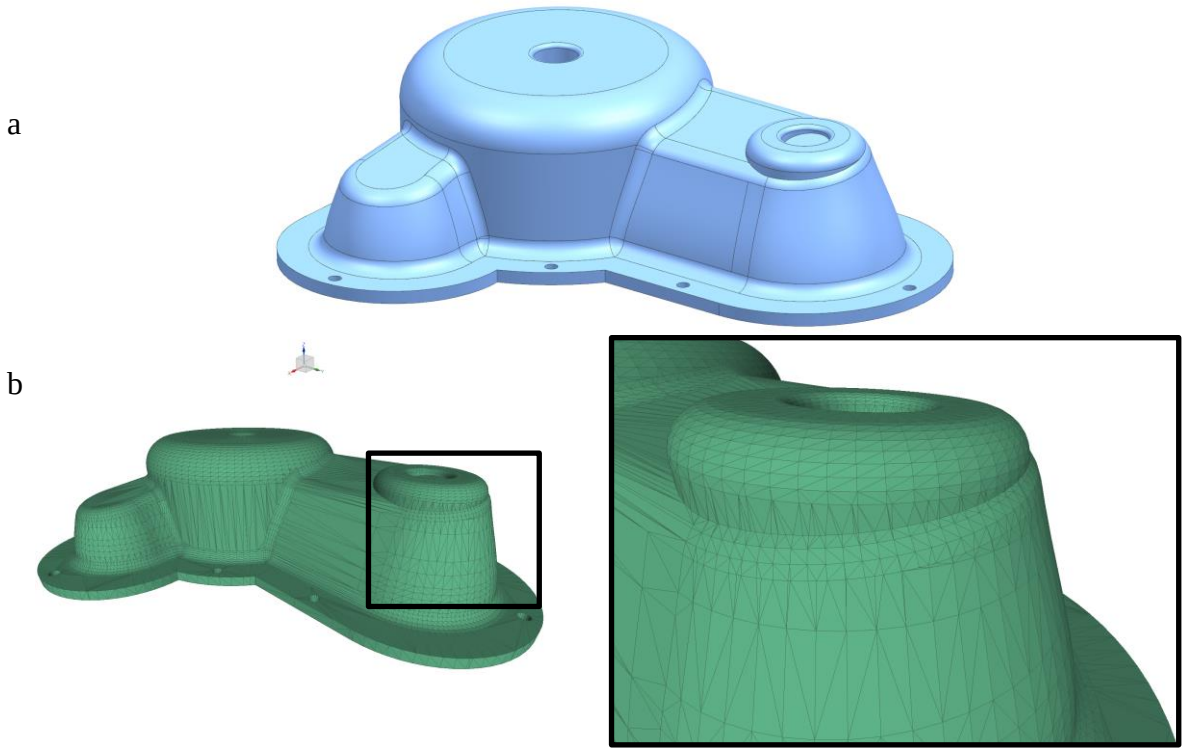


Figure 2-1 Solid models of a typical mechanical part.(a) B-rep model (b) Triangle mesh model.

From the study of solid modeling approaches, we can conclude that the best quality they possess is accuracy while their ability to provide affordable model interaction speed degrades with large number of features.

2.1.2 Vector modeling

The basic elements of vector modeling are line segments. The original vector modeling approach commonly known as *Z*-map used 2-dimensional array of line segments with each of them starting at the base plane and extending vertically up to the top surface of the modeled object. *Z*-map vector modeling is a discrete and uniform grid representation in the sense that it uses a 2-dimensional grid of line-segment elements to represent the modeling space. The objects are represented by activating the portion of the line-segments that is completely inside the object volume and deactivating the rest (Figure 2-2). This enables *Z*-map to enable interaction with the model at some constant access time irrespective of the model size. In order to avail that, the Z-map and other vector modeling approaches remove the explicit boundary representation and the associated topological connectivity information.

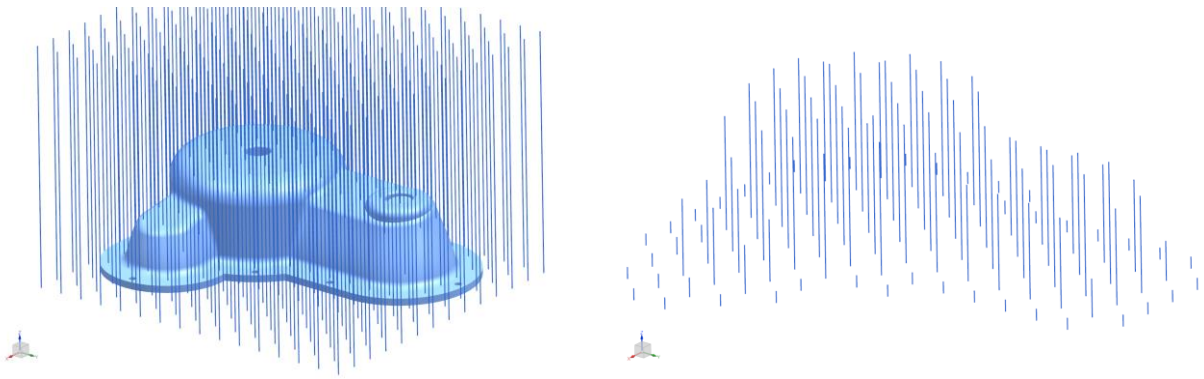


Figure 2-2 A B-rep model in a vector field (left) and the corresponding Z-map vector model (right).

The absence of explicit boundary representation does pose an issue in visualization and other model inspection activities. However, with the Z vector end points exactly sampling the object surface, *Z*-map can provide a set of exact sample points of the modeling object. These points can be suitably used for later reconstruction of the object surface.

Apart from *Z*-maps, other improved types of vector modeling are also reported. Dexels and Tri-dexels are the two main improved vector modeling methods. Dexels use a list of line segments at each grid location on the base plane. This way dexel models can handle objects with overhangs in the vertical direction (Figure 2-3c). Tri-dexels is still another improvement of vector modeling and has the ability to provide higher accuracy in all the three orthogonal directions whereas *Z*-maps and Dexels had limited accuracy for representing surfaces with normal perpendicular to the Z-vector or dexel direction. Since Tri-dexels is essentially three orthogonal dexel models, it can capture surfaces with normal along any direction with uniform accuracy (Figure 2-3d,e,f).

There are other vector modeling concepts as well such as the view dependent depth-buffer approaches and also the vectors along the surface normal direction of the reference part. These are however not having a pre-defined or uniform vector direction and hence lacks the ability to store the vectors in a uniform grid wise order. Still they are able to represent the model without use of boundary surface elements and thus eliminate the need to maintain the topological information.

Overall, the vector modeling approaches, especially the Tri-dexels, presents a valuable improvement over triangle mesh models. They are able to represent the model using basic line segments that are independent of each other but together provides a good set of sample points of the modeled object surface through the end-points of all the line segments. With this

advancement, they are able to improve the efficiency in model interaction and modification while providing the same level of accuracy as triangle mesh models. On the downside, without explicit representation of the object surface, they lack ability to readily provide the object surface for visualization and inspection.

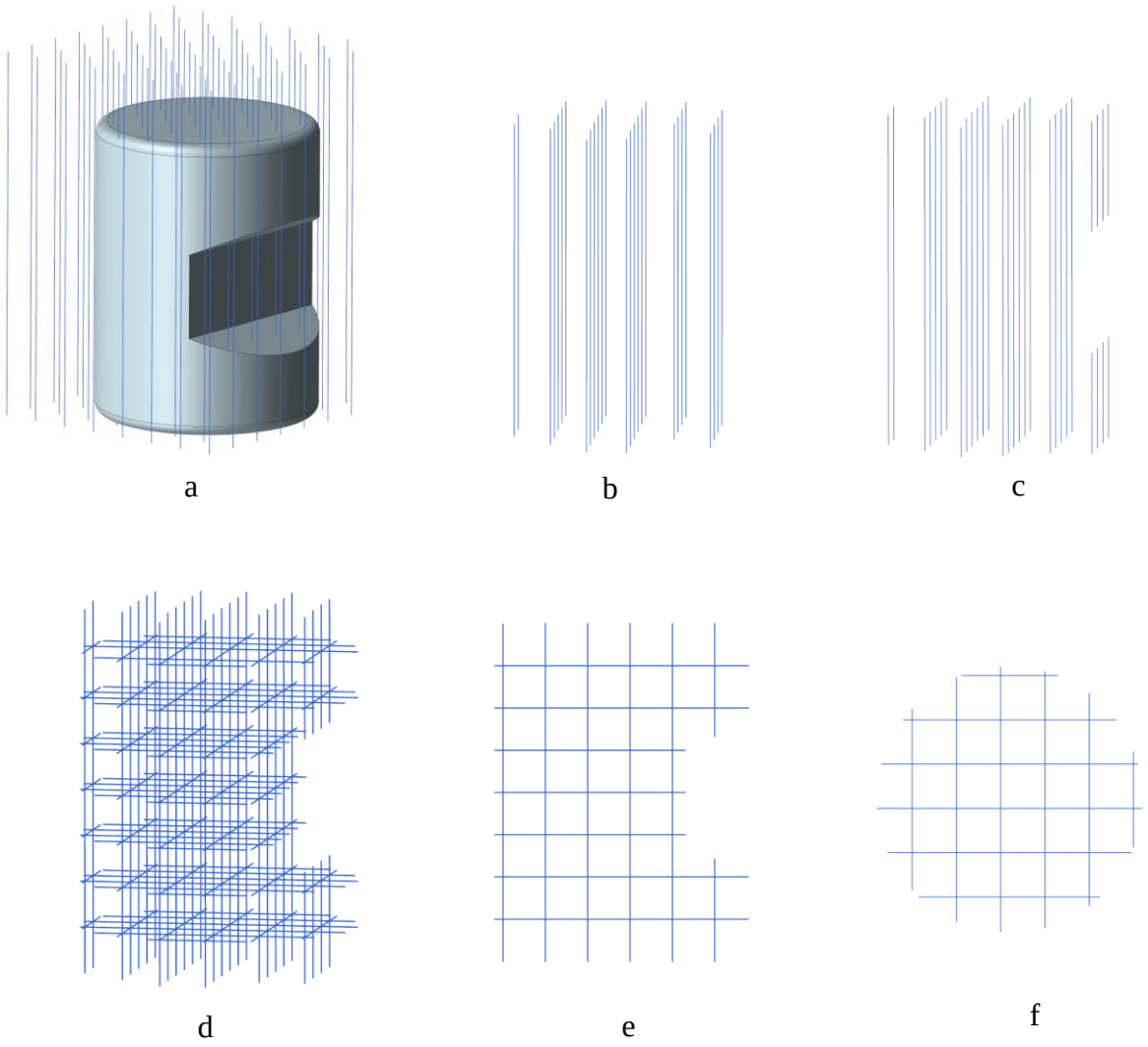


Figure 2-3 Various vector modeling types. (b) Z-map (c) Dexels (d) Tri-dexels with increased accuracy in (e) side views and (f) top view

2.1.3 Space partitioning

Space partitioning is the third class of modeling methods used for machining simulation. These methods consider the entire modeling space and divides it into smaller volumetric elements. A model is represented using many of these constituent volume elements of the modeling space. Space subdivision with uniform grid of cubical volumetric elements called voxels is the basic space partitioning method (Figure 2-4).

Voxel modeling has been mainly used in the field such as medical imaging and terrain rendering where the spatial data is available from 3D scans of a surface or a volume. Voxel models have also been used for representing computer generated models, for instance, to mix together scanned data with synthetic models [37]. Various commercial CAD systems also mention the use of voxel representation for real-time interactive modeling and collision detection, which has been investigated for tool interference detection in multi-axis machining as well [29, 30, 38]. Workpiece modeling in milling simulation has also made use of voxel representation [24, 25, 39,40].

Voxel modeling is a spatial occupancy enumeration type of solid representation scheme [41]. A voxel based solid model is composed of numerous small cubical elements called voxels. Each voxel in the 3D space is uniquely identifiable by a 3D index. Voxel modeling can be understood as activating specific voxels in the voxel space (the spatial grid of voxels) if they belong to the interior of the solid. Active voxels for a solid can include internal voxels (completely inside the solid) and surface voxels (only partially inside the solid) [42]. Length of the voxel edges decides the resolution of the voxel space and hence, the number of voxels in a model. The process of identifying the voxels that represent an object is commonly known as voxelization or 3D scan conversion. Voxelization methods to obtain a voxel model from a

continuous curve, surface and volume have been reported [37,43,44]. Voxelization of a polygon mesh has also been developed and refined to obtain a high-quality surface voxel model [45, 46].

The primary quality measures a surface voxel model shall possess are fidelity and connectivity [46]. Fidelity is the measure of how well a voxel model matches the represented continuous object. Connectivity is a topological quality measure based on the level of contact between neighboring voxels and it decides the minimality and separability [47, 45] of the voxel model. A closed surface voxel model defined by a set of voxels $\{\boldsymbol{V_s}\}$ is 26-separating if it separates the voxel space into three sets $\{\boldsymbol{V_i}\}$, $\{\boldsymbol{V_s}\}$ and $\{\boldsymbol{V_o}\}$ such that there is no possible connected voxel chain $\{\boldsymbol{V_p}\}$ between $\{\boldsymbol{V_i}\}$ and $\{\boldsymbol{V_o}\}$ without at least one common voxel with $\{\boldsymbol{V_s}\}$. The occurrence of such a possible path as $\{\boldsymbol{V_p}\}$ is called tunneling and a 26-separating voxel model is required to avoid the occurrence of tunneling. Tunneling is undesirable as it can cause inaccurate results during intersection of voxel models.

As discussed before, voxel modeling with its enumerative nature through spatial indexing, is very efficient in updating the volume removal operations arising in machining simulation. It is also robust to handle model updates due to complex tool motions as it does not have to maintain topological information explicitly. However, memory demand of basic voxel modeling is in the order of $O(N^3)$ where N is the number of voxels used to subdivide each axial direction of the Cartesian volumetric modeling space (the grid resolution). This large memory demand has been addressed through sparse representation schemes using the octree [48] (Figure 2-4) and directed acyclic graphs [49], and out-of-core algorithms [50]. Some supporting data can also be used to augment a voxel model in order to achieve sub-voxel resolution. For example, bounding planes have been used to better approximate the object surface within the voxel with significant improvement in the voxel-based visualization [48].

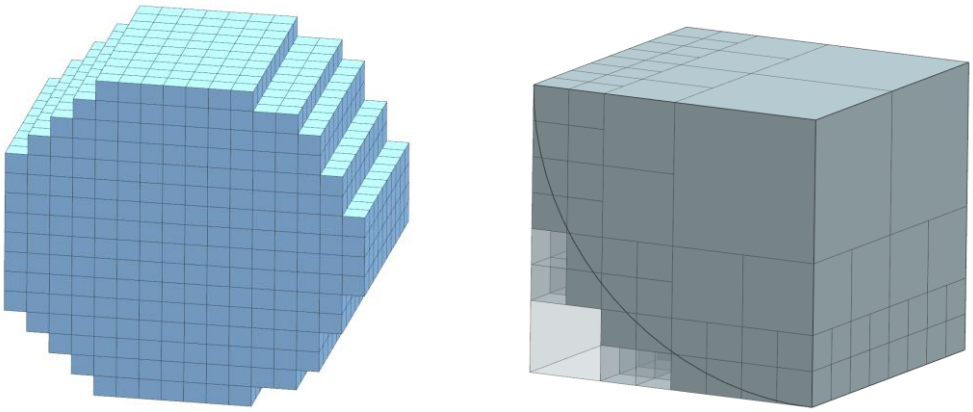


Figure 2-4 Space partitioning models: Uniform grid voxel model (left) and Octree hierarchical model (right).

As understood from the review of space partitioning methods, it can provide model representation with most efficient way to interact with. Also, the concept of multi-level representation enable space partitioning to be potentially memory efficient also. However, the approaches for improving the accuracy without much increase in model size have been unfavorable for machining simulation. A simpler technique is desired for voxel models which are being repeatedly modified. This calls for a novel sparse voxel representation coupled with a simple technique to achieve the sub-voxel resolution. Also, it should easily generate a boundary representation such as a triangle mesh from the model for visualization and data transfer as well as for analysis.

2.2 Tool swept volume representation

After a suitable workpiece representation, the next important input needed for machining simulation is the tool swept volume representation. Tool swept volume representation means the geometric model of the volume swept by the tool while it moves along a tool path trajectory.

2.2.1 Analytical definition

Analytically the tool swept volume can be defined as the union of all the tool instances along the tool path. Tool path can be defined using parametric curves [51, 52, 53] that gives the tool instance location vector as $\boldsymbol{L_t}$ and orientation as a unit vector $\boldsymbol{A_t}$ for a time like parameter *t*. Points belonging to any tool instance T_t along the tool path can be then obtained in the Global Coordinate System *G* as a homogeneously transformed set with the transformation matrix $[C_t]_G^T$ applied on the points in Tool Coordinate System *T* as follows:

$$P_t^G = [C_t]_G^T \times P^T \tag{2.1}$$

With, $\boldsymbol{F_t}$ as a unit vector perpendicular to the axis direction in the plane containing the axis and the feed direction of the tip of the tool instance T_t, and $\boldsymbol{X_t} = \boldsymbol{F_t} \times \boldsymbol{A_t}$ the transformation matrix $[C_t]_G^T$ can be defined as follows:

$$[C_t]_G^T = ([C_t]_T^G)^{-1};\ ([C_t]_T^G) = \begin{bmatrix} X_{tx} & X_{ty} & X_{tz} & -L_{tx} \\ F_{tx} & F_{ty} & F_{tz} & -L_{ty} \\ A_{tx} & A_{ty} & A_{tz} & -L_{tz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Then, the swept volume is defined as the Boolean union of all the tool instances for all the values of t from 0 to 1 along the tool path (Equation 2.3).

$$\bigcup_{t\epsilon[0,1]} T_t \tag{2.3}$$

where T_t is the volume of the tool instance for the parameter *t*.

The simple analytical definition is enough for simple 1-axis toolpaths when a vector or voxel based workpiece is used. For update with such tool paths, only the envelope surface geometry and the generating curve are required. Generating curve is the locus of the grazing

points for each tool instance along the tool path. In fact for 1-axis and up to 3-axis straight cutting tool paths, the generating curve is constant along the path which can be defined as a set of curves. Using the constant generating curve, both the envelope surface and the intersection points with vectors and voxel edges can be obtained. This enables update of the workpiece model directly from the analytic definition.

However, for general tool paths occurring in multi-axis machining, the generating curve and envelope surface are not trivial from the analytic definition. Thus, more advanced models for the swept volume are needed. Advanced analytic definition of swept volumes are indeed available for mathematical representation [54, 55, 56]. However, direct use of such definitions in simulation are computationally expensive.

2.2.2 Boundary representations

Tool swept volume as a boundary representation as shown in Figure 2-5 is required when a boundary representation workpiece is used for multi axis machining. This is because the B-rep workpiece update requires a B-rep model itself as the Boolean tool as we will see in Section 2.3. B-rep swept volume uses different surface elements to represent the various parts of the swept volume boundary. Major portions defined for a swept volume B-rep are, envelope surfaces, back instance (ingress) and front instance (egress) regions. B-rep swept volumes however have to avoid self-intersecting surfaces that arises in multi-axis machining. Special treatment of tool paths to split them into segments without self-intersection is one approach followed.

Triangle mesh approximation of the swept volume boundary is often used to eliminate the above-mentioned issues with actual B-rep based swept volume representations. The triangle mesh swept volume B-rep is generated in a number of ways. Starting from the analytic definition

of the swept volume, obtaining the grazing points of the tool instances along the path and creating a triangle mesh using the set of points is the general approach followed. Ball pivoting triangulation of point clouds, alpha shape creation, etc. has been used in the past for such processes.

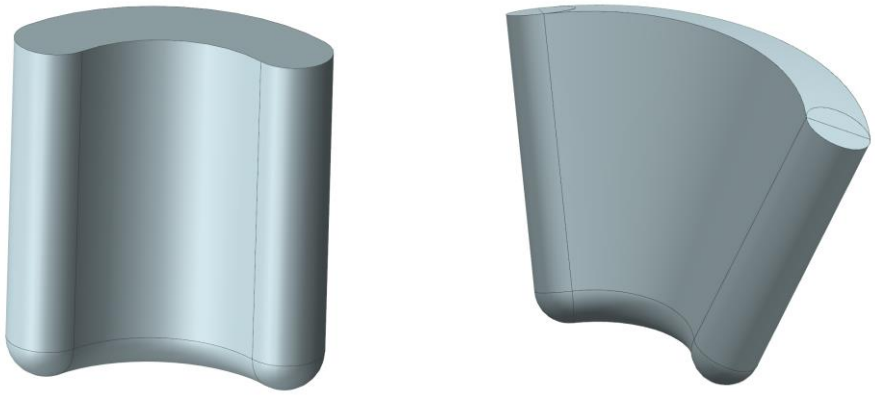


Figure 2-5 B-rep representation of tool swept volumes defined using different surfaces.

B-rep swept volumes, and especially using triangle mesh, has the benefits for being capable of updating all type of workpiece representations. 2-manifold triangle mesh swept volumes could be generated even in case of self-intersecting tool swept volumes [16].

However, the creation of such boundary representation swept volumes is an additional operation for machining simulation which can add significant amount of time as a pre-processing step of tool swept volume generation.

2.2.3 Parametric representations

Apart from the analytic definition of the swept volume in Section 2.2.1, parametric definition of swept volume has been developed in alternative forms as well. Representation of swept volume as a 2-parameter family of spheres is one such development (Figure 2-6). This has been found useful for the update of workpiece representation using *Z*-map vector modeling.

The effective points of the tool instances along the tool path that ultimately contribute to the tool swept volume are the grazing points for which the feed velocity is perpendicular to the tool envelope surface normal. Parametric representation of these grazing points has been developed [55,56] for swept volumes of simple as well as general tool paths.

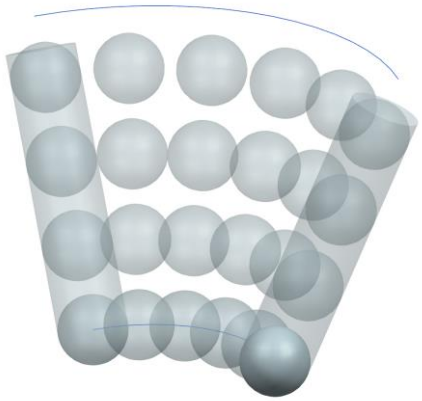


Figure 2-6 Parametric representation of tool swept volume with 2-parameter family of spheres.

2.2.4 Sampled approximations

Swept volume representation as a collection of tool instances at different sampling locations along the tool path is a practical approach useful by itself as an approximation and also as an underlying building block of many other definitions. Swept volume representation with parallel slicing (Figure 2-7) for instance have used sampled set of tool instances to generate the initial swept volume which was later sliced to obtain cross-section curves. Other approaches such as parametric definitions also ultimately uses a sampled approximation of the ideal representation for practical use.

Further, as a collection of plunging tools present along the tool path, the sampled tool instances can itself act as a representation of the tool swept volume. Instead of updating the model with a single swept volume object, all the tool instances are used one after the other with this approach. Computations to update a workpiece model with each tool instance has explicit

closed form solutions as the tool envelopes of general milling cutters has primitive surface elements [57,58]. Thus, with sampled tool instances, only consideration to make is the suitable sampling interval that can reduce the "sampling scallop" error from the ignored swept volume portion between two sampled instances.

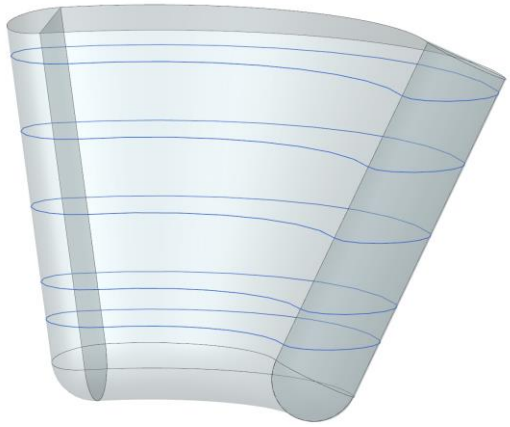


Figure 2-7 Parallel slices approximation of tool swept volume.

2.3 Workpiece update methods

Based on the methods used for workpiece geometry representation and the tool swept volume representation, the model update technique has to be different. These various update techniques is discussed in this section.

2.3.1 Boolean operations for solid models

Solid modeling approaches using CSG is the easiest to update as the model is stored as a combinatorial tree of primitives. The material removal by a tool swept volume can be easily added as another entry into the tree. However, the resultant updated workpiece geometry is not explicitly stored, requiring expensive computations for model visualization and interaction. In fact, the tree update is computationally expensive for evaluation of the surface topology though

easy to be stored.

B-rep solid model update requires trimming of the existing boundary elements and stitching of new boundary elements for the newly machined part surface. In order to add new boundary elements to the B-rep workpiece model, the tool swept volume also has to be defined as a B-rep. Thus, for general NURBS B-rep workpiece update, NURBS B-rep definition of the tool swept volume is needed and similarly for update of polyhedral workpiece representation, a polyhedral or NURBS B-rep representation of the tool swept volume is needed.

For all the above solid model workpiece and tool swept volumes, the essential process involved in the workpiece update is Boolean operation. As mentioned above, Boolean operation effectively, trims away the portions of workpiece boundary elements (faces, edges and vertices) that are completely inside the tool swept volume (Figure 2-8). It also adds new boundary elements as replacement from the portions of the swept volume boundary elements that is inside the original workpiece volume.

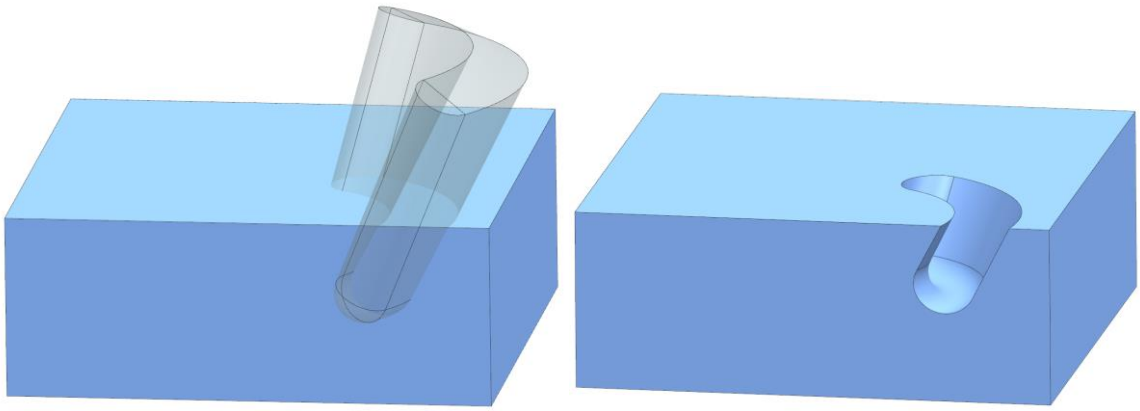


Figure 2-8 Boolean update of a B-rep workpiece model using a B-rep tool swept volume.

As it involves trimming operations that require intersection between surfaces, the Boolean update of solid models is computationally involved. Even with simpler triangle mesh models, the

number of triangles grow rapidly with more machined features and surface curvature. With high number of triangles, the time for intersection calculations also increase rapidly.

Essentially the update process for solid models are compute intensive suggesting to use an alternative workpiece or tool swept volume representation.

2.3.2 Trimming operations for vector models

Vector models of *Z*-map, dexels, and Tri-dexels, are all composed of line segments. Hence update of vector models ultimately involves the trimming of these line segments. In order to trim the line segments for update using a tool swept volume, the portion of these line segments falling inside the tool swept volume has to be identified. There are two approaches followed for this.

In one method, the vector model for the tool swept volume is generated first. Then the line segments forming the vector tool swept volume model is used to subtract overlapping portions of the line segments forming the vector workpiece model. Thus, once the vector model of the tool swept volume is available, the update of the vector workpiece model involves simple line segment clipping by end point modifications.

In another approach, the line segments forming the vector workpiece model is directly updated with the standard tool swept volume representations such as those covered in previous Section 2.2. This involves identification of the workpiece vectors crossing the tool swept volume efficiently to avoid unnecessary attempts on all workpiece vectors. Often the bounding box of the tool swept volume is used for this localization.

In either approach, intersection of line segments with tool swept volume envelope surface is involved (Figure 2-9b). Often the envelope surface do not have explicit representation for intersection with line segments. To ease the process, the various alternate swept volumes such as

polyhedral B-rep, parametric family of spheres, parallel slices etc. are used.

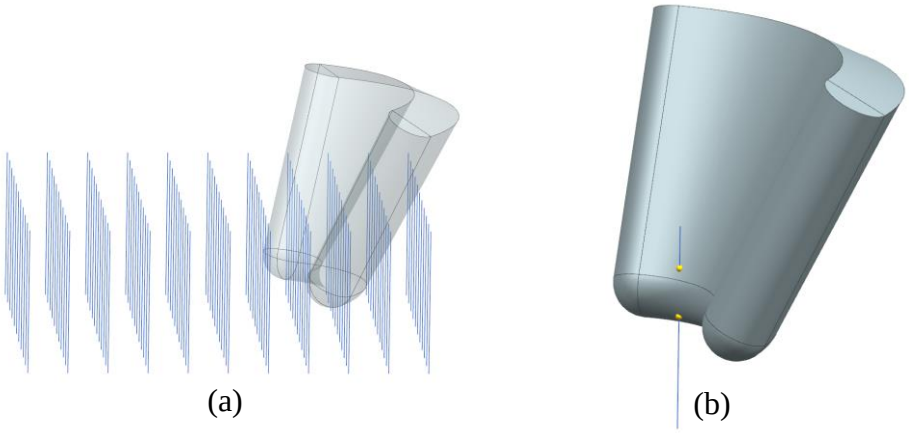


Figure 2-9 Clipping operation on line segments for vector model update using a swept volume B-rep. (a) swept volume B-rep in a vector field, and (b) one vector being clipped.

2.3.3 Binary operations for space partitioning

Space partitioning based workpiece models such as with voxels and octree are composed of volumetric elements which has a definite number of possible states. In case of binary voxel models, they can be either active or inactive. In case of octree models, the leaf nodes can be either active or inactive whereas the higher level nodes can additionally be marked as partially occupied.

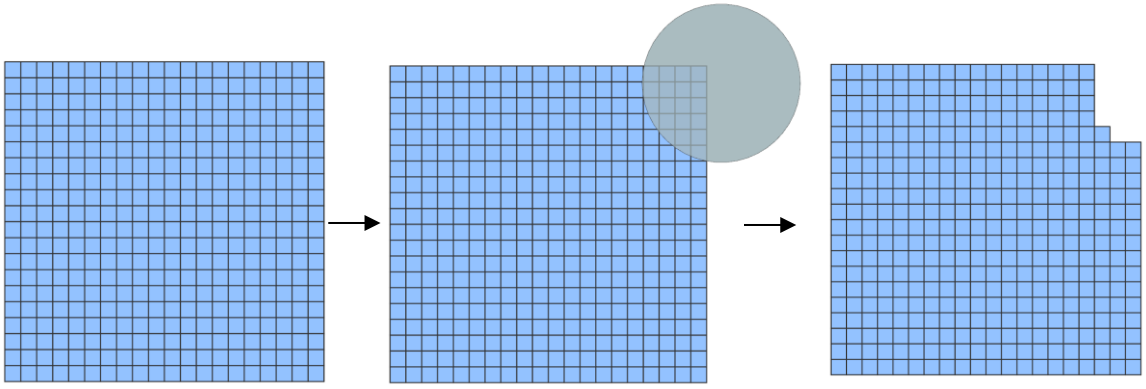


Figure 2-10 Binary update of a voxel workpiece by simple deletion of voxel elements.

As a result, update of space partitioning models is binary in nature. For a voxel workpiece, the voxels falling inside the tool swept volume are immediately turned inactive (Figure 2-10). For octree models, the leaf nodes falling inside are deleted and all higher level nodes completely emptied as a result are deleted recursively.

Alternatively for the voxel workpiece representation the voxel model of the tool swept volume can be generated first and a binary Boolean subtraction update can be done with voxel tool swept volume as the Boolean tool and the voxel workpiece as the target. Similar procedure is possible with octree creation of workpiece and tool swept volumes.

Either way, the major operation involved is to check whether a voxel or an octree node is completely within a tool swept volume representation. This involves computations to classify the voxel centre or corner points with respect to the tool swept volume. Such computations are always simpler than those involved for update of solid model and vector workpiece representations.

2.4 Workpiece surface generation

Surface generation for solid models such as NURBS and polyhedral B-rep is trivial as the geometry definition itself is based on the boundary surface elements. Usually NURBS B-rep models are tessellated to a triangle soup or mesh for easier visualization and inspection pipeline. Surface generation is of real relevance for vector and space partitioning models.

Voxel based space partitioning has been used for simulation and model visualization. Dedicated volume rendering hardware is reported as the natural choice for voxel model visualization [59]. This applies for octree model as well. However, such hardware is not yet common and alternative visualization techniques are still in need. Volume rendering with ray

casting is attempted on regular graphics hardware as well. However, this can prove to be computationally costly for view independent interactive model display. Thus, a surface generation technique is of good use for voxel and octree models.

Spatial range data collected from 3D scanning and medical imaging has been converted to surface models using the popular marching cube method [60]. They all rely upon specific distance field values at each corner point of a voxel to create the surface patch within by tri-linear or higher order interpolation techniques. In case of binary voxel models with information available only to infer the corner occupancy status, the surface generation needs to approximate the patch using the mid-points of the edges as patch boundary vertices and subsequent smoothing [61]. This can however, add unnecessary deviation onto the generated mesh. Point cloud triangulation technique using the centre points of the voxels with surface crossing is another alternative used.

The marching cube method is fast once the corner occupancy status is identified for the voxel. This is because, it utilizes a look-up table of pre-defined triangulation for each configuration of voxel corner point occupancies. The number of such configurations are definite and has many symmetry groups reducing the look-up table to only 15 different unique configurations. Thus the lookup table method is fast and has wide application. However, there are conflicting triangulations for some configurations (as one shown in Figure 2-11) which when occur in neighboring voxels cause the mesh to have holes and become non-manifold. In order to handle this, improved look-up tables and robust implementation have been developed [62, 63]. Those techniques need additional range data at the voxel corner points and are therefore not directly applicable to binary voxel models.

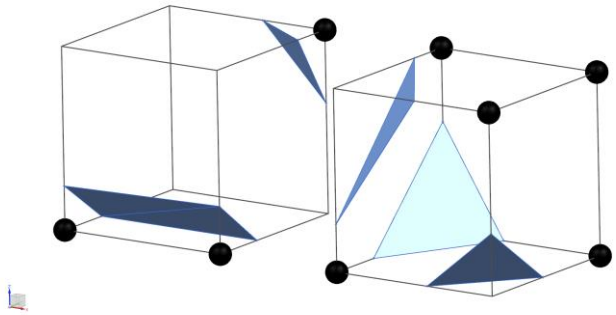


Figure 2-11 One conflicting pair of triangulation (at the shared face)
from classic Marching Cubes lookup table.

Other surface reconstruction approaches reported for vector model based simulation systems are worth considering. However, many of them are applicable to *Z*-map or dexel models only [64, 65, 66]. A recent work for generating surface representation from tri-dexel models has identified an algorithmic approach to triangulate surface patches within grid cells generated from the tri-dexel model [67]. This algorithmic approach appear to be robust to handle all practical cases and is a good alternative to use instead of the classical marching cubes for surface generation in case of voxel models as well. However, because the triangle patch has to be obtained computationally within each voxel, this prove to be a costly choice especially when triangulation on the fly for machining simulation has to be fast enough to provide enough display frames per second. A minimum of 15 frames per second display requirement leaves only about 66 milliseconds per frame. This demands the surface generation to be as fast as possible.

2.5 Summary

The various classes of workpiece geometry representation are solid modeling, vector modeling and space partitioning. Solid modeling provides the best accuracy whereas space

partitioning with multi-level representation can be the most efficient. A new approach that can merge the qualities of the two can help to achieve the next level of efficiency and accuracy combinations. Tool swept volume models can be from purely analytical to B-rep based solid models to approximate sampled tool instances. While the analytical and B-rep approaches can be exact, the sampled instances can cater to all type of tool paths. The workpiece model update depends upon the workpiece geometry and tool swept volume models and can be Boolean operations of solid models, vector trimming or binary update of voxel models. The workpiece surface representation can also be using surface or volume rendering based on the underlying workpiece geometry model.

Chapter 3: Frame-sliced voxel representation

In this chapter we will define and establish a new geometry modeling format that possess the qualities needed to be efficient and accurate enough for machining simulation. From the review of existing methods in the previous chapter, we identified the strengths and weakness of solid modeling, vector modeling and space partitioning classes of modeling methods. A better method is derived in this chapter using the lessons learned about the different methods. In the following sections the new geometry representation format is defined and discussed after identification of the salient features.

3.1 Voxel identification

The basic building element of the FSV-rep is a voxel. The numbering convention is shown in Figure 3-1 for the voxel vertices. The edges and faces can be identified based on the indices of the bounding vertices. This convention helps query for neighbors of each voxel and facilitate the involved computations.

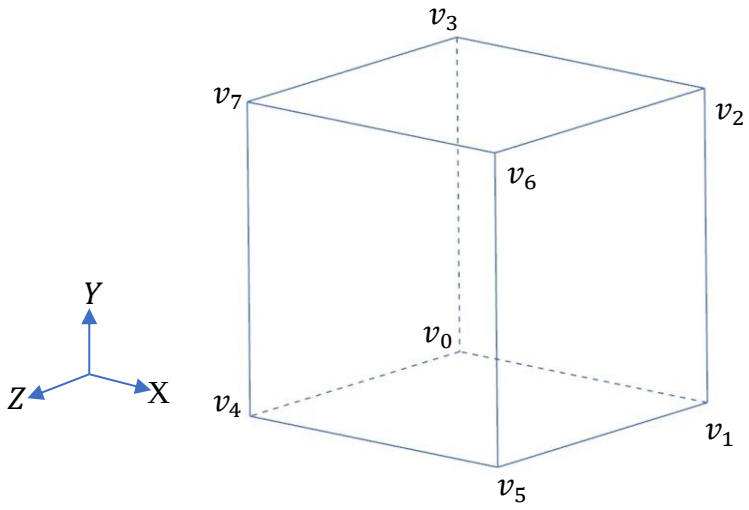


Figure 3-1 A voxel with indexed vertices.

Each voxel in the voxel space will be uniquely identified by a voxel id, V_{id}, that is obtained from the coordinates of v_0. For a voxel space with origin at $\mathbf{O}(O_x, O_y, O_z)$ and a voxel with v_0 at $\boldsymbol{p_0}(p_{0x}, p_{0y}, p_{0z})$,

$$x_{id} = floor\left(\frac{p_{0x} - O_x}{L_v}\right);\ y_{id} = floor\left(\frac{p_{0y} - O_y}{L_v}\right);\ z_{id} = floor\left(\frac{p_{0z} - O_z}{L_v}\right) \quad (3.1)$$

$$V_{id} = z_{id} \cdot N^2 + y_{id} \cdot N + x_{id} \quad (3.2)$$

where L_v is the voxel edge length and N the grid resolution of the voxel modeling space. The floor function returns the largest integer smaller than the real number argument. This unique indexing of voxels helps in enumerating the modeling space as shown in Figure 3-2 and also in the random access of any voxel at consistent query time.

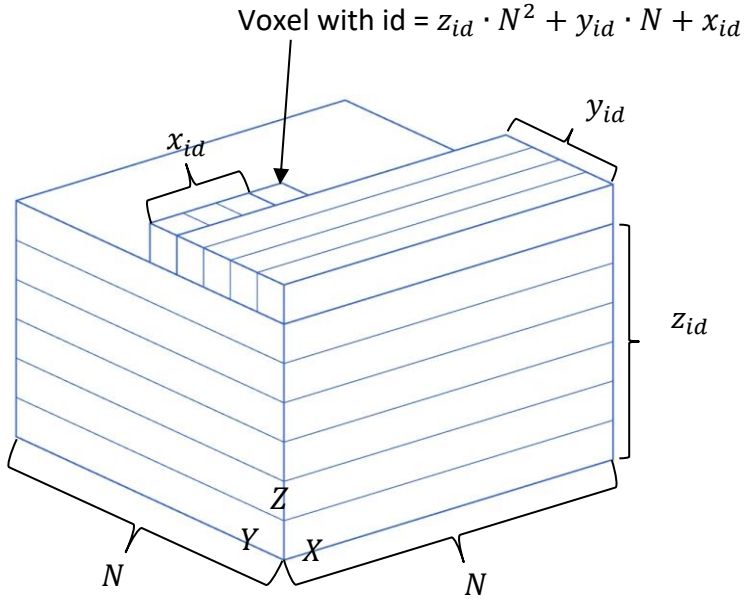


Figure 3-2 Enumeration of the voxel space with a unique index for each voxel.

3.2 Surface voxels

The FSV-rep is essentially a sparse voxel representation as it uses only the surface voxels to represent a solid. The surface voxel model for a solid object is to be made up of only the voxels through which the boundary surfaces of the object pass. As the FSV-rep uses a binary voxel representation, each voxel in the voxel modeling space can only be set to an active or inactive state. The binary voxel representation essentially limits the voxels to have only the occupancy information as their attribute and no other information such as material properties is present [42]. Since a surface voxel model is to be constructed, the voxels through which the boundary surfaces of the modeled object pass are deemed to be active while the voxels either completely inside or outside the object are deemed inactive (Figure 3-3).

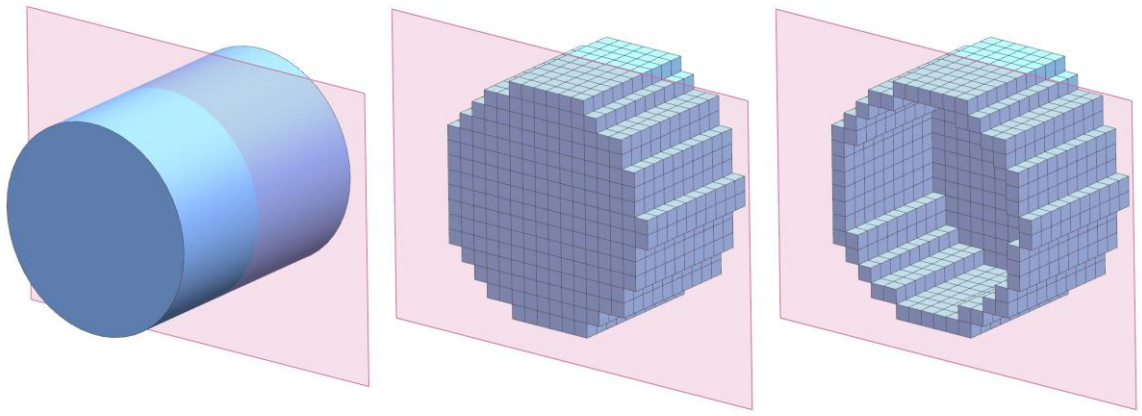


Figure 3-3 Volume voxel model (middle) and surface voxel model (right) for a cylinder.

A surface voxel can be identified using the signed distance [68] of its corner points from the object surface. If d_i is the signed distance of the ith corner point, a sign value can be assigned to that corner as

$$s_i = \frac{d_i}{|d_i|} \tag{3.3}$$

Then, a voxel is a surface voxel if

$$-8 < \sum_{i=0}^{7} s_i < 8 \tag{3.4}$$

3.3 26-separating voxel model

The FSV-rep uses a surface voxel model which is 26-separating. Such a model is needed to avoid tunneling as explained earlier as well as for proper reconstruction of the machined part surface as a closed 2-manifold triangle mesh. During the subtraction Boolean operations between surface voxel models, tunneling can cause incorrect or computationally unstable results and thus, has to be avoided.

Having a 26-separating voxel model assures that tunneling does not occur (as shown in Figure 3-4). In Figure 3-4c, an arrowed line is piercing through the cross-section of a 6-separating surface voxel model. It is observed that the voxel chain $\{V_p\}$ along the arrowed line is not intersecting (not having any voxel in common) with the $\{V_s\}$ surface voxel model. This is avoided using a 26-separating surface voxel model (as shown in Figure 3-4d) which provides a cross-section (as shown in Figure 3-4e) that always ensures an intersection.

It should be noted that the FSV-rep surface voxels (identified with Equation 3.4 above) will not include voxels with just their faces or edges intersecting the modeled object surface.

This is acceptable as the resulting surface voxel model is still 26-separating.

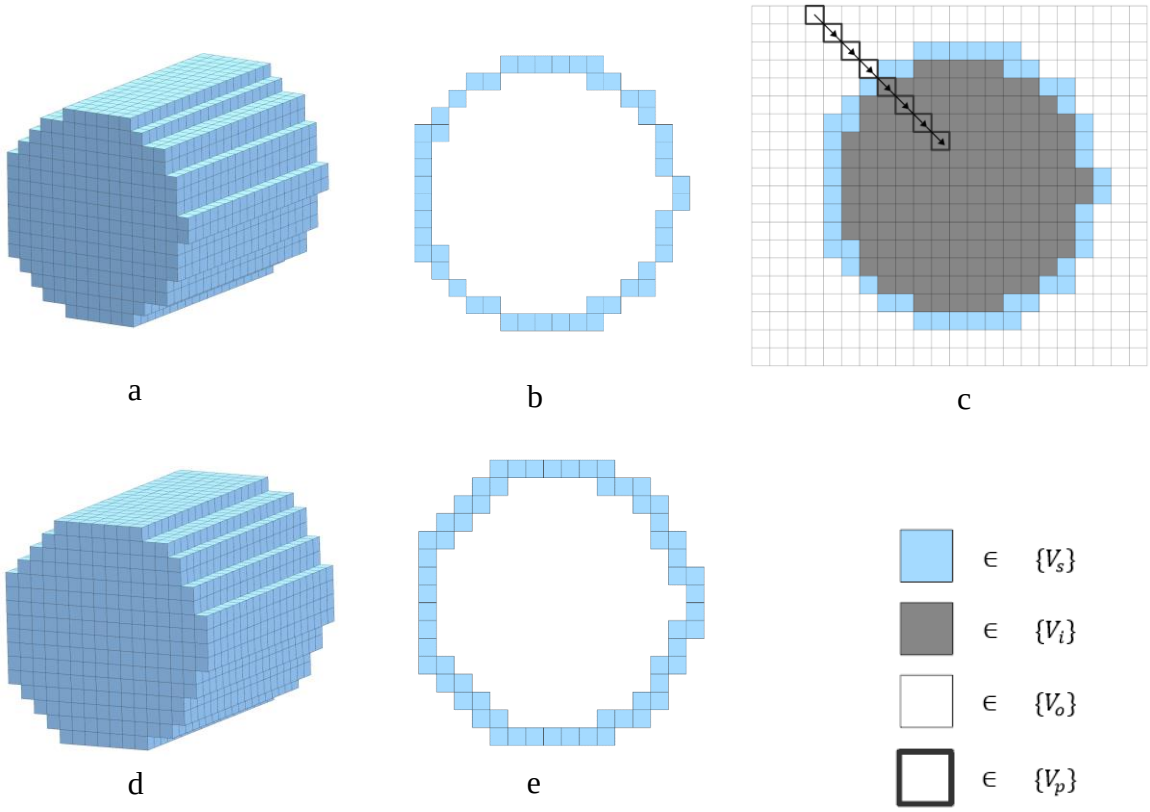


Figure 3-4 Two surface voxel representations of the same reference object (a cylinder sampled with a very low resolution grid). (a) a closed but 6-separating voxel model, (b) one cross-section of the model viewed along the cylinder axis and, (c) one possible tunneling situation for the same. (d) a 26-separating voxel model for the cylinder and (e) a cross-section of the same without any tunneling locations.

3.4 Multi-level voxel representation

Even though a single uniform grid at the required resolution can enable the volume removal operations at a computational cost independent of the model complexity, the cost is still affected by the grid resolution. For a grid resolution N, the number of voxels in the voxel space is N^3. For a model update, the number of voxels to be updated is thus of the order of $\mathbf{O}(N^3)$. Thus, increasing the grid resolution not only increases the memory requirement but also slows

down the update of the single-level voxel model. In order to address this issue, a multi-level voxel representation can be used such as the octree subdivision.

The octree subdivision will be the most efficient in collectively removing the bulk volume and then further removing at finer levels towards the boundary of the removed volume. However, for the machining simulation application where the cutting tool volume is fixed and often much less than the overall voxel space volume, it is ineffective to use the octree subdivision. This is because many of the leaf nodes corresponding to the workpiece volume removed by the tool may be bigger than the tool size. Further subdivision will be required in order to make those nodes small enough to be completely removable by the tool. Hence, the higher octree levels in effect become an overhead in repeated workpiece model update.

To avoid the overhead of traversing the octree subdivision but still reduce the computational cost of using only the finest grid resolution, the FSV-rep uses a multi-level voxel representation. The multi-level representation uses more than one voxel grids of increasing resolution to subdivide the same modeling space by a local finer grid within the voxels of coarser grid that need refinement. For the machining simulation application, the grid resolution at the coarse level can be set according to the cutting tool size. There can be one or more finer level grids to reach the desired fine grid resolution once the coarse level is set. The multi-level along with sparse voxel representation as depicted in Figure 3-5 makes the FSV-rep efficient in memory usage and potentially efficient in terms of the computational cost.

Given a surface voxel model at a particular grid resolution, a surface voxel model at a lower resolution can be obtained simply by identifying the voxels of the lower resolution grid in which each surface voxel from the higher resolution grid resides. Hence, for the FSV-rep, voxelization at lower resolutions is trivial once the surface voxels at the finest resolution are

identified. Although it is a bottom-up approach, it is simpler and faster once the finest level voxelization is done as the actual voxelization needs not be done for the coarser levels.

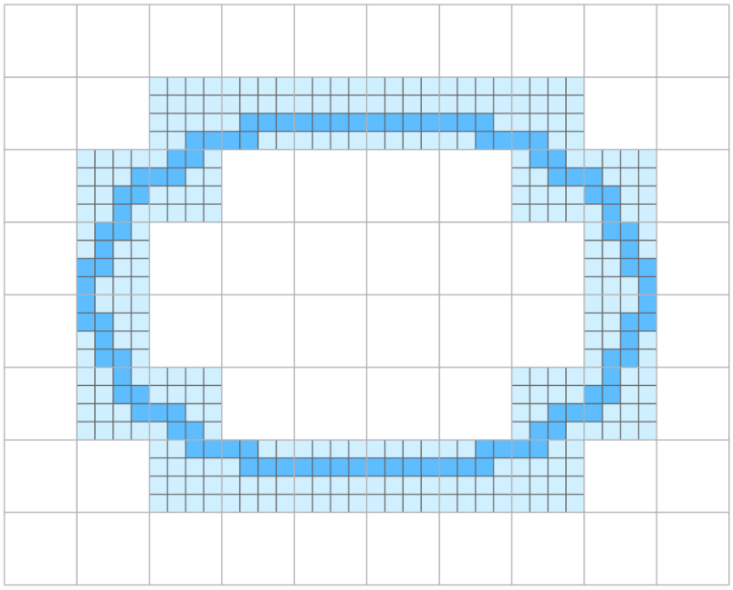


Figure 3-5 Memory-efficient multi-level sparse voxel representation of an elliptical cross-section.

For implementation simplicity, a two level sparse voxel representation is used in this thesis work for the FSV-rep model.

3.5 Frame-crossing points and frame-sliced voxels

The finest resolution that can be used for a voxel model is limited due to the computer memory and computational load restrictions. An alternative, thus, has to be developed if further accuracy is desired. The FSV-rep uses sampled points from the modeled object's boundary surface along the surface voxel edges as the alternative (Figure 3-6). These points are referred to as the frame-crossing (FC) points as they are the locations where the boundary surface of the modeled object crosses the edge-frame of the surface voxels. For an object model with a triangle-mesh representation of the boundary surface available, these points can be computed with straightforward line-triangle intersections. For NURBS-based curved surface boundary

representations, either a triangle mesh approximation is to be generated first or line-NURBS surface intersections are to be computed.

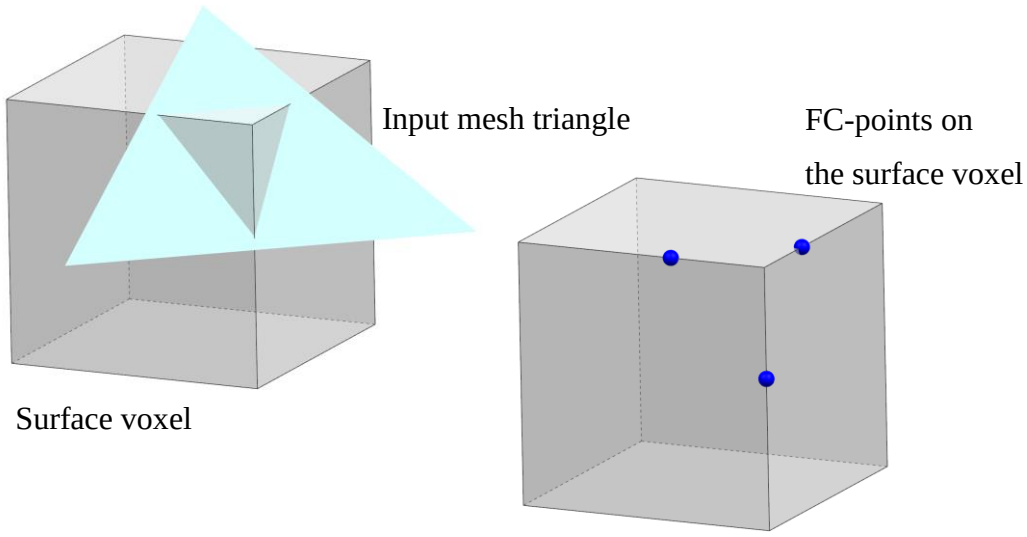


Figure 3-6 Frame-crossing (FC) points on a surface voxel from an input mesh triangle.

With the set of FC-points obtained for a surface voxel, one or more 3D loops or closed chains L_i ($i \geq 1$) can be formed from the FC-points. Each edge segment in L_i is on a face of the surface voxel and has the FC-points as its end points. A 3D piecewise linear surface with L_i as its boundary can then be defined, which is within the surface voxel. This 3D surface is named as the slice front because it can be seen as slicing the surface voxel into an interior and an exterior part. The boundary loop L_i of a slice front is the slicing loop polygon for the edge-frame of the surface voxel, dividing it into frame slices completely within the modeled object or completely outside.

The frame slice that is interior to the modeled object along with the associated slice front(s) defines a partial voxel named as the frame-sliced (FS) voxel as it is derived from a sliced frame (Figure 3-7). All such FS-voxels act as boundary elements for the modeled object and

together form a connected surface-voxel boundary representation which is named in this work as the frame-sliced voxel representation or FSV-rep in short.

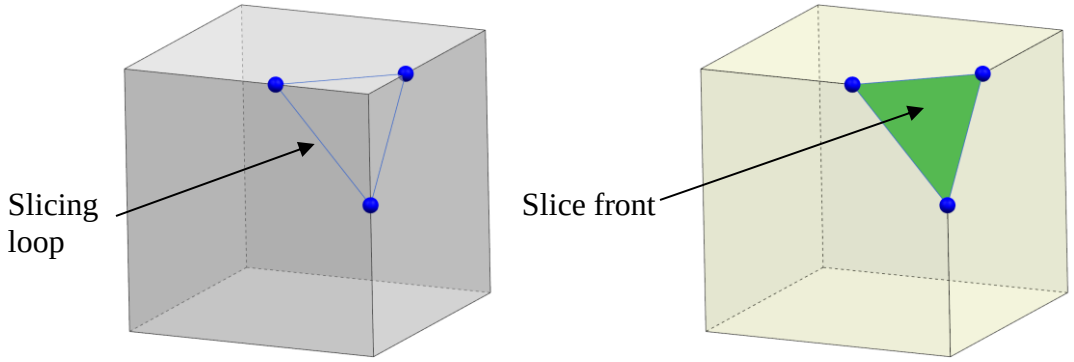


Figure 3-7 Slicing loop and slice front formation for the case in Figure 3-6.

3.6 Definition

To utilize the efficiency and robustness of a voxel model in repeated model updates and to achieve the accuracy and boundary representation similar to those of a triangle mesh, the frame-sliced voxel representation (FSV-rep) introduced in this work is defined as below:

> **Definition 1**: FSV-rep for a solid object is a collection of one or more 26-separating voxel sets with a different voxel grid resolution for each set together with a set of sample points of the object surface along the voxel edges for each surface voxel at the finest resolution.

The core of an FSV-rep model is a sparse voxel representation with more than one uniform grid of voxels to enumerate the modeling space. As discussed earlier, use of voxel models with high resolutions is undesirable due to the large memory demand. Hence, the FSV-rep does not aim at setting the finest voxel grid resolution according to the desired accuracy but use sample points from the object surface to achieve the sub-voxel resolution. These sample points are taken such

that they are also on the edge-frame of the finest resolution surface voxels. As a result, these points are named as the frame-crossing (FC) points.

3.7 Triangle mesh construction from the slice fronts

The slice fronts of each sliced surface voxel has its boundary edges on one of the 6 voxel faces. In fact, there is only one slice front within each sliced surface voxel containing the specific boundary edge. More importantly, each boundary edge of a slice front is shared with a slice front of the neighboring sliced voxel coincident on the voxel face containing the boundary edge. This is true for all the slice front boundary edges because the surface voxel model is 26-separating. As a result, the slice fronts connected along the shared boundary edges form a closed 2-manifold mesh of slice fronts. Each slice front can then be triangulated to readily obtain a 2-manifold triangle mesh (Figure 3-8).

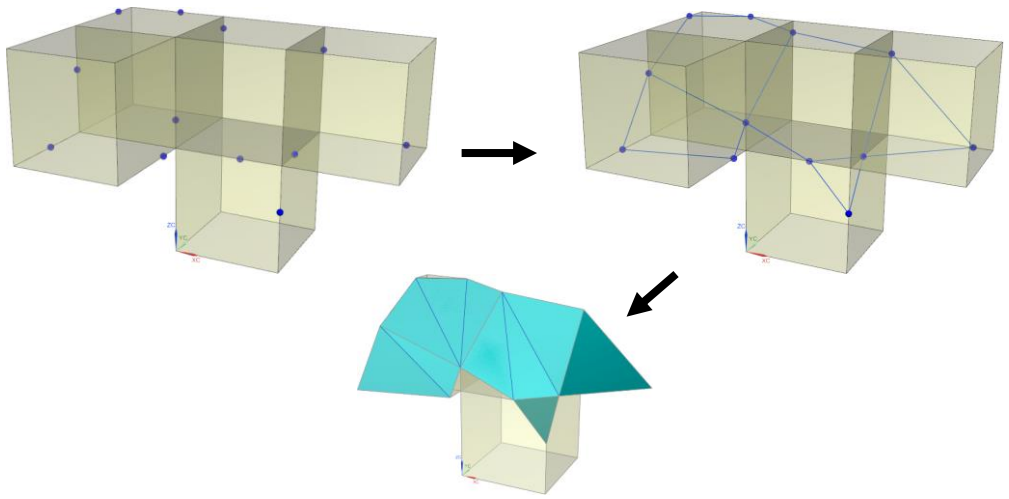


Figure 3-8 Triangle mesh construction from an FSV-rep model.

A triangle mesh has to satisfy various topological and geometric conditions in order to be a

valid representation of a physically possible object [69]. The most important topological condition for a valid triangle mesh representation of a solid object is being a closed 2-manifold mesh. It essentially means that every edge shall be shared by two triangles. An FSV-rep model guarantees to provide a 2-manifold triangle mesh. The number of triangles present in the resulting mesh corresponds to the size of the model. High-curvature features will need many small triangles in order to have a quality representation. The triangle mesh constructed from the FS-voxels is sized relative to the finest voxel size. Hence, as long as the finest voxels are small enough to capture the surface curvature, the created triangle mesh will also be able to capture it.

3.7.1 Benefits of machined surface triangulation via FSV-rep

Triangle mesh itself has been used for machining simulation [15,16] for its major advantage of being able to represent the object surface with simple triangles. With varying size of the triangle to adapt to different curvature, the model size can be optimized when using a triangle mesh directly for machining simulation. However, as pointed out earlier in Section 2, it is inefficient to update a triangle mesh directly during repeated updates of a simulated workpiece with a large number of tool paths. It is mainly due to the difficulty to identify candidate triangles for update per tool path from the entire set of triangles in the mesh. Without the use of some localization technique such as octree, the computational complexity is high for the workpiece model update as every triangle will have to be considered for update with every tool path.

Even after using an applicable localization technique to reduce the computational complexity, the update process always involves triangle-triangle intersection calculations which are still much costlier than computing operations needed for updating a voxel or vector based model. Further, integrating the newly machined triangle mesh surface with the existing

workpiece surface is also a computationally expensive step. An existing research study has confirmed that triangle mesh based machining simulation tends to be slower due to the computational tasks stated above [16]. Nonetheless, triangle mesh is deemed the most suitable format for model visualization and is, thus, widely used for this purpose in current CAM software.

On the other hand, using an FSV-rep model for the repeatedly updated workpiece and generating a triangle mesh representation only when needed for model visualization, eliminate the above mentioned drawbacks. First, with the voxel based model structure, it is computationally simpler to identify the voxels that need modification per tool path as it involves only the classification of voxel corner and center points with respect to the specific tool position. Second, the potential subset of voxels that can possibly interact with the tool can be easily estimated by utilizing the correspondence of voxel id, as identified by Equation 3.2, to its spatial location.

However, the use of FSV-rep for simulation and the generation of a triangle mesh from the FSV-rep model only at the end will result in a mesh with triangle size decided by the finest voxel size. There will be small triangles used even for surfaces with low curvature, resulting in a higher than optimal number of triangles. Existing mesh simplification techniques can be used to address this issue. Another potential issue from the use of FSV-rep is the classical chamfering effect of voxel based representations whereby the sharp edge/corner features of a model gets "chamfered". This is an artifact of FSV-rep models as FS-voxels can improve accuracy of the object surface but still cannot catch the sharp features at the interface of two surfaces. There are already developed methods to detect sharp features for triangle mesh models generated from grid-based models [70, 71] and these methods can be applied to the triangle mesh generated from

FSV-rep as well.

In summary, the FSV-rep provides a model representation format which is voxel based to retain the computational efficiency and robustness of space partitioning methods in volume removal operations and also significantly improves the modeling accuracy through the FS-voxels without being limited by the finest voxel grid resolution. An FSV-rep model can also be readily converted into a triangle mesh based boundary representation model to facilitate subsequent visualization, analysis and processing tasks of the created model.

3.8 FSV-rep data structure implementation

Figure 3-9 illustrates the basic implementation architecture to create the FSV-rep from an input shape and then produce a triangle mesh output from the FSV-rep model.

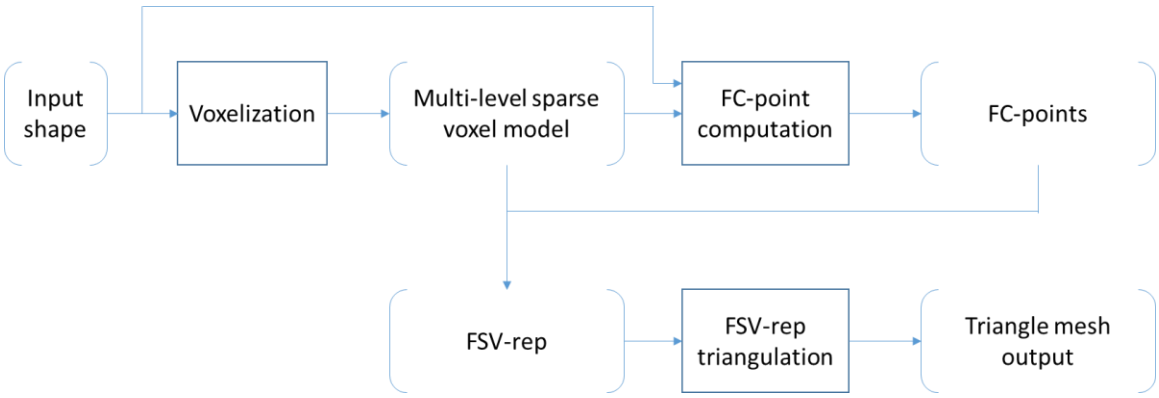


Figure 3-9 FSV-rep implementation architecture.

The FSV-rep has been implemented and coded in C++. A one-dimensional array of binary variables (bit-array) is used to represent the three-dimensional grid of voxels making the voxel space. The voxel id formulated in Equation 3.2 is used as the index of the bit-array to represent each voxel. During the surface voxelization step, the bits corresponding to the surface

voxels are set to 1 (all the bits are set to 0 initially). To hold the FC-points, a binary tree with the voxel id as the key and the set of FC-points for each voxel as the value is created.

3.8.1 Input shape

The preferred input shape is taken as a triangle mesh as it is composed of simple triangular faces and voxelization of a triangle mesh to create a 26-separating surface voxel model is comparatively straightforward as all the faces are planar. Also, triangle mesh is a common format which can be readily exported from all CAD systems used to create the related solid model. It should be noted that the triangle mesh is an approximate representation and using it will cause some approximation error in the input. However, an appropriate triangle size can be used to limit the approximation error relative to the intended voxel size. This can ensure the surface voxel model generated from the triangle mesh to be almost the same as that generated from the exact surface representation.

3.8.2 Surface voxelization

A boundary-first flood-voxelization approach for creating the surface voxel model is employed in this work. Under this approach, voxelizing a given triangle is a three-step process as depicted in Figure 3-10.

First, the three voxels in which the three vertices of the triangle reside are identified. After that, the voxels through which each of the three edges passes are obtained via binary subdivision of the edges. The voxel for the mid-point of each subdivided edge is marked until both end points of the subdivided edge are in the same voxel or the voxels are neighbors. The voxel chain for each edge is made to be 6-connected by adding extra voxels as needed. Voxelization of the

face interior is done starting with a seed voxel through which the interior of the triangle passes. The set of voxels through which the triangle face passes is obtained by propagating outward from the seed voxel into the neighboring voxels through the voxel edges that are crossing the triangle face plane and continuing the process for all such neighbors.

This set of voxels identified will be 6-connected as all neighboring voxels for all the intersecting voxel edges are used. With all such voxel sets from all the triangles in the input mesh, a 6-connected surface voxel model is generated for the boundary surface of the input shape, which is commonly the blank geometry in machining. It should be noted that the model is also 26-separating as all the voxels with edges crossing the input mesh are present in the surface voxel model.

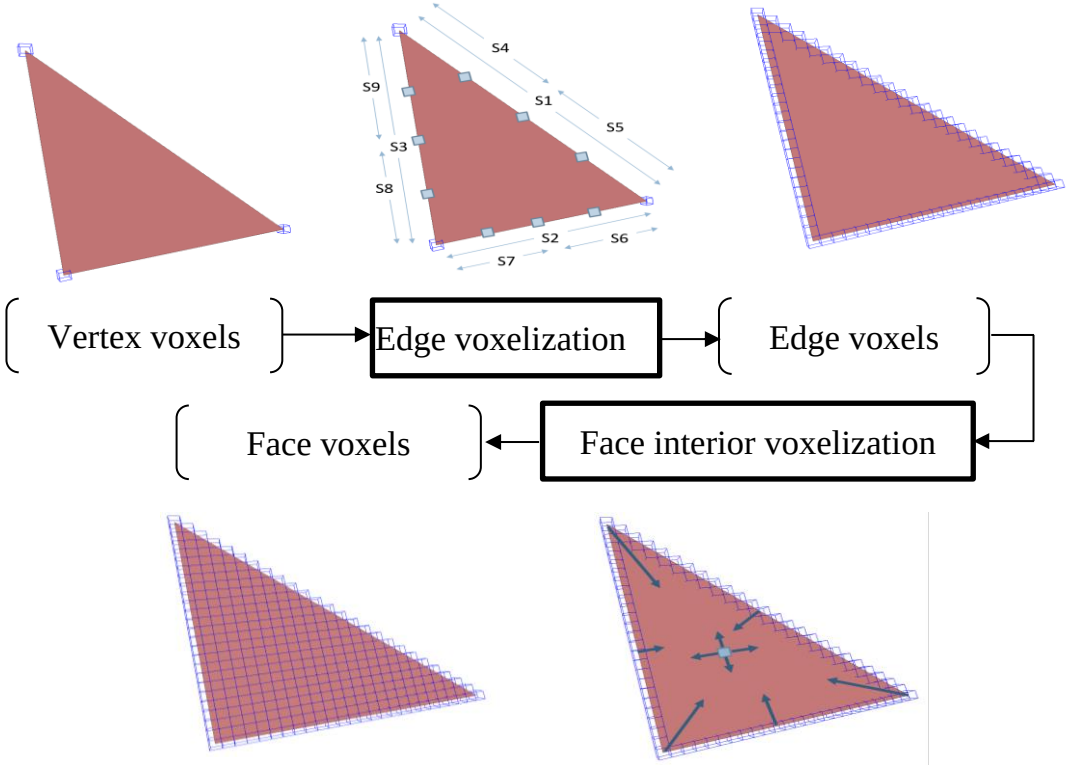


Figure 3-10 Three-step voxelization process for a single face in a triangle mesh.

The procedure followed to voxelize a triangle mesh is illustrated as Algorithm 3-1 below.

Algorithm 3-1: Triangle mesh voxelization

$\{\boldsymbol{F}\} \leftarrow$ all faces of input triangle mesh
for each face F in $\{\boldsymbol{F}\}$ **do** for-loop-1:
  for each vertex V of F **do** for-loop-1.1:
    $ID \leftarrow$ voxel identifier from vertex coordinates using Equations 3.1 and 3.2
    $\{\boldsymbol{ID\_b}\} \leftarrow \{\boldsymbol{ID\_b}\} + ID$
  end for-loop-1.1
  for each edge E of F **do** for-loop-1.2:
    $\{\boldsymbol{S}\} \leftarrow$ line segment from start vertex v_s to end vertex v_e of E
    $count \leftarrow 1$
    while $count \leq$ size of **{S} do** while-loop-1.2.1:
      $v_s \leftarrow$ start vertex of $count^{th}$ segment in $\{\boldsymbol{S}\}$
      $v_e \leftarrow$ end vertex of $count^{th}$ segment in $\{\boldsymbol{S}\}$
      $ID1 \leftarrow$ voxel identifier from v_s coordinates using Equations 3.1 and 3.2
      $ID2 \leftarrow$ voxel identifier from v_e coordinates using Equations 3.1 and 3.2
      if $ID1 \neq ID2$ **and** Voxel($ID1$) not neighbor to Voxel($ID2$) **then**
        p_{mid} = mid-point between v_s and v_e
        $ID3 \leftarrow$ voxel identifier from p_{mid} coordinates using Equations 3.1 and 3.2
        $\{\boldsymbol{ID\_b}\} \leftarrow \{\boldsymbol{ID\_b}\} + ID3$
        $\{\boldsymbol{S}\} \leftarrow \{\boldsymbol{S}\}$ + line segment from v_s to p_{mid}
        $\{\boldsymbol{S}\} \leftarrow \{\boldsymbol{S}\}$ + line segment from p_{mid} to v_e
      end if
      $count \leftarrow count + 1$
    end while-loop-1.2.1
  end for-loop-1.2
  /*Voxels($\{\boldsymbol{ID\_b}\}$) are assumed to be 6-connected in the below code*/
  $ID \leftarrow$ voxel identifier of a seed voxel entirely on face interior area
      (e.g., from face centroid)
  $\{\boldsymbol{ID\_i1}\} \leftarrow \{\boldsymbol{ID\_i1}\} + ID$
  $count \leftarrow 1$
  while $count \leq$ size of $\{\boldsymbol{ID\_i1}\}$ **do** while-loop-1.1:
    $ID \leftarrow count^{th}$ element in $\{\boldsymbol{ID\_i1}\}$
    $\{\boldsymbol{E\_V}\} \leftarrow$ edges of Voxel(ID) intersecting the face interior area
    $\{\boldsymbol{ID\_N}\} \leftarrow$ edge neighbors of Voxel(ID) through edges in $\{\boldsymbol{E\_V}\}$
    $\{\boldsymbol{ID\_i1}\} \leftarrow \{\boldsymbol{ID\_i1}\} + (\{\boldsymbol{ID\_N}\} - \{\boldsymbol{ID\_b}\})$
    $count \leftarrow count + 1$
  end while-loop-1.1
  for each ID in $\{\boldsymbol{ID\_b}\}$ not hit by above propagation **do** for-loop-1.3:
    $\{\boldsymbol{E\_V}\} \leftarrow$ edges of Voxel(ID) intersecting the face interior area
    $\{\boldsymbol{ID\_N}\} \leftarrow$ edge neighbors of Voxel(ID) through edges in $\{\boldsymbol{E\_V}\}$

```
    {ID_i2} ← {ID_i2} + ({ID_N} – {ID_b})
  end for-loop-1.3
  count ← 1
  while count ≤ size of {ID_i2} do while-loop-1.2:
    ID ← count^th element in {ID_i2}
    {E_V} ← edges of Voxel(ID) intersecting the face interior area
    {ID_N} ← edge neighbors of Voxel(ID) through edges in {E_V}
    {ID_i2} ← {ID_i2} + ({ID_N} – {ID_b} – {ID_i1})
    count ← count + 1
  end while-loop-1.2
  {All_IDs} ← {All_IDs} + {ID_b} + {ID_i1} + {ID_i2}
end for-loop-1
return {All_IDs}
```

3.8.3 Multi-level surface voxel model

FSV-rep uses a sparse multi-level representation with the finer resolution voxel grid defined only within the surface voxels of the coarser grid. The simple voxelization method described in the previous subsection is not enough to obtain the surface voxels at all the grid levels. It can be done in two ways. In a top-down approach, the surface voxelization is performed at the coarse level first and then within each resulting surface voxel, a similar process is carried out at the finer grid resolution. This, however, requires voxelization at multiple levels. Hence, an alternate bottom-up approach has been used in the current implementation. In the bottom-up approach, all the surface voxels at the finest grid resolution can be identified first using an imaginary voxel space spanning the entire modeling space with the finest resolution. The coarser surface voxels can then be identified according to where each fine voxel resides. For a given fine voxel identified with any x_{id}, y_{id} and z_{id}, the coarser voxel in which it resides is identified as

$$X_{id} = floor\left(\frac{x_{id}}{f}\right); \quad Y_{id} = floor\left(\frac{y_{id}}{f}\right); \quad Z_{id} = floor\left(\frac{z_{id}}{f}\right) \tag{3.5}$$

where f is the subdivision factor between the fine and coarse grid resolutions N_F and N_C as

$$f = \frac{N_F}{N_C} \tag{3.6}$$

In the equations above, f needs to be an integer as it corresponds to the number of subdivisions of a coarse voxel into the corresponding set of fine voxels. Further, for memory efficiency in the fine voxel sub-space within each coarse surface voxel, it is preferable if the number of fine voxels within each coarse voxel is a multiple of 8 as it can then use an integer number of bytes to represent the sub-space. This further demands f to be a positive even integer and the corresponding appropriate values for N_F and N_C.

3.8.4 FS-voxels

To generate the FS-voxels, a map between each fine surface voxel and the triangles of the input mesh passing through it, is to be kept. The FS-voxel data structure is defined as having a pair of FC-parameter values (u_1, u_2) along each of the primary edges of the voxel. In this work, the primary edges are the edges between v_0 and v_1, v_0 and v_3, and v_0 and v_4 (Figure 3-1).

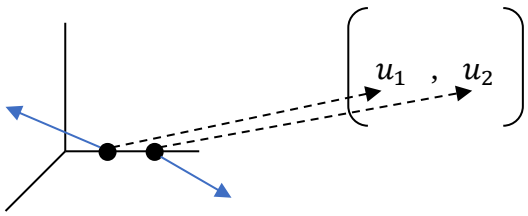


Figure 3-11 Placement of the FC-point parameter into the pair based on the surface normal (blue arrows) of generating object.

The parameter pairs are to hold the location of the FC-points with respect to the line segment of the corresponding voxel edge. As in Figure 3-11, If the intersecting triangle is such that the component of its face normal along the primary edge is positive, the parameter for the

resulting FC-point is stored as u_2. If the component is negative, the parameter is stored as u_1. These particular locations to store the FC-point parameter will later help in the proper identification of the occupancy status of the voxel corners during surface mesh generation.

It is evident that an FS-voxel permits two FC-points along each voxel edge in the implementation. It can handle four possible edge crossing cases as shown in Figure 3-12. If there are more than two crossing locations, they are simplified into two crossing locations which are chosen based on comparing the missing portions against the active portions of the voxel edge. If the sum of the missing portions is longer, then the small fragments are ignored (Figure 3-13a). If the sum of the active segments is longer, then the small gaps are ignored (Figure 3-13b). With appropriate size for fine level voxels, it can be ensured such simplification will not cause loss of any required features as will be shown later in Chapter 8.

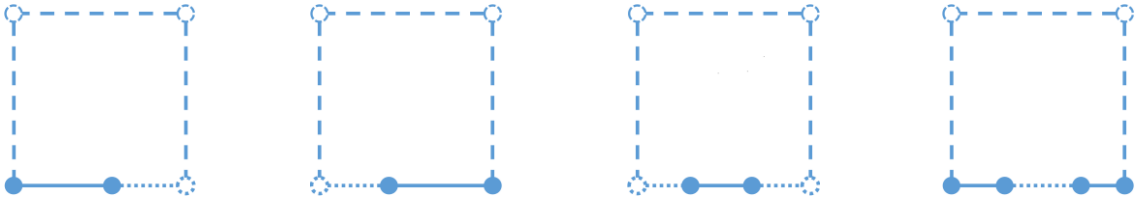


Figure 3-12 Four possible configurations of two permitted FC-points on a voxel edge.

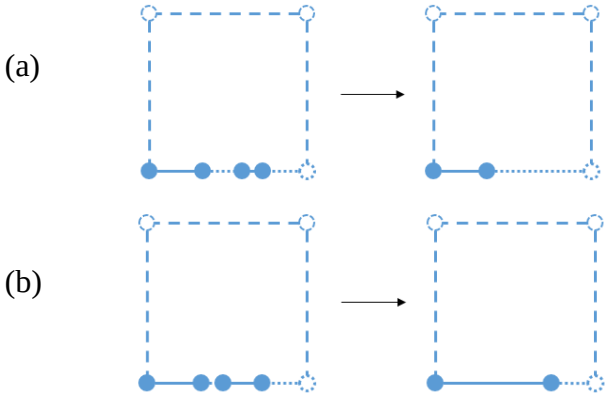


Figure 3-13 Simplifying fragmented FS-voxel edges: (a) small fragment ignored; and (b) small gap ignored.

It should be noted that the FC-point parameter pair is set only for the primary edges of a given FS-voxel. Other edges are in fact primary to some neighboring FS-voxels and thus the FC-point parameter pairs on all edges for an FS-voxel are effectively present in the data structure (Figure 3-14). Further the FC-parameter is limited to a range of [0,1) to ensure unique parent voxels for FC-points even when the point is exactly at a corner of the voxel.

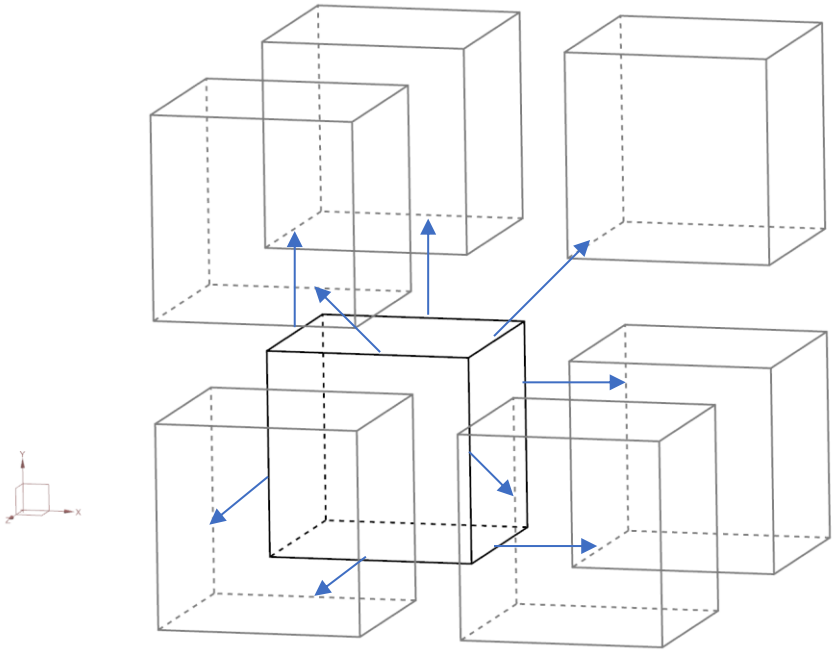


Figure 3-14 A partial FS-voxel (bottom left corner at the back) with arrows to edges of neighboring FS-voxels for FC-points on non-primary edges.

The overall procedure for the creation of an FSV-rep model from an input triangle mesh is illustrated as Algorithm 3-2 below.

Algorithm 3-2: FSV-rep model creation
$T \leftarrow$ input mesh $\{\boldsymbol{FV}\} \leftarrow$ fine surface voxels by Algorithm 3-1 $\{\boldsymbol{CV}:\{\boldsymbol{FV\_sub}\}\} \leftarrow$ map from a coarse voxel to fine voxels within it

for each fine surface voxel FV in $\{\boldsymbol{FV}\}$ **do** for-loop-1:
  $CV \leftarrow$ coarse voxel for FV using Equations 3.5 and 3.6
  $\{\boldsymbol{CV}:\{\boldsymbol{FV\_sub}\}\} \leftarrow \{\boldsymbol{CV}:\{\boldsymbol{FV\_sub}\}\} + [\boldsymbol{CV}, \boldsymbol{FV}]$
end for-loop-1
$\{\boldsymbol{CV}:\{\boldsymbol{FS}\}\} \leftarrow$ map from a coarse voxel to FS-voxels within it
for each $[\boldsymbol{CV}, \{\boldsymbol{FV\_sub}\}]$ pair in $\{\boldsymbol{CV}:\{\boldsymbol{FV\_sub}\}\}$ map **do** for-loop-2:
  for each local fine voxel FV in $\{\boldsymbol{FV\_sub}\}$ **do** for loop-2.1:
    $\{\boldsymbol{PE}\} \leftarrow$ primary edges of FV
    for each edge E in $\{\boldsymbol{PE}\}$ do for-loop-2.1.2:
      $\{\boldsymbol{u1}, \boldsymbol{u2}\} \leftarrow$ FC-parameters on E from triangles of T intersecting E
      $\{\boldsymbol{FC}\} \leftarrow \{\boldsymbol{FC}\} + [\boldsymbol{E}, \{\boldsymbol{u1}, \boldsymbol{u2}\}]$
    end for-loop-2.1.2
    $\{\boldsymbol{FS}\} \leftarrow \{\boldsymbol{FS}\} + [\boldsymbol{FV}, \{\boldsymbol{FC}\}]$
  end for-loop-2.1
  $\{\boldsymbol{CV}:\{\boldsymbol{FS}\}\} \leftarrow \{\boldsymbol{CV}:\{\boldsymbol{FS}\}\} + [\boldsymbol{CV}, \{\boldsymbol{FS}\}]$
end for-loop-2
initiate FSV-rep coarse and fine voxel-space bit-fields using the $\{\boldsymbol{CV}:\{\boldsymbol{FS}\}\}$ map

3.8.5 Triangle mesh surface generation

A closed 2-manifold triangle mesh representing the boundary of the modeled object can be easily generated from the FSV-rep model as previously described in Section 3.7. For this purpose, each FS-voxel with the FC-point parameters on all of its edges is considered individually. The closed slicing loops which define an FS-voxel are obtained from the FC-point parameters. The triangle mesh patch boundary creation algorithm used is developed later in Chapter 8. In order to create the patch boundary, the occupancy information of each voxel corner point has to be deduced from the FC-points. Using this occupancy information, a lookup table provided in Chapter 8 can be used to create the slicing loops which are in fact the patch boundaries. Considering each slicing loop as a triangle mesh patch boundary, the corresponding triangle mesh patch is created. All of these patches together form a closed 2-manifold triangle mesh. The procedure for the generation of a triangle mesh from an FSV-rep model is illustrated

as Algorithm 3-3 below.

Algorithm 3-3: Triangle mesh generation from FSV-rep

for each [$\boldsymbol{CV}$, {$\boldsymbol{FS}$}] pair in {$\boldsymbol{CV}$: {$\boldsymbol{FS}$}} map **do** for-loop-1:
  for each local FS-voxel FS in {$\boldsymbol{FS}$} **do** for-loop-1.1:
    {$\boldsymbol{ES}$} ← active edge segments along primary edges of FS
    {$\boldsymbol{NV}$} ← neighbor voxels of FS that are parent voxels of its other edges
    for each voxel NV in {$\boldsymbol{NV}$} **do** for-loop-1.1.1
      {$\boldsymbol{ES\_N}$} ← active edge segments along primary edges of NV shared with FS
      {$\boldsymbol{ES}$} ← {$\boldsymbol{ES}$} + {$\boldsymbol{ES\_N}$}
    end for-loop-1.1.1
    for each corner point CP of FS **do** for-loop-1.1.2
      if any of {$\boldsymbol{ES}$} attached to CP **then**
        set CP occupied
      else
        set CP unoccupied
      end if
    end for-loop-1.1.2
    {$\boldsymbol{L}$} ← slice front boundary loop
        (using Algorithm 8-1 in Chapter 8)
    {$\boldsymbol{t}$} ← triangulation with {$\boldsymbol{L}$} as boundary
    {$\boldsymbol{T}$} ← {$\boldsymbol{T}$} + {$\boldsymbol{t}$}
    {$\boldsymbol{FS\_empty}$} ← fine-level surface voxels neighbors through the primary edge of FS
    for each FS_empty in {$\boldsymbol{FS\_empty}$} **do** for-loop-1.1.3
      if FS_empty not in {$\boldsymbol{CV}$: {$\boldsymbol{FS}$}} **then**
        CV_parent ← coarse voxel for FS_empty
        {$\boldsymbol{CV}$: {$\boldsymbol{FS}$}} ← {$\boldsymbol{CV}$: {$\boldsymbol{FS}$}} + [$\boldsymbol{CV\_parent}$, $\boldsymbol{FS\_empty}$]
      end if
    end for-loop-1.1.3
  end for-loop-1.1
end for-loop-1
return {$\boldsymbol{T}$}

3.9 Effective memory usage

FSV-rep is voxel based and voxel sub-division of the modeling space generates N^3 voxels where N is the uniform grid resolution. For a voxel model representation, one byte can represent 8 voxels using 1 bit per voxel. For FSV-rep, the uniform grid of voxels is needed only at the coarse resolution. Hence, the memory usage for the coarse level is $N_C^3/8 = N_F^3/(8f^3)$. The rest of the memory requirement is only per coarse-level surface voxel. First, on a 64-bit machine, a map from coarse voxel ids (of 4 bytes size) to pointer (of 8 bytes size) to the coarse voxel object is needed. With n_{CS} coarse-level surface voxels, this map needs $12n_{CS}$ bytes. Each coarse voxel object has a pointer to its sub-space bit-field and a map of local fine surface voxels within it together adding $24n_{CS}$ bytes. Each sub-space bit-field needs $f^3/8$ bytes and a coarse voxel with n_{FS_sub} fine-level surface voxels within it needs $26n_{FS\_sub}$ bytes where 26 is from 2 bytes for the local id of the fine-level surface voxel and 24 bytes for the 6 FC-parameters (2 per primary edge). Thus, the effective total memory usage in bytes of an FSV-rep model is

$$M_F = \frac{N_F^3}{8f^3} + \left(36 + \frac{f^3}{8}\right) n_{CS} + 26 \sum_{i=1}^{n_{CS}} n_{FS\_sub,i} \tag{3.7}$$

In Equation 3.7, the summation in the last term yields the total number of fine-level surface voxels, n_{FS}, for the entire model. Thus, the expression reduces to

$$M_F = \frac{N_F^3}{8f^3} + \left(36 + \frac{f^3}{8}\right) n_{CS} + 26 n_{FS} \tag{3.8}$$

Further, $n_{CS} \cong \frac{n_{FS}}{f^2}$ and n_{FS} is of the order of $k \times \mathbf{O}(\mathrm{N_F^2})$, giving the approximate value for M_F in terms of the fine level grid resolution as

$$M_F \cong \left(\frac{N_F}{8f^3} + k\left(\frac{36}{f^2} + \frac{f}{8} + 26\right)\right) \times \boldsymbol{O}(N_F^2) \tag{3.9}$$

where k is a factor dependent on the shape of the object and size of the modeling space with respect to the bounding box of the object.

3.10 Case studies and discussion

Four case studies are presented in this section to demonstrate the improved accuracy in the reconstructed surface from the FSV-rep model. Case 1, 2 and 3 are from different pocket milling operations with increasing complexity in the cavity shape. Case 1 is a standard 2½-D pocket, Case 2 a cone-shaped pocket, and Case 3 a hemispherical pocket. Case 4 is a typical integrally bladed rotor which is a geometrically complex part with free-form surfaces. A triangle mesh surface is to be generated from the FSV-rep model and also from a basic voxel model, which is of the same resolution as the fine grid resolution of the FSV-rep model. In place of the exact FC-points used in the FSV-rep model, the basic voxel model uses mid-points of the edges crossing the object boundary surface as simple approximates of the FC-points. This is to illustrate the significance of the FC-points in the FSV-rep model. Connecting these approximated FC-points into a triangle mesh patch is then done for each surface voxel following the same procedure outlined in Section 3.8.5.

3.10.1 Model accuracy and memory efficiency

It is evident from Figure 3-15 and 3-16 that the FSV-rep models yield much smoother surfaces. The reconstructed triangle mesh surfaces from the FSV-rep models are also much more accurate as compared with the input mesh models. Hausdorff distance analysis was performed in Meshlab software [72] between each reconstructed mesh surface and the corresponding input mesh surface at 2 million uniformly sampled points. As can be seen in

Table 3-1, both the mean and root mean square (RMS) errors are much smaller for the meshes generated from the FSV-rep models. The maximum errors are of similar magnitudes to those in the basic voxel models due to the incapability of FSV-rep to correctly model sharp edges. This is, however, easily resolved by available existing sharp edge restoration techniques [70, 71].

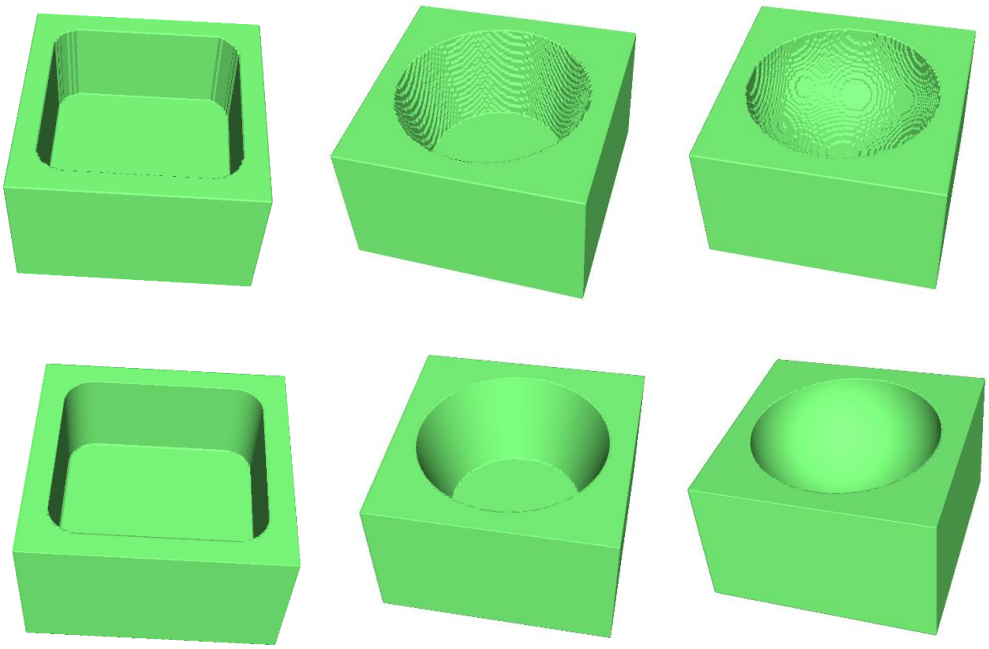


Figure 3-15 Triangle mesh surfaces generated from a basic voxel model (top) and from the FSV-rep model (bottom) for the pocket milling case studies.

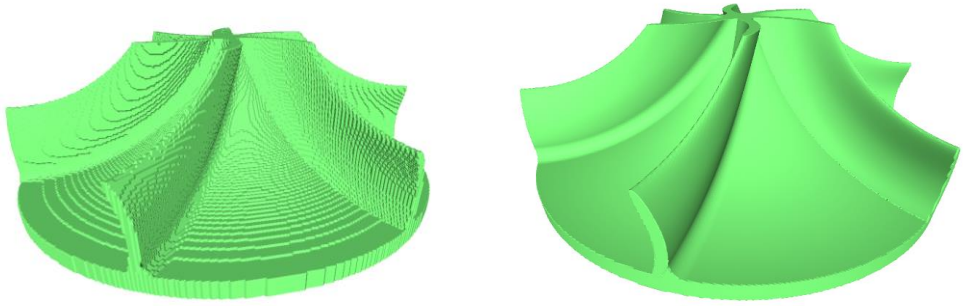


Figure 3-16 Triangle mesh surfaces generated from a basic voxel model (left) and from the FSV-rep model (right) for a complex IBR geometry.

Table 3-1 Comparison of errors in the reconstructed triangle mesh surfaces.

Case Study	Maximum (mm)		Mean (mm)		RMS (mm)	
	Basic	FSV-rep	Basic	FSV-rep	Basic	FSV-rep
1	0.900	0.751	0.234	0.005	0.242	0.036
2	0.903	0.805	0.218	0.005	0.230	0.033
3	0.925	0.785	0.214	0.005	0.228	0.032
4	0.841	0.635	0.177	0.005	0.202	0.025

The memory efficiency of the FSV-rep model compared to other voxel models can be demonstrated via a simple quantitative comparison. For a single-level basic voxel model, the grid resolution N is inversely proportional to the permitted error. As the associated memory requirement is of the order of $\mathbf{O}(N^3)$, the memory load quickly becomes unbearable if the permitted error keeps reducing. Figure 3-17 plots the memory requirement of a basic voxel model using 1 bit per voxel with decreasing error limit. It is clear from the figure that for a basic voxel model, it becomes exorbitant to further reduce the error due to the huge memory cost. The FSV-rep model, which uses sparse and multi-level voxel representation, is able to reach a desired grid resolution with a much less memory requirement.

For a cubical modeling space of a fine grid resolution of 1,024, a basic voxel model will need 128 MB. Instead, an FSV-rep model with a coarse grid resolution of 256 and a fine grid resolution of 1,024 (a sub-division factor of 4) will only need 2 MB for the coarse grid voxel representation along with additional memory for the fine-grid voxel sub-space within each surface voxel of the coarse grid. For the sub-division factor of 4, each of these voxel sub-spaces will need only 8 bytes for the bit field, 8 bytes for a pointer to the bit field, and 4 bytes for holding the coarse voxel id, totaling 20 bytes. Even with 1 million coarse surface voxels, it will only use 19.07 MB memory. This yields a significant reduction in memory usage by a factor of

6 for the FSV-rep model compared against the basic voxel model.

Since the error levels of FSV-rep reduces drastically with the addition of FC-points for each fine level surface voxels, the resolution of basic voxel models will need to be excessively high as indicated by the trend in Figure 3-17 to provide comparable accuracy. Thus, it is irrelevant to compare memory requirement of FSV-rep model of a workpiece and a basic voxel model that can provide comparable accuracy. The only conclusion we shall make is that basic voxel models cannot produce models with acceptable visual accuracy.

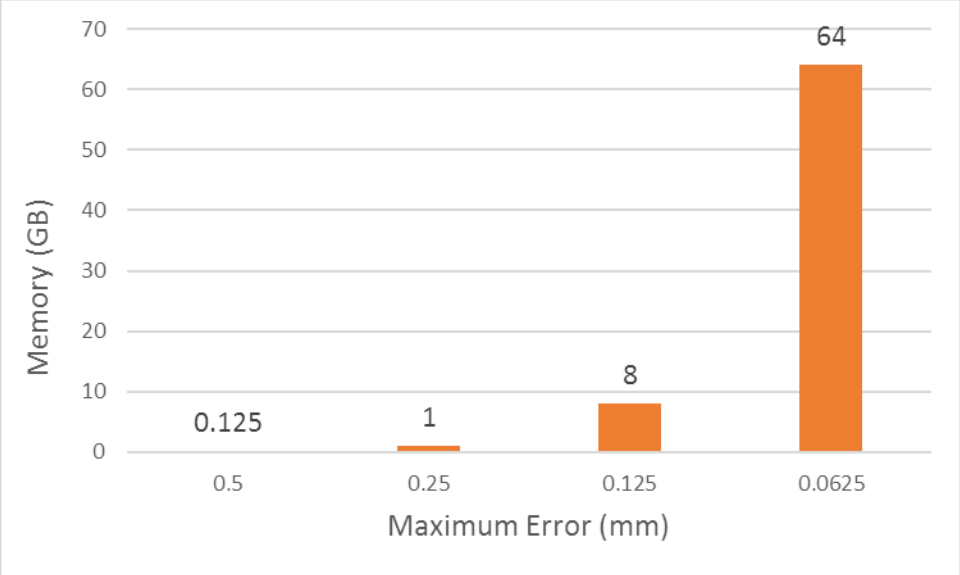


Figure 3-17 Memory requirement of a basic voxel model with decreasing error limit for a cubical modeling volume of 1,024-mm side-length.

More importantly, after reaching a limiting fine-grid resolution, the FSV-rep model uses the FC-points to define the FS-voxels which then lead to a reconstructed triangle mesh surface with much better accuracy as presented earlier. It is far superior to the common octree subdivision within the fine-grid surface voxels to improve the accuracy. For the basic voxel representation using the crossing edge mid-points for surface reconstruction, the accuracy can be

quantified as half of the voxel side-length. A simple case is shown in Figure 3-18 with the smallest octants created with 20-micron side-length within a fine-grid surface voxel of 1.28-mm side-length in order to obtain 10-micron accuracy. A quantitative evaluation can be made about the number of octants needed in this simple case. In order to reach the smallest octants across the black plane boundary within the cubical volume considered, 10,920 octants has to be created for the whole octree from the root to leaf octants. Computing the FC-points instead for the cubical volume in this case would involve only 4 point location parameters on the 4 intersecting edges of the voxel's edge-frame. This clearly shows that the FSV-rep model will be much more memory-efficient than an Octree subdivision method for improved accuracy.

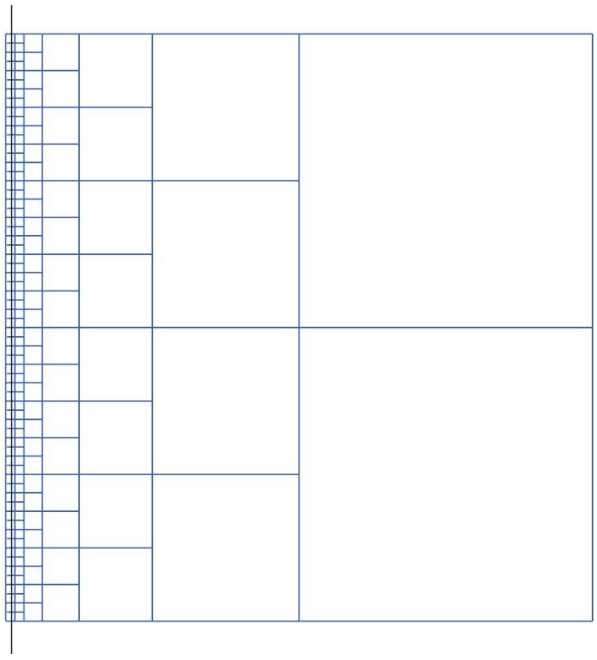


Figure 3-18 Octree subdivision to achieve a 10-micron accuracy in a cube of 1.28-mm side-length.

As shown in Table 1-1, apart from voxel based models, B-rep solid modeling and vector modeling have also been used for machining simulation. Out of computational performance limitations, B-rep is less preferred than the other classes of methods. Among the vector modeling methods, tri-dexels is a robust and accurate enough method for general milling cases.

FSV-rep is seen to be superior among the voxel space partitioning methods. FSV-rep in fact gives similar level of accuracy as tri-dexels. This is because both methods are able to sample the object surface at the intersection points with the grid lines. In case of tri-dexels, the dexels are along the grid lines and its end points are on the object surface. In case of FSV-rep, the voxel edges are along the grid lines and the FC-points on the edges of each FS-voxel are also on the object surface. Given an object, and the tri-dexels and FSV-rep models created for it, for each dexel that spans across at least one grid point (a point where three orthogonal grid lines intersect), the end points will have their corresponding FC-points in one of the FS-voxel of the FSV-rep model. For dexels shorter than the fine voxel edge of FSV-rep and not crossing any grid point, the corresponding frame-edge segments in the FSV-rep model may get ignored or fused with another frame-edge segment as shown in Figure 3-13 and described in Section 3.8.4. This can cause some dexel end points to have no corresponding FC-points but it is statistically insignificant as seen from our extensive computational tests. For the cases shown in this section, only 1 out of 54,128 dexel end points in Case 3 and 3 out of 294,562 dexel end points in Case 4 are missing the corresponding FC-points in the FSV-rep model.

The tri-dexel model size in terms of number of dexel end points has been reported to be roughly of the order of $k_T \times \mathbf{O}(N_F^2)$ where N_F is the fixed grid resolution, same as the finest grid resolution of FSV-rep, and k_T is a factor similar to k in Equation 3.9 [73]. $k \cong k_T = 6$ is true or an ideal case of an axis-aligned cube just fitting in the tight modeling space. In order to store the tri-dexel model of the cube in the ideal case using 4-byte long floating point number per end point and an $N_F \times N_F$ array of dexel list pointers per coordinate axis direction, this results in $M_T = 48 \times \mathbf{O}(N_F^2)$ bytes of memory. The effective memory usage of an FSV-rep model for the same cube using Equation 3.9 with a sub-division factor $f = 4$ gives,

$M_F \cong \left(\frac{N_F}{512} + 172.5\right) \times \boldsymbol{O}(N_F^2)$ bytes.

For all practical cases

$\frac{N_F}{512} < 10$, implying $M_F \cong 182.5 \times \boldsymbol{O}(N_F^2)$.

Thus, $M_F : M_T \cong 3.8$ is the approximate ratio of tri-dexels and FSV-rep memory usage for the ideal case. This is indicative of the ratios observed for all the practical cases in this section as listed in Table 3-2. Table 3-2 contains an additional Case 5 which is a $500 \times 500 \times 300$ mm^3 rectangular block, included to show the scalability of the methods in terms of memory usage. It is clear that even though FSV-rep takes more memory than tri-dexels, they are both of the same order of magnitude of $\mathbf{O}(N_F^2)$.

Table 3-2 Comparison of memory usage for FSV-rep and tri-dexels.

Case Study	Memory (MB)		Ratio (FSV-rep/Tri-dexels)
	Tri-dexels	FSV-rep	
1	0.57	1.68	2.95
2	0.56	1.63	2.91
3	0.56	1.60	2.86
4	3.53	14.80	4.19
5	10.49	32.79	3.12

The higher memory usage of FSV-rep compared to tri-dexels is justified by its significantly higher computational speed in model update which will be demonstrated in Chapter 7.

3.10.2 Memory usage for display

As for memory usage for display, the FSV-rep model is superior to other voxel-based

models and equivalent to the vector-based tri-dexel model. Direct voxel display without generating an envelope surface representation will have to render a large number of elements proportional to N^3 in the case of basic voxel models and proportional to N^2 in the case of hierarchical voxel models using the octree subdivision. Here, N is the grid resolution for the overall modeling space in each case. To achieve display quality comparable to that of an FSV-rep model, N will have to be much higher for these traditional voxel models as discussed earlier in this section. This will certainly affect the memory usage for the display in a similar manner. If a triangle mesh surface is generated from these traditional voxel models for improved visualization quality, the memory usage for display then becomes that of the triangle mesh being generated.

For triangle mesh based models, the model size is in general proportional to the number of triangles in the model. For FSV-rep, the number of mesh triangles generated is proportional to the number of FC-points. For traditional voxel models, the number of mesh triangles generated is proportional to the number of voxel edges crossing the modeled object surface. The involved FC-points and crossing voxel edges have essentially the same count for the triangle mesh generation. For the case of a tri-dexel model, the number of mesh triangles generated is proportional to the number of dexel end points. As identified earlier in the section, the number of dexel end points is almost the same as the number of FC-points in the corresponding FSV-rep model. As a result, the memory usage for display is deemed equivalent for all of the above methods for a given fine grid resolution when a triangle mesh is employed for display.

3.11 Summary

The FSV-rep workpiece geometry modeling method developed in this chapter is able to

combine the efficiency of multi-level sparse voxel models and the accuracy of boundary representation approaches such as triangle meshes. Thus we can have model update that are efficient irrespective of the surface complexities and achieve surface representation accuracy close to that possible by triangle mesh based B-rep models. In the following chapters, the efficient way of updating an FSV-rep model and its use for milling simulation is developed.

Chapter 4: Three step FSV-rep model update

In this chapter a general efficient method to update an FSV-rep model to reflect volume removal by a set of abstract volume removal tools is developed. Here "tool" stands for "Boolean tools" that update an FSV-rep model rather than just a "milling cutter".

4.1 Objective

Geometry of the final machined part can be accurately and efficiently computed via a three-step update process for the involved FSV-rep model. The coarse level voxels are to be updated first to quickly remove the bulk of unwanted material from the workpiece model for a given set of tool paths. The fine level surface voxels and their corresponding FC-points are then updated in sequence in order to attain the desired model accuracy without much added computational load.

4.2 Coarse update

The coarse voxels in an FSV-rep model are used for quickly removing the bulk volume from the workpiece. Such an update only needs to be approximate and the objective is to ensure that all the coarse voxels are classified (or marked) properly.

Using set of Boolean volume removal tools $\{\boldsymbol{T_{rem}}\}$ for the model update, the coarse level voxels can be classified as:

(I) definitely inside a T_{rem} (black in Figure 4-1);

(II) definitely outside all the T_{rem} (ash in Figure 4-1); or

(III) possibly intersecting the envelope surface of a T_{rem} (green in Figure 4-1).

The coarse update step will delete the category I voxels from the FSV-rep model and mark the

category III voxels as the near-field (NF) voxels for each corresponding T_{rem}. The category II voxels shall be untouched and remain in their current state.

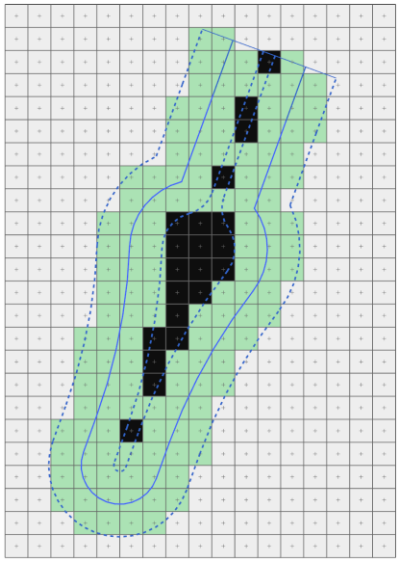


Figure 4-1 An abstract object in a voxel space with voxels classified as Near-field (green), Inner-field (black) and Outer-field (ash) based on the voxel centre point location with respect to the object surface.

In addition, the coarse update will classify voxels inside the workpiece as category I, II, or III and then perform the associated marking/unmarking operations in order to reflect the removed volume from the workpiece. The volume voxel model at coarse level can be used efficiently for this purpose. Technically the FSV-rep model has only the surface voxels as part of it. However the coarse level voxel grid is defined for the entire modeling space. Thus converting the coarse level surface voxel model bit representation does not have any additional memory cost. Also with the surface voxels already set active, the inner volume voxels can be identified easily by a linear scan along the voxel space taking one coordinate axis direction as the scanning direction. From every odd surface voxel encountered to the next even surface voxel, the voxels in between can be activated. Doing so for all the stacks of voxels along the X axis of the voxel space at every grid point on YC-ZC coordinate system plane perpendicular to X axis, for

instance, the volume voxel model at the coarse level can be obtained for the update purpose. Without the volume voxel model at the coarse level, the workpiece interior coarse voxels can still be obtained, but with more involved search methods.

The category III NF-voxels identified from the workpiece interior contain the surface voxels for the newly machined surface whereas the NF-voxels identified from the already existing coarse surface voxels are those which are just partially affected. In fact the NF-voxels identified from such voxels contain the boundary between the newly machined surface and the unaffected workpiece surface regions. A relaxed proximity check of the voxel center against the Boolean volume removal tool T_{rem}'s envelope surface is used to identify the NF-voxels. If the voxel side length is L_{vc} and the distance of the voxel center from the surface is d_{te}, a voxel is deemed to be an NF-voxel for the tool T_{rem} if

$$d_{te} < \frac{\sqrt{3}L_{vc}}{2} \tag{4.1}$$

where $\sqrt{3}L_{vc}$ is the "thickness" of the near-field region for the tool envelope which sits in the middle of the near-field region, which is between the offset surfaces in Figure 4-2.

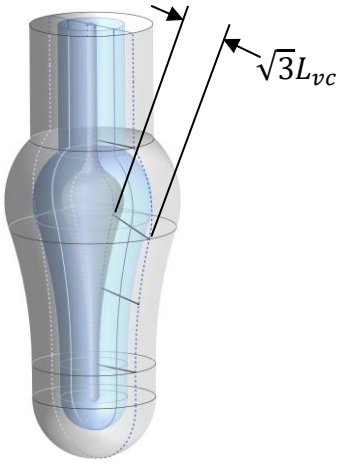


Figure 4-2 An abstract object with surfaces bounding the Near-field region.

The overall coarse update step is outlined in the Algorithm 4-1 below. The coarse update step is done in two parts. In Part 1, all the interior voxels interior to the Boolean tools are deactivated as not part of the workpiece volume. Then in Part 2, a list of near-field voxels **{NFV}** is created for each Boolean tool T_{rem} according to Equation 4.1. After the two parts are done for all the Boolean tools, the coarse voxels along the final machined part surface are present in the resulting **{NFV}** sets. It should be noted that Part 2 of the algorithm should be done after Part 1 for all the Boolean tools in order to avoid a coarse voxel from entering into the **{NFV}** of a Boolean tool if some subsequent Boolean tool will deactivate it. Such voxels do not need to be considered as they will not contribute to the final machined part surface geometry.

Algorithm 4-1. Coarse Update

```
//Part 1
{T_rem} ← set of abstract Boolean volume removal tools
for each T_rem in {T_rem} do
   {BV} ← coarse voxels inside/intersecting T_rem's bounding box
   for each V in {BV} do
      if V active and V completely inside T_rem then
         Set V inactive
      end if
   end for
end for
//Part 2
for each T_rem in {T_rem} do
   {BV} ← coarse voxels inside/intersecting T_rem's bounding box
   for each V in {BV} do
      p ← center point of V
      if V active and p inside near-field of T_rem's envelope then
         {NFV} ← {NFV} + V
      end if
   end for
end for
```

After part 1 of Algorithm 4-1, the intermediate FSV-rep model will be as shown in right side of Figure 4-3 with all the category I coarse voxels with respect to any of the volume removal tools deleted.

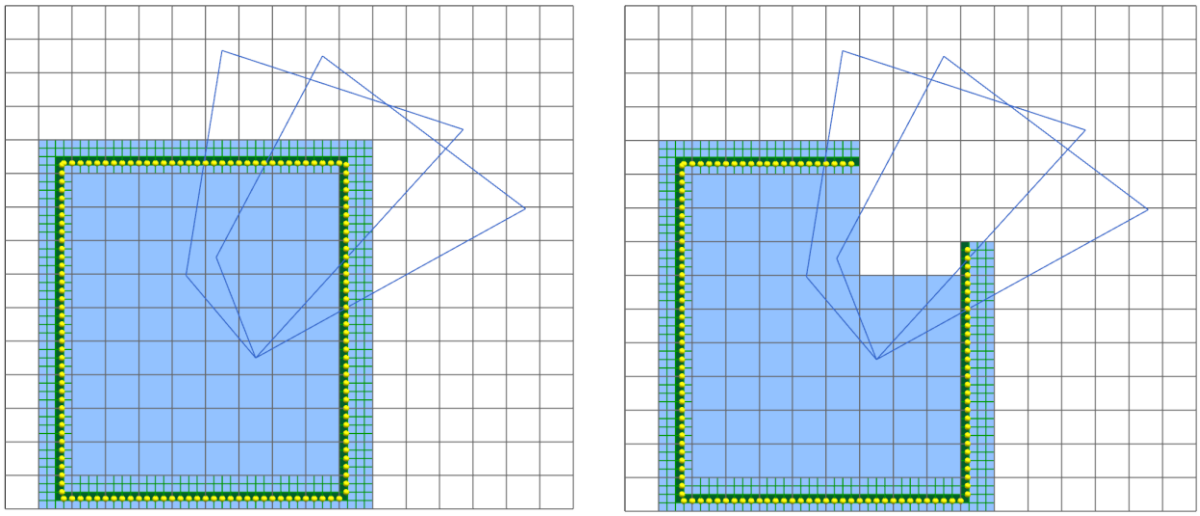


Figure 4-3 Coarse update with a set of two abstract tools for a FSV-rep workpiece. Initial FSV-rep on left and coarse updated FSV-rep on right.

Also for the coarse update, even though the coarse voxels across the entire bounding box of each of the volume removal tool T_{rem} has to be considered in Algorithm 4-1, it will not a critical concern for efficiency. This is because, the voxels associated with the bounding boxes are directly identifiable from the bounding box corner points and the additional operations to perform to classify the voxels as Category I, II and III are simple point classification operations. For the typical tools involved in machining simulation these operations are very simple as we will see in Chapter 5.

An important output of the coarse update step is the mapping of a specific T_{rem} to its corresponding **{NFV}** as shown in Figure 4-4. Part 2 of Algorithm 4-1 creates this mapping and it can now be used for the fine update step. Further to note, a given coarse voxel can be present in the {NFV} list of more than one T_{rem}.

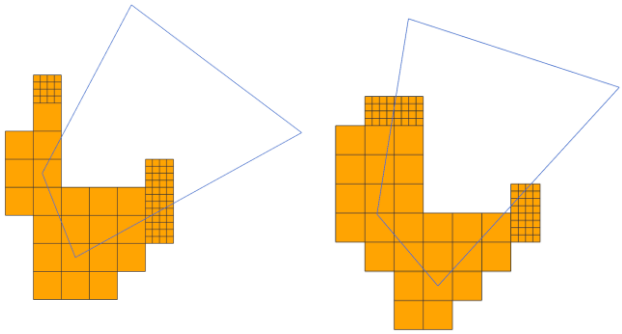


Figure 4-4 The abstract tools with NF-voxel collection for each identified after coarse update.

Recall that the coarse update is a collective voxel removal at a lower resolution. Since the model update is done at the lower resolution, the involved computational time is not linked to the accuracy-dependent finest voxel resolution. The coarse update time is only dependent on the resolution for the coarse voxel level and the number of Boolean tools. More specifically, if the ratio of the fine to coarse voxel size is 1:4, each coarse voxel removal from the model corresponds to 64 fine voxels. Thus, voxel removal at the coarse level has 64 times fewer number of voxels to consider than at the fine level. This will be the case for all the voxels falling in category I of the coarse voxel update. Thus, the majority of the coarse update is performed at an execution time faster by a cubic power of the subdivision factor from the coarse to fine levels when compared to the update of a voxel model with a single fine-level voxel grid.

It is also important to have the multi-level voxel model starting at a coarse resolution governed by the tool dimensions rather than a complete octree subdivision of the entire modeling space. This is because only those levels with voxel sizes smaller than the cutting tool size will contribute to fast removal of the model volume. In particular, only voxels completely inside the cutting tool will be marked as removed and this will not happen for any higher levels with the

voxel size larger than the tool size. Thus, if an octree model is used, even if the model size is overall smaller, all the octants in it will have to undergo subdivision to a particular level during the simulation adding extra computational load for simulation. This justifies the use of coarse level surface voxel space as the top most level for FSV-rep models.

4.3 Fine update

From all the $T_{rem} \rightarrow \{\boldsymbol{NFV}\}$ mappings obtained after the coarse update, inverse mappings of $NFV \rightarrow \{\boldsymbol{T_{rem}}\}$ is obtained between each coarse NF-voxel and the set of volume removal tools possibly crossing it.

Then, in the fine update step for each coarse NF-voxel, the finer voxels within each coarse NF-voxel have to be classified with respect to each item in $\{\boldsymbol{T_{rem}}\}$ as: (I) definitely inside (ash voxels in Figure 4-5); (II) definitely outside (white voxels in Figure 4-5); or, (III) intersecting the tool instance envelope surface (blue voxels in Figure 4-5).

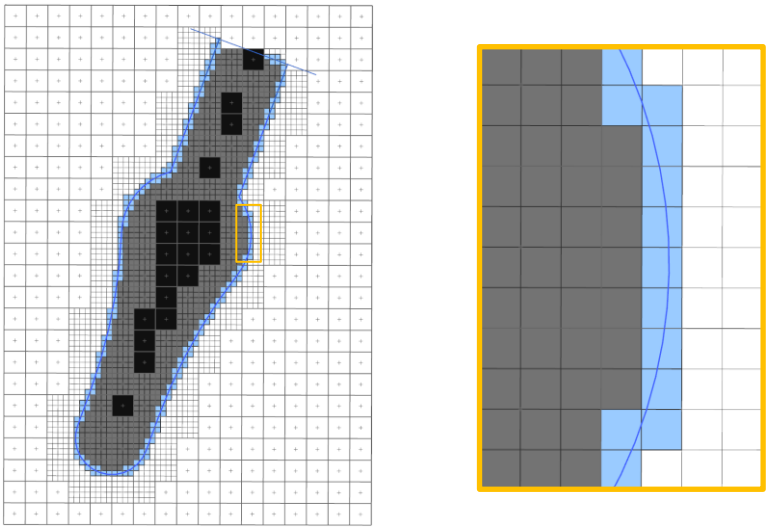


Figure 4-5 The abstract object in voxel space with actual surface voxels identified (blue).

Algorithm 4-2 outlines the overall fine update step.

Algorithm 4-2. Fine Update

```
NVR ← number of coarse voxels to refine
for i from 1 to NVR do
   CV ← ith coarse voxel to refine
   {T_rem} ← the volume removal tools crossing CV
   {FV} ← fine-level voxels within CV
   for each FV in {FV} do
      p ← center point of FV
      for each T_rem in {T_rem} do //Part 1
         if FV active and FV an interior voxel for T_rem then
            Set FV inactive
         end if
      end for
      for each T_rem in {T_rem} do //Part 2
         if FV active and p inside near-field of T_rem's envelope then
            if FV a surface voxel for T_rem then
               Add FV to surface voxel model
            end if
         end if
      end for
   end for
end for
```

Like in the coarse update step, category I of the fine voxels is removed from the model (Part 1 of Algorithm 4-2 above) and category II voxels are untouched. Category III voxels are the new surface voxels for the updated model which are now identified as those voxels with confirmed intersection with the final machined part surface.

The category III voxels in Fine update are identified by exactly classifying every corner point with respect to the volume removal tool as inside or outside. For this purpose, a signed distance value [68], d_i, from the tool surface for each corner point can been used. A voxel is

deemed as category III only if the condition,

$$-8 < \sum_{i=0}^{7} \frac{d_i}{|d_i|} < 8$$

is satisfied.

As mentioned in Chapter 3, Section 3.3, some voxels through which the tool envelope surface passes are missed in this definition of surface voxels. Those are the voxels for which the envelope surface either penetrates the faces of the voxels alone or just trims the interior portion of the voxel frame edges. This is acceptable as such intersections do not add any useful information to the FSV-rep model. Without such voxels itself the model is 26-separating. Further, for the voxels with face only penetration, there is no slicing loop or slice-front formed to define an FS-voxel. For edge-interior only trimmings, this could change the set of FC-points for the FS-voxel. However, as we will see in Chapter 8, such gaps are eventually closed for generating a surface approximation efficiently. Thus, in order to keep the fine update process also efficient, identification of these voxels is not attempted.

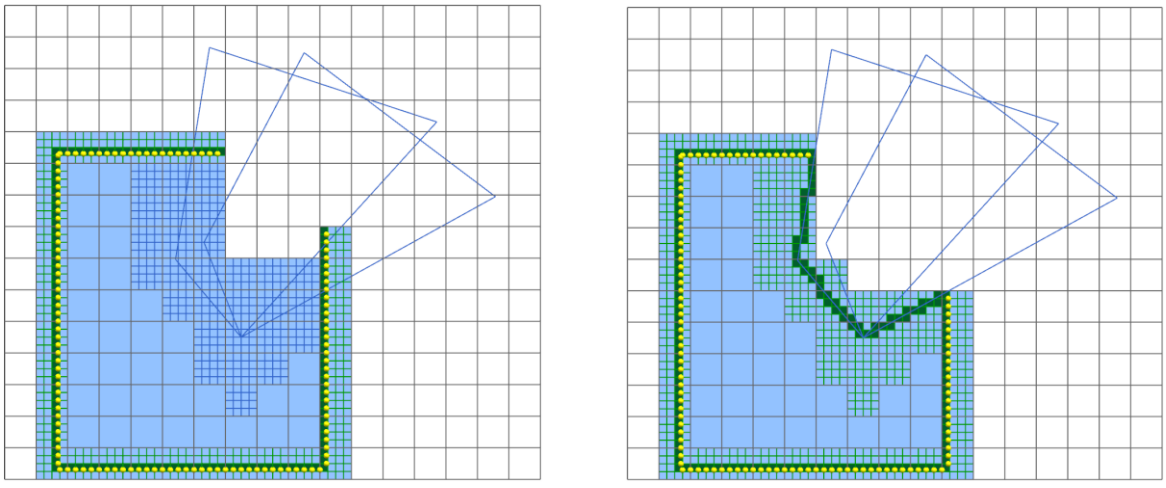


Figure 4-6 Fine update process with the abstract tools on the coarsely updated FSV-rep (left) and 26-separating fine level surface model obtained (right).

The FSV-rep model is updated (Part 2 of Algorithm 4-2 above) with the new surface voxels to get the machined part geometry represented as a surface voxel model at the finest resolution (Figure 4-6). It should be noted that the surface voxel model obtained after the fine update step is 26-separating. A 26-separating voxel model is one that ensures all the voxels which an object surface passes through by crossing the edge-frames of the voxels is part of the model. This facilitates the generation of a closed 2-manifold triangle mesh for the desired model accuracy as will be shown in Chapter 8.

4.4 Frame update

Once all of the surface voxels at the finest resolution are obtained, frame update has to be done to these fine surface voxels in order to create the FS-voxels. The FS-voxels are created by computing the FC-points for all the fine surface voxels. For a fine surface voxel edge, all the tools intersecting it can be retrieved from the $NFV \rightarrow \{\boldsymbol{T_{rem}}\}$ mapping. Line-surface intersection points are then calculated between each voxel edge and the tool item in $\{\boldsymbol{T_{rem}}\}$ as a set of potential FC-points. From that set, those intersection points not inside of any tool item are the actual FC-points on the surface of the final machined part (Figure 4-7).

For each FC-point, its location on the voxel edge has to be stored in a way to facilitate the subsequent reconstruction of the closed 2-manifold triangle mesh surface from the FS-voxels. The FSV-rep model uses a pair of point locations on each voxel edge to define the FS-voxels. An FC-point is to be stored as the first or second point in the pair according to the surface normal of the generating tool instance. If the surface normal component along the voxel edge is positive, the FC-point is stored at the second location in the pair and at the first location if the surface normal component is negative. This ensures proper orientation of the triangles in the

reconstructed triangle mesh surface.

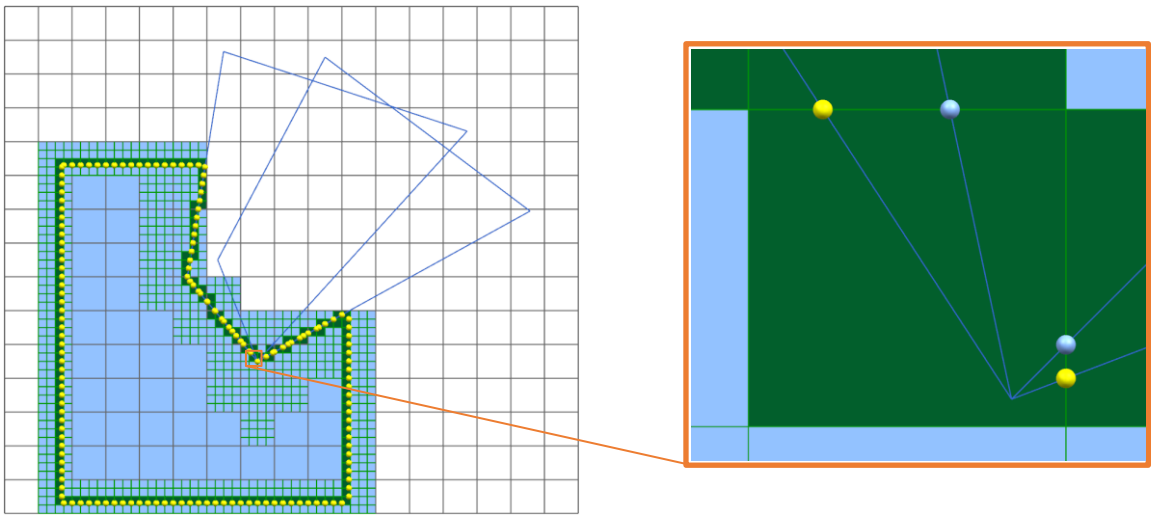


Figure 4-7 Frame-update step creating FC-points (yellow spheres) within the fine-level surface voxels. Zoomed in view shows the set of intersection points on voxel edges from intersecting tool items (yellow or pale blue).

4.5 Summary

The three-step update logic developed in this chapter is a generic approach that can be used to obtain the FSV-rep model of a part created by material removal from an initial blank workpiece. It has the quality to perform the model update in three steps utilizing the coarse, fine and frame levels of the FSV-rep model. The coarse update with all the volume removal tools before fine and frame updates helps fast model update without much influence by overlapping tools performing bulk volume removal. Further the frame update to achieve sub-voxel accuracy via FC-points involves simple line-surface intersections for the frame edges of already identified fine surface voxels.

Chapter 5: Tool swept volume representation

In the previous chapters we have developed FSV-rep, a new modeling method for machining simulation, and a three-step update process to update an FSV-rep model with a set of volume removal tools. The tool paths used for milling operations using the milling cutters has some characteristic features that can be used to develop specific models of the volume removal tools to update the FSV-rep workpiece in such cases. In this chapter we will consider various categories of milling cutters and tool paths and develop suitable representation of the volume removal tool models.

5.1 Requirement

Tool path sampling which takes tool instances along a tool path at a regular interval is one option to update the in-process workpiece in milling simulation and has been previously used to generate approximate cutter swept volumes [16,74]. Other approaches such as solid modeling [75, 76], two-parameter family of spheres [19] and analytical definitions [28] also have been used to obtain the cutter swept volume and update the in-process workpiece. Nonetheless, this work employs the method of sampled tool instances for general cases as it is a generic method that is applicable to all type of tools and tool paths and comparatively simple to implement for the voxel model update. And for the simple swept volumes, a more suitable representation named as Swept volume regions (SVRs) is developed that can provide faster and more accurate results in FSV-rep update.

The use of sampled tool instances is beneficial for tool paths involving combined translation and rotation of tool axis. For such tool paths, the swept volume definition as a boundary representation using envelope surfaces is complicated. In fact the intersection of voxel

frame edges with envelope surfaces of such swept volumes do not always have closed form solutions. Thus from practical point of view, the usage of sampled tool instances is the optimal option for such tool paths.

However, for a specific set of simple but widely used categories of tool paths, the envelope surfaces are available as surfaces with closed form solutions for their intersection with the voxel frame edges. Thus the boundary representation of the swept volumes for such tool paths can be directly used for FSV-rep workpiece update without sampling the tool paths. There are two benefits from doing so.

(1) First, it will effectively reduce the number of intersection operations needed per voxel frame edge involved in the frame update step. Instead of updating the frame edge using all the sampled tool instances crossing it, intersection with the envelope surfaces of the swept volume alone will be enough.

(2) Second, the FC-points obtained will be exact for the surfaces created by such tool paths. This is possible as the "sampling scallop" that would occur from the use of sampled tool instances is not an issue anymore for update with the swept volume itself.

5.2 Selection of tools swept volume representation

The beneficial use of swept volume for FSV-rep update is possible in case of all straight cutting tool paths up to 3-axis machining. It is possible in case of planar arc tool paths also to update FSV-rep directly with the swept volume as there exist closed form solution for intersection of frame edges with all the envelope surfaces possible in these cases.

The swept volume B-rep is applicable to certain other cases like tool paths with pure rotation of the tool axis as well as some tool paths with combined rotation and translations of the

tool path but with no translation along the screw axis. However these are not significant categories to consider with not much operations involving such tool paths. The general 5-axis milling tool path definitely have envelope surfaces that do not have closed form solution for intersection with the voxel frame edges. Thus for all these cases a general sampled tool instances along the tool path is decided to be used as the approximation.

The flow chart in Figure 5-1 below shows the suggested choice of tool swept volume representation based on the analysis above.

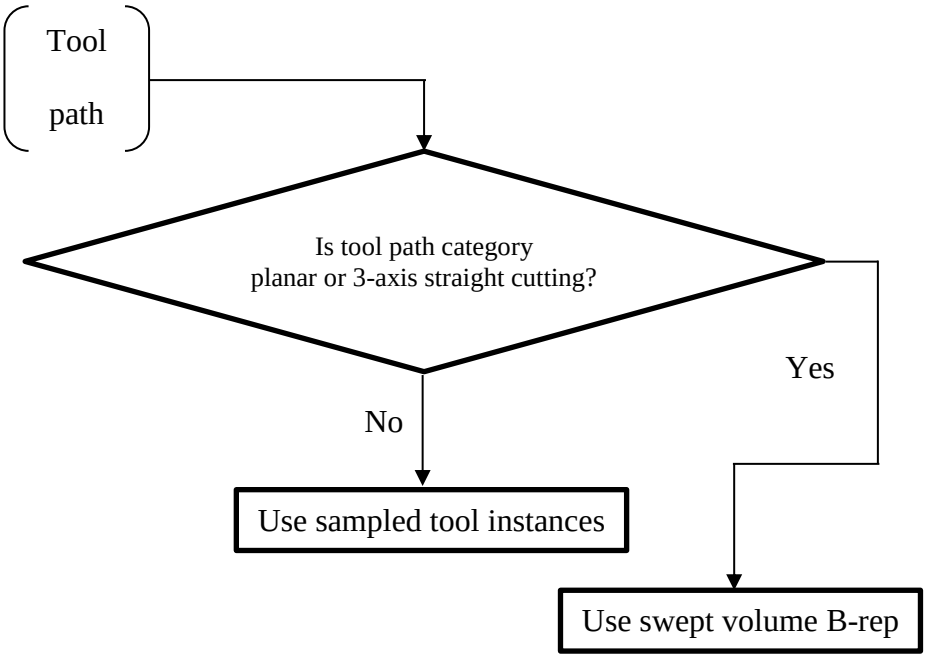


Figure 5-1 Decision diagram to select the swept volume representation based on tool path category.

For uniformity of the initial concepts developed the circular arcs are not included above for swept volume B-rep. It should be possible to extend the concepts developed to such paths as well in the future. The following sections provide the model of the sampled tool instances and the specific definition of tool swept volume as Swept Volume Regions (SVRs) for use in the update of FSV-rep with the various types of tool paths as they are fit for.

5.3 Tool instances

An easy-to-use cutter envelope formulation utilizing the distinct features of a milling cutter is employed in this work to facilitate machining simulation with an FSV-rep workpiece model. For general voxel based modeling, to perform a model update according to a Boolean operation between a target model and a tool model, the approach is to derive the voxel representation of the tool and perform a voxel space Boolean operation for the target voxel model. However, for a milling cutter, it is axis-symmetric with a parametric profile of revolution commonly defined by the APT specification [57] as shown in Figure 5-2.

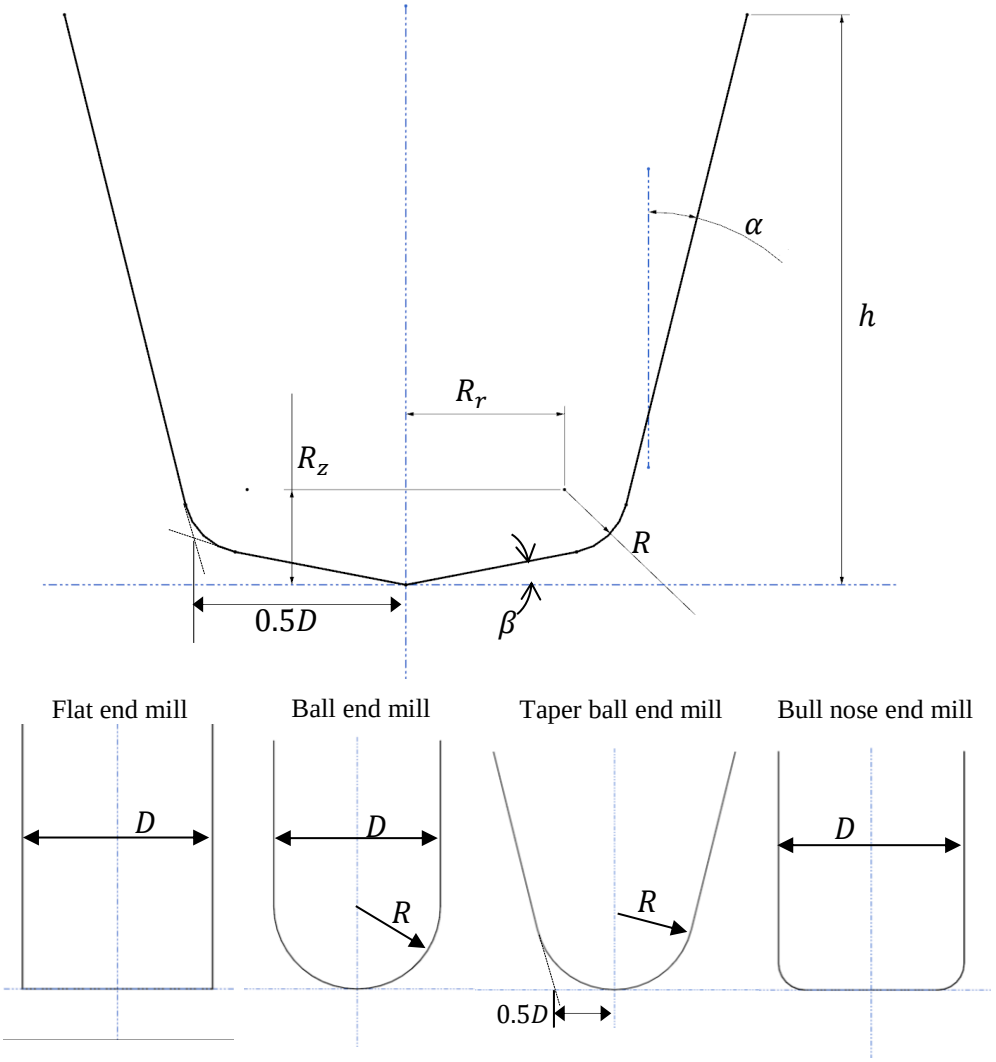


Figure 5-2 General and specific milling cutter profiles with major dimensions.

This regularity of the cutting tool profile can be efficiently used without the need to generate an explicit voxel model for the tool at every tool location along the tool path. The FSV-rep workpiece model voxels can be directly classified with respect to the tool surface envelope using simple algebraic expressions that involve the classification of voxel center and corner points in relation to the tool axis and cutting profile. Further, an axis-aligned bounding box of the milling tool can be used to reduce the search for the nearby model voxels to consider.

The basic operations to perform using the tool instances are voxel center and corner point classification and intersection of the voxel frame edges with the tool envelope surfaces.

Point classification

For a tool instance with tool location at $P(x_p, y_p, z_p)$ and axis orientation along $A_{dir}(i_a, j_a, k_a)$, the tool co-ordinate system $[C]_T^M$ is defined by Equation 2.2 (the parameter *t* is omitted here and Model coordinate system *M* is used in place of Global coordinate system *G*).

The point to be classified $P(x_p, y_p, z_p)$ can be transformed into the Tool coordinate system from the Model coordinate system as below:

$$P^T = [C]_T^M \times P^M \tag{5.1}$$

If *L* is the perpendicular distance of the point from the tool axis given by,

$$L = \left(\left(y_p^T\right)^2 + \left(x_p^T\right)^2 \right)^{\frac{1}{2}} \tag{5.2}$$

The point can be classified as follows:

$$P_{location} = \left\{ \begin{array}{cc} inside, & if\ L < R\left(z_p^T\right) \\ outside, & if\ L > R\left(z_p^T\right) \\ on\ the\ tool\ envelope\ surface & otherwise \end{array} \right\} \tag{5.3}$$

were $R(z_p^T)$ is the tool radius at height of $z' = z_p^T$ along the axis which is calculated for the

selected tools on a case specific basis in Appendix A.

Voxel classification

Similar to a point, a voxel as a whole also has to be classified according the volume removal tools as we saw in Chapter 4, Sections 4.2 and 4.3. Specifically, we are interested in classifying the voxels as near/inner/outer field voxels during the coarse update. Also in the fine update step we require further classification as surface voxels or not for the near-field voxels. Identification of a voxel as surface voxel is possible using the above point classification approach performed on each of the voxel corner points. A class value, $c_{val,i}$ can be given to the ith corner point according to the following rule:

$$c_{val,i} = \left\{ \begin{array}{ll} -1, & if\ point\ inside \\ 1, & if\ point\ outside \\ 0, & otherwise \end{array} \right\} \tag{5.4}$$

Then the voxel is a surface voxel if the following condition holds:

$$-8 < \sum_{i=0}^{7} c_{val,i} < 8 \tag{5.5}$$

In order to identify the field of the voxel as near/inner/outer field, a more involved check is required as shown below:

The idea of the near field is defined to swiftly collect a very good candidate set of voxels such that all the surface voxels are present in the set and the set size is minimal. In this work, near-field voxel identification based on the consideration of voxel center point alone is attempted. The condition followed is to consider any voxel as a near-field voxel if its center point's perpendicular distance from the tool envelope surface is less than the maximum possible value for that distance. The maximum possible distance of the center point, $d_{max,cp}$ is half of the

body diagonal,

$$d_{max,cp} = \frac{\sqrt{3}L_v}{2} \tag{5.6}$$

Now we define a voxel as a Near-field voxel if the perpendicular distance of its center point from the tool envelope surface, $d_{\perp,cp}$ is less than or equal to $d_{max,cp}$,

$$d_{\perp,cp} \leq d_{max,cp} \tag{5.7}$$

$$d_{\perp,cp} \leq \frac{\sqrt{3}L_v}{2} \tag{5.8}$$

The value of $d_{\perp,cp}$ is thus required to be computable for all the tools considered for a given voxel space. Again the coordinates of the voxel center point in the tool coordinate system, P_{vcen}^{T} is obtained from the point coordinates in the model coordinate system using the Equation 5.1. Then the expression for $d_{\perp,cp}$ in case of different tool types considered is listed below in Figure 5-3. Since the center point is transformed to tool coordinate system, this classification is easy for any tool orientation and thus application for multi-axis milling tool paths.

A voxel identified as not a near-filed voxel can be further classified as outer-field or inner-field by considering the center point classification as interior or exterior. If class value for the centre point P_{vcen}^{T} is c_{val}, a voxel is inner-field if,

$$c_{val} = -1 \text{ and } d_{\perp,cp} > d_{max,cp} \tag{5.9}$$

and it is outer-field if,

$$c_{val} = 1 \text{ and } d_{\perp,cp} > d_{max,cp} \tag{5.10}$$

Also, it is to be noted that the classification of a voxel as inner-field is only applicable provided the voxel is small enough to be completely fitted within the tool volume in any

arbitrary tool orientation. Otherwise, it is only meaningful to classify the voxel as Near-field or outer-field.

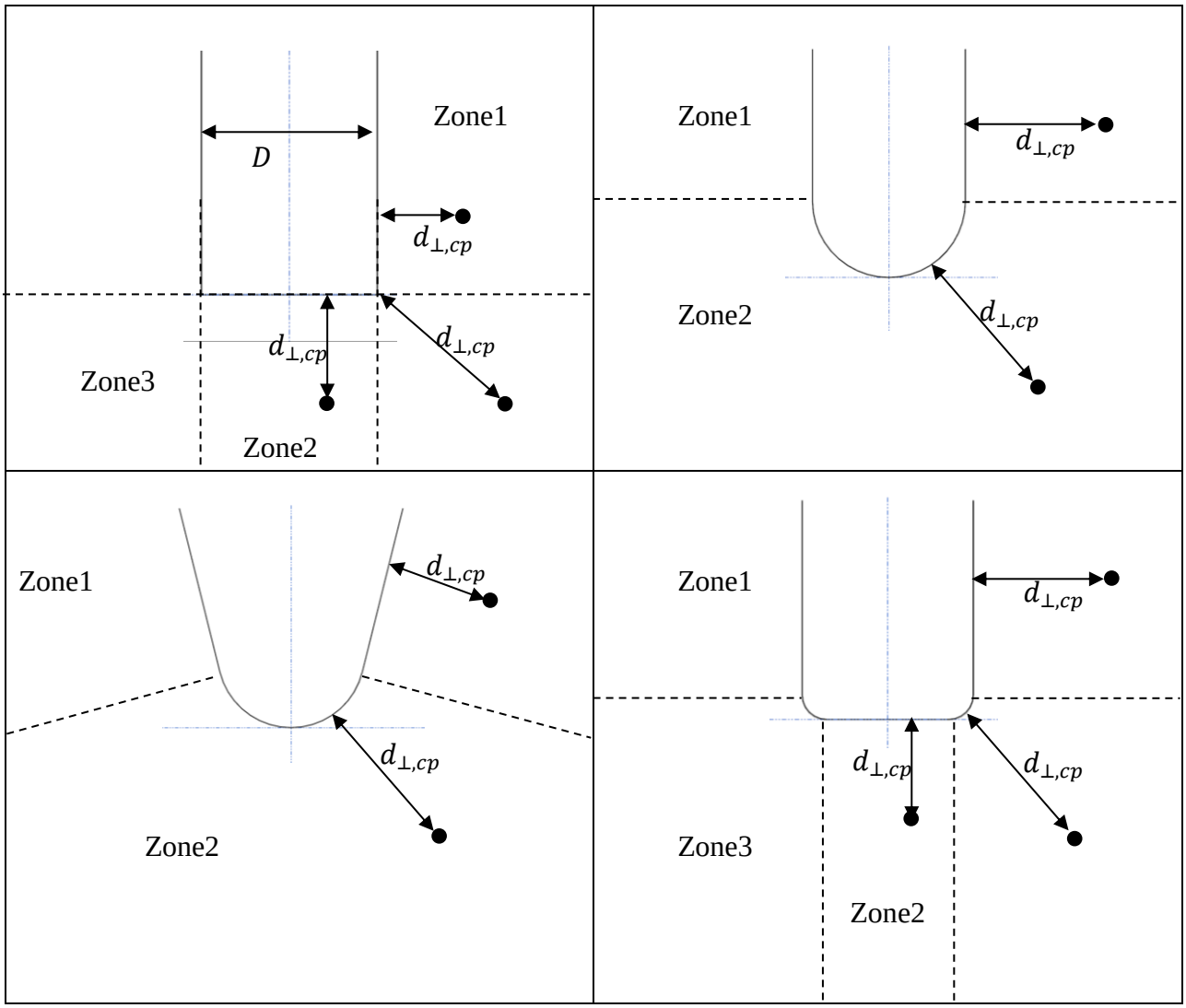


Figure 5-3 Projection distance for points in different zones for selected cutters.

Frame edge intersections

Actual intersection of the frame edges of the surface voxels with the envelope surfaces of the tool instances is required during the frame update step to create the final FS-voxels. Because

the tool instances are all volumes of revolutions of simple profile curves, the envelope surfaces are all primitive shapes with explicit expression for their intersection with the frame edges. Further, the frame edges in consideration are of a definite small size based on the fine voxel size. This can be easily used to identify which of the set of side and bottom envelope surfaces are intersecting with the frame edge in consideration. Following Figure 5-4 list out for the various tools in consideration, the potential envelope surfaces to consider for intersections with the frame edges based on the zone for the edge end points.

<table>
<tr>
<td>If one end point in zone1 and other inside*,
Intersect with side surface.
If one end point in zone2 and other inside,
Intersect with bottom.
Otherwise,
Intersect with complete tool

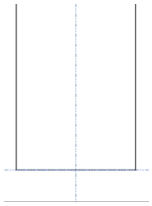
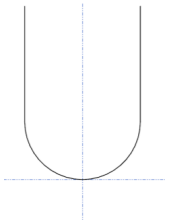
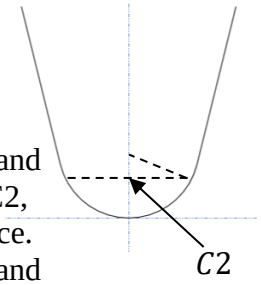
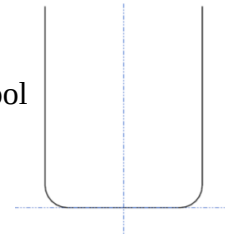
*inside implies “inside tool volume”
  </td>
<td>If one end point in zone1 and other inside above ball center,
Intersect with side surface.
If one end point in zone2 and other inside below ball center,
Intersect with bottom.
Otherwise,
Intersect with complete tool
  </td>
</tr>
<tr>
<td>If one end point in zone1 and other inside above point C2,
Intersect with side surface.
If one end point in zone2 and other inside below point C2,
Intersect with bottom.
Otherwise,
Intersect with complete tool
 
</td>
<td>If one end point in zone1 and other inside above torus center line,
Intersect with side surface.
If one end point in zone2 and other inside below torus center line,
Intersect with bottom.
Otherwise,
Intersect with complete tool
  </td>
</tr>
</table>

Figure 5-4 Selected tools and the intersection calculation needed for various types of voxel edges crossing it.

5.4 Swept Volume Regions (SVRs)

Using swept volume boundary representation directly is not the best possible option to update the FSV-rep workpiece. It will then require the operation similar to those that were involved in the generic update using the sampled tool instances. Those operations, the point classification with respect to the tool envelope volume and frame edge trimming again with the same are simple in case of a single tool instance. But for a swept volume boundary representation, the operations will be more involved as explained below.

For point classification, unlike the tool instance, the swept volume does not have an axis of symmetry in general. Therefore, the classification operation will require to consider all the envelope surfaces or at least the ruled surface generated by the tool axis swept along the path. More importantly, the later trimming operation of the voxel frame edges in order to update the FC-points becomes much more affected. For each frame edge the envelope surfaces intersecting it has to be identified. A simple swept volume boundary representation will need some additional operations to identify the candidate surfaces from all of its envelope surfaces.

Furthermore, in order to localize the workpiece voxels considered for update, the bounding box of the tool instance could be efficiently used. But for a tool swept volume with the path in 2 or 3-axis, the bounding box can become much bigger with longer tool paths. This will not be a problem for 1-axis tool paths as the bounding box is axis-aligned. But for 2-axis and 3-axis tool paths, the bounding box localization will be less effective with longer tool paths as many voxels entirely outside the swept volume will be considered. This can adversely affect the coarse update which has its efficiency greatly relying on the localization using the bounding box for bulk volume removal.

In order to address the above identified problems, we need a concept that is less intensive

than the sampling approach in terms of the number of volume removal tools involved but more effective in localization than a single swept volume boundary representation. For this purpose, we define the Swept Volume Regions (SVRs) with a given tool swept volume in a voxel space.

Swept volume of 1-axis and 2-axis linear and arc tool paths can be represented as boundary representation using envelope surfaces having parametric expressions as detailed in following sections. The boundary representation divides the surface of the swept volume into different areas characterized by the boundary elements present over there. Here boundary elements are the envelope surfaces, the edges formed by the intersection of these envelope surfaces and the vertices formed further by the intersection of these boundary edges.

For a general APT tool moving along a 3-axis linear tool path, there are 43 boundary elements to consider as shown in Figure 5.6. Out of these, 12 are surfaces, 21 are curves and 10 are points. In order to effectively use the swept volume B-rep elements for the FSV-rep update, intersection calculation of the voxel frame edges with all these surfaces has to be possible without iterative solution techniques. For the general end mill swept along a 3-axis linear tool path, 4 of the boundary surfaces are planar and 4 are conical. All of these surfaces are quadric or simpler and has reasonably simple calculation of intersection with a frame edge.

Apart from these, there are two toroidal surfaces and two envelope surfaces formed by the sweeping of the toroidal section of the cutter. These four surfaces have higher degree quartic surface representations. Still there are already available techniques in the literature with closed form solution for the intersection of these surfaces with line segments representing z-vectors [58, 77]. These techniques can be applied to voxel frame edges as well with minor modifications to apply them for the frame edges along the horizontal axes.

Thus, with the assurance that it is possible to perform direct intersections with all the

envelope surfaces of the swept volume B-reps in consideration, we now attempt to ensure these computations are done localized with the definition of Swept Volume Regions (SVRs). SVRs are defined with respect to the different boundary elements of the swept volume boundary representation as follows:

SVRs: The distinct portions of the swept volume overlapping with a set of voxels of the considered voxel space such that same boundary elements of the swept volume is present in all of these voxels and the number of these elements is the minimum possible.

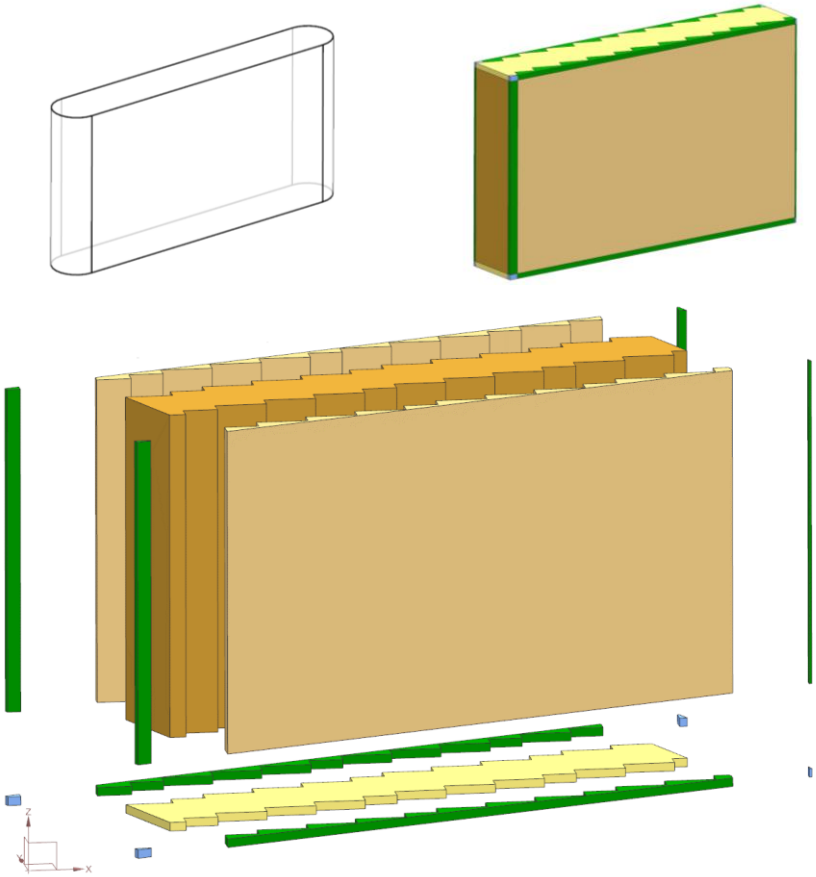


Figure 5-5 Top Left: Boundary representation of the swept volume in case of a linear 2-axis path with flat end mill. Top right: The partition of the swept volume into various regions (The outer half of the tool instances at the two ends are ignored and handled separately for simplicity). Bottom: An exploded view showing 14 of the swept volume regions (SVRs) formed by the swept volume portion considered.

The above definition of SVRs enables to consider only a minimum number of boundary elements from the swept volume B-rep during the fine and frame updates of the FSV-rep workpiece within each coarse surface voxel. More clearly, for fine and frame update of a voxel, only the portion of the envelope surface passing through it shall be considered. In other words, once a voxel is identified as associated with a particular SVR type, it is fully known what is the boundary condition inside the voxel from the swept volume passing through it (Figure 5-5). This aspect ensures that the calculations in the fine and frame updates are minimal in terms of the number of envelope boundary elements that should be considered once the SVRs and associated coarse voxels are identified in the coarse update step of FSV-rep model update.

In earlier works the Boolean operations between triangle mesh or B-rep solid models and other models have been accelerated using octree localization. Each octree node would be provided with the boundary elements from models of the Boolean targets and tools. Then in order to update the target geometry, the intersection calculation within an octree node for each of its boundary elements has to consider only the boundary elements from the tools that is present inside the same node. But this poses an issue to create the new surfaces for the updated geometry. Essentially the space subdivision methods have been used so far only to accelerate the Boolean operation. The model for the updated geometry would have to be still in a B-rep format. This would require additional stitching process and associated topology manipulations.

With the use of SVRs to update an FSV-rep model and generate the triangle mesh model for the machined surface only when it is required, the above-mentioned issues can be resolved. In the following sections, SVRs for general end mill and for usually used milling cutters will be developed.

5.5 SVRs for General end mill

The general end mill is parametrically represented by the APT definition using 7 parameters as shown in Figure 5-2. The most general tool path that need to be considered for a general SVRs case is the 3-axis linear case. SVRs for all other cases is a reduced version of the general case. For developing the general case SVRs, the swept volume boundary representation shown in Figure 5-6 can be used.

As can be seen from the boundary representation, there are 12 faces, 21 edges and 10 vertices for the B-rep (excluding those forming the top cover, which is irrelevant for model update). Corresponding to each of the boundary element, there is an SVR as well generating a total of 43 boundary SVRs. The internal volume of the swept volume also creates an additional internal SVR. Together there are 44 SVRs for the general linear tool path case.

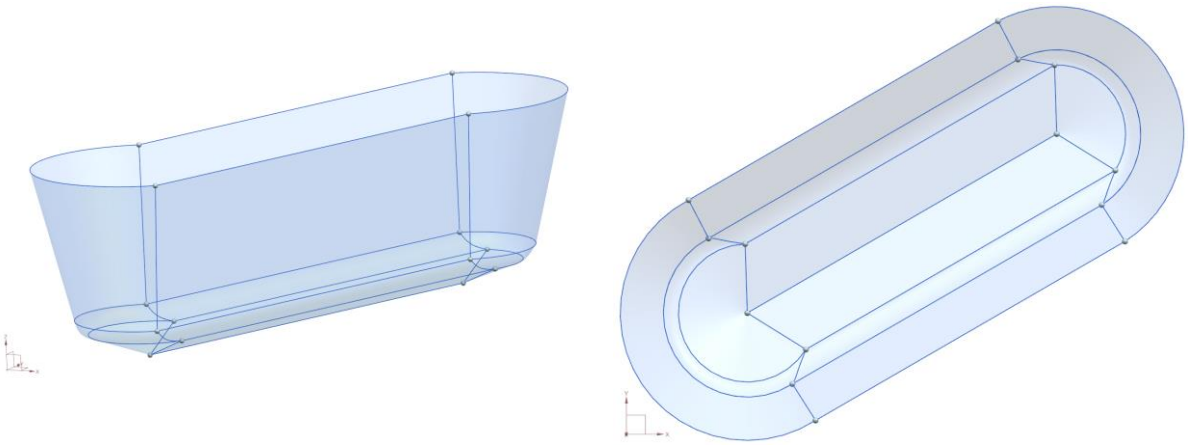


Figure 5-6 B-rep for general end mill swept along a 3-axis path.

It should be noted that there are 44 SVRs assuming the voxels of the voxel space are small enough to keep them as separate. If the voxel space grid spacing is such that there can be no voxels which can have just the portions of one face alone, then the corresponding SVR type will "merge" to other SVR types around. However, in the following section we assume that the

voxels are small enough such that the SVRs doesn't merge and all SVRs that is possible are present for the considered tool swept volume. The various SVR types and their distinguishing feature in terms of the boundary condition within the associated voxels is provided in the next section.

5.6 SVR types

The different SVRs can be categorized based on type of primary boundary element present in it. Thus as shown in Figure 5-7, there are

(1) face based SVRs,

(2) edge based SVRs and

(3) vertex based SVRs.

For **face based SVRs**, there is just one surface element of the swept volume B-rep that passes through the constituent voxels. This surface portion will divide each of the SVR voxels into two portions – one portion completely overlapping with the SVR and the other portion completely outside. Based on the envelope surface whose portion is present in the particular SVR voxel, the surface can be planar, cylindrical, spherical, conical, toroidal or a toric silhouette sweep. All these surfaces can be favorably used for updating the voxel frame edge. The same operations can be used for the fine/frame update as well within the coarse surface voxels identified as associated with the face based SVRs.

For **edge based SVRs**, there are two surfaces incident on the edge passing through the constituent voxels. Thus the portion of the voxel completely overlapping with the SVR is defined by the intersection of inner portions of both the surfaces. Here inner portion is defined for a surface element as one from which the surface normal is pointing away. In order to perform fine

and frame updates within the edge based SVRs, the specific surface to be considered to update a fine voxel or its frame has to be identified from the two surfaces incident on the edge. This can be done reasonably easily with the single valued nature of the surface elements along the axial direction.

And for **vertex based SVRs**, there can be up to 4 surfaces passing through the constituent voxels as incident on a given vertex. Consequently, the update operation within the vertex based SVRs are the most involved. However, the general approach will be similar to that possible for edge based SVRs.

The basic flow of the operations inside an SVR is same for all SVRs of a given type. Only difference will be the type of surfaces to use. As a result the conceptual definition of SVRs provide a suitable way to device the update operations at various regions of the swept volume in a localized and generic manner.

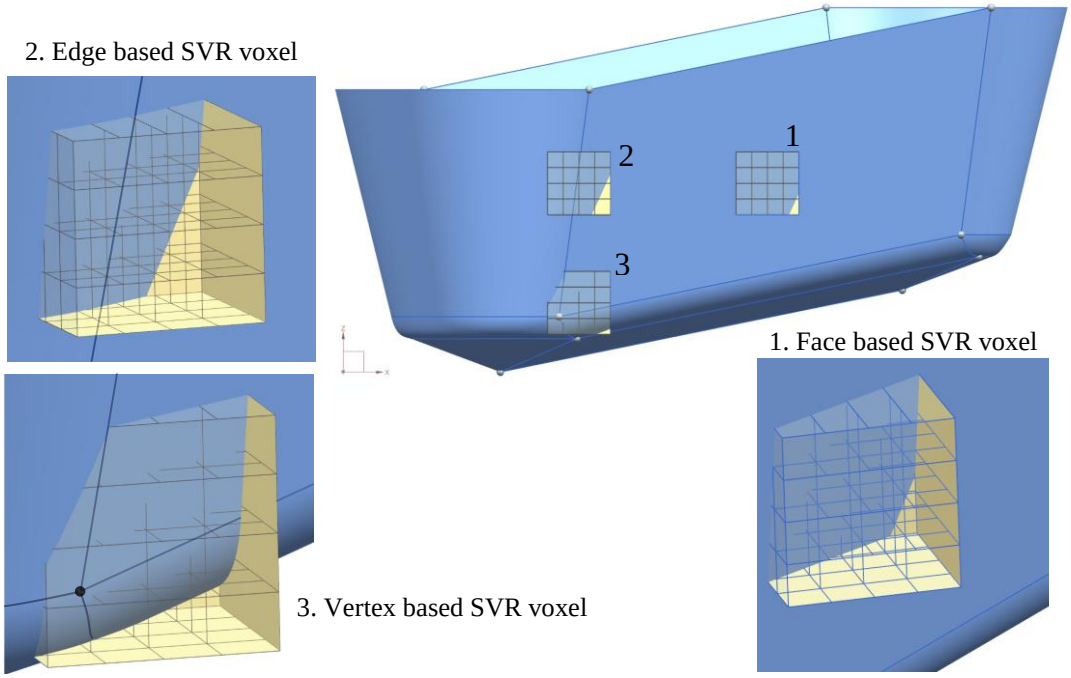


Figure 5-7 General swept volume B-rep and sample voxels overlapping with the three different SVR types.

5.7 Application to Flat end mill

The SVRs developed for general end mill can be reduced to simpler definition when the end mill geometry is simpler. This reduced definition helps for practical usage for a pre-defined tool while still following the general nature of SVRs for a tool swept volume. As shown in Figure 5-8, it can be seen that there will be 17 SVRs for a flat end mill moving along a 1-axis or 2-axis planar tool path. For a 3-axis straight cutting tool path, the flat end mill will create 19 SVRs. Out of the 17 SVRs in the planar case, 5 are face based, 8 are edge based and 4 are vertex based. For the 3-axis case, 6 are surface based, 9 are edge based and 4 are vertex based. Further the surfaces involved are planar or cylindrical alone for the planar case. For the 3-axis case, the bottom SVR has a surface that is elliptical cylinder. All these enable fast and simpler intersection calculations for FSV-rep update with SVRs.

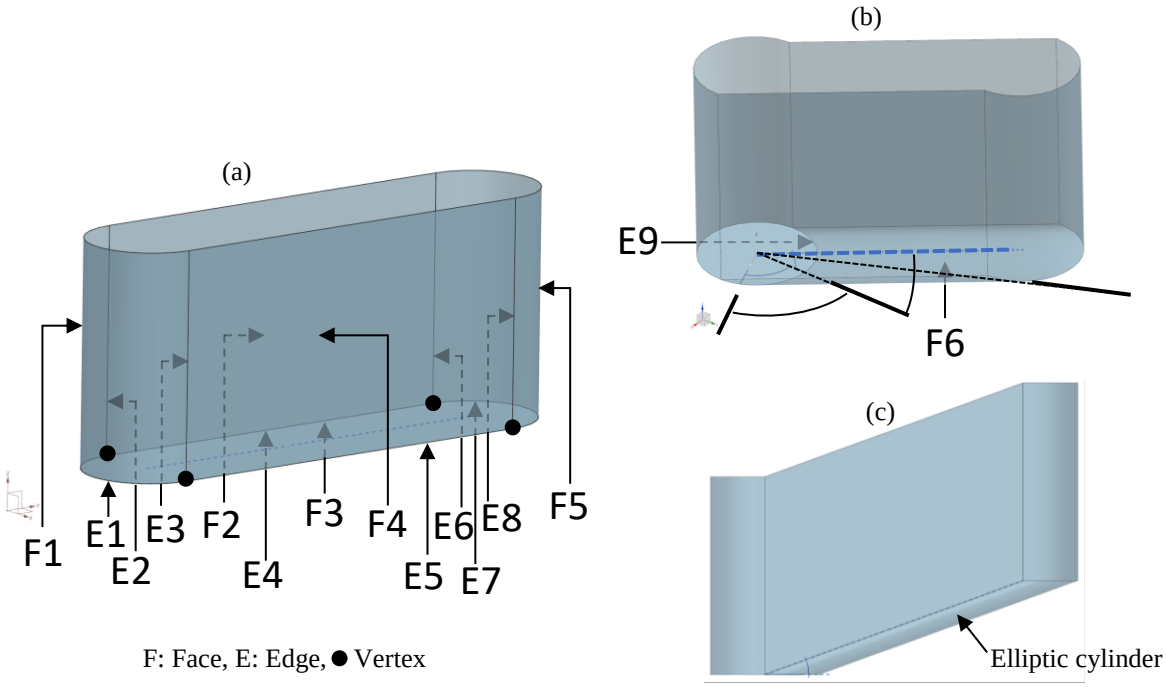


Figure 5-8 1-Axis and 3-Axis flat end mill swept volume B-reps showing different boundary elements defining the associated SVRs. (a) 1-axis, (b) 3-axis, and (c) 3-axis side view.

5.8 Application to Ball end mill

The Swept volume for a ball end mill in 1-axis tool path will consist of 21 SVRs of which 7 are surface based, 10 are edge based and 4 are vertex based. The swept volume B-rep is topologically different from that of a flat end mill in 1-axis as the bottom has to be spilt into three partitions. Unlike flat end mill, the SVRs stay the same for ball end mill on 2-axis and 3-axis tool paths (Figure 5-9). The surfaces of the swept volume B-rep can be composed of planes, cylinders, and spheres in case of ball end mill on 1-axis and 2-axis linear toolpaths. For 2-axis circular tool paths, the bottom surface is toroidal in place of cylindrical which causes more computations. For 3-axis tool paths, the bottom surface is still a cylinder as the silhouette of the spherical bottom along the toolpath direction is a circle (Figure 5-9). Thus the intersection computations involved for FSV-rep update with ball end mill up to 3 Axis linear tool paths are also simple and computationally fast as will be shown with results in Chapter 7, Section 7.3.

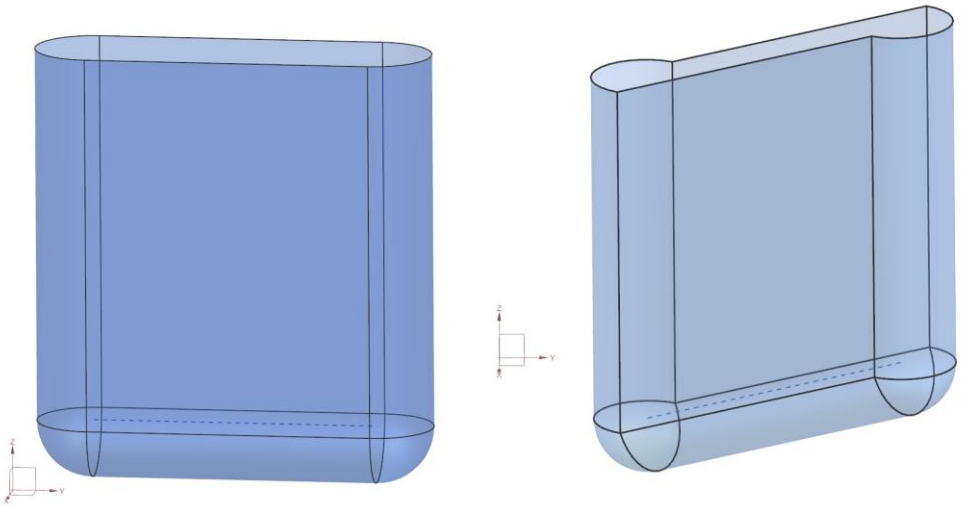


Figure 5-9 1-Axis and 3-Axis ball end mill swept volume B-reps showing the different boundary elements defining the associated SVRs.

5.9 Application to Taper ball end mill

Taper ball end mill swept volume has the same topology as that of the cylindrical ball end mill (Figure 5-10). Thus the number and type of SVRs required is also the same. However instead of the cylindrical surfaces at the ends, the taper end mill swept volume has conical surfaces. Also the two planar surfaces on the sides are no more vertical. Still the computational complexity for intersection of voxel frame edges with the surfaces is similar to that of the ball end mill thus having potential to be efficient in FSV-rep update with SVRs.

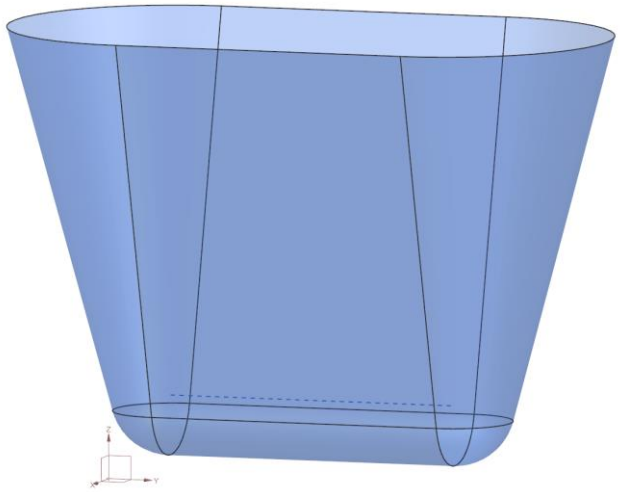


Figure 5-10 1-Axis taper ball end mill swept volume B-rep showing the different boundary elements defining the associated SVRs.

5.10 Application to Bull nose end mill

Bull nose end mill with a filleted bottom edge is another popularly used milling cutter. It has application for pocket milling with ability to generate both the rounded corners and planar face for the pocket floors.

The swept volume B-rep of bull nose end mill moving along 1-Axis or 2-Axis tool paths have simple boundary surfaces even though there are more of them compared to other simpler

milling cutters considered above. More specifically, there can be 9 face based SVRs, 16 edge based SVRs and 8 vertex based SVRs. Nevertheless, the surfaces are still primitive types of planes, cylinders and torus. Even for torus, as the tool axis is along one of the coordinate system axis, the intersection calculation with the voxel frame edges are still involving quadratic equations.

However, for bull nose end mill machining in 3-axis, the bottom surface has a fourth degree swept surface (Figure 5-11). It has been modeled as a silhouette sweep in earlier works [58] to update *Z*-map models. This procedure can be adapted for the intersection calculations involved in the FSV-rep update as well.

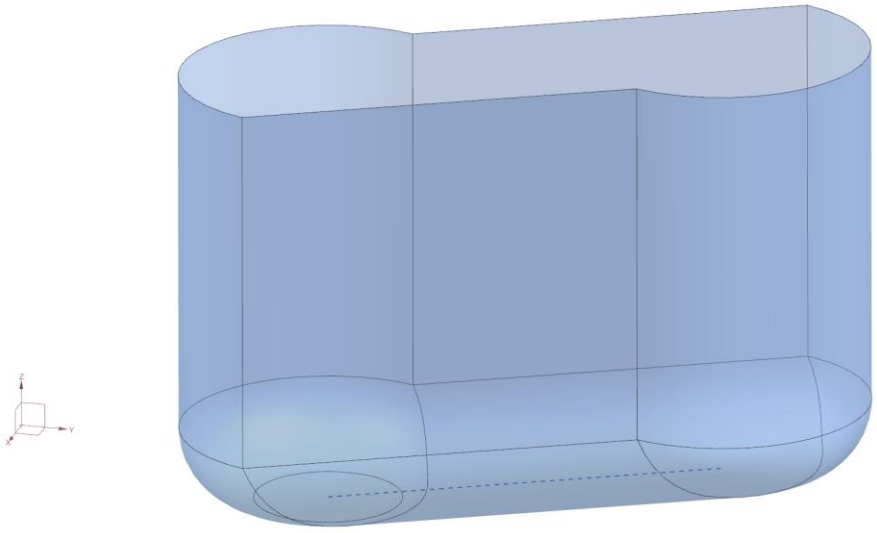


Figure 5-11 3-axis bull nose end mill swept volume B-rep showing different boundary elements defining the associated SVRs.

5.11 Summary

The tool swept volume representation has to be decided considering the possibility to update the FSV-rep workpiece model with simple and closed form equations. It should also be selected based on the accuracy required for the generated surface. From all the considerations, it

is identified that sampled tool instances along the tool path are the feasible approach for general multi axis tool paths. For simple linear tool paths, an advanced “Swept Volume Regions” definition is possible that can utilize the multi-level voxel model of FSV-rep workpiece for localized computations during the fine and frame updates.

Chapter 6: FSV-rep machining with tool swept volumes

In Chapter 4 we have seen how to perform the three-step update of FSV-rep workpiece with a set of abstract Boolean tools. In Chapter 5 we have seen that for tool paths with arbitrarily varying orientation such as with tool axis rotation and in case of curved planar tool paths the intersection operations between voxel edges and the swept volume is not direct. Thus, we decided to use sampled tool instances along such tool paths. However, for simple linear tool paths with fixed axis orientation, the sampled tool instances approach is unnecessary. First, there are explicit solutions for the intersection operations. Second, the tool instances approach is inferior to swept volume approach in case of accuracy. Further, when the explicit closed form solutions are available, the computations will be far less when one single swept volume boundary representation is used instead of a set of tool instance envelope surfaces. Thus, this chapter aims at selectively using the swept volume and the derived SVRs in case of compatible tool paths and the sampled tool instances in case of other tool paths for the FSV-rep workpiece update via the three-step process.

Hence forth, “simple swept volume” will mean swept volume for straight cutting tool paths with fixed tool orientation. When speaking about swept volumes for tool paths in general, “general swept volume” will be used.

6.1 Objective

In order to efficiently update the FSV-rep workpiece using simple swept volumes, a specific way of using it is apparently required. As seen in previous Chapter 5, using the whole B-rep for updating all the FS-voxels is not minimal in number of intersection checks needed. Thus, we had developed the concept of Swept Volume Regions in the Chapter 5 that has the inherent

property of localization to consider the least number of boundary elements of the swept volume while updating the workpiece voxels associated with it. Further to use the SVRs effectively, the three-step update process can be adapted to work for them.

Also for the update with sampled tool instances, the FSV-rep update steps can utilize the axisymmetric nature of the milling cutters to improve the calculations involved. The basic queries possible on such a tool instance such as point classification, voxel classification and frame edge intersections can be efficiently combined to have an efficient update logic with tool instances.

The basic objectives in either case are the following:

- Utilize three-step update possibility of FSV-rep to localize and minimize the intersection operations,
- Utilize swept volume representations that provide closed form solutions for the intersection calculations whenever applicable.

In the following sections, the modifications required for the three-step update process will be identified, in order to efficiently use the simple swept volumes with SVRs or the sampled tool instances as the required case may be for a particular tool path. The overall logic is laid out in Section 6.2. Then in the following sections, the specific changes required for each of the steps and the additional steps needed are developed.

6.2 Overall update logic

The essential philosophy of the three-step update process is to minimize intersection calculations, maximize the binary update of voxels and further accelerate the binary operations using coarse voxel grids for the bulk volume removal. This should be preserved for use of SVRs

and tool instances. The overall concept of the update logic selectively using SVRs or sampled tool instances is schematically shown in Figure 6-1.

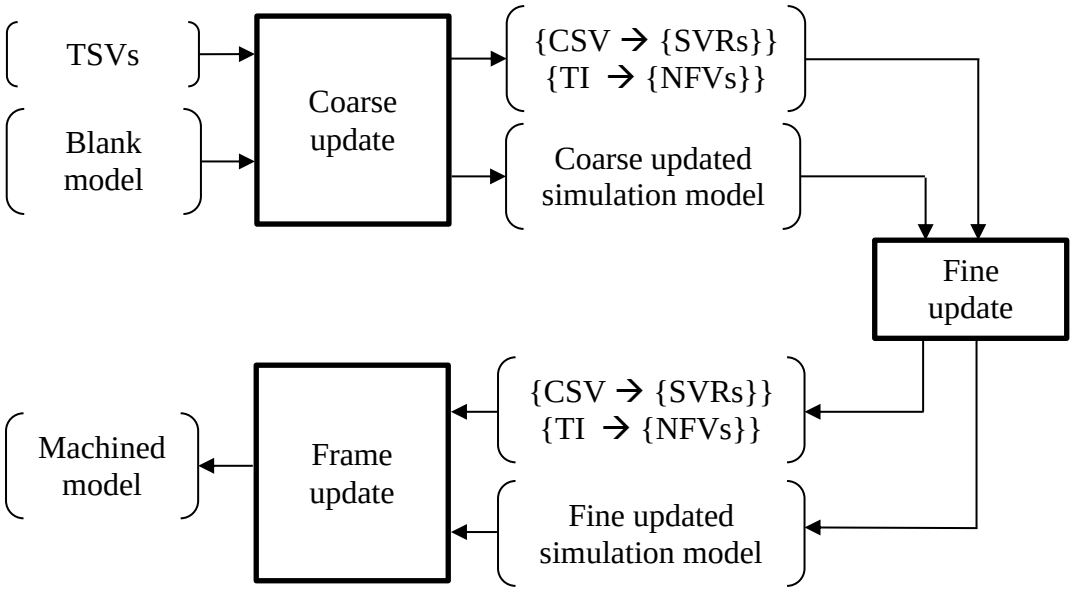


Figure 6-1 Overall update logic with SVRs and Tool instances used separately.

The swept volume representations should first perform the bulk volume removal by binary update of the coarse voxel model. For this an efficient approach is needed to identify the coarse voxels that need update. For simple swept volumes with SVRs, it is inefficient to consider the coarse voxels in the entire bounding box of the swept volume representations as the bounding volume can be very large for long tool paths even in case of 2-axis machining. Thus, a better technique to perform the coarse update is required which can identify the voxels to be updated without considering the entire axis aligned bounding box region.

In case of the sampled tool instances, the axis-aligned bounding box itself is the optimal localization option. This is because any alternative to first obtain the voxel model of the tool instance will involve intersection calculations or some voxelization approach as done for the

original triangle mesh. Directly considering all the voxels in the tool instance bounding box will avoid the intermediate voxelization step. The additional computation to perform for all the voxels in the bounding box is simple point classification operations developed in Chapter 5, Section 5.3.

Once the coarse voxel model is updated, the voxels that need fine update should be identified. In the generic update method developed in Chapter 4, the "near-field (NF) voxels" were used for this purpose to identify the candidate voxels without much overhead.

However, the NF-voxels are not necessarily having any fine level surface voxels and were utilized to avoid costlier intersection calculations or more involved checks needed to identify the actual surface voxels. In case of simple swept volume representation with SVRs, the exact coarse surface voxels (CSVs) itself can be obtained with acceptable computational load. Also with the number of boundary elements drastically reducing with use of one swept volume in place of many sampled tool instances, the overhead from intersection calculations are in a way negligible. This makes it viable to obtain actual coarse surface voxels itself for fine and frame update in case of the SVR based approach.

For sampled tool instances, due to the same reasons, use of NF-voxels itself is the efficient way for collecting the candidate voxels which will contain the potential coarse surface voxels that need further refinement.

In order to perform the fine update for the coarse voxels, a technique utilizing the specific qualities of SVRs or tool instances is to be developed. In case of tool instances, the fine update can use the axisymmetric nature of the tool envelope geometry to perform the voxel classification operations involved. However, it will have to consider all fine voxels of the subspace within a coarse voxel. This is not a requirement when using SVRs for fine update. The

directional factor of the definite envelope surfaces and the fact that all the envelope surfaces present within a coarse surface voxel together divide it into two subspace portions, one inside the swept volume and the other outside, can be used together to perform the fine update without considering all the voxels of the subspace.

Similar to the fine update, frame update can also utilize the uniformity and definiteness of SVRs to directly update the fine level surface voxels. Further to note, the frame update can still be performed with computations of same complexity as those in case of tool instances. This is courtesy of the fact that the swept volumes in considerations always have envelope surfaces with closed form solution for intersection with voxel frame edges. For tool instances as well, the frame update is computationally feasible as the tool geometry is composed of arc and line revolutions alone.

The specific details of the revised coarse-fine-frame update steps are provided in the following sections. First, the technique to easily identify the tool paths that can use SVRs is developed in Section 6.3. Next the reformed three-step update process to use SVRs and tool instances are provided separately in Sections 6.4 and 6.5 respectively.

6.3 Tool paths categorization

Identifying the category for a given tool path is crucial for the effective use of the category specific SVRs definition. Here category refers to whether the tool path is for 1, 2 or 3 axis machining, and whether the tool axis is along positive or negative direction of one of the coordinate system axis. The tool type and the path curve type should also be used along with the path category for effective implementation of the selective update using SVRs or sampled tool instances.

All the tool paths should be considered in the model coordinate system (MCS) for FSV-rep update. Therefore, we have to take into account the cases where the tool axis is along any of the three axes of MCS. Further the tool axis can be along the positive or negative direction of the particular coordinate axis. Thus, there are 6 possible alignment for the tool axis along the tool path. All these are with the prior limitation that only linear tool paths with axis aligned tool orientation is considered. The term "axis aligned" here meant along one of the MCS coordinate axes. Further, for such toolpaths, there can be translation along any of the three MCS coordinate axis directions.

For the convenience of the techniques used later in the coarse update, it is also beneficial to distinguish the three MCS coordinate axes as "feed direction coordinate axis, F_{dir}", "lateral direction coordinate axis, L_{dir}" and "axial direction coordinate axis, A_{dir}" with respect to the path as follows: In case of 1-axis tool paths, the distinction is apparent. The tool path is aligned along a particular MCS axis which becomes the F_{dir} axis. The MCS axis along which the tool is oriented becomes A_{dir} axis and the third axis becomes L_{dir}. In case of 2-axis and 3-axis paths the A_{dir} is still apparent. However, the distinction of the other two MCS axis needs further considerations as explained next.

For 2 and 3 axis tool paths, once the A_{dir} is set, the coordinate plane containing the other two MCS axis is clear. The projection of the tool path curve onto this plane, tp_{proj} gives the necessary information to identify the F_{dir} and L_{dir}. It is apparent that tp_{proj} is identical to the path curve tp itself in case of 2-axis tool paths. Here tp_{proj} is used to treat 2-axis and 3-axis tool paths in the same way. With tp_{proj} identified, the MCS axis in the projection plane to which the tp_{proj} is more inclined is the preferred scanning direction, F_{dir} and the other MCS axis has to be the Lateral sweeping direction, L_{dir} as shown in Figure 6-2.

The MCS origin for FSV-rep model is preferably set at the lower-left corner of the back extreme of the blank workpiece bounding box. This effectively ensures that all the tool paths transformed into the MCS are in the first octant. Further for the purpose of identifying F_{dir}, L_{dir}, and A_{dir}, the tool path direction along the positive and negative direction of a particular MCS axis is immaterial.

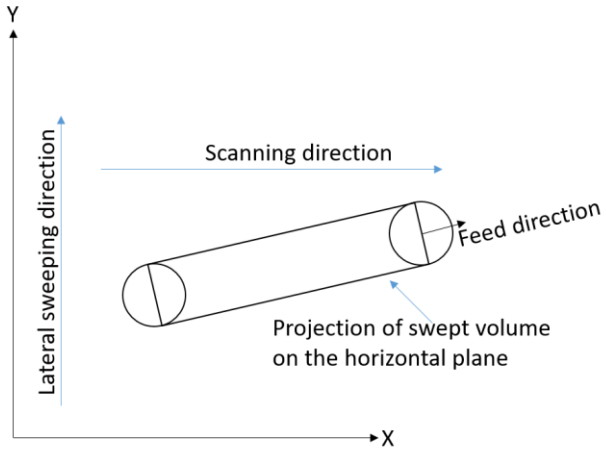


Figure 6-2 The tool path projected on the base plane (perpendicular to A_{dir}) and the F_{dir}(scanning direction) and L_{dir} (lateral sweeping direction) identified for XY as base plane.

To apply all these conditions and distinguish the MCS axes as A_{dir}, F_{dir} and L_{dir}, a four-component anaylsis of the path curves can be used as follows:

For tool path defined by two lines $c1$ and $c2$ for the trajectory of two points on the tool axis, $dX_{c1} = x\big(c1(1)\big) - x(c1(0))$, $dX_{c2} = x\big(c2(1)\big) - x(c2(0))$, $dX_{A1} = x\big(c2(0)\big) - x(c1(0))$, and $dX_{A2} = x\big(c2(1)\big) - x(c1(1))$. Then,

$$[dX] = \{dX_{c1}, dX_{c2}, dX_{A1}, dX_{A2}\} \tag{6.1}$$

$$XLabel = \delta(dX_{c1}) \times 8 + \delta(dX_{c2}) \times 4 + \delta(dX_{A1}) \times 2 + \delta(dX_{A1}) \tag{6.2}$$

where $\delta(x)$ is equal to 0 if x is 0 and is equal to 1 otherwise. Similarly, $YLabel$ and $ZLabel$ are also obtained from $[dY]$ and $[dZ]$. Then Table 6-1 provides the A_{dir}, L_{dir} and F_{dir} for the valid combinations of $XLabel$, $YLabel$ and $ZLabel$.

Table 6-1 Coordinate axis characterization using the tool paths employing SVRs.

XLabel	*YLabel*	*ZLabel*	A_{dir}	F_{dir}	L_{dir}
0	3	12	Y	Z	X
0	12	3	Z	Y	X
0	12	15	Z	Y	X
0	15	12	Y	Z	X
3	0	12	X	Z	Y
3	12	0	X	Y	Z
3	12	12	X	Y if $\lvert dY_{c1}\rvert > \lvert dZ_{c1}\rvert$ Z otherwise	Z if $\lvert dY_{c1}\rvert > \lvert dZ_{c1}\rvert$ Y otherwise
12	0	3	Z	X	Y
12	0	15	Z	X	Y
12	3	0	Y	X	Z
12	3	12	Y	X if $\lvert dX_{c1}\rvert > \lvert dZ_{c1}\rvert$ Z otherwise	Z if $\lvert dX_{c1}\rvert > \lvert dZ_{c1}\rvert$ X otherwise
12	12	3	Z	X if $\lvert dX_{c1}\rvert > \lvert dY_{c1}\rvert$	Y if $\lvert dX_{c1}\rvert > \lvert dY_{c1}\rvert$

XLabel	*YLabel*	*ZLabel*	A_{dir}	F_{dir}	L_{dir}
				Y otherwise	X otherwise
12	12	15	Z	X if $\|dX_{c1}\| > \|dY_{c1}\|$ Y otherwise	Y if $\|dX_{c1}\| > \|dY_{c1}\|$ X otherwise
12	15	0	Y	X	Z
12	15	12	Y	X if $\|dX_{c1}\| > \|dZ_{c1}\|$ Z otherwise	Z if $\|dX_{c1}\| > \|dZ_{c1}\|$ X otherwise
15	0	12	X	Z	Y
15	12	0	X	Y	Z
15	12	12	X	Y if $\|dY_{c1}\| > \|dZ_{c1}\|$ Z otherwise	Z if $\|dY_{c1}\| > \|dZ_{c1}\|$ Y otherwise

6.4 Update using SVRs

The FSV-rep model update with SVRs for linear three axis tool paths shall follow the general three-step update logic but with some necessary customization for each step as described in the following sub-sections.

6.4.1 Coarse update with SVRs

In this step of FSV-rep update, all the coarse level surface voxels of the FSV-rep workpiece model that are completely inside any tool swept volume have to be deleted. The new

coarse level surface voxels for the newly generated workpiece surface area has to be identified as well. As identified previously, the approach of considering all the voxels in the bounding box of the tool swept volume is not optimal because of the possibility that bounding box can be unfavorably big and less localizing in case of 2 and 3 axis tool paths. The most efficient way will be some operation which will explicitly solve for the set of voxels belonging to the tool swept volume. This operation of identifying the parent voxel is well defined for a point by Equations 3.1 and 3.2. For curves and surfaces, the point on the geometric element should be defined parametrically and the Equations 3.1 and 3.2 should be used for each of the point. To solve for all the voxels through which a curve or surface passes, this approach should sample enough points irrespective of the surface curvature and orientation. For the triangle mesh voxelization done in order to create the initial FSV-rep, a variant of this approach was done for the edges of each triangle. And for the face interior area, an advancing front approach could be used effectively. It could be efficient and appropriate for a triangle voxelization because the surface voxel identification for a planar element is possible without intersection calculations. In case of swept volume boundary representation, this is not guaranteed always as the envelope surfaces can be cylindrical or other shapes with curvature even in case of the simple tool paths in consideration. Thus, a computationally simpler approach is required for the purpose of coarse update with tool swept volume.

Without scanning the entire bounding box voxels, the approach essentially becomes one of a type of voxelization. Still it should be applicable to swept volumes with any boundary elements but with constant swept tool orientation. Since the tool orientation is fixed, we have a case where the voxelization approach can be done localized using the swept volume projection sv_{proj} on to the base plane perpendicular to the tool orientation (Figure 6-2). Essentially, we are moving from

axis aligned bounding box to a tighter envelope provided by the boundary of sv_{proj}.

The boundary of sv_{proj} defines a set of voxels $\{\boldsymbol{V_{SVproj}}\}$ covered by its extrusion along the tool orientation direction A_{dir}. The voxels we are interested in, specifically the surface and volume voxels of the swept volume are a subset of $\{\boldsymbol{V_{SVproj}}\}$. In order to avoid scanning through all of the voxels in $\{\boldsymbol{V_{SVproj}}\}$, we can perform a simple 2D scan in the base plane within the sv_{proj} and consider the vertical stack of voxels at each step along the 2D scan. This is convenient as our swept volumes has a fixed tool orientation and their projection onto the base plane will consistently generate a well-defined sv_{proj} without self-intersecting boundary.

The 2D scanning of shapes with simple boundary is a problem with many popular solutions of which the sweep line approach is of special attraction for its computational and data structural efficiency [78]. Thus, our approach of swept volume voxelization is an extension of sweep line algorithm into a "sweeping plane voxelization algorithm" for swept volumes of fixed tool orientation.

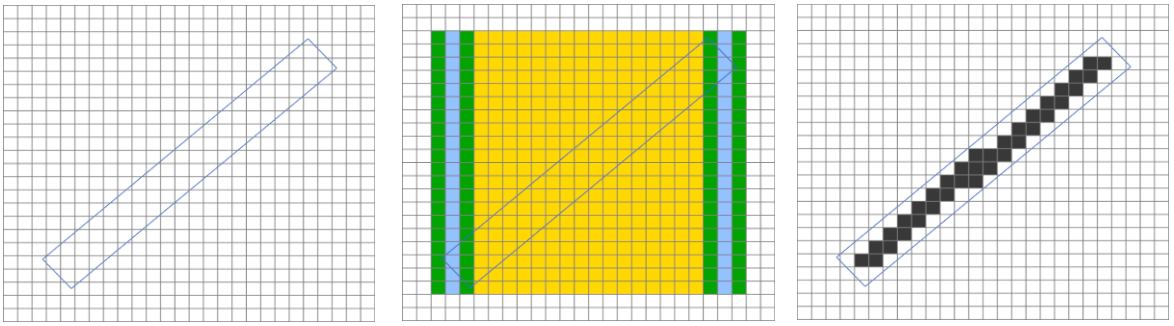
The distinction of F_{dir} and L_{dir} comes to benefit for a computationally favorable definition of the sweeping plane voxelization algorithm. The F_{dir} is the preferred primary sweeping direction and L_{dir} is the preferred secondary or lateral sweeping direction. The choice of primary and secondary sweeping directions is in fact after a trade-off consideration between computational benefit and implementation efforts and following the law of diminishing returns.

Algorithm 6-1 for the sweeping plane based coarse update in case of 1, 2 and 3 axis tool swept volumes with general and typical tools is provided below:

Algorithm 6-1: Coarse update with a swept volume
For each step *iF* along F_{dir} **For** each step *iL* along L_{dir}

$B \leftarrow$ base plane perpendicular to A_{dir}
{L} $\leftarrow$ 4 grid lines along A_{dir} at locations $\{iF, iF+1\} \times \{iL, iL+1\}$
{P} $\leftarrow$ intersection points of **{L}** with the current bottom surface element
$V \leftarrow$ voxel containing the max(***{P}***)
$Idx_A \leftarrow A_{dir}$ index of V voxel
For all voxels with A_{dir} index from Idx_A+1 to Idx_{Atop} // Idx_{Atop} is currentA_{dir}index of top cover
Delete the voxel from workpiece model
End
End
End

In the above algorithm, once the sweeping direction is identified, each swept volume will have different scanning sections based on the boundary elements at the two extremes along the lateral direction. For the 2-axis case shown in Figure 6-3a, there can be 3 to 7 sections based on the angle between feed and scanning directions (Figure 6-3b). The boundary element at the bottom can vary along the lateral direction for tools like bull nose end mill and taper ball end mill. A sub-range for the sweep along the lateral direction has to be identified to separately handle these sub-sections.



(a) (b) (c)

Figure 6-3 Different steps of the coarse update for tool paths using SVRs. (a) Top view of a planar straight cutting swept volume in voxel space, (b) different scanning regions based on bounding elements, and (c) Inner coarse voxels deleted.

The basic idea of scanning with sweeping-plane is as follows: First the range for scanning is obtained in terms of the start and end voxels layer along the primary sweeping direction. For

each layer, the bound for lateral sweeping is then identified. For each step along the lateral sweep direction, the bottom and top voxels for the stack of swept volume voxels are then identified. The coarse update then essentially deletes the workpiece voxels falling in this stack.

6.4.2 Fine update with SVRs

After the coarse update, we have the mapping of each SVR to the set of coarse surface voxels $\{\boldsymbol{CSV}\}$ of the FSV-rep model associated with it. It is noteworthy that this mapping is superior to the $T_{rem} \rightarrow \{\boldsymbol{NFV}\}$ mapping we had conceptualized as possible in Chapter 4. Similar to the inversion applied to the $T_{rem} \rightarrow \{\boldsymbol{NFV}\}$, an inversion for the current map gives a $CSV \rightarrow \{\boldsymbol{SVR}\}$ mapping. Here CSVs are actual surface voxels with definite presence of envelope surfaces inside unlike the NFVs. Further for each SVR present inside a CSV, the portion of the CSV overlapping with it is apparent from the nature and type of the particular SVR as detailed in Chapter 5, Section 5.6.

With the definiteness of SVRs, the fine update for a CSV is in a similar scanning fashion as the coarse update but within the additional bounds from the CSV subspace. Essentially the aim of fine update of each CSV is to delete the fine voxels of the subspace within it if the fine voxel is completely within any of the SVRs passing through it (Figure 6-4). The Algorithm 6-2 for fine update within a CSV is provided below:

Algorithm 6-2: Fine update within a coarse surface voxel using SVRs

For each *SVR* present in the *CSV*
  For each step *iF* along F_{dir}
    For each step *iL* along L_{dir}
      $B \leftarrow$ base plane perdicular to A_{dir}
      $\boldsymbol{\{L\}} \leftarrow$ 4 grid lines along A_{dir} at locations $\{iF, iF+1\} \times \{iL, iL+1\}$
      $Surf_{btm} \leftarrow$ the active bottom surface elements of the SVR for $\boldsymbol{\{L\}}$

```
        {P} ← intersection points of {L} with Surf_btm
        V ← voxel containing the max({P})
        Idx_A ← A_dir index of V voxel
        For all voxels with A_dir index from Idx_A+1 to Idx_Atop
           Delete the voxel from workpiece model
        End
      End
   End
End
```

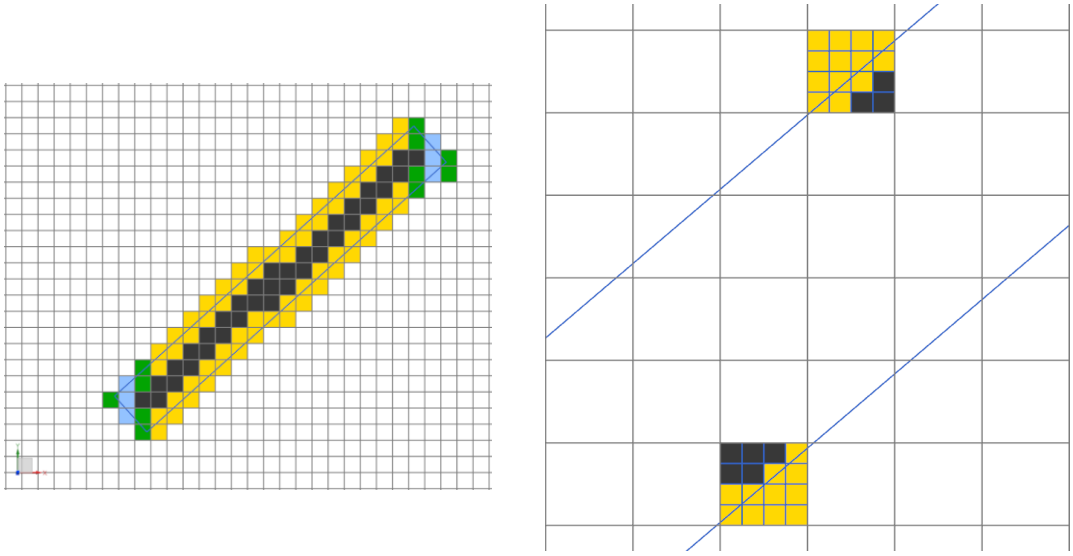


Figure 6-4 Coarse surface voxels identified for fine update using SVRs (left) and two sample coarse surface voxels after the fine update (right).

In the above algorithm the active bottom surface element is the bottom bound for the SVR within the coarse surface voxel considered that is present across the particular stack of fine voxels considered. For side face based SVRs, this can be the bottom floor of the subspace within the voxel as the side face is vertical for cylindrical end mills (flat/ball/bull-nose end mills). Also the above algorithm reduces to simpler versions based on the SVR types. For face based SVRs there is only one swept volume surface element within the associated voxels. Thus, intersection

points computation is needed only with one surface. For edge and vertex based SVRs, two or up to four surface elements may be present. The active bottom surface element has to be identified for each stack of voxels based on the location of the stack of voxels with respect to the projection of the edge elements onto the subspace floor.

6.4.3 Frame update with SVRs

The objective of frame update is to refine the final fine surface voxels. This requires intersection of the voxel edge frame with the SVRs passing through each fine level surface voxel. We can once again utilize the $CSV \rightarrow \{\boldsymbol{SVR}\}$ mapping available and used previously for fine update. After the fine update, the subspace within a particular CSV is having correct status for all the fine level voxels within. Performing frame update within a CSV only after fine update with all the SVRs present within has a merit. This ensures that a fine voxel that is a surface voxel with respect to a particular SVR shall not be frame updated if it was deleted by the fine update by another SVR within the CSV. This is beneficial as fine update requires only intersection of the voxel frame edge along the A_{dir} as we saw in previous Section 6.4.2 whereas frame update for a particular surface voxel will need intersection of all the primary edges with the SVR envelope boundary.

The common idea for frame update with any SVR within a CSV again starts with scanning over a base plane within the CSV. In this case, instead of performing delete operation for a range of voxels in each vertical stack, the surface voxels at the end are only updated with trimming operation for the frame edges (Figure 6-5). Further this is done provided the surface voxel is still an active part of the FSV-rep workpiece. The following Algorithm 6-3 detail the general update logic for various categories of SVRs:

Algorithm 6-3: Frame update within coarse surface voxels

For each *SVR* present in the *CSV*
  For each step *iF* along F_{dir}
    For each step *iL* along L_{dir}
      $B \leftarrow$ base plane perpendicular to A_{dir}
      {L} $\leftarrow$ 4 grid lines along A_{dir} at locations $\{iF, iF+1\} \times \{iL, iL+1\}$
      $Surf_{btm} \leftarrow$ the active bottom surface elements of the SVR for ***{L}***
      {P} $\leftarrow$ intersection points of ***{L}*** with $Surf_{btm}$
      {V} $\leftarrow$ surface voxels from voxels for min(***{P}***) to max(***{P}***)
      For each *V* in ***{V}***
        If *V* voxel not active
          Skip to next *V*
        Update FC-points for the *V* voxel's edge frame
      End
    End
  End
End

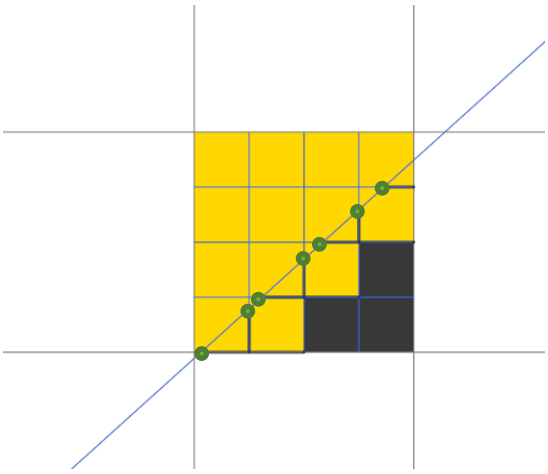


Figure 6-5 A coarse surface voxel with frame update performed from a face based SVR viewed along A_{dir}.

6.5 Update with Tool instances

In the previous sections, the specific methodology for performing the three-step update using SVRs were developed. It required some specialized techniques to enable the coarse fine and frame level updates efficiently using the salient features of SVRs for localization. In this

section, the three-step update methodology will be specialized for use of sampled tool instances having axisymmetric geometry.

Sampled tool instances are the suitable approach for approximating general multi-axis tool paths as we saw from the discussion in Chapter 5. With sampled tool instances for tool paths, swept volume is approximated with a set of tool instances sampled sufficiently along the tool path such that the union of all these tool instance volumes is as close as possible to the swept volume. Each tool instance along the tool path, T_i has a specific orientation and tip location. The swept volume V_{swept} is then,

$$V_{swept} \approx \sum \{T_i\} \tag{6.3}$$

where the approximation error depends on the forward sampling interval along the tool path and hence on the range of i in the above equation,

6.5.1 Sampling interval selection

The sampling interval along the tool path is the deciding factor of the effectiveness of sampled tool instance in closely approximating the swept volume. Any sampling interval will have an associated sampling scallop which is the error from approximation present only on the simulation part surface and not on the actual machined. The sampling scallop ϵ_{flat} for a given sampling interval, L_s or conversely the minimum sampling interval for a given sampling scallop limit is easy to derive in case of a flat end mill (of radius R_t) creating a planar face by side milling:

$$\epsilon_{flat} = R_t - \sqrt{R_t^2 - \frac{L_s^2}{4}} \tag{6.4}$$

However, for general tool paths and cutter types, it is complicated to obtain due to the varying curvature of the machined surface and the and tool radius along the axis. Thus, for general cases a different guideline for sampling interval is required. In this work the sampling interval is set to be less than or equal to the spacing of the grid for which the tool path is sampled – coarse grid spacing for coarse update and fine grid spacing for the fine and frame updates.

A sampling interval less than or equal to the relevant grid spacing ensures that almost all the voxels that are affected by an ideal swept volume are collected by the set of sampled tool instances as well. This also gives a very much acceptable sampling scallop in the case of planar face machining by side milling considered above:

In this work, the maximum voxel size for the coarse grid is set via the cutting tool size as

$$L_{vc} < \frac{D}{\sqrt{3}} \approx R_t \tag{6.5}$$

where *D* is the diameter of selected cutting tool. This is to ensure the voxel is small enough to be completely inside a tool instance and thus be removed by the coarse update part 1. With this value, the sampling interval and the resultant sampling scallop in the above case becomes,

$$L_s = L_{vf} = \frac{L_{vc}}{f} = \frac{R_t}{f} \tag{6.6}$$

$$\epsilon_{flat} = \frac{R_t}{2f}\left(2f - \sqrt{4f^2 - 1}\right) \tag{6.7}$$

The sampling scallop as a percentage of cutter radius, R_t for typical values of sub-division factor f are given in Table 6-2. Even though this is the value for planar side machining, it is quite representative as the value for simulation of a convex surface machining will be lesser than this. For simulation of concave surface machining, the sampling scallop will be higher related to the

curvature of the concave surface. This can however be reduced by sampling along the cutter contact curve instead of the cutter location curve.

Table 6-2 Sampling scallop for flat side milling at various FSV-rep subdivision factors.

f	ϵ_{flat} as % of R_t
2	3.17
4	0.78
8	0.19
16	0.05

6.5.2 Coarse update with tool instances

The main operation involved in the first part of coarse update to bulk remove the coarse voxels completely inside any tool instance as shown in Figure 6-6 is to classify the voxel with respect to the tool instances as inner-field or otherwise. Equations 5.7-5.10 developed in Chapter 5, Section 5.3 gives the field classification of a voxel with respect to a tool instance. This can be used for the purpose.

The second part of coarse update to collect the Near-field voxels of each tool instance that is active part of FSV-rep workpiece after the first part can also use the same set of equations.

Thus, the coarse update Algorithm 4-1 developed for the set of general volume removal tools in Chapter 4, can be used with the appropriate equations in place as below:

Algorithm 6-4. Coarse Update with tool instances
{TP} ← list of the milling tool paths **for** each TP in **{TP} do** //Part 1

```
    {TI} ← set of sampled tool instances along the TP
    for each TI in {TI} do
      {BV} ← coarse voxels inside/intersecting TI's bounding box
      for each V in {BV} do
        if V active and V completely inside (Equation 5.9) TI then
          Set V inactive
        end if
      end for
    end for
end for
for each TP in {TP} do //Part 2
    {TI} ← set of sampled tool instances along the TP
    for each TI in {TI} do
      {BV} ← coarse voxels inside/intersecting TI's bounding box
      for each V in {BV} do
        p ← center point of V
        if V active and p inside near-field (Equations 5.7) of TI's envelope then
          {NFV} ← {NFV} + V
        end if
      end for
    end for
end for
```

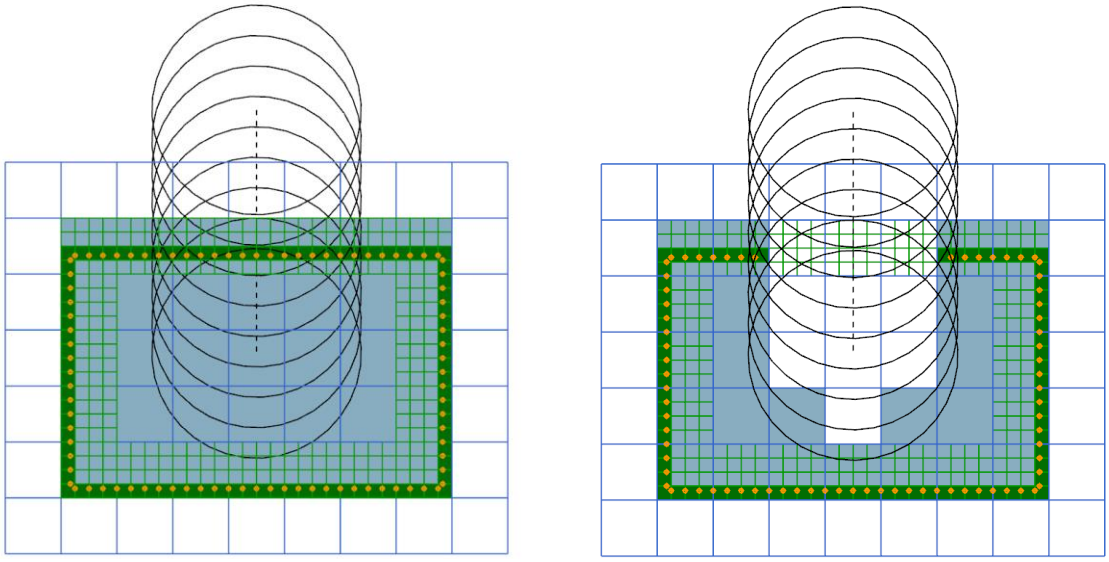


Figure 6-6 Coarse update with a set of sampled axisymmetric tool instances along a tool path.

6.5.3 Fine update with tool instances

Fine update for the NF-voxels has to perform essentially the same activity as the coarse update but now within each NF-voxel using the set of tool instances passing by them (Figure 6-7). Algorithm 4-2 developed in Chapter 4, Section 4.3 is to be reused with changes to utilize the exact equations available for tool instances. Also, all the fine voxels within the coarse NF-voxel has to be checked against each tool instances unlike the fine update using SVRs. This is because the tool instance orientation can be in any arbitrary direction and there is no fixed A_{dir} as that could be used for SVRs.

Unlike the coarse update which had to only classify voxels as near-field and inner-field, fine update has to further categorize the near-field voxels as surface voxels or not. Equation 5.5 developed in Chapter 5, Section 5.3 can be used for this.

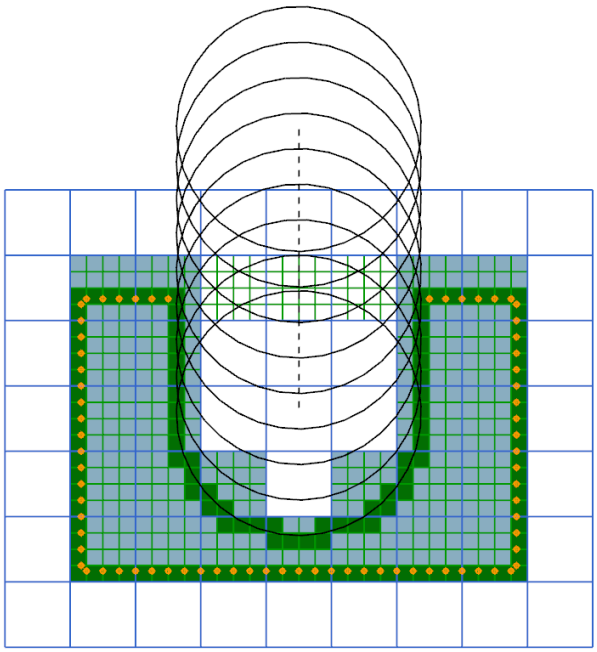


Figure 6-7 Fine update with set of sampled tool instances for a tool path, creating the fine level surface voxels.

The modified fine update algorithm for sampled tool instances is given below:

Algorithm 6-5. Fine Update

```
NVR ← number of coarse voxels to refine
for i from 1 to NVR do
   CV ← ith coarse voxel to refine
   {TI} ← tool instances crossing CV
   {FV} ← fine-level voxels within CV
   for each FV in {FV} do
      p ← center point of FV
      for each TI in {TI} do //Part 1
         if FV active and FV an interior voxel (Equation 5.9) for TI then
            Set FV inactive
         end if
      end for
      for each TI in {TI} do //Part 2
         if FV active and p inside near-field of TI's envelope (Equation 5.7) then
            if FV a surface voxel (Equation 5.5) for TI then
               Add FV to surface voxel model
            end if
         end if
      end for
   end for
end for
```

6.5.4 Frame update with tool instances

The frame update concept developed in Chapter 4, Section 4.4 can be used with exact equations for the intersection of frame edges with tool instances. The axisymmetric nature of the milling cutters provides quadric or toroidal surfaces for the tool envelope. The intersection calculations are all feasible and thus frame update can be done in definite steps.

The representation process of frame update for a set of sampled tool instances is given in

Figure 6-8. Each tool that is crossing a fine level surface voxel is used to obtain the intersection points on the frame edges that tool instance is intersecting. Finally the intersection point that is not inside any of the tool instance is kept as the FC-point on that frame edge as shown in the zoomed in view in Figure 6-8.

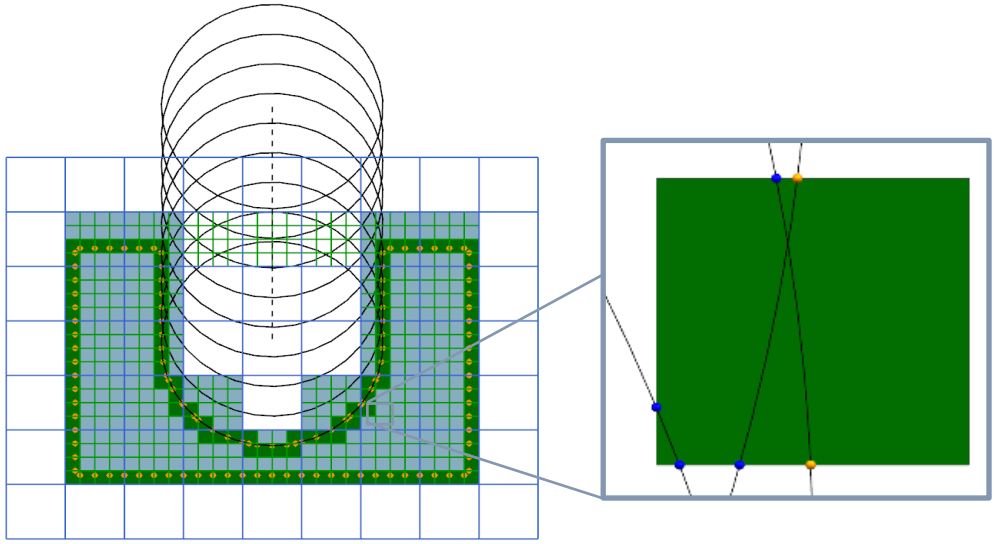


Figure 6-8 Frame update for the fine level surface voxels creating the FC-points (yellow spheres) from the intersection points on the frame edges (yellow or blue sphere in the zoomed in view).

6.6 Summary

In this chapter we could identify the appropriate ways in which the FSV-rep workpiece should be updated using sampled tool instances for general multi-axis tool paths and SVRs for linear three-axis tool paths. In both cases, the efficient three-step update logic could be suitably employed. This ensures potential mixed use of the two type of swept volume representations for appropriate tool paths in a general machining operation.

Chapter 7: Simulation system implementation and case studies

In order to evaluate the performance of FSV-rep in use for machining simulation a prototype system was implemented and a series of case studies were done. The prototype was developed with C++ programming in Qt environment using Microsoft VC++ compiler. All case studies were done on a Windows 10 PC with 8 GB RAM and 3.3 GHz processor. The following sections provides the details of the implemented simulation system for machined part geometry computation and also the discussion of result for various case studies.

7.1 Implementation details

The FSV-rep model with two levels of voxel grids (a coarse grid and a fine grid) was employed in the implementation of this work. The ratio of the fine-grid voxel size to the coarse-grid voxel size was set as 1:4. A fine grid spacing of 1 mm was used resulting in a 4 mm coarse grid spacing. One-dimensional array of binary variables (bit-array) was used to represent the 3-dimensional grid of voxels making the voxel spaces. As stated previously, the coarse grid is to span the entire modeling space and the fine-level voxel grids are needed only within the coarse surface voxels that need refinement.

In the FSV-rep model, an integer ID is used to identify the bit in the bit-array corresponding to a specific voxel, thereby achieving access to any voxel for activation or deactivation with a constant computing time. Since a single bit is used for a voxel, the voxel ID has to be converted into a (byte ID, bit ID) combination as below for using bit operators to access the corresponding bit for a voxel:

$$byteID = floor\left(\frac{voxelID}{8}\right) \tag{7.1}$$

$$bitID = voxelID - byteID \times 8 \quad (7.2)$$

Since bit operations are very fast and the above conversions are simple, this does not pose a noticeable computational load in voxel bit access. On the other hand, the memory requirement to represent the voxel space reduce by a factor of 8 compared to using one byte per voxel.

For the surface voxelization of the original input workpiece shape, the bits corresponding to the surface voxels are set to 1 after setting all the bits to 0 initially (Figure 7-1). To facilitate the model update process, a volume voxel model for the coarse voxels is also needed which is generated by setting the bits corresponding to the voxels inside the model volume to 1 as well.

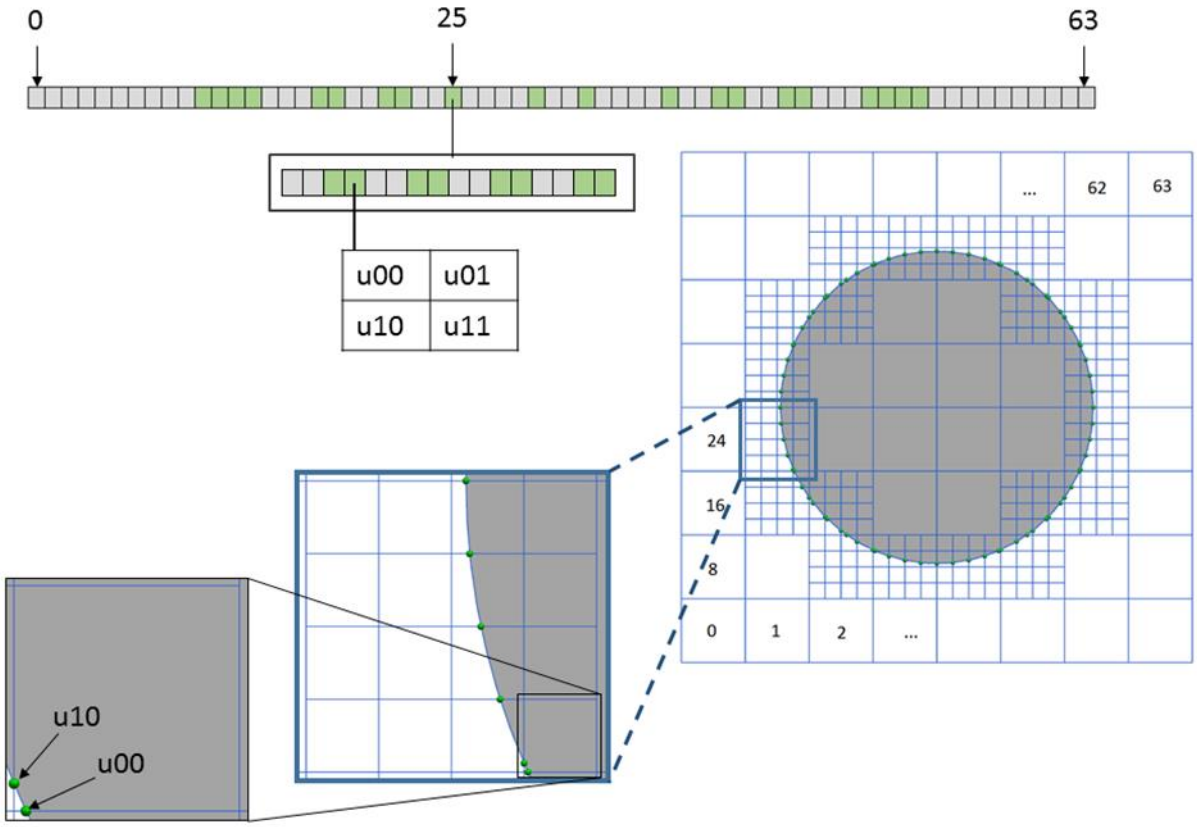


Figure 7-1 A sample 2D analogy of FSV-rep model and the corresponding bits and FC-points pair for a particular FS-voxel and its parent coarse surface voxel.

All the tool paths specified in the milling operation are processed to identify which tool

path can use SVRs and which shall use sampled instances. Those requiring sampling are sampled individually with the sampling distance sufficiently small to make sure that all of the affected voxels are included in the model update process as described in Chapter 6, Section 6.5.1. A tool path is to be defined by the trajectories of two points on the tool axis with one point being the tool tip and the other being the point along the tool axis at a particular height from the tool tip [51]. The sampling interval length on the tool path trajectories has to be set equal to or less than the voxel edge length in order to capture all of the affected voxels. The tool path is then sampled to follow this on both of the trajectory curves between two sampled tool instances. The list of tool instances and swept volumes as appropriate from all the tool paths for the entire milling operation is thus generated and used to update the FSV-rep model created for the original input workpiece.

During the update process of an FSV-rep workpiece model at the coarse level, the bits corresponding to the coarse voxels completely inside any tool instances or SVRs are set to 0 in the bit-array. The NF-voxel lists are created for the tool instances according to Algorithm 6-4 and Coarse surface voxels lists are created for SVRs according to Algorithm 6-1. Both of these list are inverted to obtain the mapping from NF voxel to tool instances passing by them, NFV$\rightarrow\{\boldsymbol{T_i}\}$ and the mapping from Coarse surface voxels to SVRs passing through them, CSV$\rightarrow\{\boldsymbol{SVR}\}$. Both these mappings are created as STL maps using binary search trees that has $O(\log n)$ complexity for element wise operations such as insertions and search. This ensures that the creation and later use of these mappings are optimal in the system.

Once the mappings are obtained, the bit-array representations of the fine-level voxel grid within each coarse voxel having an entry in one of the mappings is also updated similarly according to Algorithm 6-2 or 6-5. Using Algorithm 6-5 an active bit in the bit-array for the fine-

level voxel grid is set to 0 if the fine voxel is inside any of the tool instances in the list. Using Algorithm 6-2 similar operation is done for fine-level voxels within all the voxels having an entry in the CSV→{$\boldsymbol{SVR}$} mapping.

After the fine updates using all the tool instances and SVRs, the frame update is done for all the fine level surface voxels finally active within the NF or CSV voxels using the tool instances or SVRs from the corresponding {$\boldsymbol{T_i}$} or **{SVR}** lists.

Computing the FC-points for all the edges of a fine surface voxel is not an efficient task as the computation will be repeatedly done for the same edge from all the four incident voxels. To avoid the redundant computations, the FS-voxel holding the FC-points only on the primary edges is to be used. In essence, an edge is deemed as the primary edge with respect to only one voxel. As a result, every intersecting voxel edge will be the primary edge for just one of the fine surface voxels. In order to calculate the FC-points for a primary edge, a wire body corresponding to the portion of the primary edge inside the original workpiece volume is defined first. Then, a Boolean subtraction operation is performed on that wire body using the solid bodies of the tool instances or SVRs crossing the primary edge as the Boolean tools. The end points of the resultant wire bodies (excluding those coinciding with the voxel corners) are the FC-points for the primary edge. All these Boolean operations for FC-points computation are done with line-surface intersection operations.

7.2 Simulation cases with tool path sampling

Machining simulation using FSV-rep outperforms other voxel based methods in terms of computational performance as well other than accuracy and memory efficiency. This is intuitively apparent from the fact that with more elements to update, the computational time also

increases. All basic voxel modeling methods will require significantly higher resolution to achieve model accuracy comparable to that of FSV-rep. As noted in the Section 3.10.1 of Chapter 3, an octree sub-division of a fine-level voxel needed 10,920 octants in place of 4 FC-points of an FS-voxel to achieve the comparable accuracy. It will, thus, be much more time consuming as well to update such an octree sub-division compared to updating the FS-voxel.

A series of case studies have been carried out using sampled tool instances to demonstrate the improvement in the computational time of the present method based on the FSV-rep modeling to compute the milling part geometry as compared to that of the existing method based on the tri-dexel modeling. The tri-dexel method is employed as a comparison benchmark as it has been recognized as providing the best combination of modeling accuracy, robustness, and computational speed among the reported methods in the literature.

In all the case studies in this section, sampled tool instances are used for all the tool paths. A separate set of case studies is later done with SVRs in Section 7.3. This is to first understand the performance of FSV-rep vs Tri-dexel for same type of swept volume approximation in both cases. Sampled tool instances is the appropriate approximation equally applicable to both FSV-rep and Tri-dexels and also suitable for multi-axis milling simulation. Similar to the way the FSV-rep workpiece model is updated, the tri-dexel workpiece model is also updated using sampled tool instances along the milling tool paths. However, as there had been no development of multi-level representations of tri-dexels, the model update had to be done at the finest resolution with incremental updates of the affected dexels with each sampled tool instance.

Three basic case studies were devised to illustrate the increasing complexity of the milling tool paths (Figure 7-2). In all these cases, the machining was done by flank milling with a flat-end mill of 12 mm diameter. Only one flat-end mill was used to make sure that the computing

time results were not dependent on the tool type but on the tool paths and tool orientations. In Case I, the tool axis was always vertical along the tool paths in the machining of the T-section part. The bounding box for each tool instance was the minimal in this case. Case II used a tilted tool with a constant orientation along each tool path and the tool axis being parallel to one of the axial planes of the workpiece coordinate system. The tool bounding box became larger in this case. Case III had the tool axis changing along each tool path and thus attaining an arbitrary 3D orientation. Compared with cases I and II, case III had the largest bounding boxes for the involved tool instances. Here Cases II and III involve multi-axis toolpaths.

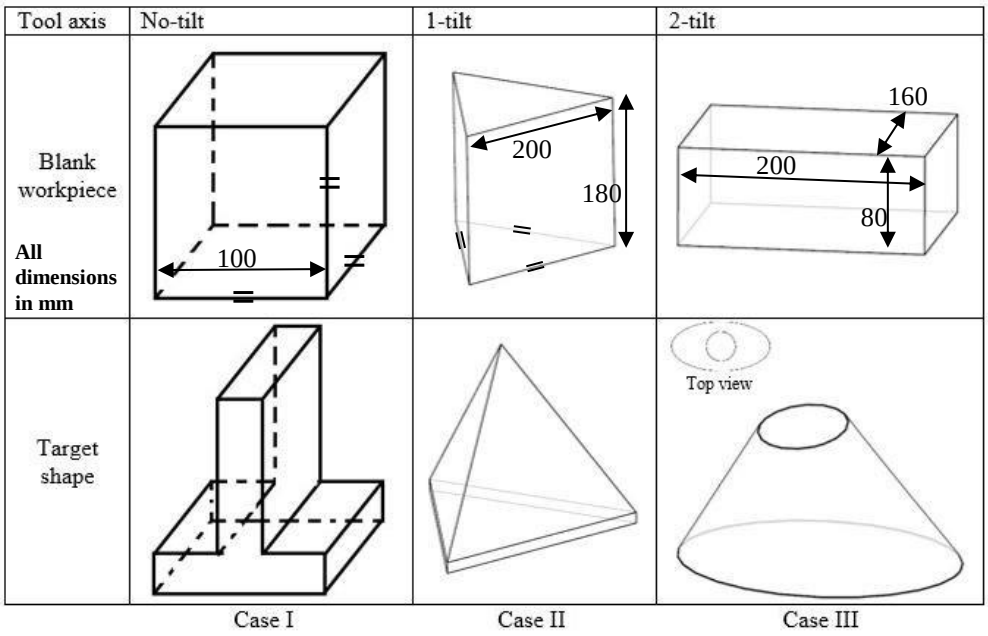


Figure 7-2 Basic case studies: (I) fixed vertical tool orientation; (II) fixed tool orientation but tilted in one axial plane; and (III) arbitrary and varying tool orientation.

Figure 7-3 shows the computational time of the FSV-rep and tri-dexel methods for the three basic milling cases. It can be seen that the FSV-rep method gives faster performance in all the three cases and the faster performance is more pronounced from case I to case III. The improvements are primarily from two factors as shown in Figures 7-4 and 7-5. Figure 7-4 shows

the execution time for simulating the machining of the T-section (case I) part with the increasing value of the total axial depth of cut h_c. For very small values of h_c, the tri-dexel method is faster as the multi-level coarse and fine update of the FSV-rep method does not have much advantage. However, after h_c is larger than the coarse voxel grid spacing, the FSV-rep method becomes faster and as h_c further increases, the advantage of the FSV-rep method becomes evident. The FSV-rep method achieves this via the collective volume removal by batch processing at the coarse voxel level first before moving to the fine voxel update and FC-points computation. This facilitates the bulk material removal simulation at a much faster rate as compared to the sole procedure of intensive intersection calculations of the tri-dexel method to reach the final machined surface geometry. With the coarse update and identification of the NF-voxels of the FSV-rep method, only those coarse voxels in the vicinity of the final machined surface are considered for the subsequent fine and frame update steps. As a result, fine surface voxels and FC-points are computed only within the coarse voxels relevant to the final machined part surface.

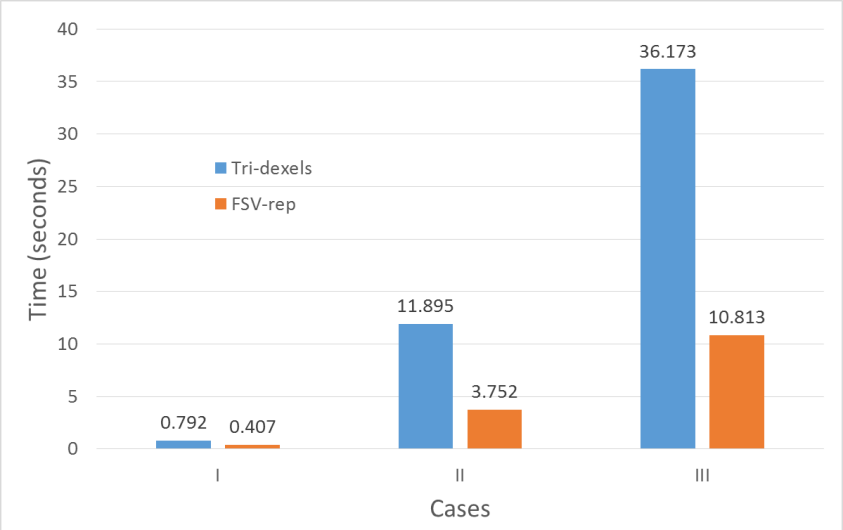


Figure 7-3 Execution time comparison for computing the machined part geometry.

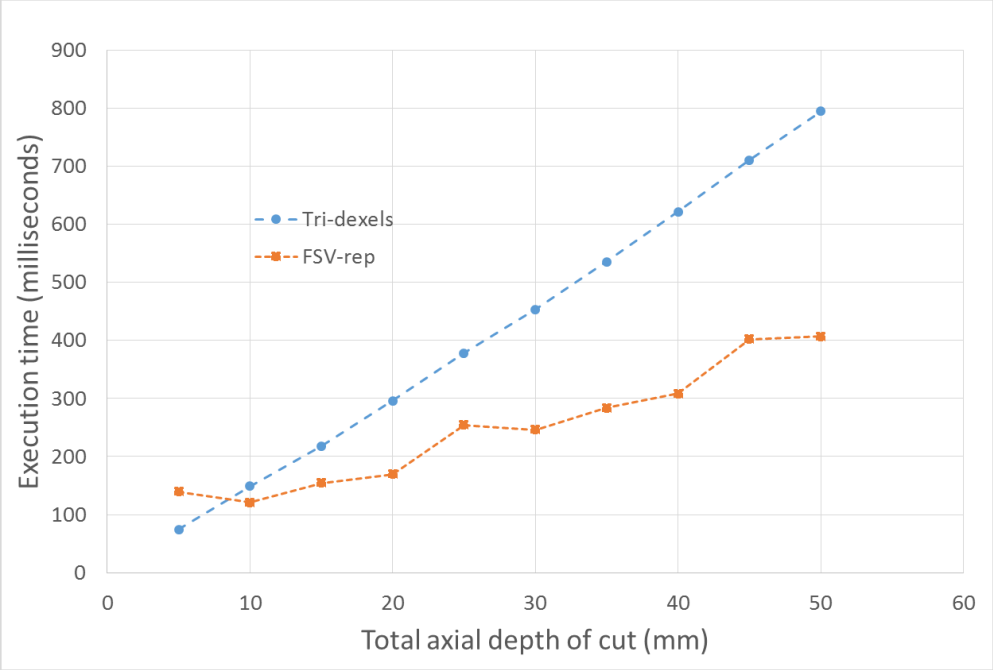


Figure 7-4 Execution time with the increasing total axial depth of cut for the T-section part.

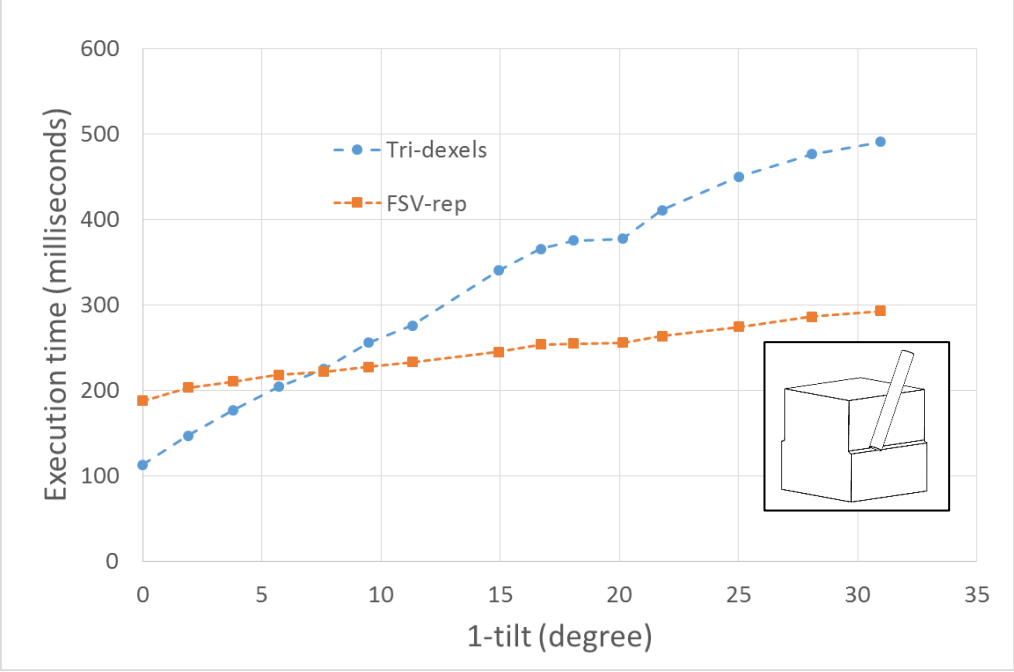


Figure 7-5 Execution time with the increasing forward tilt of the flat-end mill in the half-immersion side cuts for the T-section part.

An analysis has been performed to know the separate execution time for the coarse, fine and frame update steps in order to have a better understanding of the proportional workload of the three different steps. As seen in Figure 7-6, most of the execution time is spent on the coarse update. Since the coarse update step mostly involves a simple binary marking/unmarking operation, the large proportional workload gives the reason to the much faster performance of the FSV-rep method. The results also confirm that to obtain the FS-voxels via the frame update to yield the higher model accuracy, the computational time needed is relatively acceptable after the coarse and fine updates.

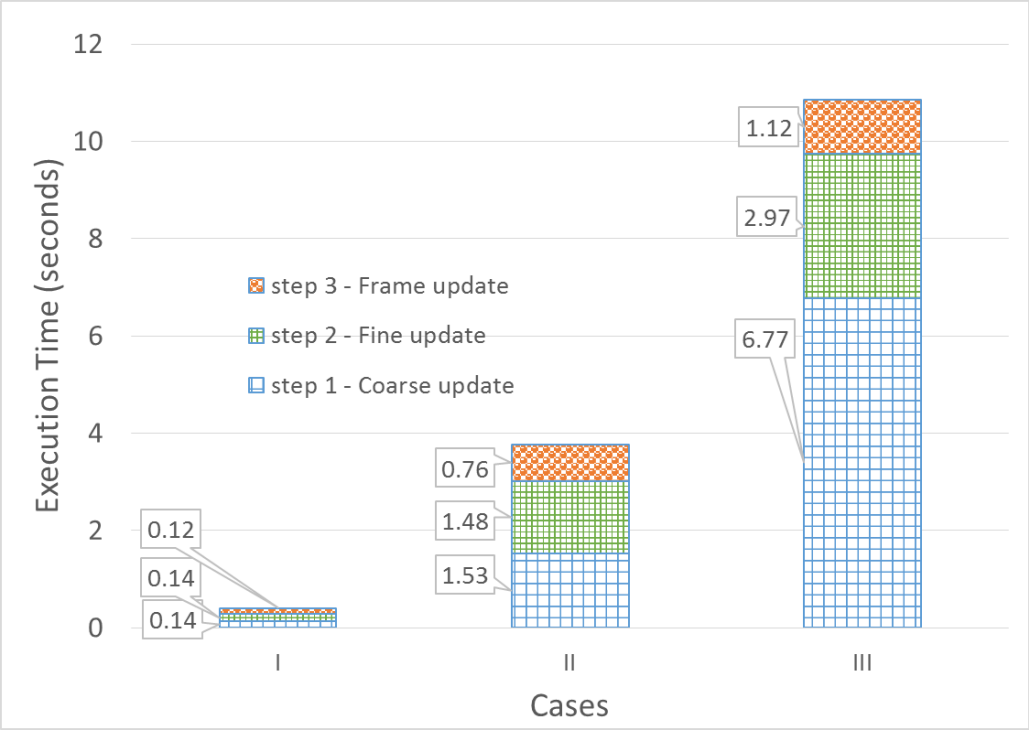


Figure 7-6 Time-splits among the coarse, fine and frame update steps in the FSV-rep method.

Figure 7-5 illustrates the second factor contributing to the observed performance improvement of the FSV-rep method. It shows the execution time for simulating the machining

of the T-section part with the width of the side cuts being only half of the tool diameter. The machining simulation was done for different values of the forward tilt angle of the flat-end mill along the tool path. It should be noted that the side cuts were completed using only one half-immersion tool pass with no tool path overlap. Hence, there is no advantage present for the FSV-rep method from the aspect of bulk volume removal. As can be seen from Figure 7-5, after a particular forward tilt angle of the tool axis, the FSV-rep method becomes faster than the tri-dexel method and the time difference gets bigger with further increase in the tool tilt. This is due to the volume increase of the tool bounding box as the end mill becomes more and more tilted. With a larger bounding box, more elements (dexels or voxels) need to be considered and processed. Nonetheless, in the case of the FSV-rep method, rapid check at the coarse voxel level is attainable and hence, the effect of the increased bounding box volume is much less. Furthermore, only point-to-tool distance classifications are involved in the identification of near-field and surface voxels whereas for the tri-dexels, actual intersection calculations on those dexels covered by the bounding box are needed to even confirm the intersections.

Figure 7-7 depicts the matching of the FC-points determined in the FSV-rep method with the end points of the dexel line segments from the tri-dexel method for cases I and III with two representative zoom-in views. Close to perfect matching was obtained with virtually all of the FC-points coincident with all of the tri-dexel end points except for some rare cases. Specifically, all of the tri-dexel end points were attained by the FSV-rep method in case I, and only 6 out of 103,270 tri-dexel end points were not attainable via the FSV-rep update process in case II and 4 out of 60,448 unattainable in case III. The minute difference is caused by an implementation restriction in the FSV-rep method which limits a maximum of two FC-points to be stored on a voxel edge. The restriction is put in place for easier data management and subsequent

identification of the surface orientation in reconstructing the triangle mesh from the FS-voxels. The number of mismatch is seen to be fairly insignificant in general as noted from the extensive computational tests.

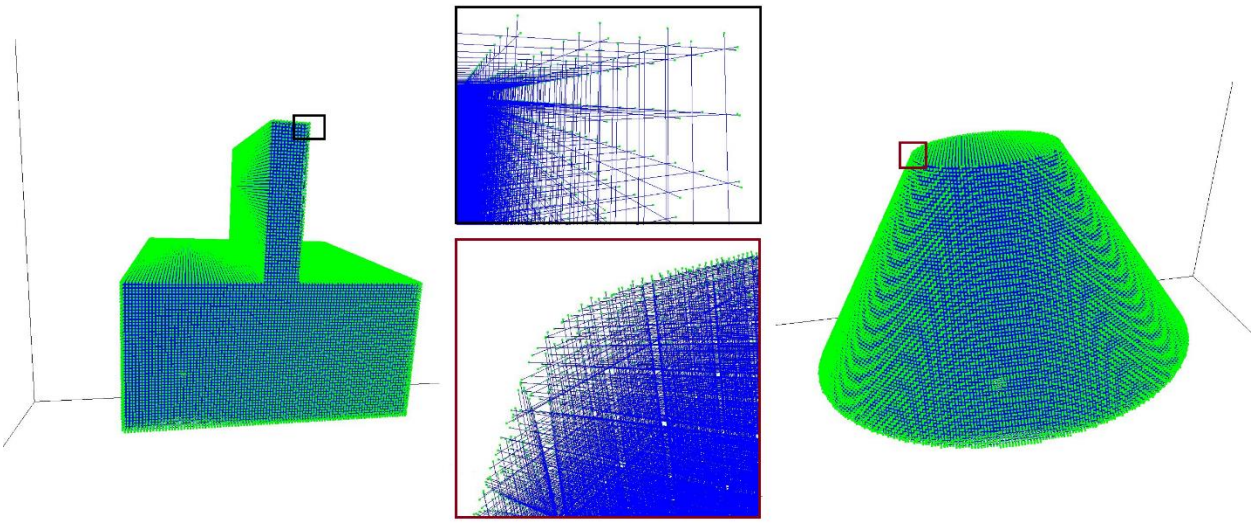


Figure 7-7 Matching of the FC-points (green) from the FSV-rep method with the end points of dexels (blue lines) from the tri-dexel method for case I (left) and case III (right).

Generation of a triangle mesh surface for the machined part geometry is quite straightforward from an FSV-rep model as we will see in Chapter 8. The triangle mesh surfaces obtained for the machined part geometry in the three test cases are shown in Figure 7-8. The meshes are all of good quality and thus useful for the visual verification of the associated machining operations. More importantly, the meshes will be useful when preforming a quantitative comparison against their reference design models for identifying potential machining errors such as gouging and undercuts. It should be pointed out, however, that the triangle mesh models obtained do not have sharp machined edges between faces. The improved accuracy of the FSV-rep model over the basic voxel model is due to the triangle mesh surface

generated from the FS-voxels and the associated FC-points. The FS-voxels are still not sensitive enough to capture the sharp machined edges and corners that are not coincident with the voxel edge-frame. This is in fact a well-known issue for the discrete dexel or voxel representations. Since the deviation is only along the sharp edges of the machined part, this is a localized issue and only affects a relatively small area of the model. It can thus be easily resolved by a variety of triangle mesh processing methods, for example, the method developed and demonstrated by Ren et al. [67] or Wang et al. [71].

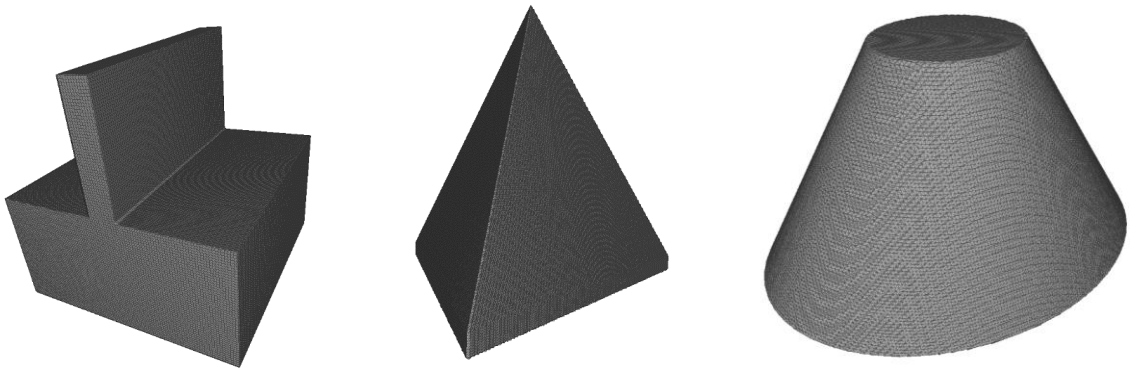


Figure 7-8 Triangle mesh surfaces generated from the FSV-rep models.

The case studies presented above only involve workpiece model updates with a flat-end mill. Nonetheless, the overall model update process is general and all types of milling cutters can be used. As the use of sampled tool instances along a tool path for the workpiece model update represents an approximation to the exact tool swept volume, it will result in a series of 'sampling scallops' left between sampled tool instances as we saw from the discussion in Chapter 6, Section 6.5.1. It is evident from Figure 7-8 that with a conservative value for the sampling interval length, the resulting sampling scallop size will be relatively small and not visible.

The basic cases considered above were all creating convex shapes. In order to demonstrate the machining of concave surfaces, two pocket machining cases are done. Table 7-1 shows the simulation time for Cases 1 and 2 from Chapter 3 to compute the final machined part geometry. It also shows that FSV-rep is up to 2 times faster than tri-dexels. Here, Case 1 is machined with simple 2½-D linear tool passes whereas Case 2 uses five-axis contour machining for the conical side wall. Both machining cases were simulated with a flat-end mill of 12-mm diameter. The level-based 2½-D milling of Case 1 (machining a $80 \times 80 \times 40$ mm^3 square pocket) used a 5-mm cutting depth per level. Each level was completed via contour-parallel milling tool paths with a 2-mm side step. For Case 2 (machining a cone-shaped pocket of 40-mm depth, 80-mm top diameter and 60-mm bottom diameter), the employed milling operations included: (1) level-based zig-zag 2½-D milling tool paths for roughing as well as for forming the bottom face; (2) a circular tool path to finish the perimeter of the bottom face; and (3) five-axis surface contouring tool paths for finishing the side wall. The zig-zag roughing tool paths used a 4-mm side step. The surface contouring was also completed in multiple steps of 5-mm vertical depth each.

Table 7-1 Comparison of simulation time for FSV-rep and tri-dexels in pocket milling.

Case Study	Simulation Time (ms)		Ratio (FSV-rep/Tri-dexels)
	Tri-dexels	FSV-rep	
1	550	399	0.72
2	3,253	1,564	0.48

In order to demonstrate the applicability of FSV-rep based simulation in a real industrial scenario, another case study was done to compute the in-process workpiece (IPW) geometry of an integrally bladed rotor (IBR). The case study was to obtain the IPW geometry after the machining operations to create one blade. Figure 7-9 shows the blank workpiece as the initial

input and the IPW geometry as a triangle mesh generated from the updated FSV-rep model. The case study involved three milling operations using three ball-end mills (with roughing tool diameter of 13 mm and two finishing tools of 7 mm and 6 mm diameter) and involving 41,616 tool motion commands. The tool motions were mostly multi-axis. The simulation execution time for both FSV-rep and tri-dexel based IPW generation is listed in Table 7-2. It can be seen that the execution speed of FSV-rep is about 2.3 times faster than that of tri-dexels for this case. The portions of execution time spent for the three FSV-rep model update steps are also given in Table 7-2. The time split across the three update steps is consistent with the general trend observed for the basic case studies in Figure 7-6. Also, it is worth noting that the improvement in execution speed for FSV-rep is mainly from the tilted tool orientation as depicted in Figure 7-5. The other factor due to bulk volume removal as depicted in Figure 7-4 has less effect here. This is because the machining operations created more surface area per unit volume removed, thereby effectively having less bulk volume removed. Nevertheless, contributions from both factors give a combined faster performance.

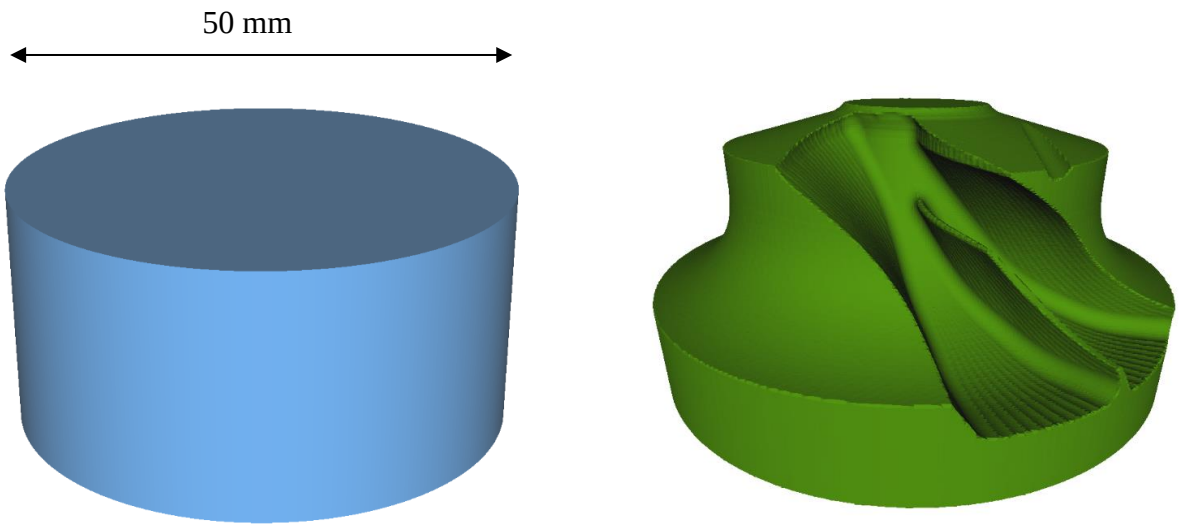


Figure 7-9 Industrial case study: (a) blank workpiece; and (b) in-process workpiece of an IBR with one blade machined.

Table 7-2 Execution time comparison for the industrial case study.

Modeling Method	Execution Time (Second)	
Tri-dexels	Total	28.198
FSV-rep	Coarse update	4.731
	Fine update	4.866
	Frame update	2.516
	Total	12.113

As for model accuracy in terms of the sample points on the machined part surface, the FSV-rep based IPW in the above case is very much comparable to tri-dexels with only 830 out of 196,794 dexel-end points not matched with the FC-points in the FSV-rep model. The discrepancy is higher than that observed in the basic cases. The reason is mainly due to the relatively large scallop areas produced by the ball-end mill. The tip of the scallop may create a small hanging voxel frame segment (shorter than the edge length of the fine-level voxel in FSV-rep and not attached to any voxel corner point). These hanging frame segments are ignored in FSV-rep if no other portion of the edge-frame of that particular FS-voxel is active. This is permitted as a computational compromise in the implementation of the FSV-rep model update. It leads to the small difference of only 0.42% in the complex industrial case. Ignoring such small hanging segments does not create much impact on the model geometry. Also, any sharp features that are lost due to this will be restored via post-processing the generated triangle mesh.

7.3 Simulation cases with SVRs

In order to demonstrate the improvements SVRs can bring to FSV-rep simulation, different

cases are shown in this section that employ a certain type of tool path alone in each case but with increasing complexity of the SVRs involved. For each case the performance is compared with FSV-rep update using sampled tool instances in each case. As SVRs based update is specifically designed for improving FSV-rep update in case of simple tool paths, it is appropriate to compare its performance with FSV-rep update using a conventional approach. And, if it were compared with other methods such as Tri-dexels updated by tool instances or by another swept volume approximation appropriate for them, it would not be a pure one-to-one comparison. These aspects justifies the evaluation of SVRs update performance by comparison against sampled tool instances used for FSV-rep itself.

Since it is to demonstrate the performance improvement in case of pure translational tool paths, each case study devised is to involve purely of such tool paths. As shown in Figure 7-10, case 1A is to machine the T-section part that was used as a case study in previous section as well. This case involves 1-axis tool paths alone and requires only flat end mill. Hence this is the first and the simplest case study. Case 2A is a rotated T-section and require all tool paths to be involving 2-axis motions. Case 2AV is again involving flat end mill and 2-axis motion but in this case, one of the axis of motion is along the vertical tool axis direction. This change the envelope surfaces for the bottom to an elliptic cylinder and causes the associated SVRs to have more complex boundary elements. Case 3A involves a ball end mill moving along 3-axis tool paths and has the most involved operations for the update steps among the four cases.

From the resulting machined part geometry after FSV-rep simulation in each case using sampled tool instances along the tool paths and using SVRs shown below in Figure 7-10, it is visually noticeable that the SVRs based update is improving the generated surface quality as it no longer has the sampling scallop. The computational performance comparison in terms of the

execution time is also shown in Table 7-3 to further emphasis the relevance of SVRs. The computational speed increases by up to an order of magnitude in case of 1-axis and 2-axis machining cases.

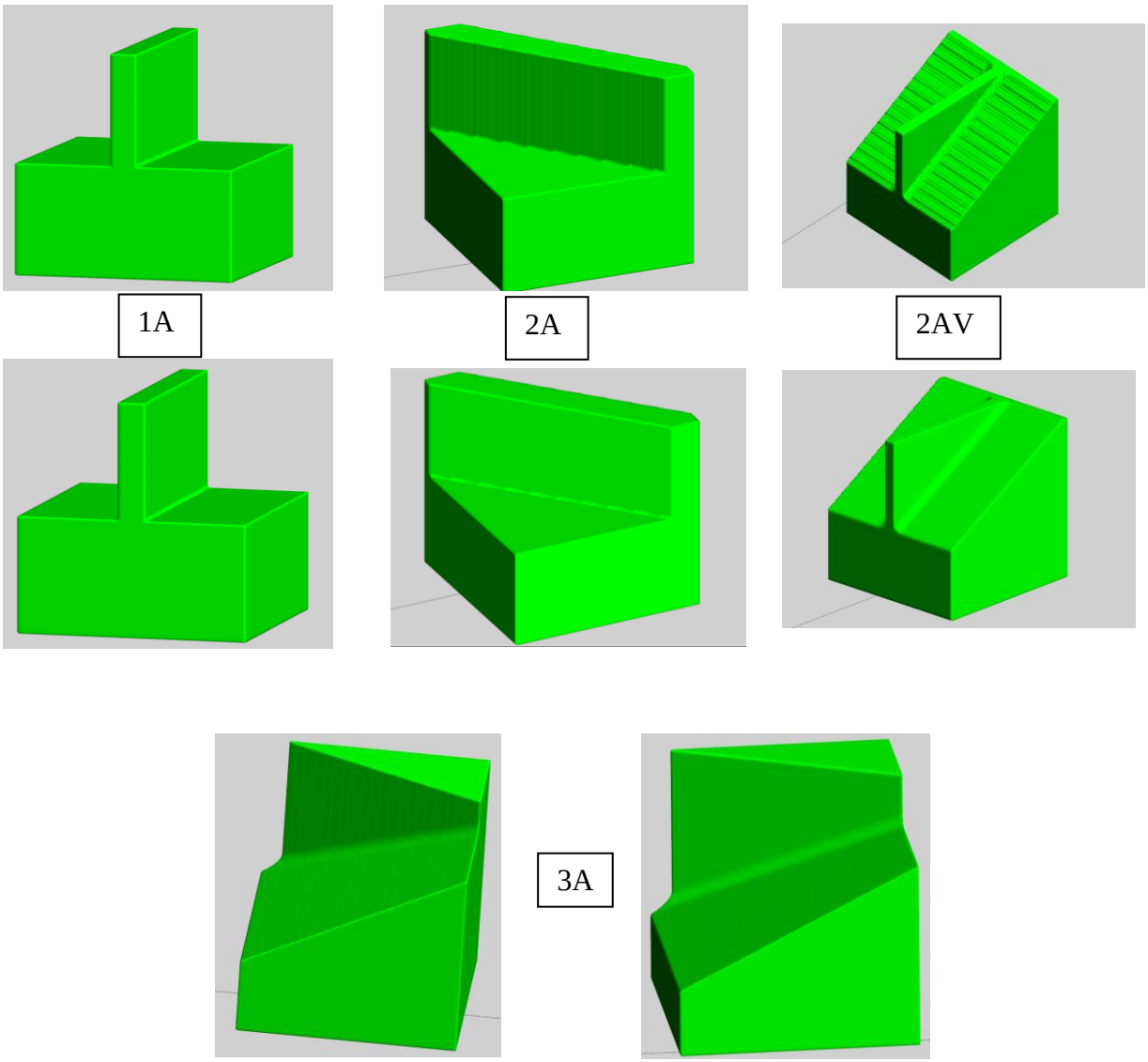


Figure 7-10 Results of case studies to compare performance of FSV-rep with SVRs (bottom figure for 1A, 2A and 2AV and right side figure for 3A) instead of sampled tool instances (the other figure in each case).

It is worth noting that the performance improvement measured as ratio of time for update

by sampled instances and by SVRs shows a decrease in value from case 1A towards case 3A. This is as expected as the involved envelope surfaces and SVRs boundary elements increase in complexity in that direction.

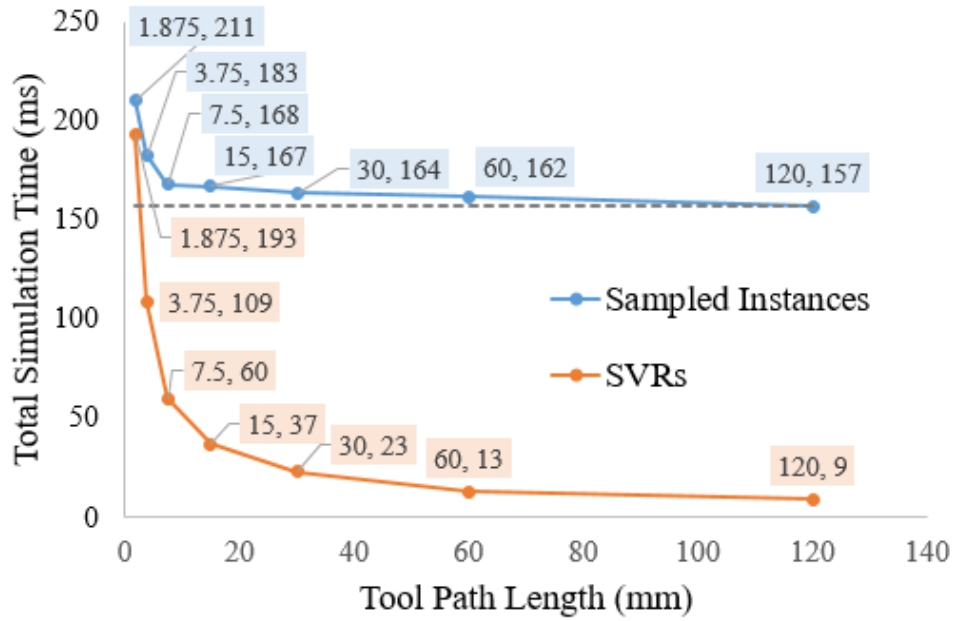
Table 7-3 Execution time comparison for cases shown in Figure 7-10.

Case	sampled instances (T1 ms)	SVRs (T2 ms)	T1/T2
1A	364	18	20.50
2A	409	27	14.95
2AV	186	17	10.91
3A	192	27	7.23

Another aspect to evaluate is the change in the ratio of the computation times for a single case itself but with reducing length of the tool paths involved. This study is done for 1-axis and 2-axis tool paths by repeating each with tool paths bisected for every iteration. Thus, each has the same machining work load but has double the number of shorter tool paths used compared to its previous case. As shown in Figure 7-11, SVRs is faster for both 1-axis and 2-axis cases even for small tool paths. There is however one unhandled aspect for the sampled tool instance approach that is partially responsible for this consistent outperformance of SVRs as explained next.

The sampled tool instances approach loses part of its speed when the coarse update is done at sampling interval shorter than the coarse grid spacing. Ideally, the total tool path length over consecutive tool paths should be used for the sampling purpose when all the tool paths are sampled. However, the results in above figures are using independent sampling for each tool path. This causes the sampling approach to slow down once the tool path length reaches the coarse grid spacing.

(a)



(b)

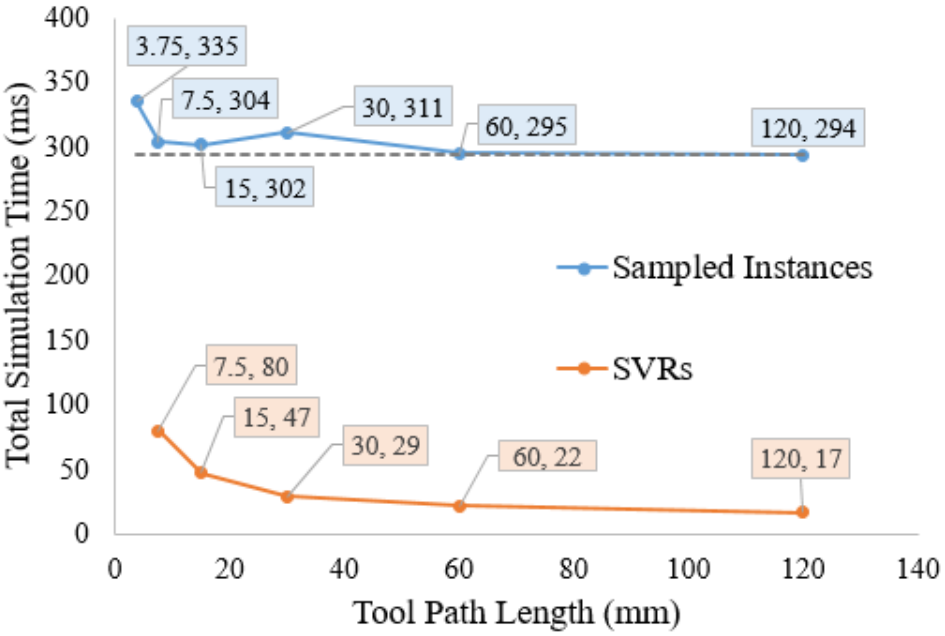


Figure 7-11 Performance comparison between FSV-rep update with sampled instances and SVRs for different length per tool path. (a) in 1-Axis (b) in 2-axis

We can however make a simple approximate measure from the existing chart itself to understand the performance of sampled tool instances approach with sampling considering the cumulative tool path length. When the sampling is ideal using cumulative path length, the performance of the sampled instances approach should be independent of the individual path length barring some minute overheads. The computational time of the case with the original longest tool paths itself will be the time for sampled tool instances approach at each iteration. Thus, the horizontal dashed line shown in each graph is a very good approximation for performance of sampled tool instances with cumulative sampling.

The SVRs methods does cross the sampled instances approach with ideal cumulative sampling as seen in the chart. This indicates, that SVRs is computationally favorable when the tool paths are of some minimum length. The SVRs are still useful from accuracy point of view at all tool path lengths nevertheless.

However, the approach of obtaining the SVR→{$\boldsymbol{CSVs}$} mapping in the coarse update part 2 using the sweeping plane we developed in Chapter 6, Section 6.4.1 is applicable only when the path length is large enough to avoid the boundary elements at two extreme ends along the F_{dir} direction occurring in the same scanning section. For shorter tool paths, some SVR types can vanish and requires more rigorous identification techniques and may not be feasible at all. Thus we can only tell confidently that the SVRs based update is faster than sampled instances approach when the individual path lengths are larger than a particular threshold which is different for each tool path category and tool type. Nevertheless the observed threshold minimum tool path length is small enough to enable SVRs to advantageously handle a wide range of planar straight cutting roughing operations.

As it is seen from the 2-axis case that the SVRs generation technique is applicable only

after a threshold path length, the analysis of the trend in case of 3-axis linear tool paths for different path length is irrelevant. This is because, long 3-axis linear paths are not common in machining operations. Only applicability of SVRs for 3-axis linear paths would be to approximate 3-axis helical and freeform paths as many linear 3D segments and apply SVRs based update for those linear segments. This could improve the accuracy by reducing the sampling scallop. However, the minimum threshold length of tool path is greater than the tool radius as per observation. As the linear segments needed to approximate the curved tool paths are much shorter, the SVRs approach would not be applicable for that purpose. Hence the valuable information we can have is that the SVRs approach is applicable to majority of the tool path lengths in planar straight cutting operations.

7.4 Summary

The case studies in this chapter has shown that FSV-rep model update with sampled tool instances outperforms the best known methods using Tri-dexels. It is primarily because FSV-rep model update can utilize the coarse update for faster bulk volume removal and the frame update for easy addition of FC-points on to the fine level surface voxels. For simple linear three axis tool paths, the FSV-rep model update could be made further efficient and more accurate with the use of SVRs.

Chapter 8: FSV-rep surface generation

FSV-rep is a model exclusively for simulation purpose. It is designed such that the characteristics suit efficient and accurate update during simulation of a machining process. However, the quality of the model can be appreciated completely only when a visual representation is available. The need of a visual representation is crucial in case of process verification as visual inspection of the simulated part geometry is an important step in validating the process. Thus, in this Chapter we develop a method that can generate a model representation that can be rendered from the FSV-rep.

8.1 Requirements and objective

Triangle mesh models are widely used in graphics and display modules for its suitability to render any geometry with acceptable accuracy. The constituent elements being many triangles and often small in size enables the rendering logic to perform many of the operations in parallel on multi core Graphics Processing Units (GPU). This is very beneficial as rendering is somewhat a compute intensive operation that can take significant time if done sequentially for many large geometric elements. Thus, in this work also, the aim is to ensure that a triangle mesh model can be created from FSV-rep workpiece without much additional processing requirements.

Generating a triangle mesh from FSV-rep has other merits as well. Triangle mesh is a well-established representation format for which analysis and processing algorithms have been developed and improved to a mature level after decades of research and work in the area. Many of the analysis steps required to evaluate the machined part geometry obtained from simulation can make use of the available methods if the model is represented as a triangle mesh. Machining error analysis by surface comparison is one such activity. Data transfer to other systems and

reverse engineering to make design changes is another scenario that can make use of a triangle mesh of the machined part geometry. Thus, the generation of a triangle mesh model from the FSV-rep workpiece after simulation is a very beneficial step.

In order to cater to the different purposes listed above, the generated triangle mesh should be satisfying various conditions as identified in Chapter 3, Section 3.7. Being a closed 2-manifold is one basic requirement so that the mesh is acceptable for downstream operations such as error analysis. Post processing operations on the mesh model in order to restore sharp details that is required for the surface models obtainable from FSV-rep model will also benefit from a 2-manifold mesh [71].

As we saw from the review done in Chapter 2, Section 2.4, the existing algorithms for mesh generation from a grid structured information have shortcomings or do not have direct applicability to the data FSV-rep representation provides. Thus, this chapter develops a technique specifically to serve for mesh generation from an FSV-rep model. It is inspired by the classic marching cubes [60] technique and is indeed an extended version suitable for models generated by machining operations. There are certain assumptions that can be made about such models as we will see in Section 8.2. Those assumptions help to resolve many ambiguous cases that arises while using the technique. Later in the results of Section 8.7, we will see that the assumptions are indeed safe to be made and does not create any global deviation for the shape.

8.2 Assumptions

We can assume that there are no thin “through gaps” in the shape such as shown in Figure 8-1. These are gaps with the gap width smaller than the edge length of the fine level voxel. The machining cases we are to deal with are limited to use a set of tools which have some

characteristic size parameters. The parameter D pertaining to the "tool diameter" is the major dimension we have to take into consideration. It is the only radial size factor present for the major milling cutters like flat , ball and bull nose end mill. For other major types like taper ball end mill this parameter still provides the lower limit of the radial size of the tool except towards the tip.

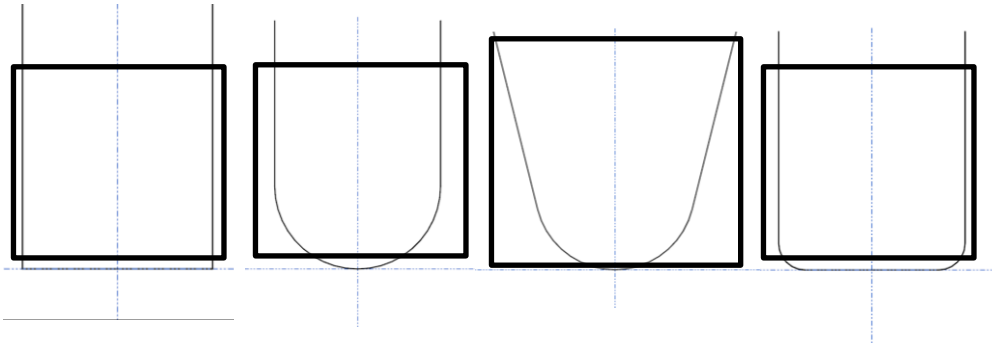


Figure 8-1 Through gaps created by different milling cutters inside voxels of comparatively large size.

Even at the tip of the tool, the tool has no way to create a through gap of width smaller than the fine level voxel edge length. If at all there can be a through gap feature that is machined for a voxel, it will be of width at least equal to a minimum value based on the tool type as below:

$$w_{taperGap} = \frac{2R(1-sin\alpha)}{cos\alpha-2sin\alpha} \text{ for taper ball end mill} \quad (8.1)$$

$$w_{gap} = D \text{ for flat, ball and bull nose end mill} \quad (8.2)$$

The above expression shows the voxel size has to be greater than or equal to the main diameter of the tool for a through gap to be formed. From Chapter 6, Section 6.5.1, we have decided the coarse grid spacing based on D (Equation 6.5). The fine gird spacing and hence the fine level voxel size are definitely smaller based on the subdivision factor (Equation 3.6). All the

triangulation is done within each fine level surface voxel as we will see in the following sections. Since the voxel size is much smaller than the characteristic tool size *D*, it is safe to assume there are no through gaps of width smaller than the edge length of the fine level voxel. This is further supported by the value we get for the gap width, w_{gap} in terms of tool radius and subdivision factor for a single voxel layer penetration along the axial direction of a ball end mill:

$$w_{gap} = 2R_t\sqrt{1-\left(1-\frac{1}{f}\right)^2} > \frac{R_t}{f} \tag{8.3}$$

Other assumptions that we make without affecting majority of the machining cases are as below:

1) The partial gaps and floating segments on the FS-voxels (as shown in Figure 8-2) can be ignored. This is not a major feature of the machined surface and ignoring such gaps does not alter the global surface a lot.
2) The surface of the machined part is such that the valid feature edges are not many compared to the smooth surface area. This means the sharp features and thin faces are less compared to smooth areas. This is indeed the case in most of the mechanical parts with flat, freeform or blended surfaces.
3) The thin features are wide enough to be occupying more than one fine level voxel along the feature width dimension. This can be assured by setting an appropriate fine grid spacing based on the thin features that could be machined or present on the workpiece.

8.3 Input features

There are certain features of FSV-rep model that can be used favorably to generate a

closed 2-manifold mesh model. First of all, with the composition of independent FS-voxels making up the boundary of the model, it is potentially possible to perform a triangulation on a per FS-voxel base. Further for each FS-voxels there are the frame details which can be used as further aid in the process. Of the frame details, it is the FC-points present on the frame edges that is essential here. Moreover, the FC-points are stored as a pair for each frame edge, which can be used to infer the orientation of the surface patch within the voxel.

Since each frame edge can have up to 2 FC-points and there are 12 frame edges for a voxel, there can be 2^{24} theoretically possible configurations for the FC-points collection of an FS-voxel if just the status of the FC-point of being present or absent is only considered. Alternatively, if we take the configuration possible for each frame edge independently, there are 6 configurations as shown below in Figure 8-2. Thus, with total of 12 frame edges for an FS-voxel, there are 6^{12} theoretical configurations for the frame edges segments forming an FS-voxel.

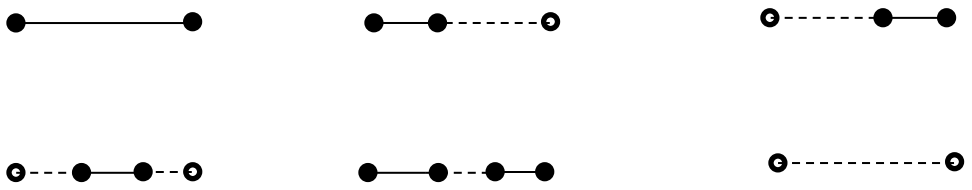


Figure 8-2 Six possible frame edge configurations possible with maximum two FC-points.

But in practice the actual configurations that can occur is limited from the following condition:

> **Condition 8-1**: For a corner point of an FS-voxel, and considering only the frame edge segments on that particular FS-voxel frame, there will be either three frame edge segments connected to it or none.

This condition occurs from the more global condition we see when all the 8 voxels sharing the corner point is considered together. There are 6 voxel edges incident on that corner point. For FS-voxels corresponding to the 8 voxels, there can be only 6 frame edge segments or none connected to the corner point. This condition leads to Condition 8-1 when applied together with fact that each voxel incident at the corner point has 3 of its edges connected to it. Here "voxel edge" is a fixed line segment between two voxel corner points. "Frame edge segments" are portions of it that can be active or inactive. "Frame edge" corresponding to a "voxel edge" is composed of all the "frame edge segments" lying on the "voxel edge".

Condition 8-1 gives us an alternate way to look at the FS-voxel: from the perspective of the corner points as active or inactive. Irrespective of the condition of the frame edge in between two corner points, the corner points can only be either active with three edge segments of the FS-voxel in consideration connected to it or inactive with no edge segments connected to it. Thus with 8 corner points there are 2^8 =256 configurations with respect to the corner point status.

However, each of the 256 configurations correspond to many frame segments configurations even with the practical limitation from Condition 8-1. For instance, a simple configuration with all corners inactive has many frame segment configurations as shown in Figure 8-3.

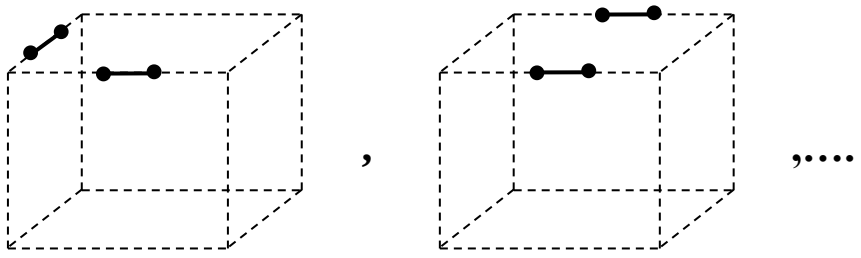


Figure 8-3 Different possible configurations for an FS-voxel with all corners inactive.

In fact this is the case for all of the 256 configurations based on corner point status. It arises from the fact that for a given frame edge, with the corner points at its ends given a status there are more than one possible configurations for the frame edge segments in between as shown in Figure 8-2 above. However, the total number of 6^{12} for the possible FS-voxel configurations is without considering the practical limitation from Condition 8-1. We can compute the total practical number of FS-voxel configurations possible in a more elaborate approach as shown below.

For a given configuration of corner point statuses, we can designate the frame edges as $type_{00}$, $type_{01}$, $type_{10}$ and $type_{11}$ types. Here $type_{00}$ has both end corner points inactive. $type_{01}$ has start point inactive and end point active. $type_{10}$ has start point active and end point inactive and $type_{11}$ has both end points active. The 6 configurations of frame edges can be mapped to the types as shown in Figure 8-4.

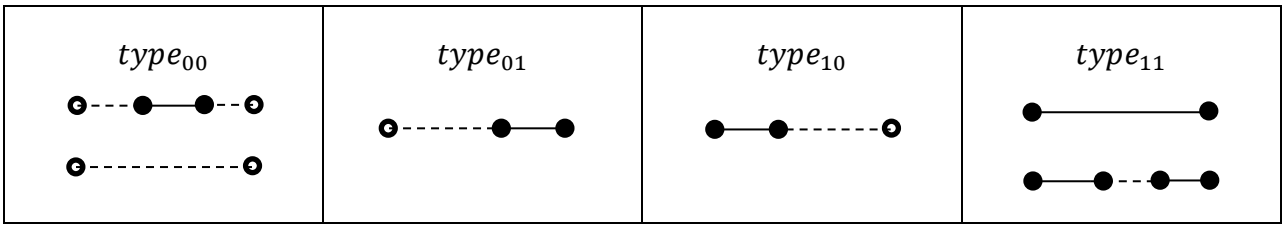


Figure 8-4 Frame edge configurations mapped to the edge corner status types.

Thus $type_{00}$ and $type_{11}$ types frame edges has two possibilities each and $type_{01}$ and $type_{10}$ types has one each.

Now for a given corner points configuration for an FS-voxel, the number of frame edges falling into each type is fixed and can be counted as n_{00}, n_{01}, n_{10} and n_{11} where $n_{00} + n_{01} + n_{10} + n_{11} = 12$. Thus, the total number of FS-voxel frame segments configurations corresponding to this particular corner points configuration is essentially

$$n_{FSconfigs} = 2^{n_{00}} \times 2^{n_{11}} \times 1^{n_{01}} \times 1^{n_{10}} \quad (8.4)$$

Thus, running Equation 8.4 for all the 256 corner points configurations, the total practical number of frame segments configurations is

$$nTotal_{FSconfigs} = \sum_{i=1}^{256} n_{FSconfigs,i} \quad (8.5)$$

where $n_{FSconfigs,i}$ is the number of frame segments configurations corresponding to ith corner points configurations.

Actual computation of Equation 8.5 leads to the number as 36,450. This is less than the value of 6^{12} predicted previously though still a large number.

Hence to have a definite situation which can handle almost all of the practical machining cases, following two simplifications are applied to the frame edge segments:

1) Gaps are filled
2) Floating segments are deleted

With the above simplifications, we have a situation where there is only one frame segments configuration corresponding to each of the 256 corner status configurations. This is because the number of variants for $type_{00}$ and $type_{11}$ types have also reduced to 1 each and Equation 8.5 gives $n_{FSconfigs}$ as 1 for all the 256 corner point configurations.

The above simplifications are justified by some other aspects as well: The triangulation of the slice fronts within an FS-voxel with gaps in the frame edges is ineffective without further information such as the surface normal or so. The features causing such gaps and also the floating segments are in fact insignificant as they only add the minor dents and sharp features. As the fine level voxel size is already about 12.5% of the tool diameter with a subdivision factor of

4, the details we may lose from making the above simplification are minor. The case studies in the later Section 8.7 will be validating this justification. Also it is challenging to preserve the floating frame edge segments which will require advanced algorithmic or excessively extended treatment of the configurations possible.

8.4 Look-up table definition

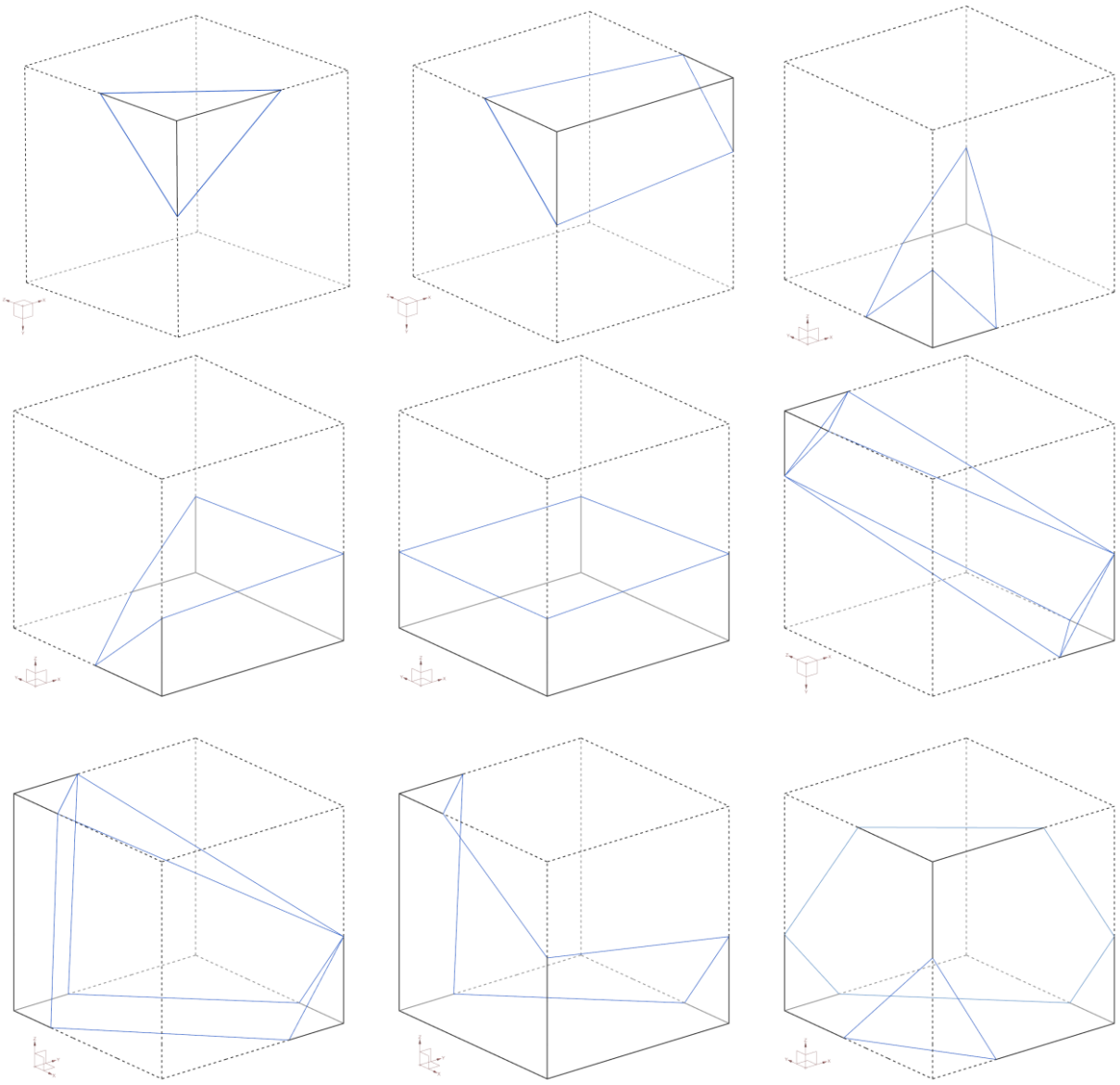
From the review done in Section 2.4 of Chapter 2 we have seen both look-up table based and algorithmic approaches for generating triangle mesh from grid based models. The look-up table based approach currently available (MC-15 and its improvements) are in fact very efficient as they have a direct triangulation ready once the specific configuration status of the particular voxel is identified. However, there are still ambiguous cases for the classical look-up table and the improved approaches require additional information not readily available from the FSV-rep model. And for the algorithmic approaches the computational time involved is significant compared to a look-up table approach even though the algorithmic approach is capable of handling more cases.

In order to improve upon the current lookup table and algorithmic approaches in generating a well-defined triangle mesh for machined part geometry and to eliminate the additional time for algorithmic methods, in this work we identify a new look-up table for the unique FS-voxel shapes possible.

The 256 configurations we counted as possible are the elaborate set of FS-voxel shapes possible after the two simplifications for the frame edges. However, after rigorous analysis of the 256 configurations, it is identified that these configurations are in fact, variants of 22 basic configurations. All the FS-voxel shapes can be generated by rotational and mirror

transformations of the 22 basic configurations.

Figure 8-5 below lists the 20 partial FS-voxel shapes that act as the basic set for generating all the other partial FS-voxels. The other two shapes are one with no frame segments present thus a voxel completely outside the shape and another with all the frame edges active and thus a voxel completely inside the shape.



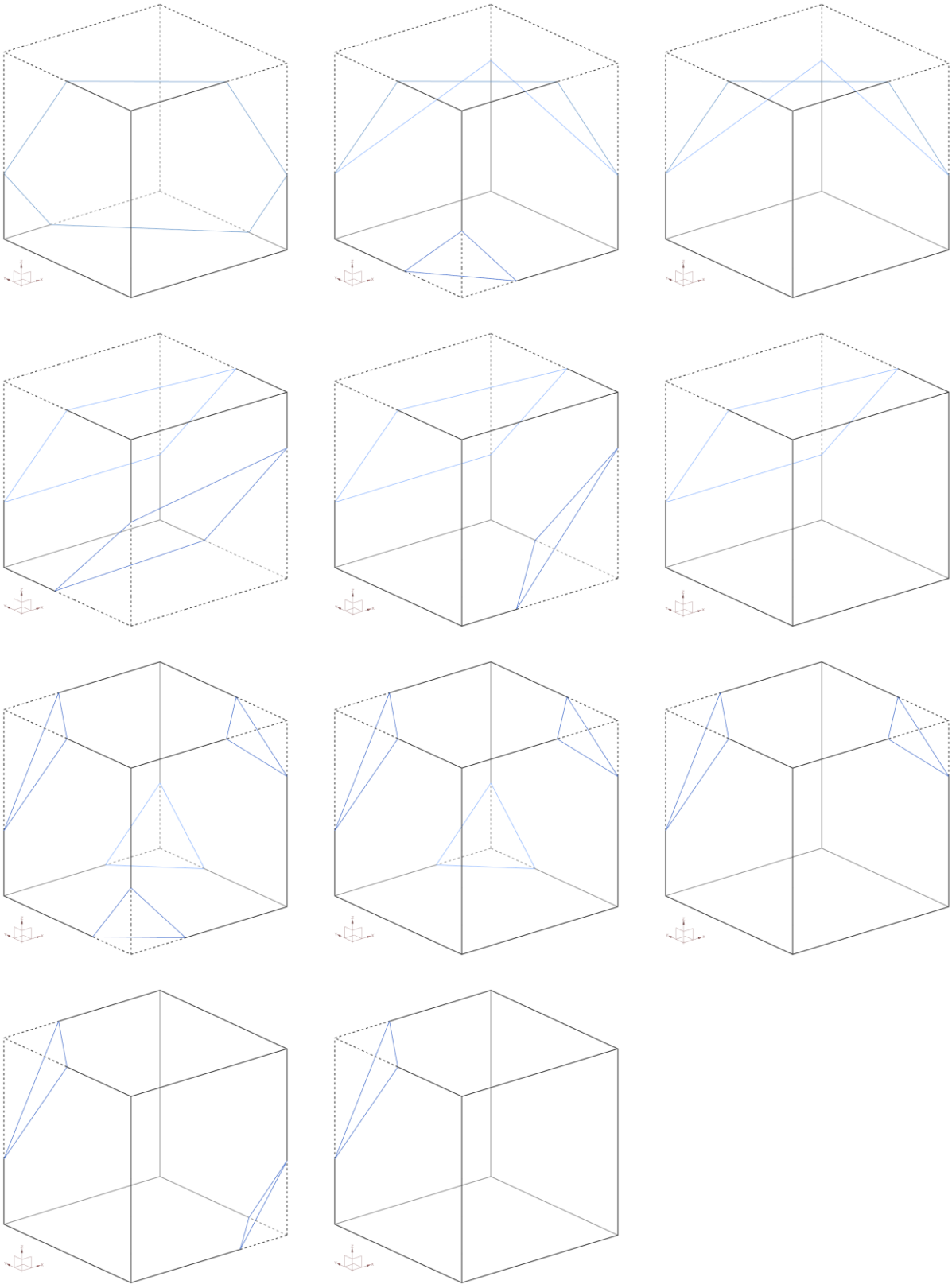


Figure 8-5 Set of 20 basic partial FS-voxel shapes with associated slicing loops.

As seen from the table above, there are some ambiguous cases for which we have made a choice based on the most probable condition that can occur on machined surfaces. Specifically, for all the FS-voxels with the following possible face boundary configuration, the one on the right in Figure 8-6 is chosen.

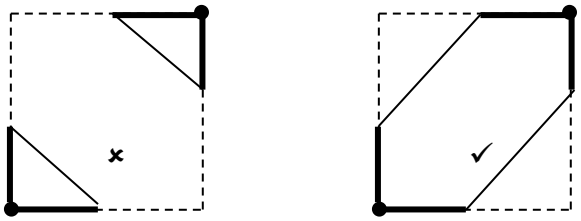


Figure 8-6 Selection of a suitable face boundary from two options for a particular corner points configuration.

The selection is made based on two criteria:

1) Most volume preserving shape is preferred to enable more feature preservation.
2) The selected shapes are more suitable for ensuring conformality between the shapes possible for neighboring FS-voxels as will be shown in next section.

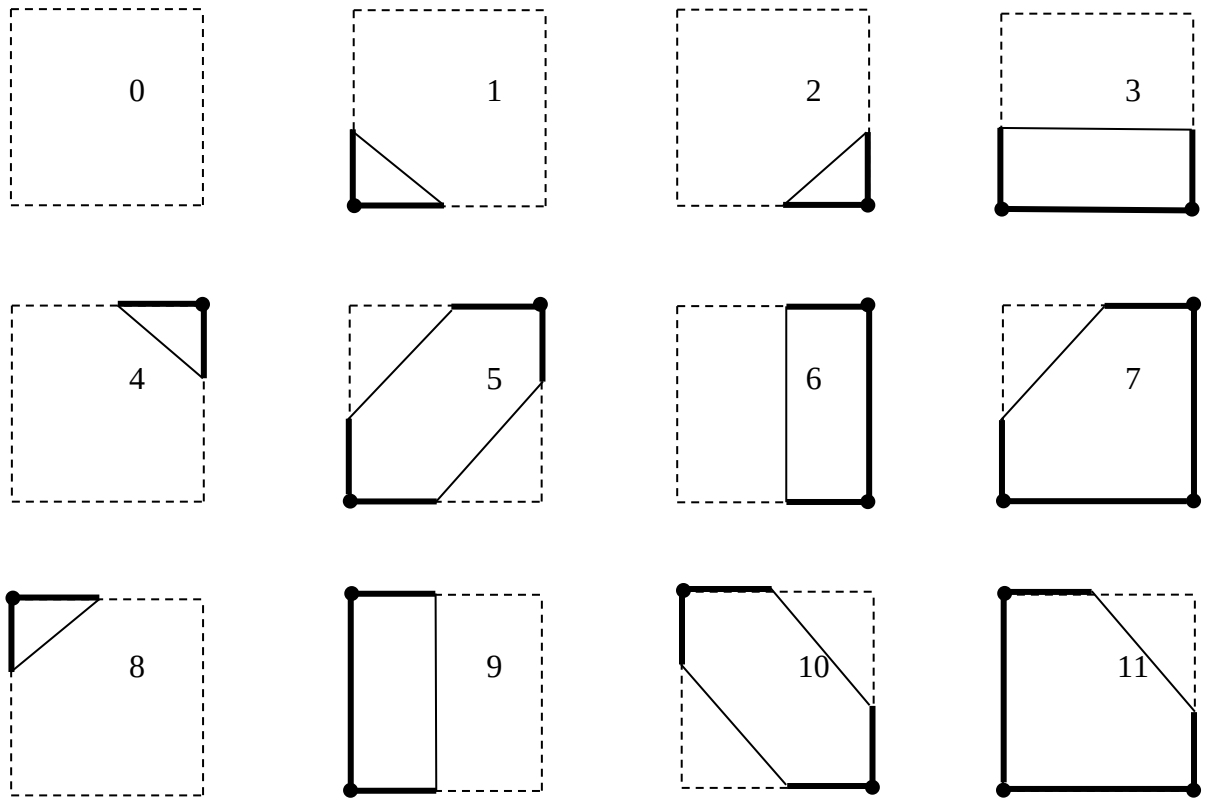
8.5 Proof of applicability

It can be proven that the developed FS-voxels look-up table will ensure closed 2-manifold triangle mesh generation for all FSV-rep models after the two simplifications are applied. This can be done in two steps: First, by verifying the condition within each FS-voxel is valid for generating a 2-manifold mesh. And then by verifying the interfaces between each neighboring FS-voxels are having matching topology.

The triangle patch within all the FS-voxels are certainly composed of triangulation that is well-defined. This is guaranteed as all the 256 FS-voxel shapes are rotational and mirror

transformations of the basic 22 shapes. Rotational transformations only change the global orientation of the FC-points without changing their relative locations. That is, there exist a coordinate system with respect to which the relative location of the FC-points of the latter FS-voxel is identical to that of the base one. Mirror transformations only change the FC-points orientation such that the relative location is preserved if a left-handed coordinate system is used instead of the a right handed system. Thus it is guaranteed that all the possible 256 FS-voxel shapes have valid triangulation once the basic 22 FS-voxel slice front triangulations are valid.

The next requirement is to have matching situation across the shared faces of two neighboring FS-voxels. This is ensured by the particular set of FS-voxel shapes chosen as the basic shapes. It can be appreciated by considering the possible cases for each FS-voxel face. For the FS-voxel faces, there are only 16 possible situations as shown in Figure 8-7.



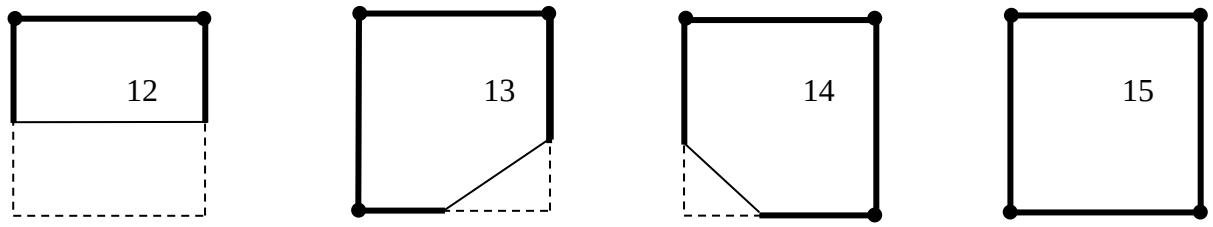


Figure 8-7 16 configurations for the frame segments on an FS-voxel face.

This ensures that for a given face of a FS-voxel the configuration is such that the configuration for the coincident face of a neighboring FS-voxels is conforming as shown below in Table 8-1. The numbers in the table points to the different FS-voxel face configurations in Figure 8-7. Each pair of numbers separated by ⇔ symbol are for the possible FS-voxel face configurations that can be present on the coincident faces of two neighboring FS-voxels.

Table 8-1 Possible frame segment configurations on pair of coincident faces of two neighboring FS-voxels.

0 ⇔ 0
1 ⇔ 2, 1 ⇔ 8
3 ⇔ 3, 3 ⇔12, 12 ⇔12
4⇔8, 4⇔2
5⇔10
6⇔6, 6⇔9, 9⇔9
7⇔11, 7⇔14
11⇔13, 13⇔14
15⇔15

The above conformity is available from the fact that a point acting as a voxel corner is

either active for all the voxels incident on it or inactive for all of them and also from the fact that the FS-voxel shapes is set so that the faces always have one of the above configurations in Figure 8-7.

Finally, the fact that the FSV-rep model is 26-separating ensures that all the FS-voxels has a face-neighbor across a face with at least one slicing loop segment.

All the three points above together guarantee that the triangle mesh models generated from FSV-rep using the 22-bases lookup table are always, closed and 2-manifold.

The applicability of triangle mesh model from a simplified set of FS-voxels is only complete if it can reasonably represent all the machining cases that may arise. The main simplifications we apply are to connect the gaps and ignore floating frame edge segments. Closing of gaps causes some concave features to disappear. Ignoring floating segments remove very sharp edge regions or pointed tips. All these lost features are however with characteristic dimensions less than the fine level voxel size. Thus, it is appropriate to assume such features form only very small fraction of the machined part geometry and does not add to the functional aspects of the part that needs to be preserved. The case studies in the later sections will further justify these views.

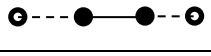
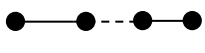
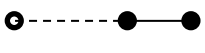
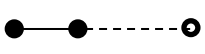
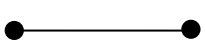
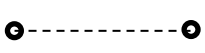
8.6 Implementation

In order to perform triangulation of the FSV-rep model, by the marching cube like per voxel processing, all the information necessary for a voxel need to be collected together. As described in the implementation of FSV-rep data structure (Chapter 3, Section 3.8), the FS-voxels are stored as partial elements with FC-points held only on the primary edges. For identifying the specific configuration of a particular FS-voxel, the FC-points on all the edges

have to be obtained. The neighbors shown in Figure 3-14 are the relevant neighbors to query. From the binary search tree holding the FC-points information per FS-voxel, the data from the neighbors can be obtained in $O(\log n)$ time cost per FS-voxel and thus $O(n \log n)$ time cost for the whole model.

With the FC-points available for all the frame edges, the occupancy status of each corner point can be identified. Based on the relative value of the FC-point parameters for a frame edge, the frame edge segments can be deduced as in Table 8-2 for parameter pair $[u_1, u_2]$ on each frame edge. For each frame edge configuration, solid dots and hollow dots on the extreme ends are active corner points and inactive corner points respectively. The solid dots in between are FC-points.

Table 8-2 Deduction of frame edge configuration from the FC-points parameter pair.

$\{u_1$ and $u_2\} \epsilon [0, 1)$ and $u_1 < u_2$	
$\{u_1$ and $u_2\} \epsilon [0, 1)$ and $u_1 > u_2$	
$u_1 \epsilon [0, 1)$ and $0 > u_2$	
$u_2 \epsilon [0, 1)$ and $0 > u_1$	
$\{u_1$ and $u_2\} \geq 1$	
$\{u_1$ and $u_2\} < 0$	

This deduction is possible as the FC-points are stored as parameters at particular location in the pair based on the surface normal of the generating tool as shown in Figure 3-11. Using the above mapping table to deduce the frame edge segments on all the voxel frame edges, the frame segments and corner points to which any active segments are attached are identified. Any corner with a segment attached is set active and others are set inactive. A completely alive edge

($type_{11}$) is assumed between to active corners and a completely absent edge is assumed between two inactive corners ($type_{00}$). This way the simplification of closing the gaps and ignoring the floating segments is implicitly done when the occupancy status is identified.

Once the occupancy status is obtained, the corresponding FS-voxel and the slice front boundary can be obtained front the lookup table defined in Section 8.4 with appropriate rotations and mirroring.

8.7 Case studies

In order to demonstrate the applicability of the identified 22-bases lookup table, a number of mechanical parts with simple and freeform shapes were converted to FSV-rep format from an input STL and the surface generation from FSV-rep was done using the approach described above. Additionally, the feature detection technique developed by Wang et al. [71] was performed on the generated model to reconstruct sharp edge features. The results of the mesh generated directly from FSV-rep model and the processed mesh with sharp features is provided in the Figures 8-8 to 8-11 below.

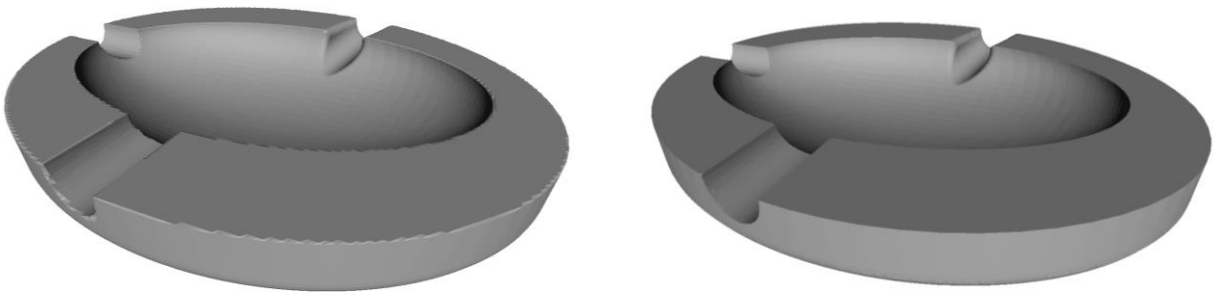


Figure 8-8 Ashtray model FSV-rep surface mesh (left) and edge restored mesh (right).

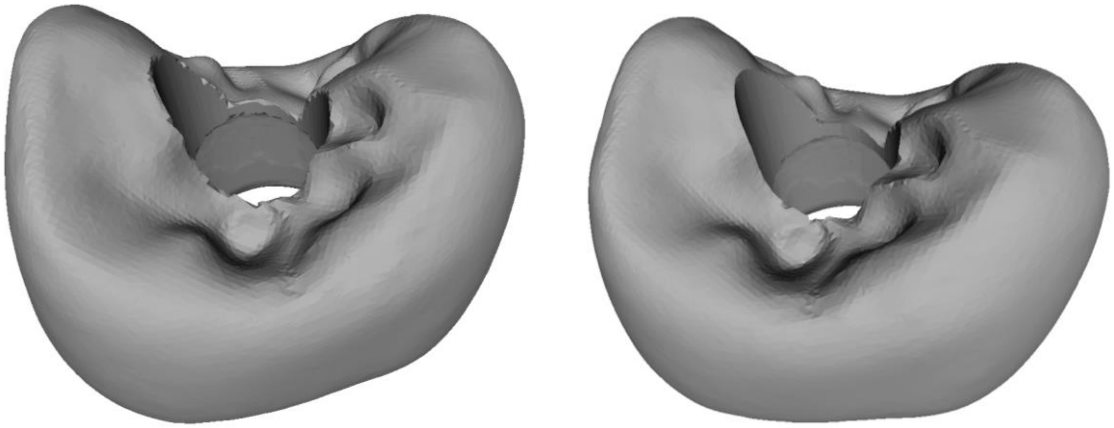


Figure 8-9 Dental part FSV-rep surface mesh (left) and edge restored mesh (right).

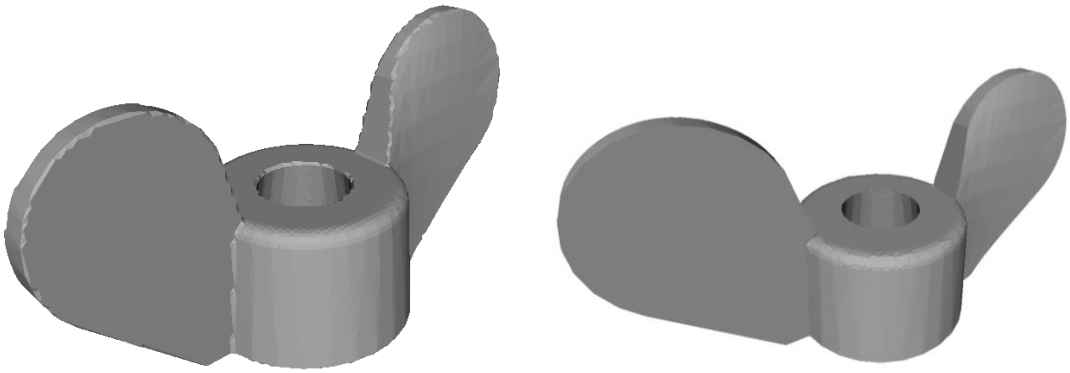


Figure 8-10 Nut model FSV-rep surface mesh (left) and edge restored mesh (right).

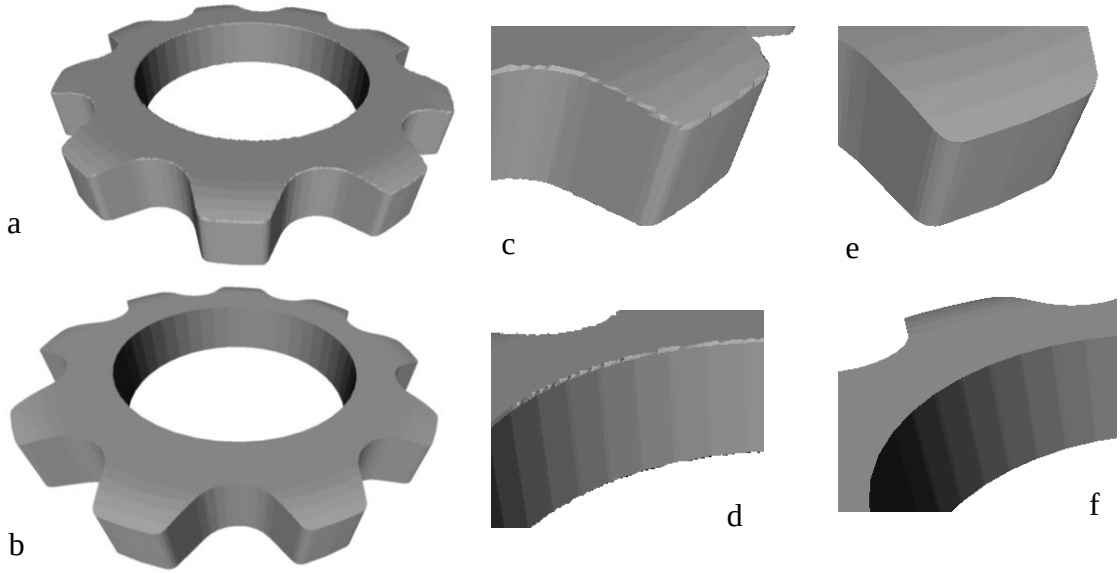


Figure 8-11 Gear model. (a) FSV-rep surface mesh. (b) edge restored mesh. (c)-(d) zoomed in view of two areas on (a). (e)-(f) zoomed in view of the corresponding areas on (b).

In order to evaluate the computational gain possible from use of the 22-bases look-up instead of the algorithmic approaches available, its performance is compared against that of method the method by Ren et al. [67]. The algorithmic approach by Ren et al. has shown to be generating valid manifold meshes in most practical cases. It can be seen from Table 8-3 below, the 22-bases lookup table approach is consistently faster by a factor of 2 or more.

Table 8-3 Comparison of Triangle mesh generation by algorithmic and 22-bases lookup table approaches.

model	grid spacing (mm)	number of triangles	number of vertices	Algorithmic triangulation, A (ms)	22-bases triangulation, M (ms)	M/A
ashtray	1	139,024	69,516	160.31	77.96	0.49
gear	0.5	138,552	69,276	159.45	75.84	0.48
dental	0.1	81,280	40,654	101.80	45.14	0.44
nut	0.5	39,708	19,856	46.17	20.90	0.45

8.8 Summary

The look up table based triangle mesh generation from FSV-rep is fast and robust in generating a closed 2-manifold triangle mesh. It is fast as the triangulation is readily available from a pre-defined lookup table for which only the status of the voxel corner points as active or inactive is needed for its usage. This information is directly obtainable from the FC-points. Further from the FS-voxels shapes selected such that the slice front boundaries on the shared faces of neighboring FS-voxels are always coincident, the generated triangle meshes are proven to be closed 2-manifold as well.

Chapter 9: Conclusions and Future research options

Following conclusions can be made about the FSV-rep model based machining simulation developed in this thesis work. The potential future research options are also identified later in the chapter.

9.1 Conclusions

A review of the existing modeling methods used for machining simulation has shown that the voxel-based space partitioning approach is the most computationally efficient. However, this approach is limited by the need of a very large grid resolution to attain a reasonable accuracy, which makes voxel modeling infeasible in terms of the resulting huge model size and memory requirement. The FSV-rep model introduced in this thesis work overcomes this issue by using a multi-level sparse voxel model representation and the FS-voxels. The FSV-rep model is deemed a very suitable modeling method for machining simulation due to its modeling accuracy and memory efficiency.

To use FSV-rep workpiece in machining simulation, an efficient three-step update process has been identified. After evaluation of various tool path categories, a new concept of swept volume regions (SVRs) was developed to update FSV-rep workpiece with straight cutting tool paths in case of 1-axis to 3-axis machining. And for other tool paths, a conservative sampled instances approach was adopted. Customized three-step update process in case of using SVRs or sampled instances could also be developed to exploit specific qualities of those representations.

The developed FSV-rep method has demonstrated to be faster than popular methods such as with tri-dexels in case of update with sampled instances for tool path. This is a very common scenario in multi-axis machining and thus becomes very relevant. The observed better

performance is due to two primary factors: (1) the bulk volume removal can be made faster with the coarse update of the FSV-rep model; and (2) the tilted tool orientation has less effect on updating the FSV-rep model again thanks to the coarse update. Further, the FSV-rep method is able to carry out the majority of the voxel model update steps involving only simple point classifications and binary marking/unmarking operations. The computationally demanding intersection calculations are just used for the frame-update step, thereby limiting the calculations to the final machined surface.

Use of SVRs for straight cutting planar and 3-axis tool paths shows further improvement for FSV-rep update. It is possible due to drastic reduction in number of surfaces to perform when sampled instances are replaced by a single swept volume. The quality of SVR types to immediately deduce the specific surface elements from the swept volume forming it further help in avoiding consideration of all the swept volume boundary elements while updating a particular voxel or its frame. This localization effect of SVRs also contribute to the enhanced FSV-rep update performance. SVRs concept is also shown to be applicable with better performance to a wide range of tool path lengths occurring in planar machining. Further, all intersection calculations are ensured to be with closed form solutions in general for FSV-rep update.

The triangle mesh model for the machined part surface is a straightforward output from the FSV-rep model using the FS-voxels. An improved lookup table based triangle mesh generation could be developed which can provide topologically valid surface representation from the FSV-rep model for the machined part geometry. The computational cost is also minimal from the use of direct access lookup tables.

Two issues occur on the simulated machined part geometry based on the FSV-rep model. First, the triangle mesh obtained from the FSV-rep model has missing sharp machined features.

A robust triangle mesh post-processing algorithm could be applied to restore the sharp features on wide variety of mechanical parts created by milling processes. Second, sampling scallops are present on the triangle mesh surface due to the use of sampled tool instances to approximate the exact tool swept volume. To attain visually smooth surfaces, the tool path sampling interval has to remain small but this can hurt the overall computational efficiency. With the use of SVRs this issue is partially addresses at least for many planar milling operations. However, for other tool path types, the scallop size observed is not critical with the affordable grid resolutions used thus alleviating this issue to some extent.

9.2 Future research options

In the future, machining simulation with FSV-rep workpiece can be used for assisting mechanistic machining simulation with the input of CWE maps. Current thesis work focused on generating the final machined part geometry. To obtain CWE maps, the developed method itself can be used in theory with the model finalized at every point along the tool paths a CWE map is needed. However, this may affect the efficiency of update available from the bulk volume removal at the coarse level as the CWE maps are usually required at very small forward steps. Still the multi-level nature of FSV-rep and simple operations involved for voxels can be potentially used for CWE region identification in an efficient way. Further research is needed to get a feasible methodology to exploit this aspect of FSV-rep for fast and accurate CWE map computations. Once an incremental FSV-rep update and surface generation technique which is computationally affordable is developed, apart for CWE maps, FSV-rep workpiece can also be used in a complete machining simulation environment for the visual geometric process animation. Also, in the present work, a simple two-level grid structure has been employed for the

FSV-rep model. To take full advantage of the multi-level voxel representation, an octree type of sub-division from the coarsest to the finest grid level will be needed in order to make the modeling format scalable.

References

[1] Altintas Y. Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design. 2nd ed. Cambridge University Press; 2012.

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

[3] Zhang Y, Xu X, Liu Y. Numerical control machining simulation: a comprehensive survey. International Journal of Computer Integrated Manufacturing. 2011;24(7):593–609.

[4] Oliver JH, Goodman ED. Direct dimensional NC verification. Computer-Aided Design. 1990;22(1):3–9.

[5] Du J, Yan XG, Tian XT. The avoidance of cutter gouging in five-axis machining with a fillet-end milling cutter. International Journal of Advanced Manufacturing Technology. 2012;62(1–4):89–97.

[6] OuYang D, Feng H-Y, van Nest BA, Buchal RO. Effective gouge-free tool selection for free-form surface machining. Computer-Aided Design and Applications. 2009;6(6):839–49.

[7] Ren Y, Lai-Yuen S, Lee Y-S. Virtual prototyping and manufacturing planning by using tri-dexel models and haptic force feedback. Virtual and Physical Prototyping. 2006;1(1):3–18.

[8] Erkorkmaz K, Katz A, Hosseinkhani Y, Plakhotnik D, Stautner M, Ismail F. Chip geometry and cutting forces in gear shaping. CIRP Annals - Manufacturing Technology. 2016;65(1):133–6.

[9] Lee SW, Nestler A. Mechanistic Model Based on the Actual Removal Volume during Simultaneous Five-Axis Milling. Advanced Materials Research. 2011;223:713–22.

[10] Luo S, Dong Z, Jun MBG. Chip volume and cutting force calculations in 5-axis CNC machining of free-form surfaces using flat-end mills. International Journal of Advanced Manufacturing Technology. 2017;90:1145–54.

[11] Jerard RB, Hussaini SZ, Drysdale RL, Schaudt B. Approximate methods for simulation and verification of numerically controlled machining programs. The Visual Computer. 1989;5(6):329–48.

[12] El Mounayri H, Spence AD, Elbestawi MA. Milling Process Simulation—A Generic Solid Modeller Based Paradigm. Journal of Manufacturing Science and Engineering. 1998;120(2):213–21.

[13] El-Mounayri H, Elbestawi MA, Spence AD, Bedi S. General geometric modelling

approach for machining process simulation. International Journal of Advanced Manufacturing Technology. 1997;13:237–47.

[14] Spence AD, Abrari F, Elbestawi MA. Integrated solid modeller based solutions for machining. Computer-Aided Design. 2000;32(8–9):553–68.

[15] Aras E, Yip-Hoi D. Geometric modeling of cutter/workpiece engagements in three-axis milling using polyhedral representations. Journal of Computing and Information Science in Engineering. 2008;8(3):31007.

[16] Gong X, Feng H-Y. Cutter-workpiece engagement determination for general milling using triangle mesh modeling. Journal of Computational Design and Engineering. 2016;3(2):151–60.

[17] Roy U, Xu Y. Computation of a geometric model of a machined part from its NC machining programs. Computer Aided Design. 1999;31(6):401–11.

[18] Li Z-L, Wang X-Z, Zhu L-M. Arc-surface intersection method to calculate cutter-workpiece engagements for generic cutter in five-axis milling. Computer Aided Design. 2016;73:1–10.

[19] 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. 2011;52(9–12):913–27.

[20] Fussell BK, Jerard RB, Hemmett JG. Modeling of cutting geometry and forces for 5-axis sculptured surface machining. Computer Aided Design. 2003;35(4):333–46.

[21] Stifter S. Simulation of NC machining based on the dexel model: A critical analysis. International Journal of Advanced Manufacturing Technology. 1995;10:149–57.

[22] Benouamer MO, Michelucci D. Bridging the gap between CSG and Brep via a triple ray representation. In: Proceedings of the fourth ACM symposium on Solid modeling and applications. 1997. p. 68–79.

[23] Lee SW, Nestler A. Virtual workpiece: workpiece representation for material removal process. International Journal of Advanced Manufacturing Technology. 2012;58(5–8):443–63.

[24] Jang D, Kim K, Jung J. Voxel-based virtual multi-axis machining. International Journal of Advanced Manufacturing Technology. 2000;16:709–13.

[25] Wou SJ, Shin YC, El-Mounayri H. Ball end milling mechanistic model based on a voxel-based geometric representation and a ray casting technique. Journal of Manufacturing Processes. 2013;15(3):338–47.

[26] Karunakaran KP, 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. 2010;46(5–8):465–90.

[27] Frisken SF, Perry RN, Rockwood AP, Jones TR. Adaptively sampled distance fields: a general representation of shape for computer graphics. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. 2000. p. 249–54.

[28] Sullivan A, Erdim H, Perry RN, Frisken SF. High accuracy NC milling simulation using composite adaptively sampled distance fields. Computer Aided Design. Elsevier Ltd; 2012;44(6):522–36.

[29] Ding S, Mannan MA, Poo AN. Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces. Computer Aided Design. 2004;36(13):1281–94.

[30] Ilushin O, Elber G, Halperin D, Wein R. Precise global collision detection in multi-axis NC-machining. Computer-Aided Design. 2005;37:909–20.

[31] Lee Y-S, Chang T-C. 2-Phase approach to global tool interference avoidance in 5-axis machining. Computer-Aided Design. 1995;27(10):715–29.

[32] Erkorkmaz K, Altintas Y, Yeung CH. Virtual computer numerical control system. CIRP Annals - Manufacturing Technology. 2006;55(1):399–403.

[33] Merdol SD, Altintas Y. Virtual cutting and optimization of three-axis milling processes. International Journal of Machine Tools and Manufacture. 2008;48(10):1063–71.

[34] Yousefian O, Tarbutton JA. Prediction of cutting force in 3-Axis CNC milling machines based on voxelization framework for digital manufacturing. Procedia Manufacturing. Elsevier B.V.; 2015;1:512–21.

[35] Mujber TS, Szecsi T, Hashmi MSJ. Virtual reality applications in manufacturing process simulation. Journal of Materials Processing Technology. 2004;155–156(1–3):1834–8.

[36] Ong SK, Yuan ML, Nee AYC. Augmented reality applications in manufacturing: a survey. International Journal of Production Research. 2008;46(10):2707–42.

[37] Wang SW, Kaufman AE. Volume sampled voxelization of geometric primitives. In: Proceedings of IEEE Visualization. 1993. p. 78–85.

[38] Tang TD. Algorithms for collision detection and avoidance for five-axis NC machining: A state of the art review. Computer-Aided Design. 2014;51:1–17.

[39] Yau HT, Tsou LS, Tong YC. Adaptive NC simulation for multi-axis solid machining. Computer-Aided Design and Applications. 2005;2(1–4):95–104.

[40] Liu C, Esterling DM, Fontdecaba J, Mosel E. Dimensional verification of NC machining profiles using extended quadtrees. Computer Aided Design. 1996;28(11):845–52.

[41] Requicha AG. Representations for Rigid Solids: Theory, Methods, and Systems. ACM Computing Surveys. 1980;12(4):437–64.

[42] Kaufman A, Cohen D, Yagel R. Volume Graphics. Computer. 1993;26(7):51–64.

[43] Karabassi E-A, Papaioannou G, Theoharis T. A fast depth-buffer-based voxelization algorithm. Journal of Graphics Tools. Natick, MA, USA: A. K. Peters, Ltd.; 1999 Dec;4(4):5–10.

[44] Kaufman A. Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes. ACM SIGGRAPH Computer Graphics. 1987;21(4):171–9.

[45] Huang J, Yagel R, Filippov V, Kurzion Y. An accurate method for voxelizing polygon meshes. In: Proceedings of the 1998 IEEE Symposium on Volume Visualization. 1998. p. 119–26.

[46] Kaufman A. Efficient algorithms for scan-converting 3D polygons. Computers and Graphics. 1988;12(2):213–9.

[47] Cohen-Or D, Kaufman A. Fundamentals of Surface Voxelization. Vol. 57, Graphical Models and Image Processing. 1995. p. 453–61.

[48] Laine S, Karras T. Efficient sparse voxel octrees. IEEE Transactions on Visualization and Computer Graphics. 2011;17:1048–59.

[49] Kämpe V, Sintorn E, Assarsson U. High Resolution Sparse Voxel DAGs. ACM Trans Graph. 2013;32(4):101:1--101:13.

[50] Baert J, Lagae A, Dutré P. Out-of-core construction of sparse voxel octrees. In: Proceedings of the 5th High-Performance Graphics Conference. 2013. p. 27–32.

[51] Langeron JM, Duc E, Lartigue C, Bourdet P. A new format for 5-axis tool path computation, using Bspline curves. Computer-Aided Design. 2004;36(12):1219–29.

[52] Fleisig R V, Spence AD. A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining. Computer-Aided Design. 2001;33(1):1–15.

[53] Sencer B. Five-axis trajectory generation methods. Master Thesis. The Univresity of British Columbia; 2005.

[54] Blackmore D, Leu MC, Wang LP. The sweep-envelope differential equation algorithm and its application to NC machining verification. Computer-Aided Design. 1997;29(9):629–37.

[55] Lee SW, Nestler A. Complete swept volume generation, Part I: swept volume of a piecewise C1-continuous cutter at five-axis milling via Gauss map. Computer-Aided Design. 2011;43(4):427–41.

[56] Lee SW, Nestler A. Complete swept volume generation — Part II: NC simulation of self-

penetration via comprehensive analysis of envelope profiles. Computer-Aided Design. 2011;43(4):442–56.

[57] Altintas Y, Engin S. Generalized modeling of mechanics and dynamics of milling cutters. CIRP Annals - Manufacturing Technology. 2001;50(1):25–30.

[58] Chung YC, Park JW, Shin H, Choi BK. Modeling the surface swept by a generalized cutter for NC verification. Computer-Aided Design. 1998;30(8):587–94.

[59] Pfister H, Hardenbergh J, Knittel J, Lauer H, Seiler L. The VolumePro Real-time Ray-casting System. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. 1999. p. 251–60.

[60] Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. SIGGRAPH Computer Graphics. 1987;21(4):163–9.

[61] Cohen-Or D, Kadosh A, Levin D, Yagel R. Smooth Boundary Surfaces from Binary 3D Datasets. In: Volume Graphics. Springer London; 2000. p. 71–8.

[62] Chernyaev E V. Marching cubes 33: construction of topologically correct isosurfaces. CERN Report, CN/95-17. 1995.

[63] Lewiner T, Lopes H, Vieira AW, Tavares G. Efficient implementation of marching cubes' cases with topological guarantees. Journal of Graphics Tools. 2003;8:1-15.

[64] Leu MC, Peng X, Zhang W. Surface reconstruction for interactive modeling of freeform solids by virtual sculpting. CIRP Annals - Manufacturing Technology. 2005;54(1):131–4.

[65] Zhang W, Peng X, Leu MC, Zhang W. A novel contour generation algorithm for surface reconstruction from dexel data. Journal of Computing and Information Science in Engineering. 2007;7(3):203–10.

[66] Zhu W, Lee Y-S. A visibility sphere marching algorithm of constructing polyhedral models for haptic sculpting and product prototyping. Robotics and Computer-Integrated Manufacturing. 2005;21(1):19–36.

[67] 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. 2008;5(6):932–41.

[68] Osher S, Fedkiw R, Piechor K. Signed distance functions. In: Level Set Methods and Dynamic Implicit Surfaces. 2003. p. 17–22.

[69] Mäntylä M. Topological analysis of polygon meshes. Computer-Aided Design. 1983;15(4):228–34.

[70] Huang H, Ascher U. Surface mesh smoothing, regularization, and feature detection.

SIAM Journal on Scientific Computing. 2013;31(1):74–93.

[71] Wang S, Chen JSS, Joy J, Feng H-Y. Machined sharp edge restoration for triangle mesh workpiece models derived from grid-based machining simulation. Computer-Aided Design and Applications. Accepted..

[72] Cignoni P, Callieri M, Corsini M, Dellepiane M, Ganovelli F, Ranzuglia G. Meshlab: an open-source 3d mesh processing system. In: Sixth Eurographics Italian Chapter Conference. 2008. p. 129–36.

[73] Muller H, Surmann T, Stautner M, Albersmann F, Weinert K. Online sculpting and visualization of multi-dexel volumes. In: Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications. 2003. p. 258–61.

[74] Ferry W, Yip-Hoi D. Cutter-workpiece engagement calculations by parallel slicing for five-axis flank milling of jet engine impellers. ASME Journal of Manufacturing Science and Engineering. 2008;130:51011.

[75] Weinert K, Du S, Damm P, Stautner M. Swept volume generation for the simulation of machining processes. International Journal of Machine Tools and Manufacture. 2004;44(6):617–28.

[76] Yang Y, Zhang W, Wan M, Ma Y. A solid trimming method to extract cutter--workpiece engagement maps for multi-axis milling. International Journal of Advanced Manufacturing Technology. 2013;68(9):2801–13.

[77] Shmakov SL. A universal method of solving quartic equations. International Journal of Pure and Applied Mathematics. 2011;71(2):251–9.

[78] Fortune S. A sweepline algorithm for Voronoi diagrams. Algorithmica. 1987;2:153–74.

Appendix

Appendix A

Expression for the cross section radius of the selected cutters along the axis as a function of axial height, *z*.

Flat end mill:

$$R(z) = \frac{D}{2} \; \forall \; z$$

Ball end mill:

$$R(z) = \begin{cases} \frac{D}{2} \; if \;\; z \geq \frac{D}{2} \\ \sqrt{\frac{D^2}{4} - \left(\frac{D}{2} - z\right)^2} \;\; if \; 0 < z < \frac{D}{2} \end{cases}$$

Taper ball end mill:

$$R(z) = \begin{cases} Rcos(\alpha) + \left(z - \; R(1 - \sin(\alpha))\right)\tan(\alpha) \;\; if \; z \geq R(1 - \sin(\alpha)) \\ \sqrt{R^2 - (R - z)^2} \;\; if \; 0 < z < \; R(1 - \sin(\alpha)) \end{cases}$$

Bull nose end mill:

$$(z) = \begin{cases} \frac{D}{2} \; if \;\; z \geq \frac{D}{4} \\ \frac{D}{4} + \sqrt{\frac{D^2}{16} - \left(\frac{D}{4} - z\right)^2} \;\; if \; 0 < z < \frac{D}{4} \end{cases}$$