SSI-DT46205206

Doctor Dissertation

USING FOUR-DIMENSIONAL GEOMETRIC MODELS FOR REPRESENTING DYNAMIC MACHINING PROCESS

Tong Zhang

June, 2024

Course of Systems Science and Informatics
Graduate School of Information Science and Technology
Hokkaido University

USING FOUR-DIMENSIONAL GEOMETRIC MODELS FOR REPRESENTING DYNAMIC MACHINING PROCESS

Tong Zhang

Abstract

In the trend of Industry 4.0, the relationship between cyber field and physical field is gradually being established and deepened. In the field of processing and manufacturing, cyber field can be used to simulate the processing. This simulation can guide the real processing from multiple aspects and be applied to various processes.

In this study, the authors tried to propose a geometric modeling system for cutter-workpiece engagement (CWE) of the milling process based on four-dimensional (4D) spatio-temporal space. Different from the traditional geometric process simulation based on three-dimensional (3D) models, this study pioneered the introduction of 4D geometric models for CWE analysis.

4D spatio-temporal space has three spatial coordinate axes (x, y, z) and one time coordinate axis (t). Therefore, the 4D geometric model in the 4D spatio-temporal space also integrates all the data in space and time. This means that a 4D geometric model can represent the shape of a 3D object, as well as the movement and morphological changes of the 3D object over a period of time. In traditional 3D methods, such a dynamic process often requires a discrete data set or is derived using physical equations. The method using a 4D geometric model can integrate this dynamic process into a 4D static model, which undoubtedly improves the integrity of the data. At the same time, it also provides a more convenient method for a series of computer-based and AI-based post-processing processes, such as display and deep learning training cases. In this study, the main focus is on representing the dynamic process of tool movement and material removal in CWE. The authors believe that this process is very suitable for analysis and representation using 4D geometric models.

This dissertation describes the details of this study in detail. Chapter 1

summarizes the general outline of the paper. Chapters 2 and 3 introduce the research background and foundation in the field of machining simulation and 4D geometric modeling. Chapter 4 introduces the more special tools used to build the system. Chapters 5, 6, and 7 are the main chapters of the paper, which elaborate on three CWE analysis methods based on 4D geometric models. These three methods are T-map method, 4D imprint method and 4D set operation method, and they each have different advantages and disadvantages. The entire paper will be summarized in Chapter 8. The author hopes that this dissertation can propose a new solution to CWE analysis to a certain extent, and put forward some thoughts on the application of 4D models for industry filed and graphics field.

Keywords: 4-dimensional model, spatio-temporal space, cutter-workpiece engagement, milling simulation, computation geometry

Table of Contents

List of figures

List of table

Chapter 1. Introduction

1.1 Research background

Since more than ten years ago, the laboratory of Digital Systems and Environments (DSE Lab) at Hokkaido University has been committed to the development of 4-dimensional (4D) modeling system and its application. The 4D space refer to a Four-dimensional Euclidean space. Four axis are X-, Y-, Z- and T-axis, where T refers to the time corresponding to space [1].

In 2008, Professor Onosato of our laboratory has conducted a series of related research on the use of 4D geometric models in the cyber field [2]. The paper pointed out 4D geometric model can be regards as an integrated static representation of dynamic shape, orientation, position of a target 3D object. It can replace traditional 3D approaches dependent on applications with independent description in a certain degree[3,4].

From 2008 to 2010, based on the Triangulation Table used in high-dimensional space proposed by Bhaniramka et al[5]., Onosato et al. started from the three-dimensional (3D) marching cube method and extended it to the four-dimensional space. They proposed a method based series of 3D voxel models on continuous time periods to generate a 4D mesh models, which has been called as 4D marching cube method[6]. Meanwhile, they realized the importance of GPU parallel processing applications in 4D algorithms. It can most conveniently reduce the problem of long calculation time caused by the explosion of four-dimensional data in the absence of better algorithms[7,8]. In addition to these processing methods for 4D model, they also propose an idea to obverse the 4D models by cutting it along a 3D hyperplane[9].

From 2011 to 2013, Otomo et al, proposed a simple and high effective approach to construct 4D mesh models for a moving scenario of dynamic 3D rigid body shapes[11,12]. In 2019, Yabuki et al. proposed another method based on 4D ball pivoting to generate 4d meshes[13]. Figure 1.1 list some 4D modeling technology developed in DSE Lab.

Since 2011, Kasai et al. started to analyze and process the intersection

condition of 4D models[14].These series of studies were not basically completed until 2016 by Matsunaga et al. They proposed a series of methods to solve the intersection determination of 4D mesh models by categories[15].

Since 2012, DSE Lab has begun to explore and discuss the practical application of 4D models. Kameyama et al. first proposed applying 4D mesh model to express the five-axis machining process. And they combined the method of Otomo et al. to determine the

expression method of tool path[16]. Kiita et al. proposed represent the transmission of ocean waves by 4D mesh model[17].

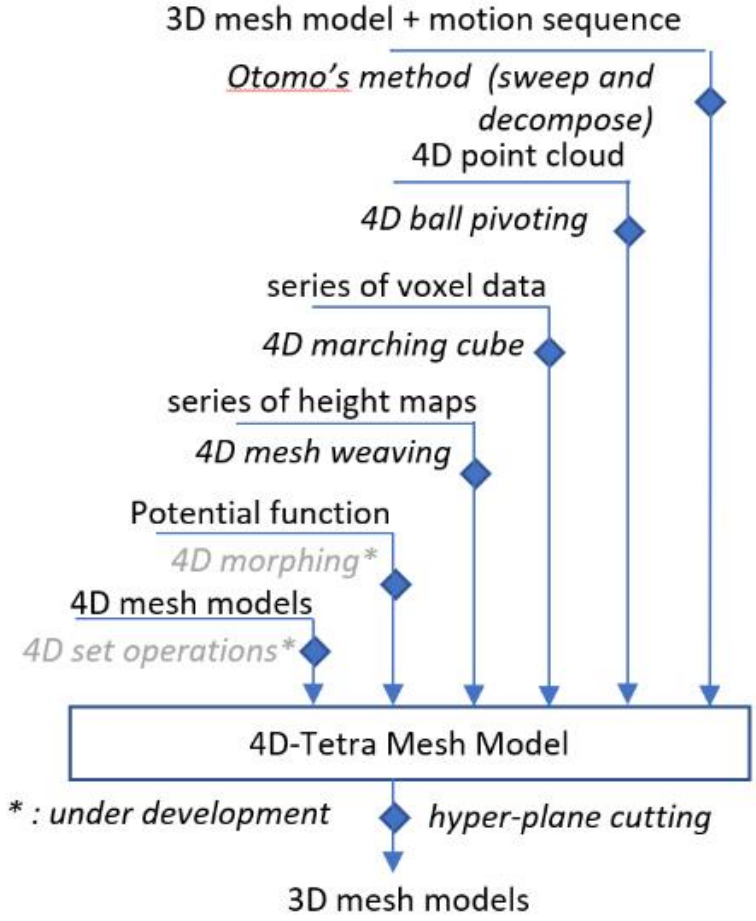


Fig. 1.1 4D modeling technology developed by DSE Lab

Although more than ten years have passed, research on the application of 4D geometric models is still in its infancy. With the widespread application of AI, the analysis and understanding of 4D mesh models will become easier. The emergence of advanced equipment such as quantum computers can also effectively solve

problems such as computing speed[18]. The author believes that the future prospects of 4D models are very optimistic.

The author's academic background is the combination of manufacturing and computer graphics. In The author's master's period, author's study constructed a system to analyze milling process of a given workpiece in 3D space and accelerate it by parallel processing by GPGPU[19]. Therefore, the author selects to continue to develop algorithm of 4D geometric models in the author's doctoral study. Meanwhile, the author tries to apply it to machining simulation and analysis based on graphics.

In the machining production process, the interaction between the cutting tool and the workpiece, known as cutter-workpiece engagement (CWE), is essential for assessing the cutting process's quality. To achieve a more precise evaluation, three-dimensional (3D) geometry milling simulation systems have been extensively developed and utilized. These systems are designed to estimate the quality and efficiency of the cutting process beforehand, with a particular emphasis on the CWE. [20, 21].

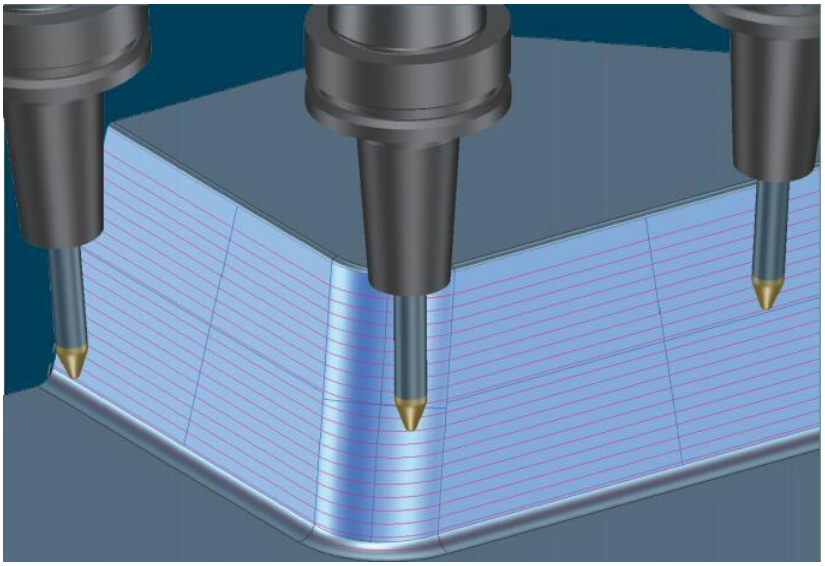


Fig. 1.2 Traditional 3D CWE analysis algorithm, based on a series of 3D snapshots.

Xun Gong and colleagues categorized the existing methods for determining CWE into three fundamental strategies. [22, 23]. The first strategy entails analytically calculating the CWE using parametric spatial curves. This method can be quite time-consuming due to the extensive computational effort involved.

Generating the cutter swept volume (CSV) and updating the workpiece shape during the cutting process is the second strategies.

However, this method necessitates the updating of an increasing number of CSVs as the tool progresses along its path. The final strategy involves calculating the CWE by tool positions between intermediate steps along the tool path. Aras et al. reduced the volume of calculation by utilizing the removal volume of the workpiece for extracting the CWE [24, 25]. All of the above strategies are based on 3D geometric models and share similar limitations.

The fundamental concept of representing a dynamic process in 3D involves a series of discrete snapshots of 3D shape that is changing over time, such as Fig.1.2. Therefore, the motion of the object is consistently represented by the position and shape data of each snapshot. For intricate or precise motion processes, users need to use shorter time intervals, which leads to a greater number of iterations and extended calculation times. [26].

The authors think that 4D description approach is effectively applied to CWE analysis. After interpolation by 4D modeler, all required data can be obtained by one time computation, which can replace computing based on contexts in the traditional 3D method. This feature makes sure the continuous of 4D result and the user can observe it at any axis and any value to analyze CWE quality. It also makes the 4D results are more suitable for computer and AI process. Hence the 4D result is an integral whole and contain all information.

Comparing to 3D representation of machining 4D representation is an integration. Common 3D representation of a CWE process can be consisted by a series snapshots of moving 3D shapes or 3D model with it moving trajectory. Regardless of the method, expressing the geometric changes of CWE requires some discrete data, and it is also discrete in time. At the same time, the state of the intermediate time needs to be extrapolated or interpolated from the known time state, and this part of the calculation is unavoidable. Therefore, if you want a computer or AI to understand this kind of data, a sufficient number of calculations are necessary. The 4D model technology can integrate the entire machining process into one geometric model. In other words, all data about shape and motion can be represented integrally by a 4D geometric model. These 4D geometric models are continuous in each direction, and they can reflect all movement and deformation state of the object

at every moment [7]. As a model that can contain all required data without additional calculations, it is more suitable and can be applied to computers and AI faster than 3D representations that require additional calculations. Through The four-dimensionalization of 3D dynamic process, the complex motion and deform of cutting process can be geometricized. This also allows us to approach the machining process with a more homogeneous and simple geometric method which is suitable to AI and computer.

In addition, the 4D model gives us a broader perspective and a richer geometric based CWE analysis method. The author can use computers to analyze the 4D model as a whole, or you can use the 3D representation 4D models to provide different focuses of observation and analysis, such as simple XYZ, YZT, XZT hyperplane slices or projection which perpendicular to a certain coordinate axis [9]. For example, by performing XZT hyperplane slicing, the author can observe the time-varying state of the XZ 2D section at a fixed Y. This is very useful for analyzing the touch condition of the tool and the workpiece at a fixed position. Therefore, our analysis directions for four-dimensional models can also be diverse and are not limited to the positive time direction of 3D representation [27]. Such analysis may be difficult for humans, but for computers and AI it is just a geometric model. Meanwhile the 4D model is good at represent the deform process and the deform with displace which not suitable for represent by 3D models. If the meaning of each axis is changed for example the fourth axis is used to indicate temperature[28]. A representation of a wider range of situations can be realized with a 4D model.

The author hope that this research can serve as a catalyst for 4D models in digital manufacturing fields such as CWE and even CAM. It is also expected that 4D geometric models can be more widely used in various fields.

Since the difficulty and complexity of accumulating tool occupied region as 4D shape, it is necessary to develop more efficient algorithm using GPU processing to accumulate 4D regions While developing these challenging functions of the target framework, the authors have studied some alternative approaches which enable CWE evaluation approximately by introducing some limited conditions.

1.2 Research objective and position

The research objective is constructing a series of CWE analysis system based

on 4D geometric model in spatio-temporal space. This system consists of applications of different 4D models, corresponding processing algorithm and a flow of analyzing process of CWE.

The author expects this research can become the bridge of 4D geometric model in spatio-temporal space and its machining application. In addition, the research also enable to serve as a starting point for future research which based on 4D geometric model, 4D space and corresponding algorithms.

1.3 Research outline (Implemented outline)

Figure 1.3 is the basic outline of this system and research. The complete system is envisioned to consist of five distinct parts. We use A to E to differentiate.

As the part A, a kind of 4D models called tool-occupied region (TOR) is constructed. It generates by a part of the tool model which extract from 3D cutting tool shape library with the rotation and translation in a very short period of time. It is essentially a differentiation of the tool's 4D movement trajectory. Additionally, 4D tool models can be stored in a moving tool shape library. Part B serves as an

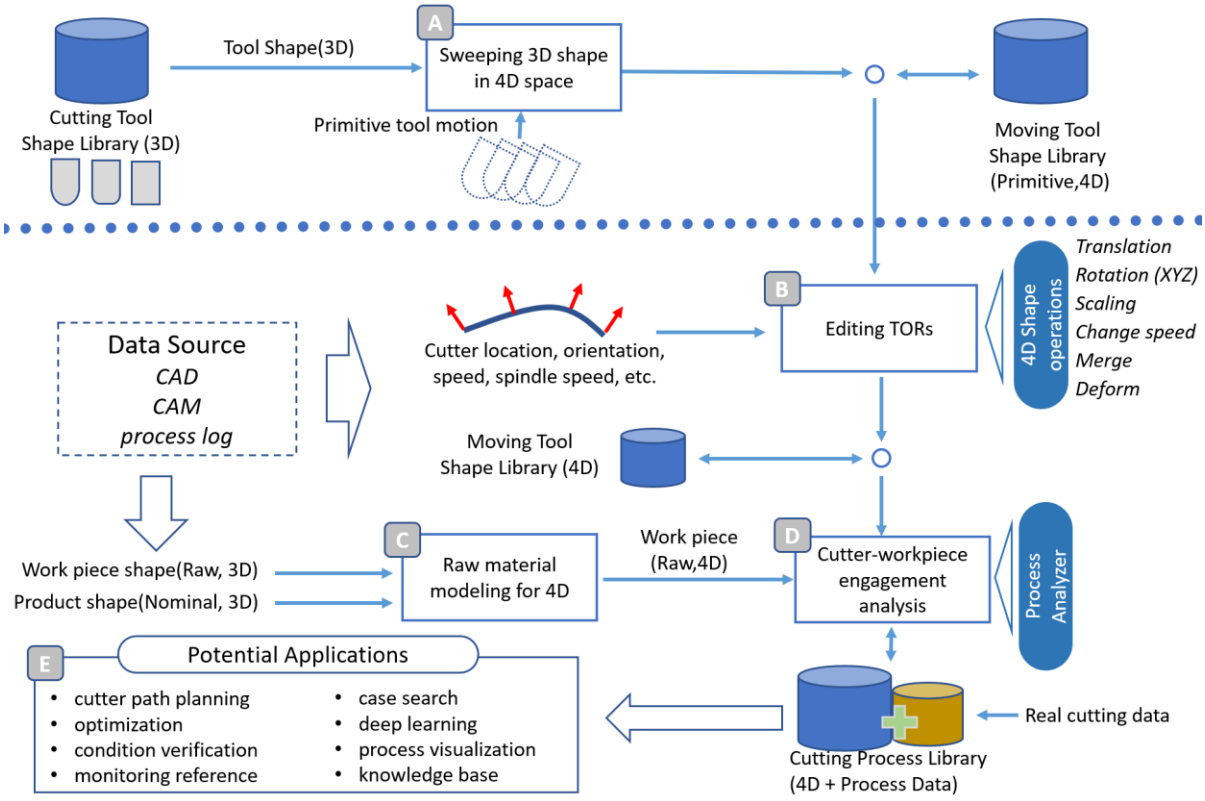


Fig. 1.3 Research frame

editor for users who wish to design the TOR by configuring the tool's location, orientation, speed, and other parameters.

Apart from inputting tool data, Part C involves inputting a 3D workpiece model for computation alongside the TOR. Additionally, a nominal product model is inputted for reference.

Part D constitutes the pivotal segment of the entire research. The process involves determining CWE in Spatio-Temporal context, aiming to identify the intersection between the edited TOR and a 3.5D workpiece model generated by sweeping a static 3D workpiece along the T-axis. The resulting data is extracted and fed into a 4D cutting process library, applicable across various scenarios.

Part E list a part of possible potential applications, such as case search, condition verification, cutter path planning, optimization, monitoring reference, case search, deep learning, knowledge base, process visualization.

As mentioned above, part D is the core technical part of this research. So far, the author has developed a total of three CWE technologies. They are T-map, 4D imprint and set operation based method. Limited by the huge amount of data in the 4D model, these three methods each have their own advantages and disadvantages, and they need to be selected according to the situation. The process is shown in fig.1.4.

The T-map method emphasizes continuity in the time axis direction and abandons continuity in 3D space. In contrast, the 4D imprint method gives up the continuity of the time axis direction and instead focuses on the continuity of the 3D space. These two methods, due to their partial discretization, have faster calculation speed and lower accuracy.

Set operation based method is a general method. Its basic principle is very simple, solving CWE situations through set operations between geometric models. This method ensures continuity in all directions and also withstands the data explosion caused by the introduction of 4D models, so it requires an extremely long computing time.

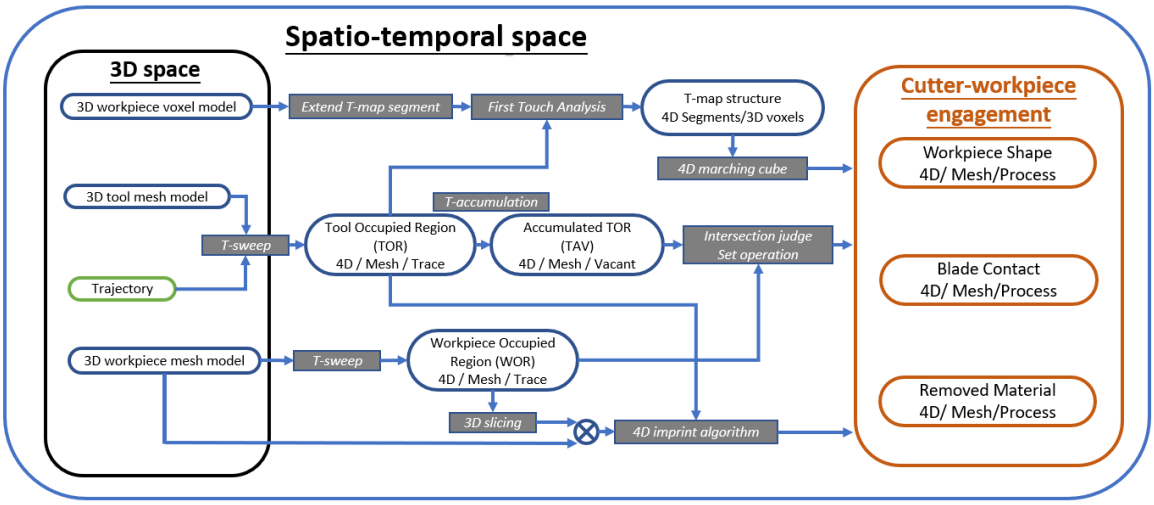


Fig. 1.4 CWE strategies

1.4 Dissertation outline

In addition to the chapter 1, this dissertation has other 6 chapters.

The chapter 2 mainly introduces the traditional 3D modeling representation of machining process. The machining process refers to a common 5-axis milling process. It is very common, generally and representative machining process. Then the 3D representation of CWE is introduced and discussed. The author would like to analyze the disadvantage of 3D representation of machining process and discuss the necessity of bring 4D mesh model to this field.

The chapter 3 introduces the 4D modeling and related technology. Meanwhile, it systematically summarizes the precursor and basic research of all 4D research used in this study, such as the part A and B in the research outline. Chapters 2 and 3 enable to be regarded as detailed description of the background of this study.

For reader understanding, the chapter 4 introduces tools which is used in this study.

Chapter 5 and 6 is the core part of the dissertation. They introduce in detail and systematically all the 4D model based CWE analysis methods proposed in this study.

Chapter 7 will conclude the dissertation. Three different approaches will be compared and discussed. The authors' contributions will also be described in detail. Finally, future plans and expectations are put forward for the unfinished parts of this study. The potential applications in part E of research outline are also mentioned in this chapter.

Chapter 2. Machining process and its 3D modeling representation

2.1 Typical 5-axis milling

5-axis machines feature a tool capable of movement in five distinct directions: along the X, Y, and Z axes, as well as rotation around the A and B axes, as illustrated in figure 2.1. This versatility enables operators to approach a part from multiple angles within a single operation, eliminating the need for manual repositioning of the workpiece between operations. This not only enhances efficiency by saving time but also renders 5-axis Computer Numerical Control (CNC) machining particularly well-suited for crafting intricate and precise components commonly encountered in industries such as medical, oil and gas, and aerospace. It's crucial for product teams to acquaint themselves with various types of 5-axis machines, including indexed 5-axis CNC machines, continuous 5-axis CNC machines, and mill-turning CNC centers. [29].

Indexed 5-axis CNC milling is similar to 3-axis CNC milling, where the

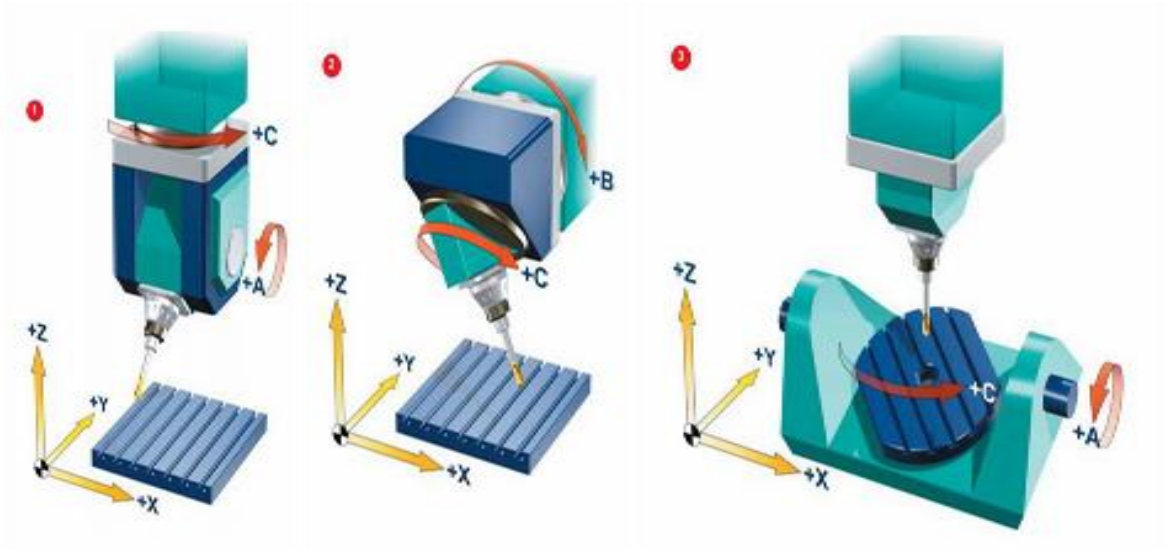


Fig. 2.1 Explanation 5-axis system
https://waykenrm.com/blogs/5-axis-cnc-machining/

cutting tool translation along three axes without maintaining continuous contact with the workpiece. Nevertheless, during operations, both the machining table and tool head can automatically swivel in two directions. Indexed 5-axis machining proves especially valuable in the fabrication of jigs, housings and fixtures. It offers a balance among the speed, precision, and complexity handling of 3-axis CNC milling and continuous 5-axis CNC machining[30].

In the process of continuous 5-axis CNC milling, both the workpiece and the cutting tool can rotate and move simultaneously. This not only saves time but also enables the manufacture of complex geometries with organic surfaces. Continuous 5-axis CNC milling provides enhanced speed, surface finish and dimensional stability, although it comes with a higher cost per part. Figure 2.2 is a 5-axis CNC system.

Fig. 2.2 A typical 5-axis milling system

Mill-turn CNC centers resemble CNC turning machines but are outfitted with CNC milling capabilities. The workpiece is affixed to a spindle that can either rotate or stay stationary while cutting tools chip away material. By amalgamating the capabilities of CNC lathe machines with milling tools, mill-turning CNC centers provide exceptional precision and geometric flexibility, rendering them well-suited for manufacturing parts with intricate rotational symmetries, such as camshafts or centrifugal compressors.

Types of 5-axis CNC milling equipment above not only offer higher accuracy when process deep parts and hardened materials but also provide higher yields and faster milling speeds. However, the specialized equipment and expertise required for 5-axis machining contribute to its higher costs.

Another commonly used 5-axis machining system is fixed 5-axis machining. fixed 5-axis machining is a technique where three-axis milling program is carried out with the cutting tool locked in a fixed direction using the two rotational axes of traditional 5-axis machine. Hence, it is called fixed 5-axis machining. This method is also known as "3+2axis machining" because the fourth and fifth axes are used to orient the cutting tool at a fixed angle rather than manipulating it continuously during the machining process. This is the major differences of 3+2 machining and simultaneous 5-axis machining [31].

2.2 Cutter-workpiece engagement of machining process

Milling is a dynamic machining process where the geometry of the workpiece being processed, and the cutting conditions of the cutter change continuously. In manufacturing field, a typical milling process simulation generally contains two parts: physical modeling and geometric modeling. Typically, the simulation process commences with geometric modeling, with its outputs often serving as inputs for subsequent physical modeling.

The CWE is one of an main output of the geometric modeling and can become bridges of the two successive modeling processing. There is an example of CWE in milling process in fig. 2.3. According to Gong et al., CWE refer to the instantaneous touch geometry between the cutter and the processing workpiece in milling

process[22]. In typical physical modeling and its simulation, CWE serves as the fundamental parameter for predicting cutting forces and analyzing the onset of chatter. Simulation of CWE enables the subsequent calculation of predicted cutting forces and torques in physical modeling and simulation [32].

In geometric modeling process of this dissertation, CWE represents the geometric shape morphing, how every blade sweep the workpiece and remove materials as it rotates and moves during milling process. This process considers all shapes of cutter, tool paths and processing workpiece shape containing even self-intersections. The difficulty and challenge of this kind of CWE analysis require to overcome is following three points: first, the complexity of the changing geometry of processing workpiece. Second is curved moving path of cutter. The final challenge is the complexity of cutter shapes representation, the shapes and arrangements of multiple blades, as well as the shapes and positions of different cutter heads and holders.

In preceding research, the utilization of these established studies has consistently been confined to simplistic cutter geometries and/or tool paths. Previous studies have proposed three fundamental strategies for determining CWE:

The initial approach computes CWE analytically at each tool position in the entire tool path, employing a series of time steps. The cutting edges are represented as curves in 3D space, and for each direction of the cutter, the intersections among these curves and the workpiece are calculated to determine the segments undergoing cutting. Larue and Altintas [33] introduced an approach to evenly distribute the cutter on tool path, resulting in a group of intersection curves subsequently utilized for CWE determination in 2005.

The second strategy involves the construction of the cutter swept volume (CSV), primarily utilized to update the processing workpiece. Subsequently, CWE is computed based on computing intersection condition of either the workpiece or the cutter with CSV. Aras et al. [25] utilized the removal volume (RV) which from the workpiece to extract CWE in 2008. RV represents the material on the workpiece swept or removed by the CSV based on a specific tool path segment, which is then mapped onto cutter's surface as CWE.

The final approach involves calculating CWE at intermediary cutter positions along a given tool path. Concurrently, Ferry et al [34] introduced a method

employing RV by a modeler. Their contribution is notable for introducing the "RV update" concept to address self-intersecting tool paths, a common issue in 5-axis machining.

These strategies outlined above encompass the majority of existing methods. Broadly speaking, the first method necessitates extensive computations to determine intersections at each direction of the cutter. The second strategy bases the concept of CWE geometry, use to physical simulation and modeling directly. Nonetheless, it leads to a proliferation of Cutter Swept Volumes (CSVs) and demands numerous computations between these volumes and the workpiece shape.

In contrast, the last approach utilizes concept of RV, which is significantly smaller than model of workpiece. Moreover, the RV aligns with the processing workpiece update procedure. While CSVs are generated update the workpiece, the RV can be concurrently obtained. At every designated cutter position along the tool path, this RV undergoes an update first, followed by the extraction of CWE[35].

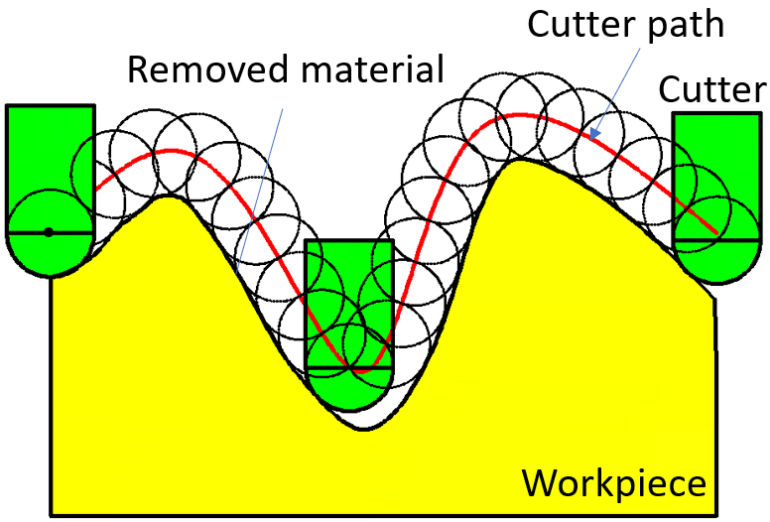


Fig. 2.3 A 2D analogy of CWE of milling process

2.3 3D representation of cutter-workpiece engagement

In Gong et al.'s paper, they select a specific geometric modeling method for implementation, after deciding the CWE determination strategy. Adaptability, modeling simplicity and accuracy are three important aspects when we select a

adapt geometric modeling method. Modeling the CWE geometrically is a crucial task, and it has been the focus of active research for many years.

Envelope method represents as the common approach to generating Cutter Swept Volume (CSV), it depicted cutter through explicit or implicit expressions. The fundamental principle underlying this method is that CSV envelope surface remains tangential to cutter surface throughout its movement. It serves as the cornerstone of CSV generation, initially pioneered by Wang et al [36], with subsequent enhancements by Chiou et al. [37,38] and Du et al. [39]. Envelope method lays down a robust conceptual foundation for CSV generating; however, it entails considerable computational expenses. In a bid to alleviate this computational burden, an alternative method was proposed also known as the imprint method [40,41]. At every tool position, the curve of tool surface, delineating the contribution of the cutter to the resultant CSV. It called as the generating curve. It epitomizes the cutter imprint on the machined surface. To adapt to a broader spectrum of tools, they devised a new parametric approach based on surfaces [42,43]. Obviously, a latest Gauss map-based approach successfully addressed the tool path self-intersection problem, albeit with restricted applicability to specific types of milling tools [44,45].

The aforementioned research endeavors rely on analytical methodologies, constrained by their limitations and assumptions. Attaining the CSV as a fully closed model in mathematical is often unfeasible, with most methods incapable of addressing tool path self-intersection issues. Consequently, approximate modeling methods emerge as a pragmatic choice for common applications.

Approximate CSV modeling methods usually use triangle mesh to model the CSV envelope [46]. This kind triangle mesh is regarded as representation with great promise, boasting several advantages: Firstly, it models a surface through interconnected triangles, thereby converting the 3D boundary model to a segmented 2D representation, facilitating simplified process of calculating intersection. Secondly, it enables the modeling of the cutter in a closed region, without limitations on surface configuration. Thirdly, it seamlessly integrates with most of commercial CAD software. Consequently, this work selects to use triangle mesh representations to tool and CSV modeling.

On other hand , there is a critical aspect of geometric modeling involves the processing workpiece representation. The instantaneous cutting dynamics of every

cutter position depend on the interacting situation between the cutter and the processing workpiece. Previous research efforts have tackled workpiece modeling using solid modeling as well as approximate modeling approaches. The solid modeling method is renowned for its precision in the aspect of accuracy.

In milling processes, the global tool path may contain thousands of segments, tool path segment data generation difficult to realize. Consequently, vary suitable modeling approaches for the processing workpiece have been developed, such as dexel, voxel, Z-map, polyhedral representations, and normal vector [47]. Among these methods, the Z-map approach stands out for its simplicity in implementation and robustness [48,49]. The Z-map approach represents the workpiece based on a collection of segments along Z-axis. Consequently, it don't good at modeling geometry and presents inaccuracies in depicting sharp edge or vertical walls features. In response to these limitations, Jerard [50] and Park [51] turned to normal vectors to refine workpiece modeling. A more recent advancement in discrete modeling is the triple directions dexel model [52], an extending concept of the single direction dexel (essentially akin to the Z-map). The triple directions dexel technique employs discrete segments in three vertical directions to represent the workpiece. These dexels undergo trimming with every milling tool path segment, following which processing workpiece is re-meshed by polygon mesh model.

Currently, researches highlight the complexity associated with in-process workpiece modeling, underscoring persisting challenges. Among the various modeling methods explored, one particularly promising approach involves modeling the processing workpiece as a triangle mesh model. Not similar to alternative suitable modeling techniques, the triangle mesh method stands out as the sole approach directly representing the workpiece as a closed model. This method boasts geometric isotropy, enabling the representation of surface details in all directions without constraints. Furthermore, when both the workpiece and CSV are modeled as polygon meshes, the set operation involves solely triangle-to-triangle calculations. these intersections may generate to some sharp features in workpiece, facilitating the generation of an accurate representation.

2.4 Disadvantages of 3D representation compare to 4D

When we are representing dynamic process of 3D shapes, a series of discrete snapshots of the shape have to be achieved. The motion of this shape is always represented by the location and shape data in snapshots. Therefore, for precise and complex processes, researchers have to use shorter time intervals, that lead to larger iteration times and longer calculating cost.

Comparing to 3D representation of machining 4D representation is an integration. Common 3D representation of a CWE process can be consisted by a series snapshots of moving 3D shapes or 3D model with it moving trajectory. Regardless of the method, expressing the geometric changes of CWE requires some discrete data, and it is also discrete in time. At the same time, the state of the intermediate time needs to be extrapolated or interpolated from the known time state, and this part of the calculation is unavoidable. Therefore, if you want a computer or AI to understand this kind of data, a sufficient number of calculations are necessary. The 4D model technology can integrate the entire machining process into one geometric model. In other words, all data about shape and motion can be represented integrally by a 4D geometric model. These 4D geometric models are continuous in each direction, and they can reflect all movement and deformation state of the object at every moment. As a model that can contain all required data without additional calculations, it is more suitable and can be applied to computers and AI faster than 3D representations that require additional calculations. Through The four-dimensionalization of 3D dynamic process, the complex motion and deform of cutting process can be geometricized. This also allows us to approach the machining process with a more homogeneous and simple geometric method which is suitable to AI and computer[53].

In addition, the 4D model gives us a broader perspective and a richer geometric based CWE analysis method. We can use computers to analyze the 4D model as a whole, or you can use the 3D representation 4D models to provide different focuses of observation and analysis, such as simple XYZ, YZT, XZT hyperplane slices or projection which perpendicular to a certain coordinate axis. For example, by performing XZT hyperplane slicing, we can observe the time-varying state of the

XZ 2D section at a fixed Y. This is very useful for analyzing the touch condition between the tool and the workpiece at fixed position. Therefore, our analysis directions for four-dimensional models can also be diverse and are not limited to the positive time direction of 3D representation. Such analysis may be difficult for humans, but for computers and AI it is just a geometric model. Meanwhile the 4D model is good at represent the deform process and the deform with displace which not suitable for represent by 3D models. If the meaning of each axis is changed for example the fourth axis is used to indicate temperature. A representation of a wider range of situations can be realized with a 4D model.

We hope that this research can serve as a catalyst for 4D models in digital manufacturing fields such as CWE and even CAM. It is also expected that 4D geometric models can be more widely used in various fields.

Chapter 3. Introducing 4D modeling in spatio-temporal space

3.1 Geometry in spatio-temporal space

In various domains such as robotics, manufacturing, construction, and medical science, there is a growing need for 4D modeling to accurately depict and analyze dynamic objects that undergo continuous changes in shape and position over time. Traditionally, a widely adopted method for 4D modeling involves creating a sequence of 3D frames, each representing the object at specific time intervals as static 3D geometric models. This conventional approach to 4D modeling segments the continuous spatio-temporal domain into discrete 3D subspaces, each capturing the object's scene without temporal variations.

Dynamic object process is captured discretely through a series of static 3D snapshot, akin to films used inmovie and animation videos. To assess properties depend on time such as motion and deformation, relevant scenes at specific time points are retrieved, enabling the reproduction of object changes from multiple 3D models. Remarkably, the handling of the temporal dimension within the spatio-temporal realm diverges from the treatment of the remaining trio of axes (X, Y, and Z) within a 3D subspace, and effectively assessing temporal-related characteristics is highly contingent upon the manipulation of model data through programming.

Introducing 4D geometry modeling allows for the representation of dynamic objects within a 3D space as static objects within a 4D space. In this approach, the time axis is consistently treated on par with the other three axes, facilitating a more cohesive representation of temporal dynamics.

3.2 4D mesh model and te4 file

To introduce 4D geometric modeling, we begin by explaining geometric elements in four-dimensional space. The simplest shape in 3D space is a tetrahedron, that is also the 3-simplex in 3D space, with 4 vertices (V), 6 edges (E), and 4 faces (F). The most commonly recognized polyhedron is the cube in 3D space. And every

polyhedron adheres to the formula $V - E + F - C = 0$ in 3D space. Moving to 4D space, the 4-simplex is referred to as a pentachoron, consisting of 5 vertices, 10 edges, 10 faces, and 5 cells (C). The 4D equivalent of a cube is a hypercube, which features 16 vertices, 32 edges, 24 square faces, and 8 cells. Similar to 3D space, every 4-polytope in 4D space follows equation $V - E + F - C = 0$.

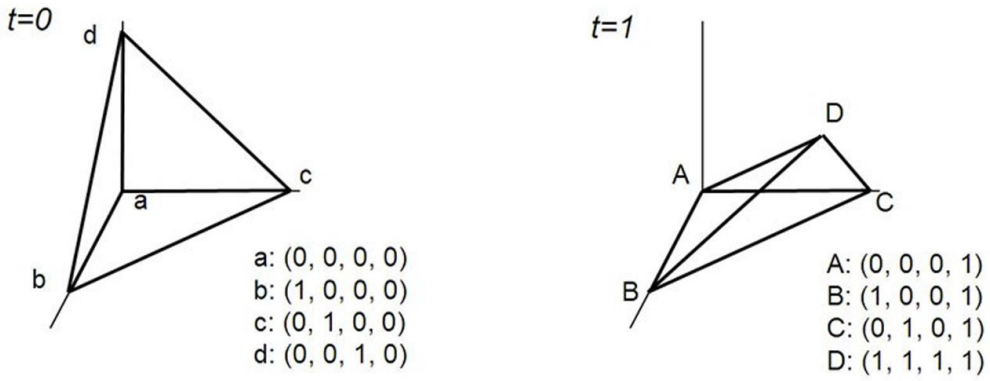


Fig. 3.1 Deformation of tetrahedron

In the realm of 4D space, every 4-polytope is encompassed by cells, akin to how every polyhedron in 3D space is surrounded by some faces. These cells within the 4D domain are polyhedra situated within 3D subspaces, commonly called as hyperplanes. These cells can be further subdivided into tetrahedrons. Consequently, a 4-polytope in 4D space can be depicted by a collection of tetrahedron forming its boundary. This representation, termed as a 4D mesh model in this study, provides a comprehensive portrayal of the 4-polytope's structure and geometry.

Four points placed in 4D space $p_i = (x_i, y_i, z_i, y_i) \in R^4, (i = 0, 1, 2, 3)$ are used to construct a tetrahedron on the hyperplane that normal n is computed based on normalizing 4D vector m by $m = m'|m|$.

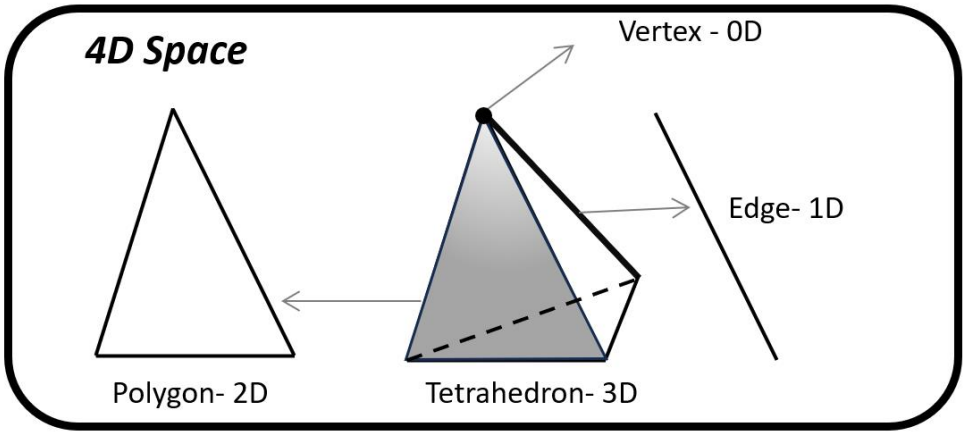
$$m = \begin{vmatrix} e_x & e_y & e_z & e_t \\ x_1 - x_0 & y_1 - y_0 & z_1 - z_0 & t_1 - t_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 & t_2 - t_0 \\ x_3 - x_0 & y_3 - y_0 & z_3 - z_0 & t_3 - t_0 \end{vmatrix}$$

Where e_{axis} is the unit vector of corresponding axis .

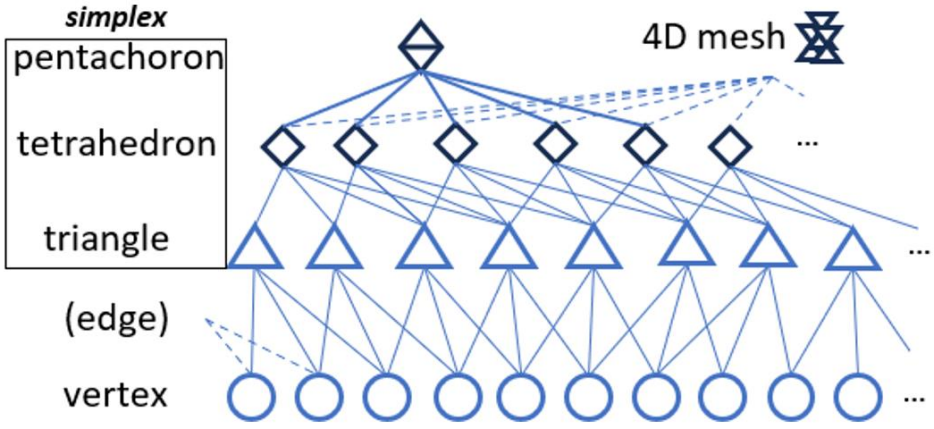
Consistently determining which half-space lies inside or outside each boundary tetrahedron relies heavily on organizing the tetrahedron's four vertices in a serialized manner. For instance, altering the sequence of vertices from 0-1-3-2 to 0-1-2-3

results in an inversion of the normal vector. This careful consideration of the normal vector direction in 4D space is analogous to the importance of differentiating between clockwise (CW) and counterclockwise (CCW) loops in 3D triangle mesh modeling.

In the provided example illustrated in Fig.1.4, a tetrahedron undergoes deformation. At the initial moment, denoted as time 0, the tetrahedron's four vertices are situated at 4D coordinates labeled as a, b, c, and d. By t = 1, vertices a, b, and c maintain their coordinates from t = 0, while vertex d, initially at (0, 0, 1, 0), moves to D (1, 1, 1, 1) as depicted which is shown in fig. 3.1. Consequently, the tetrahedron experiences deformation between time 0 and time 1. Subsequently, let's examine how this deformation process is depicted in 4D mesh modeling, as shown in Fig. 3.2 (b).



(a) Topologic elements in 4D space.



(b) Topologic relation in a tetrahedron.

Fig. 3.2. Geometric structure of 4D mesh model

In the 4D mesh model representing the deforming tetrahedron, vertices are derived from the vertices of original tetrahedron: A, B, C, D, a, b, c and d. The model comprises 14 cells of tetrahedron that define this 4D structure boundary, including the initial tetrahedrons abcd and ABCD, along with additional ones generated during the time interval. Each tetrahedron shares its triangular faces with four adjacent tetrahedrons, resulting in a total of 28 triangular faces and 22 edges.

The detailed data structure depicted in Figure 3.2 underscores the complexity of representing even a basic 4D object. For reducing a 4D mesh model's data size, faces and edges can be excluded from the model structure, as these elements and their relationships can be uniquely inferred from the cell data.

Introducing neighborhood links connecting adjacent tetrahedra facilitates efficient retrieval of information about faces and edges [6].

In our laboratory, we usually use a kind of file name extension is te4 to store 4D mesh model data. Te4 file is a binary file. The te4 including:

1. Three 32 bits integers, represent "dtypeID", vertices number and tetrahedrons number. The "dtypeID" indicate if the file has normal vectors data of tetrahedrons.
2. An array of 32 bits floating number. Each group of four floating numbers represents the coordinate value of a vertex. The length of the array is four times the number of vertices.
3. An array of 32 bits integers. Each integers represents the index of a vertex. Each group of four integers represents 4 vertices of tetrahedron. The index of vertex is corresponding to the sequence of part 2. The length of the array is four times the number of tetrahedrons.
4. If "dtypeID" = 2, there is an array of 32 bits floating number. Each group of four floating numbers represents normal vector of a tetrahedron. The length of the array is four times the number of tetrahedrons. If "dtypeID" = 1, there is nothing and the normal vector can be calculated by the sequence of 4 vertices of tetrahedron in part 3, in general case.

3.3 Observe 4D mesh model

3.3.1 Observe 4D meshes by hyperplane slicing

In the author's conceptualization of the study, the observation of the 4D model does not have any effect on the before and after process. All 4D models should be left to the computer and only the results should be visualized.

In such a process, the method of visualizing the 4D model is still necessary to ensure that the relevant state of the 4D model is obtained intuitively for the user and the results are presented. Since our visual system can only provide a 2D planar image of the 3D model, the approach to visualize the 4D model is generally to obtain a 3D model that can represent it as well as possible. Generally there are two methods for viewing 4D models, a slicing method and a projection method [54].

Below the author would like to explain slicing method with the mesh model as the main body.

The essence of the slice method is to show a situation in a subspace of the current space, where only the graphics within that subspace are shown. When using the slicing method this subspace is referred to as a slice, which is usually only one dimension lower. For example, if we slice a 3D model through a 2D planar space, we can get a 2D mesh model in that 2D planar space, which will be surrounded by 1D end-to-end Edges. Let's take this up to a 4D model. Obviously, a slice of a 4D

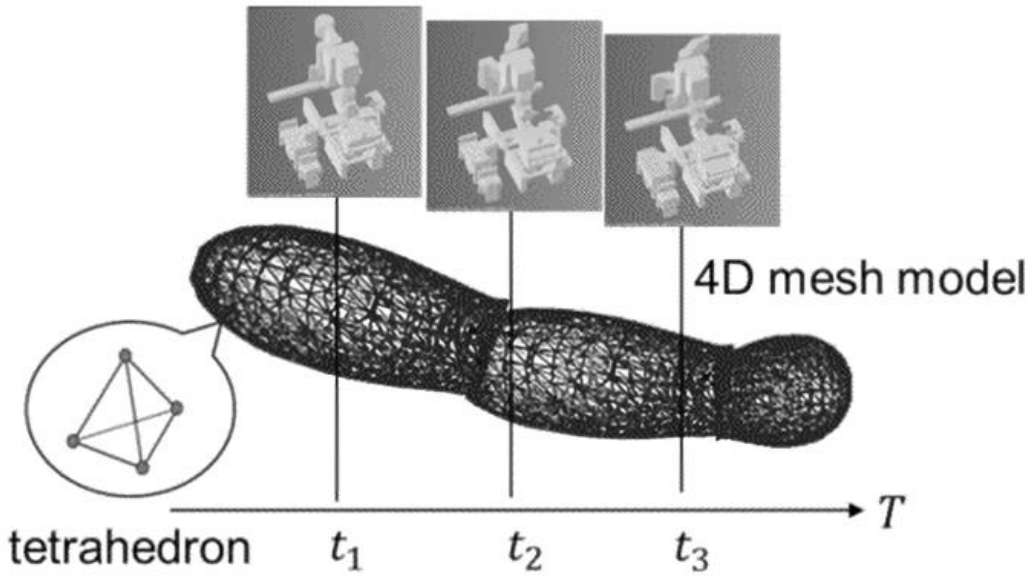


Fig. 3.3 Analogy of Slicing method

model is a 3D subspace, and a 3D subspace can be interpreted as an upscaling of the 2D planar space, which can also be called a 3D hyperplane. When we use a 3D hyperplane to slice a 4D model. We can get a 3D mesh model surrounded by 2D

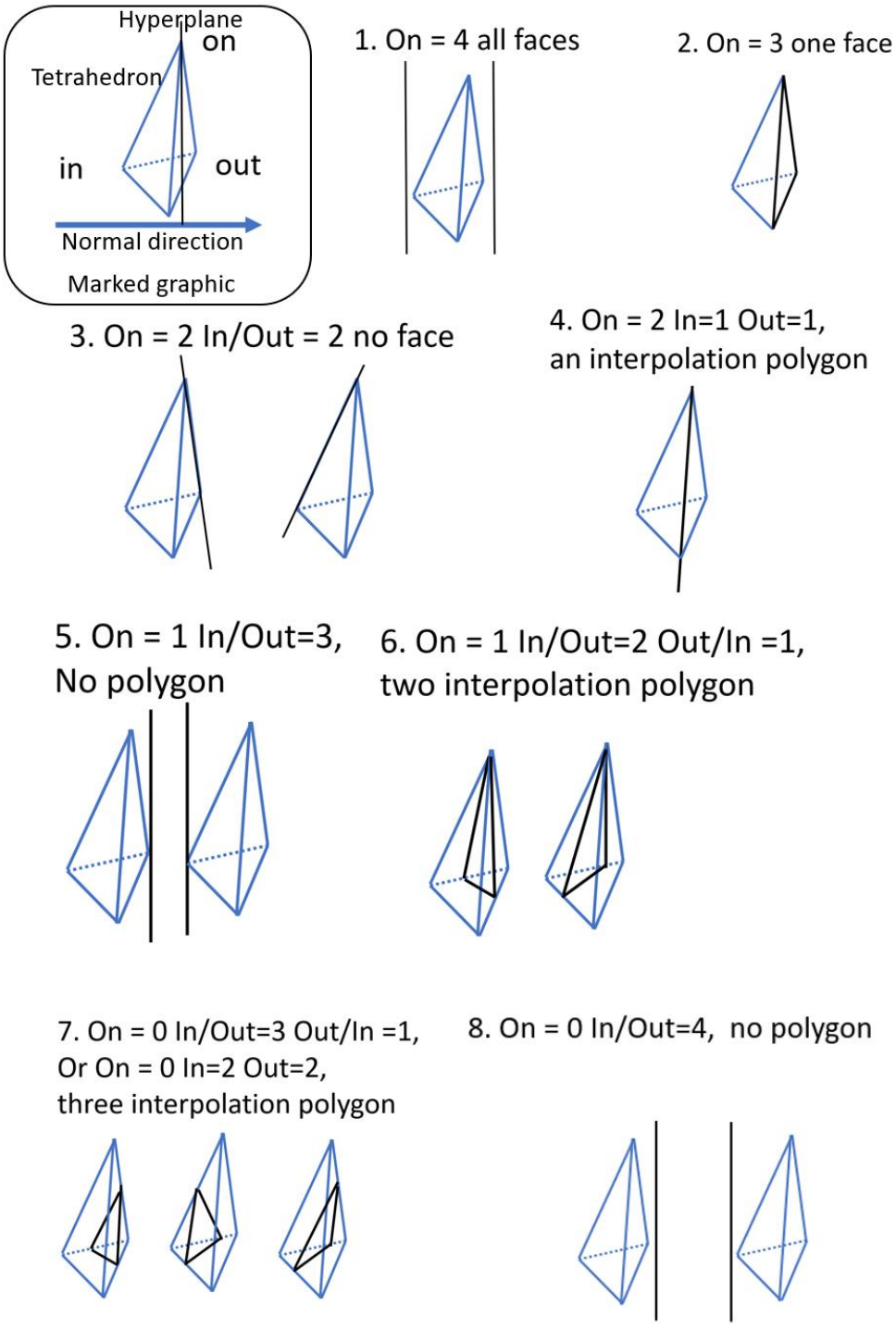


Fig. 3.4 Cases of slicing method

polygons on that 3D hyperplane. This process illustrates in fig. 3.3.

In general, the 3D hyperplane used in slicing method is usually parallel to the X- ,Y-, Z- and T- axes in order to get more explicit meaning and results. Such 3D hyperplanes perpendicular to the axes can represent the 3D shapes in YZT, XZT, XYT, and XYZ spaces located at the determined X,Y,Z, and T values. In the CWE process, perpendicular to the T-axis, for example, this means that it represents the machined shape and the machining when the time is t. Perpendicular to the Z-axis can be used to observe the XY2D plane over time.

The basic idea of the algorithm is to discuss the situation separately, case by case. When a 3D hyperplane cuts a mesh model that exists in 4D space, it can be microscopically represented as a 3D hyperplane cutting a tetrahedron, a fundamental element of a 4D mesh model. a tetrahedron being cut by a 3D hyperplane will likely exist in the following situations (the relevant arguments and detailed descriptions will be shown in section 7.2):

For ease of representation, when a hyperplane cuts the tetrahedron, there will be three cases of vertices on the tetrahedron. “On” means that the vertex lies on the hyperplane. “In/Out” means that the vertex is on both sides of the hyperplane. In general, when we assign a normal vector to a hyperplane, ‘In’ means the negative direction of the normal vector, and ‘Out’ means the positive direction.

Figure 3.4 shows all usually used cases in hyperplane cutting, and list below:

1. All four vertices of the tetrahedron are on the 3D hyperplane. on = 4. This will completely represent the shape of the tetrahedron on the hyperplane.
2. Three vertices of the tetrahedron are on the 3D hyperplane. on = 3. Since any three vertices of the tetrahedron can form a polygon, the polygon will be represented on the hyperplane.
3. Two vertices of the tetrahedron are on the 3D hyperplane, and the other two vertices are on the same side of the 3D hyperplane. On = 2, In/Out = 2. Only one Edge on the tetrahedron is on the hyperplane, so only one 1D Edge is shown on the hyperplane.
4. Similar to case 3. In this case, two vertices of the tetrahedron are on the 3D hyperplane, but the other two vertices are on either side of the 3D hyperplane. On = 2, In = 1, Out = 1. The tetrahedron is then sliced into a polygon, which shares a single Edge with the tetrahedron, and the last

5. vertex is obtained through interpolation.
6. Only one vertex is on the 3D hyperplane and the other three vertices are on the same side of the 3D hyperplane. on = 1, In/Out = 3. There is only one vertex, which is represented on the hyperplane.
7. Only one vertex is on the 3D hyperplane, and the other three vertices are on both sides of the 3D hyperplane. On = 1, In/Out = 2, Out/In = 1. The displayed polygon will have one vertex shared with the tetrahedron, and the remaining two vertices will be obtained by interpolation.
8. No vertices are in the 3D hyperplane, and four vertices are on either side of the tetrahedron. On = 0, In/Out = 3, Out/In = 1 or On = 0, In = 2, Out = 2. All three vertices of the polygon which is displayed will be interpolated.
9. No vertices are on the 3D hyperplane and four vertices are on the tetrahedron side. On = 0, In/Out = 4. the hyperplane is not intersected by the tetrahedron.

3.3.2 Observe 4D meshes by projection

If the slice method represents a localized 3D situation, the projection method can represent both the local and the overall 3D situation. The comparison can see Fig. 3.3 and 3.5. Figure 3.6 is an instance of two displaying methods of 4D Tool-occupied region (TOR) model.

In this study, we use parallel projection to project a 4D model onto a selected virtual 3D hyperplane. If a parallel projection is performed on a 3D mesh model, we

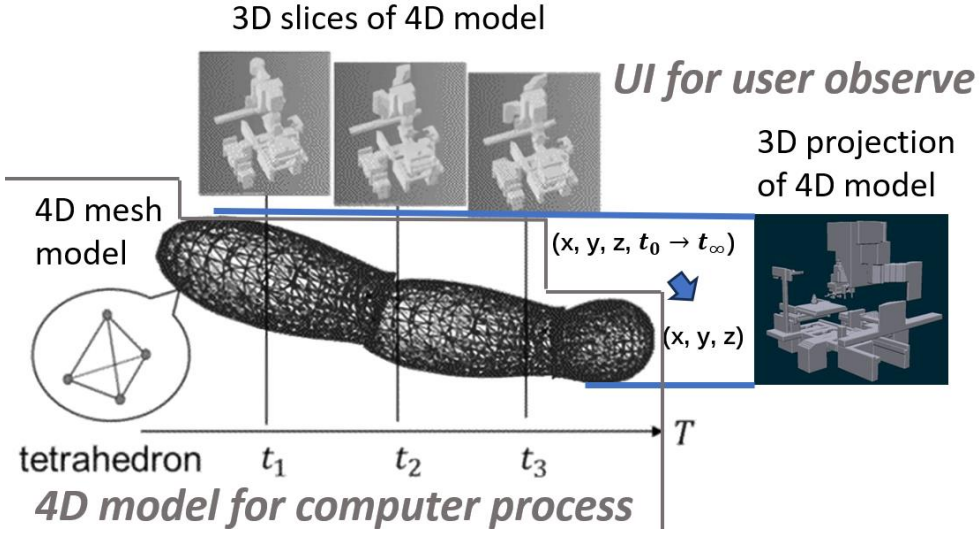


Fig. 3.5 Comparison between two displaying methods

can obtain a 3D figure. And by performing a parallel projection on a 4D mesh model, we can get a 3D shape. Similar to the slicing method, this 3D hyperplane usually

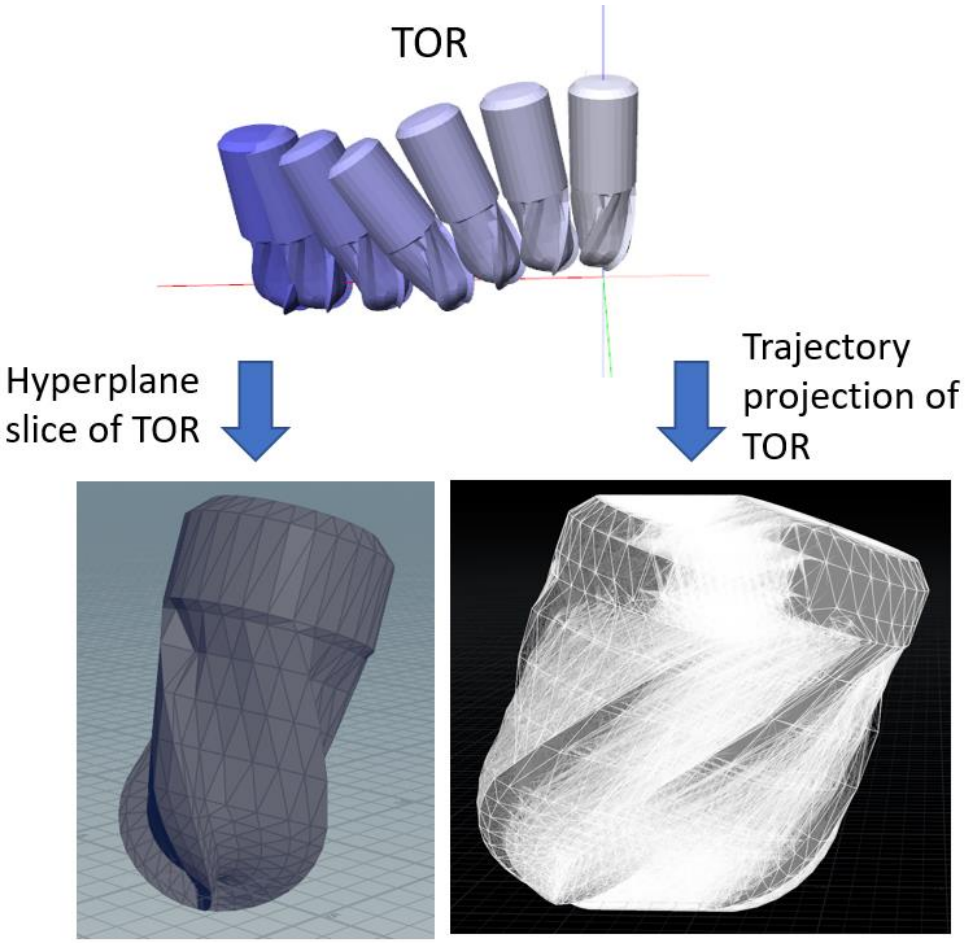


Fig. 3.6 Example of two displaying methods

needs to be perpendicular to the X-, Y-, Z-, and T-axes in the CWE representation in order to gain meaning. For example, a parallel projection on the XYZ hyperplane perpendicular to the T-axis can usually be used to represent the trajectory of an object over a period of time in CWE. Such a hyperplane may or may not intersect the model to obtain different local and global 3D models. The method is also simple to compute, and does not need to discuss and derive the interpolation and polygon generation for each case as in the slicing method. In most cases, it only needs to remove the values of the perpendicular axes, such as (x,y,z,t) to (x,y,z) and deduplicate the generated redundant vertices.

3.4 4D mesh modeling

3.4.1 4D marching cube

One of the most effective methods for constructing 4D mesh models is via iso-surfacing 4D voxel data, as proposed by Bhaniramka et al. in 2000[5]. This approach extends Lorensen et al.'s 3D marching cubes algorithm to 4D space[55]. The 4D marching cubes technique transforms 4D voxel into a group of tetrahedrons by referencing a 4D triangulation table that contains 2^{16} patterns in a hypercube. Every pattern can be record as a 16 bits signifying in/out flag, ranging from 0 to

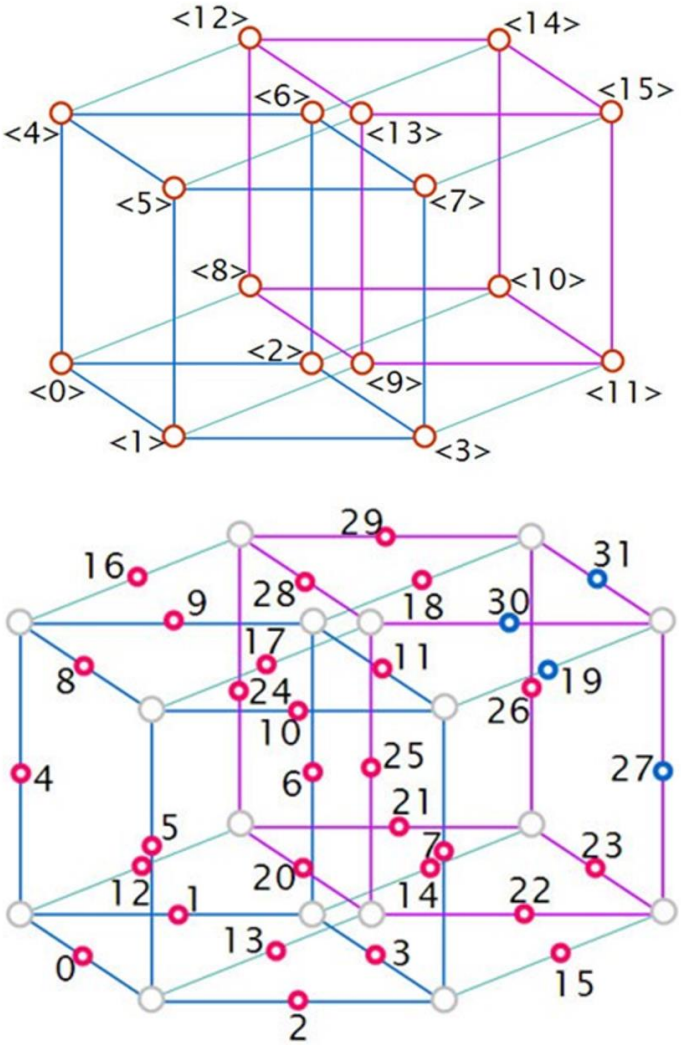


Fig. 3.7 Vertices index numbers of hypercube (upper) and Edges index numbers of hypercube (below)

65,535 (indicating all voxels are outside to inside). In the 4D marching cubes approach, the vertices of the generated tetrahedrons are situated along the edges of a hypercube and are denoted by edge index numbers, which shown in fig. 3.7.

The 4D triangulation table establishes a correlation between the corresponding and set of tetrahedrons hypercube pattern codes are created. By this method, a hypercube pattern code like 10 (which is consisted by 2^3+2^1), indicating that vertices No. 3 and 1 have inside flag while the remaining vertices are outside, may generate three tetrahedron patterns with edge numbers such as 3-0-5-13, 7-3-5-13 and 15-3-7-13. The mapping table covers 65,536 patterns in a hypercube and results in the generation of 856,960 tetrahedrons, which can be classified into 8,036 patterns where vertices are located on edges. As fig. 3.8, the hypercube patterns of this 4D triangulation table generally average match to 13.1 tetrahedral elements., with a maximum of 26 tetrahedrons per pattern. Consequently, the 4D marching cubes approach often produces a large number of tetrahedrons from 4D voxels shape, presenting challenges in its application to large-scale time-varying datasets [56].

Onosato et al. have created a kind of algorithm based on advancing frontier edge for 4D marching cube, which eliminates the need for vertices merge processing since each vertex is generated only once. To illustrate the algorithm, they provided a simplified instance of mesh generation from a sequence of one-dimensional (1D) voxel models, as shown in Figures 3.9 and 3.10.

A time series comprising 1D voxel models, that save 0/1 values, is loaded into main memory at a time. The initial voxel model (t=0s) is primed for model termination, with all values are set to zero. If size of the model is $N+1$, N boxes

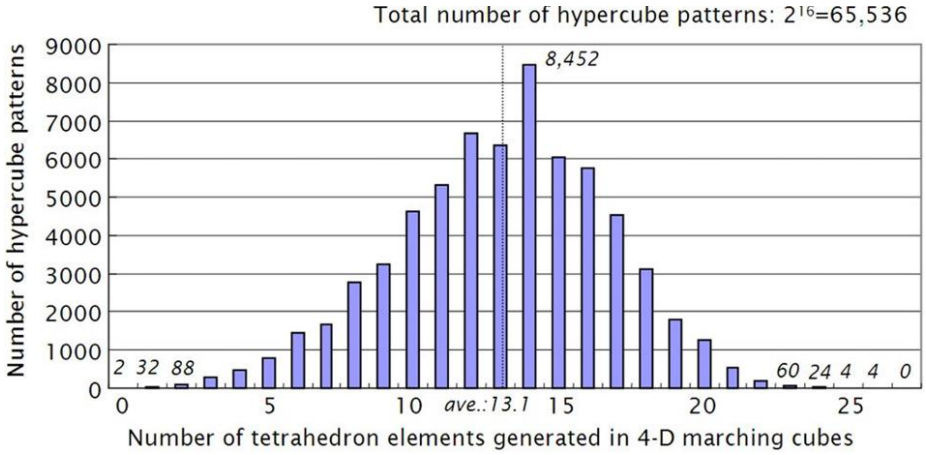


Fig. 3.8 Distribution situation of elements in 4D triangulation table

are set in line and prepared. Coordinate values of each box is (x, t) and index value can be calculated by $k = x + N_x$. Four vertices Exist in each box, labeled a, b, c, and d, illustrated in Fig. 3.10. The "advancing frontier edges" is edges cd and bd, because they do not have neighbor boxes once processed. As depicted in Fig. 3.9, a box consists of two levels: the lower level containing vertices a and b, and the upper level containing vertices c and d. Vertices on the lower level retain values from the preceding voxel model, while vertices on the upper level hold values from the processing model. The method set Voxel value of current model to every vertex of two neighbor boxes. The vertex d in the box indexed has same value to vertex c in the box indexed $k + 1$, which is shown in fig. 3.9. For differentiating the vertices created on an edge, the notation like (box index):(edge index) is introduced. One of the vertex located on the right edge is assigned an edge index of 0, whereas a vertex situated on the top edge is designated an edge index of 1. The left edge also employs an edge index of 0, akin to the right edge, yet it relies on the index of the adjacent left box. Then, a vertex located on the bottom edge is referenced by the box index number of the previous layer. The indexing approach allows for the creation of vertices without duplication. Figure 3.10 depicts the procedure of line generation utilizing the advancing edge frontier algorithm. Upon completing the line generation of one layer, the vertices on the upper and lower levels are swapped, and the upper level is reset to become a new layer. Execute alternating and reusing process of these two levels, the main memory space required is minimized to only one layer.

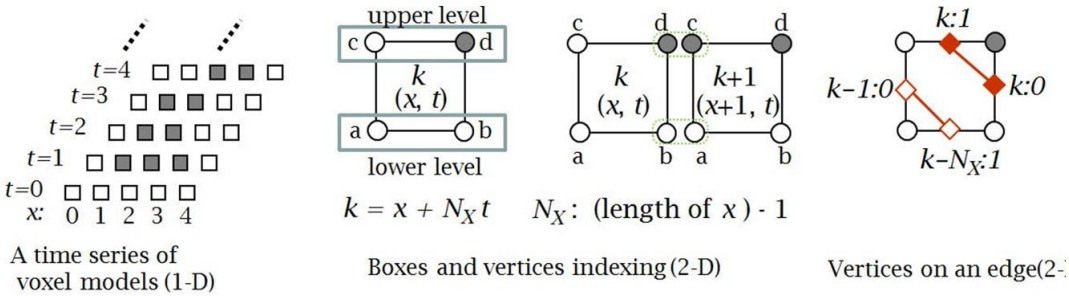


Fig. 3.9 A simplified 2D example of indexing concept of boxes and vertices

The method outlined in 2D spatial context naturally extends to 4D spatial dimensions, evolving into the advancing frontier edge strategy tailored of 4D marching hypercubes. We note these edges as No. 19, 27, 30, and 31, and are indicated as 0, 1, 2, and 3, respectively. Every hypercube with a coordinate (x, y, z, t) in 4D space and the hypercube index is defined by $k = x + N_x (y + N_y(z + N_z t))$, where N_x, N_y, N_z are the numbers of hypercubes along different axis (x, y, z) respectively. They give the definition as:

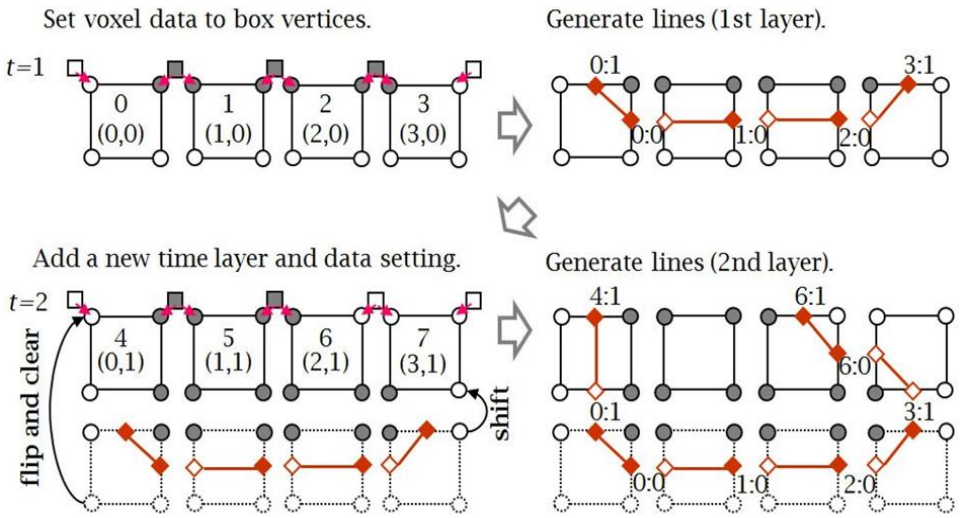


Fig. 3.10 A simplified 2D example of line generation process

$$k = x + dz + N_x (y + dy + N_y(z + dz + N_z(t + dt)))$$

Vertices created and notated in the form $k{:}pn$ can be settle to a vertex list, each vertex is assigned to an index number by this list.

3.4.2 4D ball pivoting

BPA[57] is a method of 3D surface reconstruction in which a 3D point cloud is scanned by a sphere of appropriate radius and a triangulation is constructed from the points in contact with the sphere. Yabuki et al. extended this method to 4D and proposed 4D BPA[13] , which reconstructs a 4D surface by scanning a 4D point cloud with a hypersphere of appropriate radius and constructing a tetrahedron with the points in contact with the hypersphere. Hereafter, the hypersphere S^3 refers to the boundary of the 4D sphere in 4D real Euclidean space.

The process flow of 4D BPA is shown below, where $\vec{V}(E^4)$ is the vector space associated with the 4D real Euclid space E^4. A illustration also provide as Fig.3.11.

1. Seed Tetrahedron Search Process : Select 4 points from the 4-dimensional point cloud $P \doteqdot \{\vec{p} \in \vec{V}(E^4)\}$, such that the hypersphere S^3 of radius r_{gl} is in contact and the interior of S^3 contains no other vertices. 4 triangles constituting a tetrahedron are "Active" states.

2.Ball-pivoting process: Select a triangular T_{pv} that is active and rotate S^3 in the direction away from S^3 to the remaining point $\vec{p}_{op}$ of the tetrahedron, using the plane containing T_{pv} as the rotation plane. The first point of contact $\vec{p}_{new}$ and T_{pv} are then used to construct a new tetrahedral element. The newly obtained triangles are assumed to be active. The newly obtained triangle is assumed to be Active, and T_{pv}, which was fixed by the rotation, is updated to the "Frozen" state.

3.Termination process: The ball-pivoting process is repeated until there are no more triangles in the Active state. Finally, the tetrahedrons obtained from the **Seed Tetrahedron search** and **ball-pivoting processes** are collected to form a 4D mesh model by aligning the orientation of the tetrahedron elements.

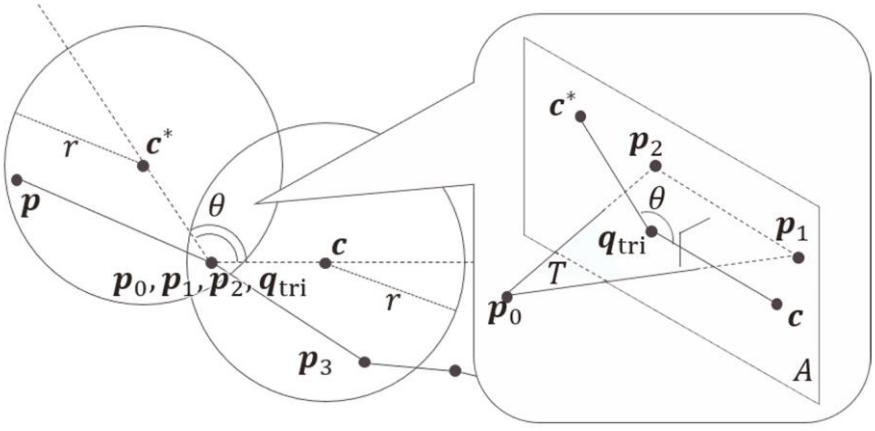


Fig. 3.11 Schematic diagram of 4D BPA

3.4.3 Direct method of moving rigid body based on time sequence (Otomo's method)

Otomo et al. developed a direct 4D mesh modeling method for a moving three-dimensional (3D) rigid body, which is effective and convenient, albeit limited in its ability to represent shape deformation of the object.

Otomo's method tracks the movement of the polygons in a primitive 3D model over a time interval. It gives a polygon the time information and record its new coordinate after the time interval. Then it connects corresponding vertices of primitive polygon and new polygon to generate a triangular prism in Spatio-Temporal space. Each prism is meshed to 14 tetrahedrons, and there are three kinds of tetrahedrons in one prism. This structure makes 4D modeling convenient and shown in Fig. 3.12. Compared to other 4D model construction method, such as 4D marching cube method. The Otomo's method's feature, construct prism and subdivision, allow it to generate a controllable number of tetrahedrons while retaining sufficiently high accuracy of 4D model. While preventing data explosion, it is only suitable for expressing the movement of rigid bodies. These movements can include uniform and variable speed translations as well as all rotations. This is exactly suitable for representing tool path models that do not deform but only move during the CWE process. Meanwhile, it is different from existing 4D modelers, such as the F-rep 4D modeling tool by Pasko et al [12]., Otomo's method tries constructing 4D meshes instead of B-spine with a new scalar value.

The method mainly consists of following four steps:

Step 1. Begin by tetrahedralizing the interior of a 3D mesh shape in the initial state;

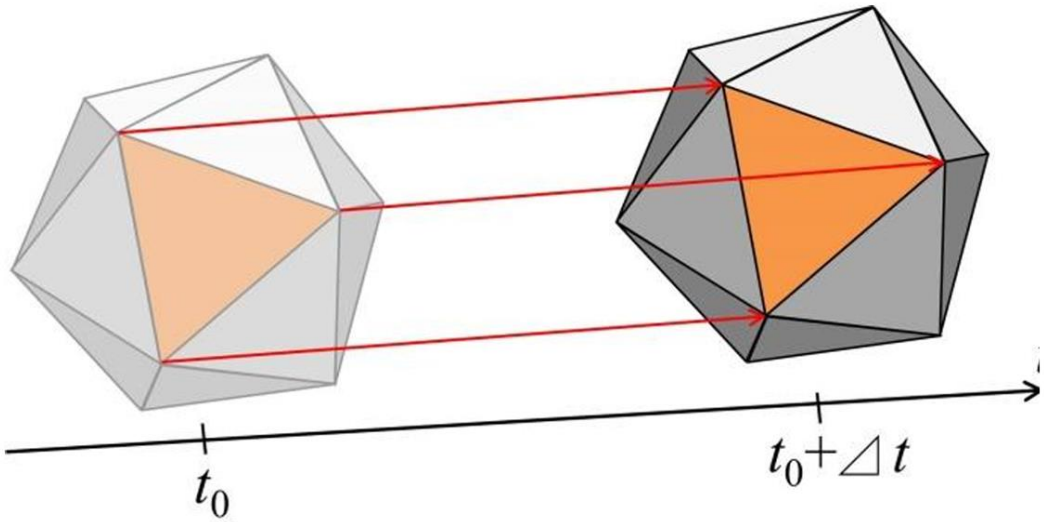


Fig. 3.12 Swept shape of a triangular mesh along the spatio-temporal axes

Step 2. Moving this 3D mesh model sequentially and repeatedly tetrahedralize result of 4D swept shapes;

Step 3. Tetrahedralize the interior of this 3D mesh model in its final state;

Step 4. Create 4D meshes by merging the tetrahedron generated in Steps 1, 2, and 3.

Steps 1 and 3 play vital roles in delineating both ends normal to the t-axis of a 4D mesh model through a series of tetrahedra. Thus, it becomes imperative to tetrahedralize the interior of a 3D mesh model to achieve this representation. Consider this situation, the addition of new vertices becomes imperative due to the understanding that certain shapes cannot be tetrahedralized without this augmentation [1, 14]. Additionally, it is crucial not to add extra vertices on the boundary shapes, while ensuring the preservation of these shapes. Therefore, the Constrained Delaunay Tetrahedralization method [15] is employed. During the tetrahedralization process, this approach regards the boundary shapes as constrained conditions. When tetrahedralizing a 3D mesh model, it is imperative to verify that

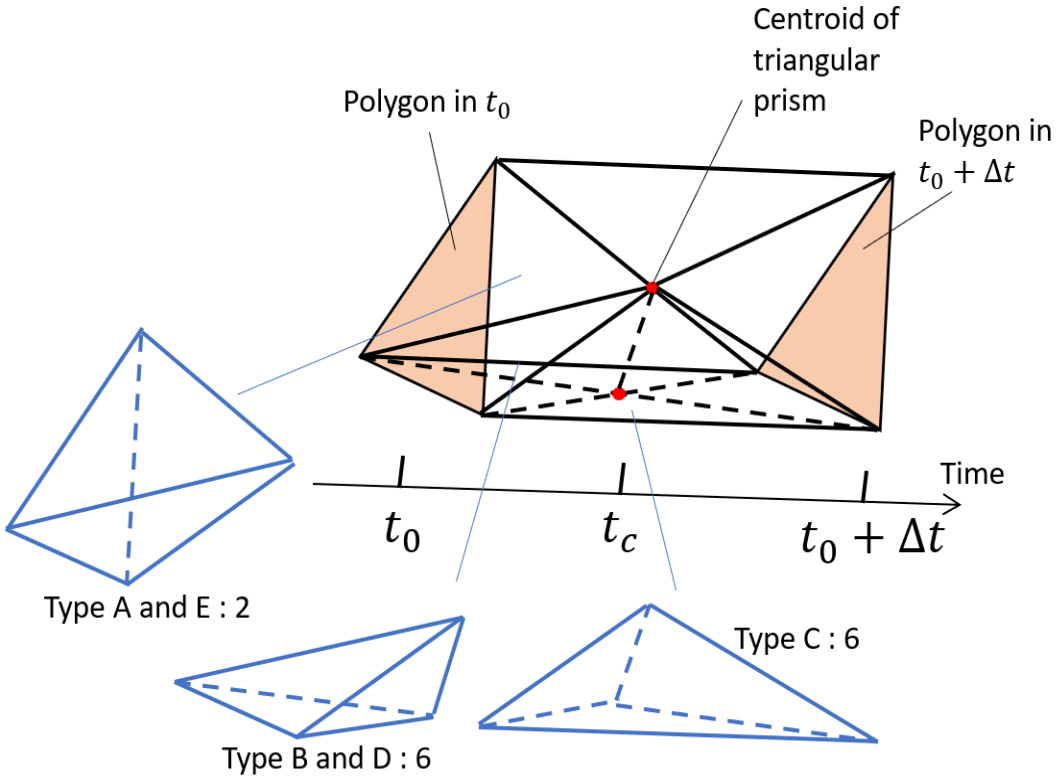


Fig. 3.13 Tetrahedrons type in Otomo's method

there are no missing or overlapping meshes within the model. Failure to do so may result in inaccuracies in the outside/inside determination of 3D mesh model and could lead to incorrect tetrahedralization outcomes[58].

Regarding step 2, the swept shape of a mesh along the spatio-temporal axes bears resemblance to a triangular prism, as depicted in Fig. 3.13. To tetrahedralize inside of triangular prism, they introduced four new vertices: one is the centroid of the triangular prism and three is the centroids of three square face. This results in the generation of two tetrahedra from the triangular faces and the prism's centroid, while each square face yields four tetrahedra, incorporating the prism's centroid and the face's centroid as vertices. Consequently, a triangular prism can be divided into 14 tetrahedrons.

Table 3.1 5 kinds tetrahedrons in prism

Tetrahedrons types	Vertices on t_0	Vertices on t_c	Vertices on $t_0 + \Delta t$	Number
A	3	1	0	1
B	2	2	0	3
C	1	2	1	6
D	0	2	2	3
E	0	1	3	1

In Otomo's study, the necessity for a clear correspondence between vertices before and after movement restricts the focus to movement without topological changing only. But the author think the implementation of Otomo's method can extend to all dynamic process which topological structure of 3D model is not change. As long as the pair of corresponding polygons can be found, even if they have different shapes, we can connect their vertices in sequence and tetrahedralize the resulting prism.

There are five kinds of tetrahedrons exist in a prism, showed as A,B,C,D,E in table 3.1 and fig.3.13. These tetrahedrons have some features:

- 4D mesh model is consisted by many time intervals
- In the period of time interval, there are only three different T-value (t_0, t_c,

$\boldsymbol{t_0 + \Delta t}$) of all vertices.

- Type A, B, D and E tetrahedron exist between two adjacent T-values.
- Type C tetrahedron stretch across all of three T-values.

3.5 Using 4D models to represent CWE in spatio-temporal space

The CWE condition of a machining process consists of deforming of workpiece shape, contacting of tool's blade and removing material. All of them are a process and represent by a series of snapshot in 3D method. Computing CWE in Spatio-Temporal space is similar to 3D but more directly. Naturally, the CWE enable to achieve by the set operation between the Tool accumulated volume (TAV) and workpiece-occupied region (WOR) such as the Eq(3.1) and (3.2) [59].

$$Vol(Removed\ material) = TAV \cap WOR \quad \text{Eq (3.1)}$$

$$Vol(Machining\ process) = WOR - TAV \quad \text{Eq (3.2)}$$

$Vol(Removed\ material)$: 4D meshes of removed material.

$Vol(Machining\ process)$: 4D meshes of remained volume.

3.5.1 Tool occupied regions (TOR)

Tool-occupied regions (TORs) are derived from models of tool shapes and fundamental tool movements. They constitute a form of 4D representation, embodying standard cutting process parameters such as basic tool trajectory, velocity, and tool axis alignment. The structure and creation process of TOAs are depicted in Fig 3.14.

In other word, it can be regarded as a small differentiation of the 4D tool sweep volume. A TOR can be modified by changing some parameters, such as resetting tool's location, orientation, speed and so on. Meanwhile, extend the process time of

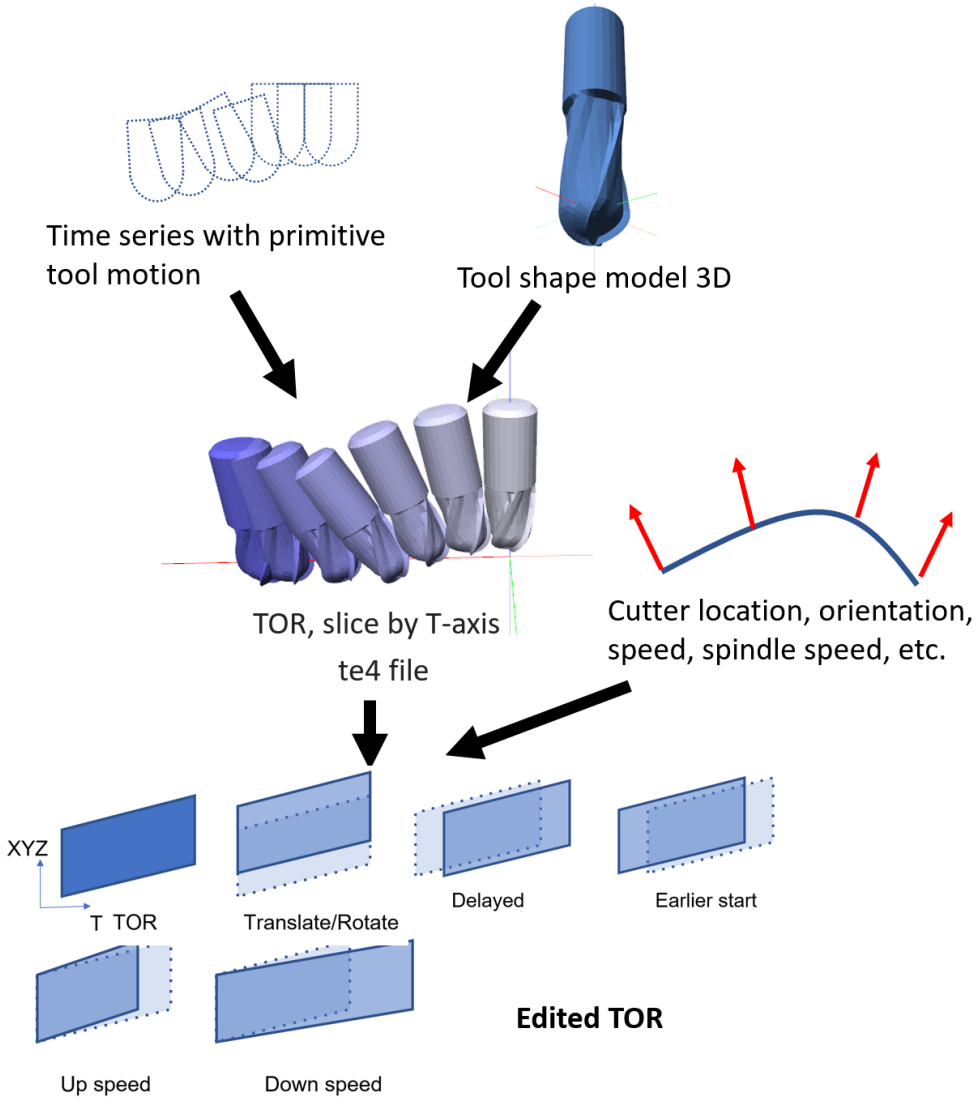
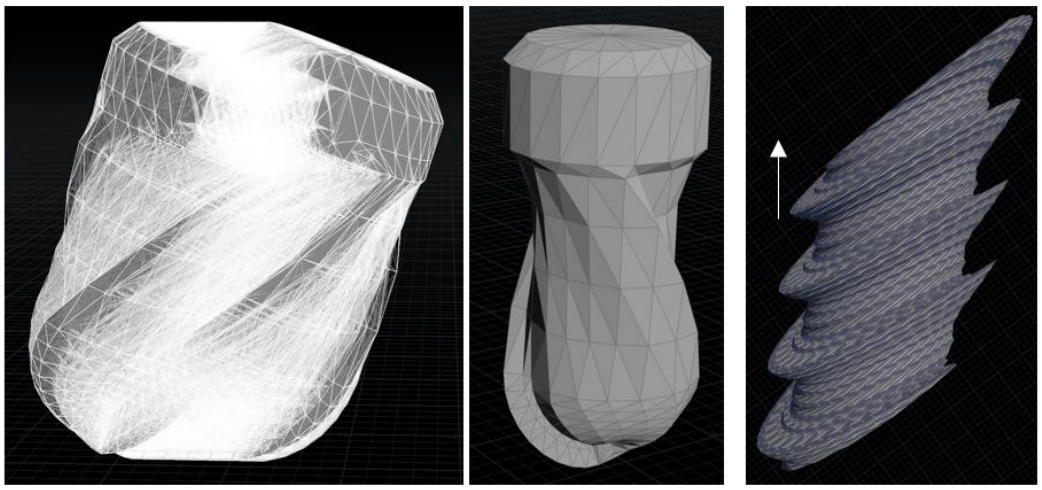


Fig. 3.14 Analogy of TOR constructing process

TOR enable to generate the whole machining process. Figures 5 and 6 illustrate TOR's construction and its diagram in 3D space. Tool shape can be regard as a rigid body in this study. Therefore, TORs can be generated easily by Otomo's direct method. Figure 3.15 is an example of TOR shape.



(a) Projection (b)slice at t=1.0 (c) slice at z = 1.0

Fig. 3.15 A TOR example

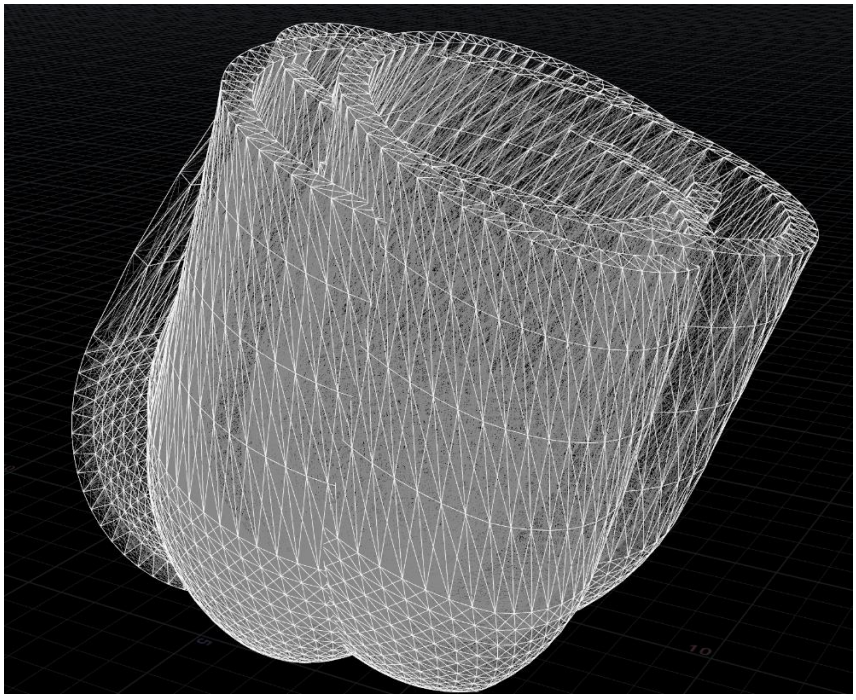


Fig. 3.16 Projection of BSR

3.5.2 Blade occupied regions (BSR)

Balde shaving occupied regions (BSR) is a replacement for TOR.

Comparing to TOR, BSR only contain its part of blade on the tool which is used to remove material. Therefore, BSR have less tetrahedrons than the TOR. As a simplified model it can reduce large computation capacity. Figure 3.16 is a projection of BSR shape.

3.5.3 Workpiece occupied regions (Time-invariant WOR)

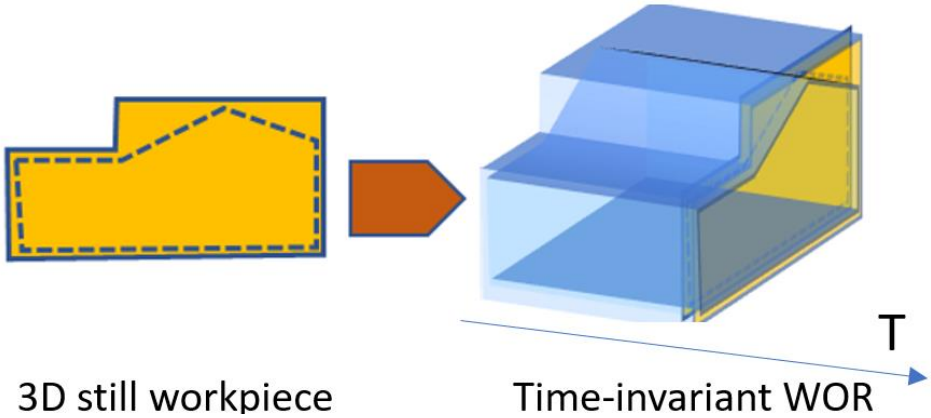


Fig. 3.17 Analogy of WOR constructing process

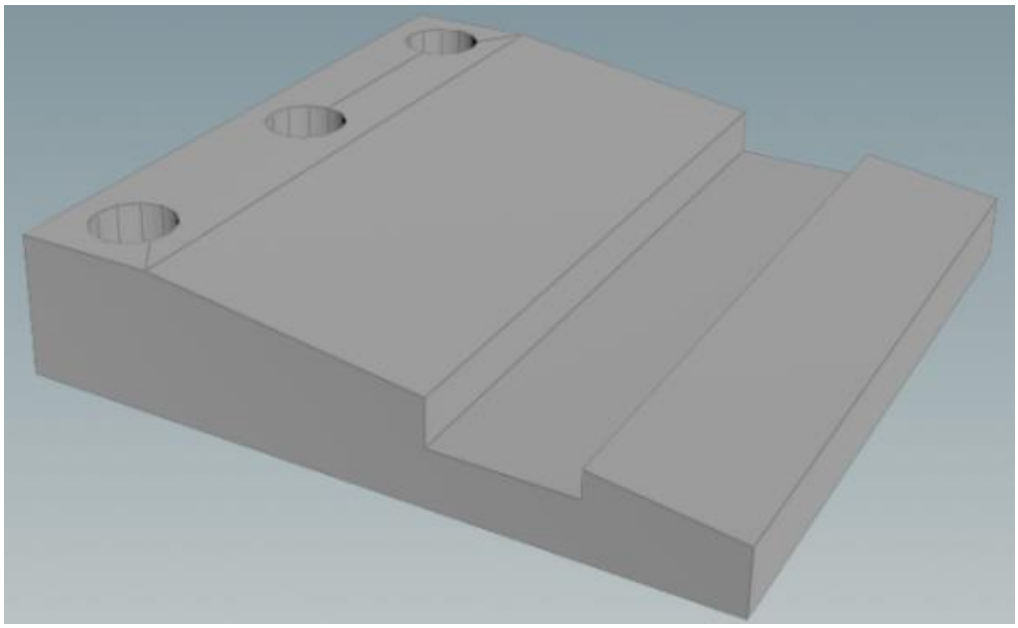


Fig. 3.18 An example of WOR

On other hand, a 3D workpiece model as a raw material and a 3D nominal product model are also prepared. These models will be swept along positive T-axis for representing a still object in 4D space, such as fig.3.17. Since this 4D WOR's 3D slicing along T-axis is not change, it is called as Time- invariant WOR or 3.5D WOR. Figure 3.18 is an example of WOR, its slices and projection should same to the original shape of workpiece.

3.5.4 Accumulated volume (TAV)

Although the TOR integrates the trajectory and shape information of tool in the process, it does not represent the entire machining process. Therefore, TOR is transformed to TAV, which represents the history of tool occupation during the machining process. The comparison of TOR and TAV is shown in Fig. 3.19.
TAV enable to be generated by two methods to generate:

The first approach is a recursive way and from the definition of TAV. The TOR is split by time interval and the volume processed in the previous step is accumulated for current step. Finally, getting the accumulation of all steps.

The second approach is a direct way. The Time-invariant model generate from tool's position of each time interval. Then generate the TAV by compute the union of these Time-invariant model.

$$TAV_{t_{0,1}} = TOR_{t_{0,1}} \cup Vol^{t_{0,1}}_{3D\ tool\ shape} \quad \text{Eq. (3.3)}$$

$$TAV_{t_{0,n}} = TAV_{t_{0,n-1}} \cup TOR_{t_{n-1,n}} \cup Vol^{t_{n-1,n}}_{TAV\ slice} \quad \text{Eq. (3.4)}$$

$$TAV_{t_{0,n}} = TOR_{t_{0,n}} \cup Vol^{t_{0,n}}_{TOR\ slice} \cup \ldots \cup Vol^{t_{n-1,n}}_{TOR\ slice} \quad \text{Eq. (3.5)}$$

$TAV_{t_{0,n}}$: TAV model from t = 0 to t = n.

$TOR_{t_{0,n}}$: TOR model from t = 0 to t = n.

$Vol^{t_{n-1,n}}_{TAV\ slice}$: Time-invariant model, extend from TAV slice at t = n-1 to t= n.

Fig. 3.20 (a) and (b) illustrated both of two approaches, and Eq. (3.3), (3.4) indicate the first approach and Eq. (3.5) indicates the second approach The TAV enable to be represented by the union of existing regions in the figure. Among them, Vol refers to the Time-invariant model of the specified 3D slice over a period of

time.

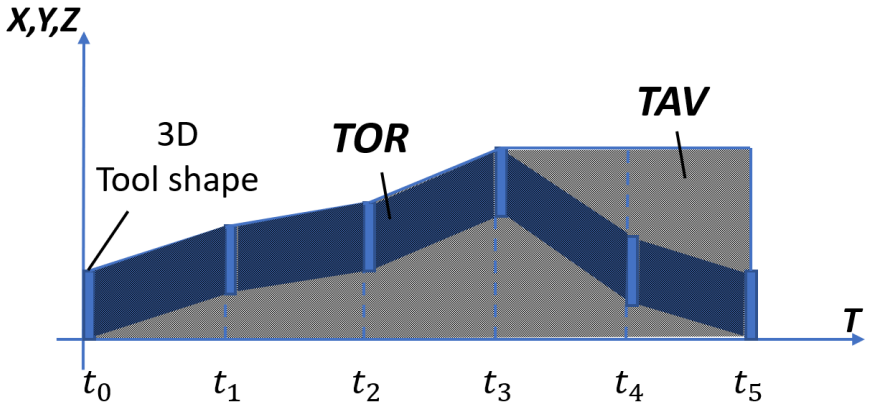
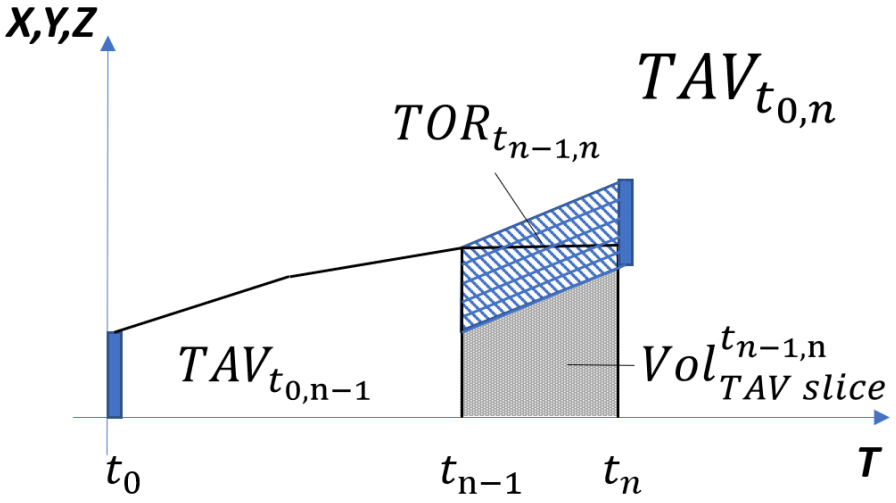
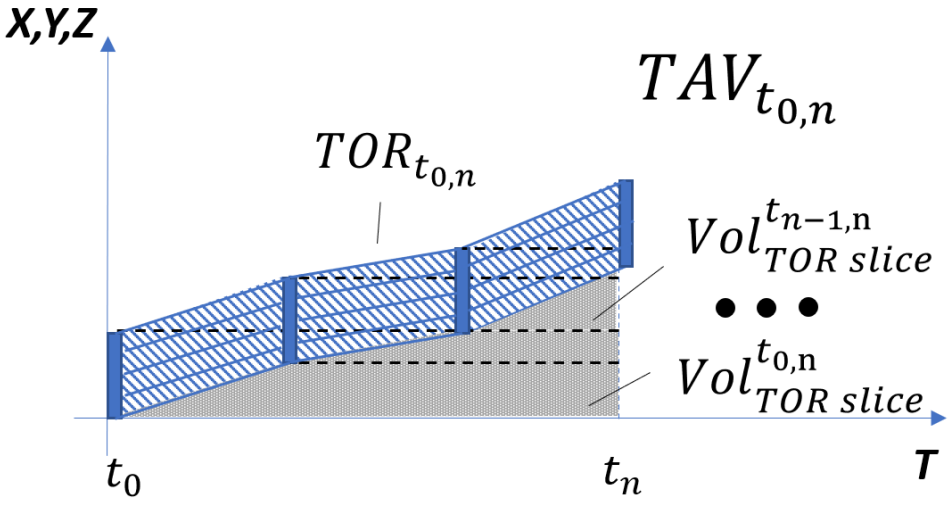


Fig. 3.19 2D analogy of TAV



(a) TAV generation based on recursive method



(b) TAV generation based on direct method

Fig. 3.20 Two methods of generate TAV

Comparing to pervious research by Nakamoto et al [60], the introductions of TOR, WOR and TAV make the preparation for analyzing 4D CWE continuously. Meanwhile, for solving data explosion of 4D meshes, authors plan to introduce GPGPU for accelerating[61,62].

Figure 3.21 is a TAV slices at the end of swept time. This TAV is generated by T-map model and marching cube method which are introduced in Chapter 5.

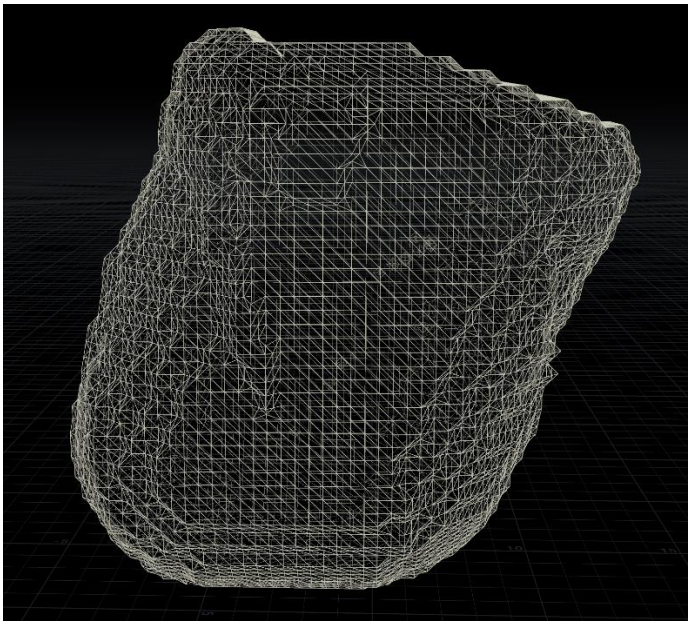


Fig. 3.21 A TAV slice at the last time step generate by marching cube

3.6 potential applications of 4D models in manufacturing industries

Compared to traditional 3D models, 4D models have two significant advantages in representing dynamic processes. First, as a static geometric model, a 4D model integrates all the information of the dynamic representation process of a 3D model, and it even includes more information due to the implicit pre-computation of interpolation. A 4D model is a continuous model from which we can extract all the information that can be extracted or calculated in the dynamic representation process of a 3D model. Moreover, due to the integrity of the 4D model, we can obtain data from various perspectives and aspects, such as XYT, YZT, and other

hyperplanes. As a geometric model, it can also apply geometric operations suitable for it, allowing for the analysis and processing we need through different means. The second advantage is that 4D models are more suitable for processing by AI. Unlike human users, 4D models are closer to being designed for AI processing. Human users, existing in four-dimensional spacetime, cannot easily observe and process 4D models, whereas AI does not have this problem.

This study is a preliminary exploration of 4D models in the processing direction. We hope that this research can play a role in inspiring researchers to apply 4D models in various directions such as manufacturing and even industry. At present, our laboratory believes that 4D models are more suitable for expressing complex combinations of deformation and motion.

Among the many current processing methods, forging is a process that is more suitable for simulation using 4D models. During the forging process, the material mainly deforms, but the topological structure generally does not change. This change process is very suitable to be represented by a continuous 4D model as a whole. The 4D model is continuous in time, and the analysis process of deformation results is more detailed and accurate.

And because the fourth axis of the 4D model can not only represent time, it can also represent a continuous variable, such as temperature, hardness, elasticity, etc. Therefore, the 4D model can also express a series of temperature-related processing processes such as forging and heat treatment without changing the meaning of the fourth axis.

In industrial fields other than manufacturing, we believe that flexible robots with complex motion situations are very suitable for simulation using 4D models. Most flexible robots will undergo displacement and deformation at the same time during operation. A 4D model can include all its forms and movements within a static model.

In the potential application part of our research scheme, the author also listed potential applications of the 4D models analyzed in this study, such as building databases, using deep learning to tool path planning, case analysis and optimization, etc. The basic idea of these potential applications is to use non-human analysis methods such as computers and AI to process 4D data. 4D data is not intuitive enough for human being. Humans can only intuitively experience 3D space. On

other hand, 4D data provides computers and AI with a complete and more macroscopic representation, which is more suitable for them to analyze it. Compared with traditional 3D data sets, 4D data can be analyzed from a more comprehensive perspective, and it is easier to compare data and obtain geometric or variable relationships that are difficult to detect in 3D discrete data sets.

Chapter 4. Common tools for processing 4D models in the research

4.1 Using Houdini to display

4.1.1 Create node for represent 4D mesh model by python

Houdini, a commercial 3D animation software application, was originally developed by SideFX, a Toronto-based company. It was adapted from the PRISMS suite of procedural generation software tools. These procedural tools serve various purposes, including generating intricate reflections, animations, and particle systems. Remarkably, some of its procedural features have been in use since 1987.

Primarily renowned for its proficiency in visual effects creation, Houdini is extensively utilized in the film and television industry. It is the preferred choice for major VFX companies such as Walt Disney Animation Studios, Pixar, DreamWorks Animation, Double Negative, Industrial Light & Magic (ILM), MPC, Framestore, Sony Pictures Imageworks, Scanline VFX, Method Studios, and The Mill.

Houdini has played a significant role in numerous feature animation productions, including Disney's acclaimed films such as Fantasia 2000, Frozen, Zootopia, and Raya and the Last Dragon. Additionally, it has contributed to Blue Sky Studios' Rio and DNA Productions' Ant Bully.

For aspiring artists and enthusiasts, SideFX offers Houdini Apprentice, a free-of-charge limited version of the software tailored for non-commercial use.

SideFX offers the Houdini Object Model (HOM), an application programming interface (API) designed to enable users to access information from and manipulate Houdini using the Python scripting language. HOM serves as a replacement for Houdini's previous command language, HScript.

In Python, the hou package serves as the apex of a hierarchy comprising modules, classes and functions that define the HOM. This hou module is automatically imported when using expressions within the parameter editor and the Python command-line shell. [64].

4.1.2 Nodes list for 4D Display

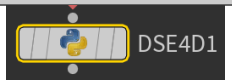
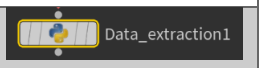
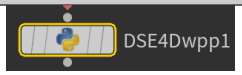
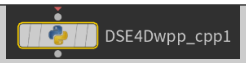
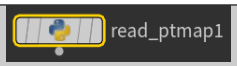
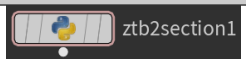
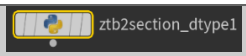
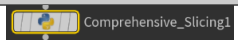
Based on the hou package, the author developed a series of python script nodes to display 4D models. the author lists some of them below, and in Table 4.1:

1. Node “4D mesh model”: This node can read and display input 4D mesh model (.te4 file). It provides projection of tetrahedron mesh or frame wire model. It also can display the original 3D model for Otomo's model.
2. Node “Simple subResult te4”: This node can read and display input 4D mesh model(.te4 file). Especially the subdivided result from set operation process is very suitable for the node. The user can specify an arbitrary index range of tetrahedrons to be displayed.
3. Node “DSEslicer”: This node is used to cutting and displaying the hyperplane slice of the 4D model . It can connect with input Node "4D mesh model" or "Simple subResult te4" and so on .te4 reading node. It has two parameter "slice position" and "cutting axis".
4. Node “Extraction”: Because of node “DSEslicer” is too slow to achieve the result. the author divide it to two nodes. One is “DSEslicer”, extract required data to Houdini shared class. That make data only read 1 time which is not like “DSEslicer” read at every update.
5. Node “4D slicer with preliminary process”: Another is this node. It connects to Extraction. Others same to DSEslicer.
6. Node “4D slicer with preliminary process_CPP”: C++ edition of "4D slicer with preliminary process". For reducing process time.
7. Node “read Tmap”: Read Tmap(voxel arrange uniform in an Axis-Aligned Bounding Box (AABB)), and display remain points cloud before input time value.
8. Node “read ptmap”: Read Tmap (a points cloud in any shape), and display remain points cloud before input time value.
9. Node “Diff HP tmap”: Read Tmap(voxel arrange uniform in an AABB), and display remain points cloud before input axis value.
10. Node “AC Tmap”: Read Tmap (voxel arrange uniform in an AABB), and display inverse points cloud before input axis value. Can be regard as a local TAV
11. Node “ztb2section”: Read .ztb file(tetrahedron boundary intersection data),

and display the intersection tetrahedron. Parameters include .ztb file, Model1.te4 and Model2.te4

12. Node “ztb2section dtype1”: Read .ztb file(tetrahedron boundary intersection data), and display the intersection tetrahedron. Parameters include, .ztb file, Model1.te4 and Model2.te4. te4 file “dtype” is 1
13. Node “Comprehensive Slicing”: Compute set operation in Houdini and display by hyperplane slicing. Parameters include, Slice position, slice axis and set operation type.

Table 4.1 Node list

Node name	Parameters	Input	Output	Display	Icon
4D mesh model	File path, Display method	none	4D mesh model data	projection or original model	
Simple subResult te4	File path, Display range	none	4D mesh model data	Projection of tetrahedrons	
DSEslicer	Cutting axis, cutting position	4D mesh model data	3D slice data	Slice result	
Extraction	None	4D mesh model data	Houdini shared data	None	
4D slicer with preliminary process	Cutting axis, cutting position	Houdini shared data	3D slice data	Slice result	
4D slicer with preliminary process_CPP	Cutting axis, cutting position	Houdini shared data	3D slice data	Slice result	
read Tmap	File path, Time value	none	T-map data	Points cloud with color	
read ptmap	File path, Time value	none	T-map data	Points cloud with color	
Diff HP tmap	File path, hyperplane axis, axis value	none	T-map data	Points cloud with color	
AC Tmap	File path, Time value	none	T-map data	Points cloud with color	
ztb2section	File path of model 1, File path of model 2, File path of file ztb	none	Two 4D mesh model data and ztb data	Projection of Intersection part	
ztb2section dtype1	File path of model 1, File path of model 2, File path of file ztb	none	Two 4D mesh model data and ztb data	Projection of Intersection part	
Comprehensive Slicing	Set operation type, cutting process, cutting position	Two 4D mesh model data and ztb data	3D slice data	Slice result after set operation	

4.2 Parallel processing technology based on GPGPU

4.2.1 Structure of GPGPU

General-purpose computing on graphics processing units (GPGPU, or less commonly GPGP) refers to the utilization of a graphics processing unit (GPU), which traditionally handles computation solely for graphics, to perform computations typically handled by the central processing unit (CPU). Using multiple video cards in one computer or a large number of graphics chips, the inherently parallel nature of graphics processing can be enhanced even further.

Fundamentally, a GPGPU pipeline involves parallel processing across one or more GPUs and CPUs, treating data as if it were in the form of images or graphics. Although GPUs operate at lower frequencies, they usually have a considerably higher number of cores, allowing them to handle a larger volume of images and graphical data per second than conventional CPUs. Transforming data into graphical form and leveraging the GPU for scanning and analysis can significantly boost processing speed.

Initially conceived in the early 21st century for graphics processing, specifically to enhance shaders, GPGPU pipelines were discovered to be well-suited for scientific computing needs. This realization spurred further advancements in this domain.

Originally, data moved in a single direction from the central processing unit (CPU) to the graphics processing unit (GPU), and finally to a display device. However, as time advanced, it proved advantageous for GPUs to hold more complex data structures, which could then be sent back to the CPU for further analysis. This data could represent images or scientific data in 2D or 3D formats comprehensible to a video card. The GPU's capability to access each draw operation allows for quick data analysis, while a CPU must process every pixel or data element more slowly due to the slower access speeds between the CPU and its larger random-access memory pool (or, in the worst case, a hard drive). By moving the actively analyzed portion of the dataset into GPU memory in formats easily readable by the GPU, such as textures, the processing speed is enhanced. The defining characteristic of a GPGPU design is its ability to transfer information bidirectionally between the GPU and the CPU. Ideally, high data throughput in both directions creates a multiplier

effect on the speed of specific high-use algorithms. GPGPU pipelines have the potential to improve efficiency, particularly when dealing with large datasets or data containing 2D or 3D imagery.

CPUs are tailored to execute a small number of more complex tasks, while GPUs are designed to handle a large number of simpler tasks.

While GPGPU is primarily a software concept rather than a hardware concept, it denotes a specific algorithmic approach rather than a physical piece of equipment. Nevertheless, specialized hardware designs can further optimize the efficiency of GPGPU pipelines, which typically execute a relatively small set of algorithms on extensive datasets. Tasks dealing with massively parallelized, vast datasets can undergo further parallelization through specialized configurations like rack computing.

Rack computing involves the integration of numerous similar, highly tailored machines into a rack configuration. This setup adds another layer of parallelism, with each computing unit employing multiple CPUs to correspond to multiple GPUs. This approach allows for even greater parallelization and scalability, particularly for

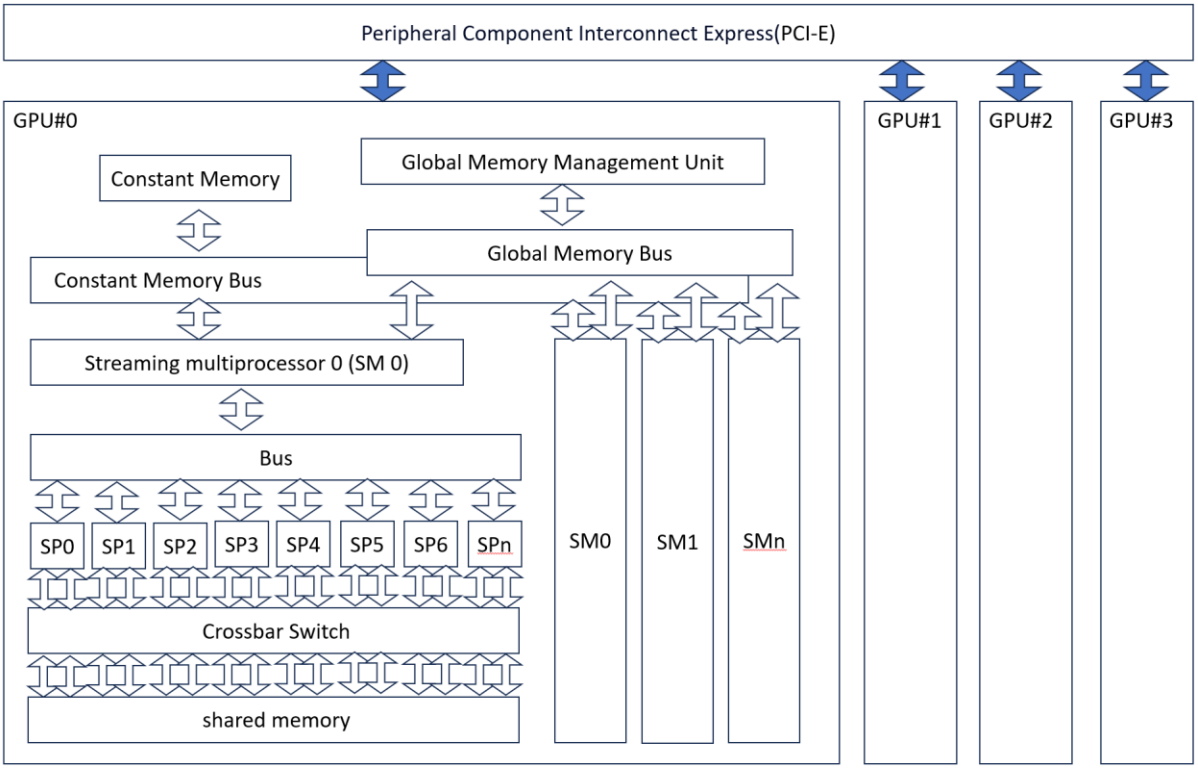


Fig. 4.1 A standard GPU structure

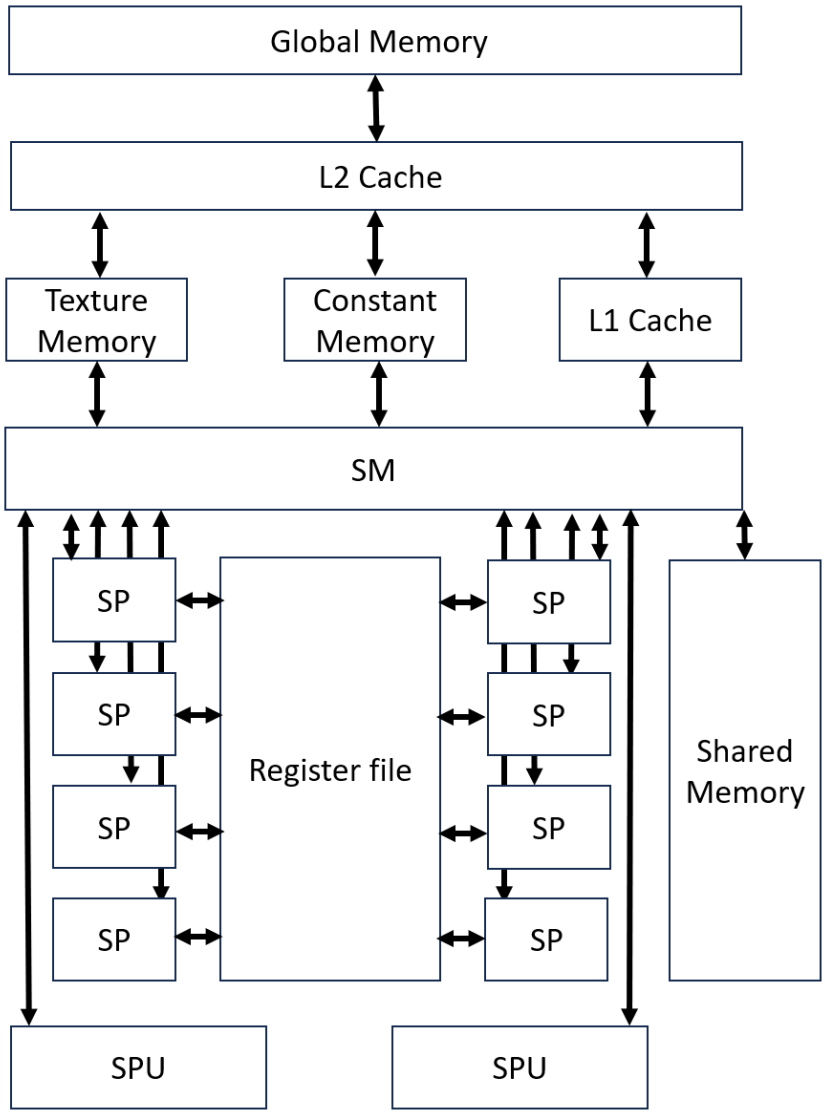


Fig. 4.2 A standard SM structure

tasks requiring extensive computational resources and handling massive datasets[65].

A common GPU hardware is consisting of three main modules, Memory, Streaming multiprocessor (SM) and Streaming processor (SP). The GPU can be regarded as a SM array which illustrate in fig. 4.1. In a typical GPU architecture, as illustrated in fig. 4.1, there are numerous Streaming Multiprocessors (SM), with each SM containing multiple Streaming Processors (SP). Similar to arithmetic and logic units (ALUs), SPs can execute computing tasks individually and concurrently. This architecture is well-suited for intensive and parallel data processing, as depicted in fig. 4.2.

The construction of a GPU primarily involves assembling multiple Streaming Multiprocessors (SMs). Each SM comprises various key components. Typically, it consists of multiples of 8 Streaming Processors (SPs), each linked to a shared register file. This register file, operating at the same speed as the SPs, serves as a memory unit for storing and managing the active registers running on the SPs. Additionally, there exists shared memory within the SM, accessible exclusively by the SM internally. This shared memory acts as a "program-controllable high-level cache" and can be fully manipulated by the programmer. Furthermore, each SM incorporates two or more Special-Purpose Units (SPUs), which specialize in executing specific hardware instructions, notably for high-speed 24-bit sine/cosine/exponential functions. [67].

4.2.2 Structure of CUDA

Compute Unified Device Architecture (CUDA) is a proprietary parallel computing platform and application programming interface (API) that enables software to utilize specific types of GPUs for accelerated general-purpose processing, a concept known as General-Purpose computing on Graphics Processing Units (GPGPU). The CUDA API, along with its runtime, extends the capabilities of the C programming language, allowing for the specification of thread-level parallelism and GPU device-specific operations, such as data movement between the CPU and GPU. CUDA acts as a software layer that grants direct access to the GPU's virtual instruction set and parallel computational elements, enabling the execution of compute kernels. Alongside drivers and runtime kernels, the CUDA platform encompasses compilers, libraries, and developer tools aimed at assisting programmers in accelerating their applications by leveraging the computational power of GPUs. [68].

In the official website of CUDA, the CUDA Toolkit is introduced as below: “The NVIDIA CUDA Toolkit provides a development environment for creating high-performance, GPU-accelerated applications. With it, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms, and supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library [69]”.

CUDA's hardware scheduling uses an interesting and novel model called “Single Program, Multiple Data (SPMD)”. In some ways, this scheduling choice is based on the underlying hardware architecture of the GPU seniority in our previous section 4.2.1. At the core of such parallel programming is a concept called “thread”. The software level thread corresponds to the hardware level SP. Corresponding to the GPU structure, CUDA can start a large number of threads concurrently and execute same instruction with different multiple data respectively under an organized thread network. This architecture is called single instruction multiple thread (SIMT). This parallel process method can also be called as Sing Instruction, Multiple Data (SIMD).

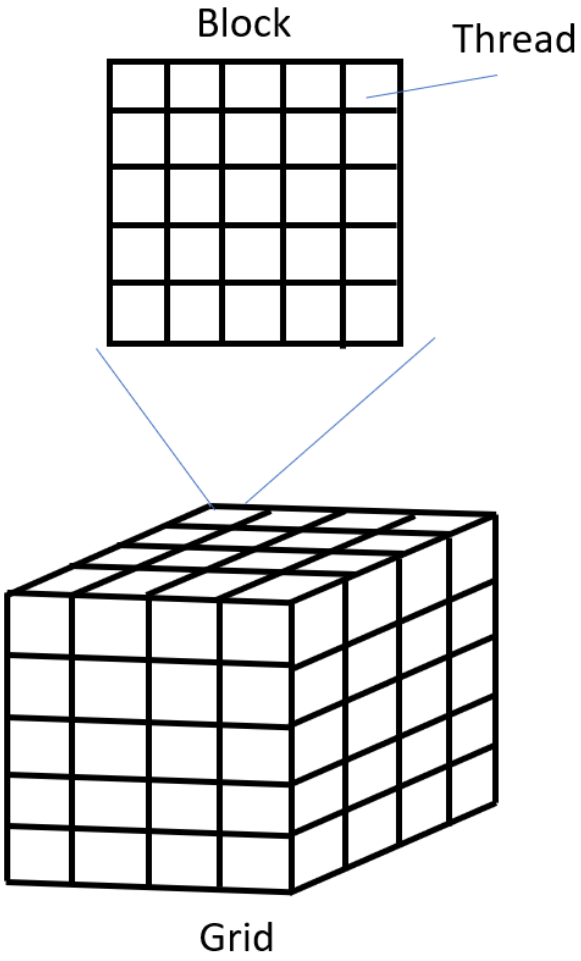


Fig. 4.3 CUDA threads structure

In general, many threads are organized as thread blocks in CUDA. thread blocks are generally a 1D or 2D structure. the number of threads in a thread block is based

on the GPU, e.g., for a GTX 1070, 1024 threads can exist in each thread block. and many Thread blocks can be organized into thread grids. Thread grids can be organized into a structure that is 3D and below. The upper limit of thread blocks in a Thread grid typically equal to 65535×65535[70]. This structure can be regarded as shape in fig. 4.3.

4.2.3 Embarrassingly parallel

In parallel computing, an embarrassingly parallel workload or problem, also referred to as embarrassingly parallelizable, perfectly parallel, delightfully parallel, or pleasingly parallel, demonstrates a minimal need for effort in dividing into multiple parallel tasks. This ease of parallelization stems from the absence or minimal dependency on communication between these tasks or their outcomes.

These problems contrast with distributed computing challenges, which necessitate communication between tasks, particularly the exchange of intermediate results. Embarrassingly parallel tasks are more readily executed on server farms that lack the specialized infrastructure found in true supercomputer clusters. They are well-suited for large-scale, internet-based volunteer computing platforms. Conversely, inherently serial problems, which cannot be parallelized, represent the antithesis of embarrassingly parallel problems.

A classic instance of an embarrassingly parallel problem is 3D video rendering conducted by a graphics processing unit (GPU), where each frame (using the forward method) or pixel (employing the ray tracing method) can be processed independently without interdependence. Certain forms of password cracking also illustrate embarrassingly parallel tasks, which can be effortlessly distributed across central processing units (CPUs), CPU cores, or clusters [67].

4.2.4 Dynamic parallel

CUDA dynamic parallelism expands the CUDA programming model by enabling kernels to invoke other kernels. This capability allows each thread to dynamically uncover work and initiate new grids based on the newly discovered workload. Additionally, it facilitates dynamic allocation of device memory by threads. This structure is like fig. 4.4.

While dynamic parallelism offers numerous benefits in terms of improved workload balance and programmability, it's crucial to consider the potential drawback of underutilizing GPU resources when launching grids with a small number of threads. To mitigate this issue, a general recommendation is to launch child grids with a significant number of blocks, or at the very least, blocks with a substantial number of threads if the total number of blocks is small. This approach helps maximize GPU resource utilization and enhances overall performance[71].

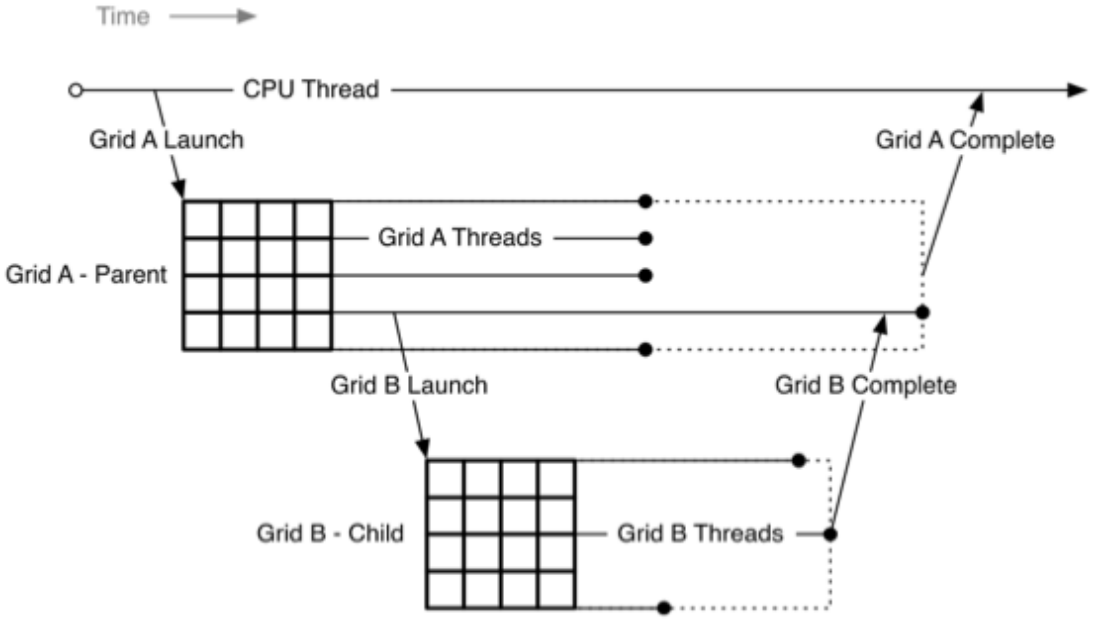


Fig. 4.4 Dynamic parallel

4.2.5 Implementation of parallel processing for accelerating 4D model operations

When we talk about the shortcomings of 4D models, its huge data for processing is a big problem. For example, when we use Otomo's method to generate a 4D model of a moving object over a period of time. Each time step will generate 14 tetrahedrons from two polygons. Excluding the shared part, it will generate 30 more polygons and one extra vertex. Although there is only one more vertex data, all 14 tetrahedrons are considered during the processing. And the entire movement usually consists of dozens of time steps.

At the algorithm level, there is still no better optimization solution that can reduce the amount of computation while maintaining good accuracy. Therefore, the

author tried to introduce parallel processing to speed up the calculation process. In the research, the author mainly consider some commonly used graphical representation models such as mesh model and voxel model. Each of their elements are discrete from each other, such as different tetrahedrons or voxels. Generally these elements do not affect each other. By designing parallelism in data structures and algorithms, we can easily transform the required processing algorithm strategies into embarrassingly parallelism or dynamic parallelism[72,73,74].

Chapter 5. Local continuous methods of CWE analysis

5.1 Algorithm outline of T-map method

In the author's doctor period, the research direction shifted from the application of 3D graphics in the machining process to the application of 4D graphics in the machining process. The T-map method is considered as the first project to solve the 4D CWE representation. It is a method that extends the traditional Z-map or one-direction Dexel method. It is very easy to understand and master for people who are familiar with 3D graphics algorithms.

Different to record the height of Z-map, T-map method records time. A tradition Z-map model is based on 2D grids which represent the workpiece bottom on XY-plane. Z-map segments extend from grids along Z-axis to the top surface of workpiece. This Z-map segments set can represent the workpiece shape and cut off when cutter model sweep over them. Z-map algorithm records the height of model (length of Z-map segments) to represent the machining process. Similar to Z-map, T-map is based on XYZ-hyperplane grids (voxel) and extend T-map segments along T-axis. When T-map segments swept over by TOR, T-map method records the length of T-map segments. This length represents the time period that the 3D voxel exists on the workpiece model and the author called it "first touch time".

Therefore, the first touch analysis is the core of T-map algorithms. In addition to the core, some components have been added to complete the method. The completed method is illustrated in Fig.5.1.

Two models are planned to input to the T-map system. TOR represents the 4D trajectory of cutter. A 3D voxel workpiece model is the base of T-map segments. After the first touch analysis, the result of T-map structure is achieved. Although it can be display by points cloud, a 4D representation is also required to make it standardized and prepare other 4D process.

I introduced 4D marching cube to generate a tetrahedralization of T-map structures. The result is 4D meshes of workpiece changing process. This process is

a 4D mesh model that can represent CWE[75].

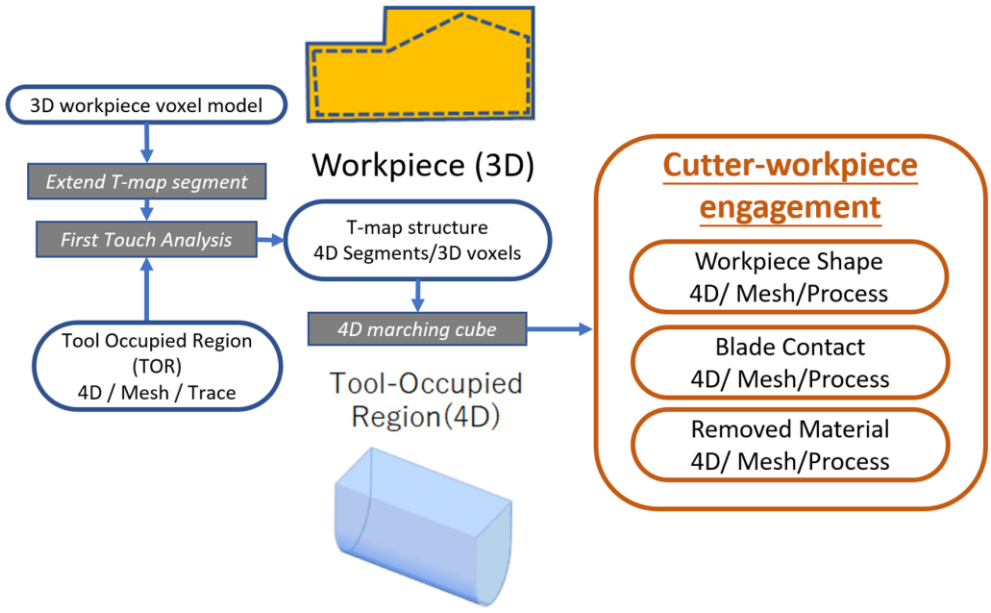


Fig. 5.1 Outline of T-map method

5.2 Z-map structure and T-map structure

5.2.1 Traditional Z-map structure

The Dexel model is a discrete representation of solid shapes. The Dexel model is always composed of a set of line segments perpendicular to the XY, YZ or XZ plane, see fig. 5.2. According to the corresponding number of directions, Dexel model can be classified as triple directions Dexel model and single-direction Dexel model. Compared with the single direction Dexel model, the triple directions Dexel model could express a more precise shape. For most of milling process simulation, single Dexel model is enough and much faster than triple directions Dexel [76]. Fig. 5.3 illustrates the triple-direction Dexel model.

To convert a polygon model into a Dexel model, consider orthogonal line segments on a plane from each point of intersection. A ray extends until surfaces of a model as a segment. The higher density of these segments, the higher the accuracy of Dexel model. Figure 5.4 illustrates the conversion from a polygonal model into a Dexel model.

Z-map model is introduced as the workpiece representation for the cutting simulation. A Z-map model which can be considered as a special case of Single

direction Dexel model is defined on a two-dimensional regular square grids on the XY-plane. From each grid point, an upward segment along the Z axis extends until it reaches the top surface of workpiece, see fig.5.5 upper. The heights of segment are recorded. This operation converts the cylindrical blank shape to a set of vertical segments for the cutting simulation.

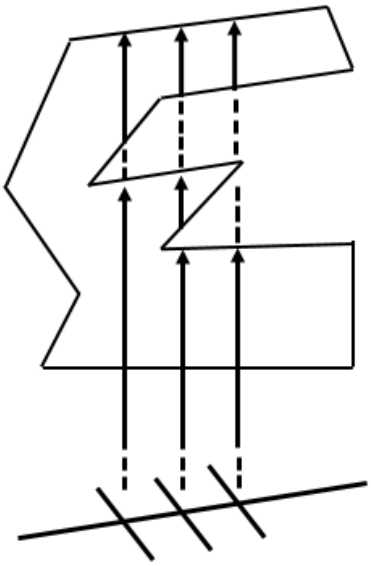


Fig. 5.2 Definition of Dexel model

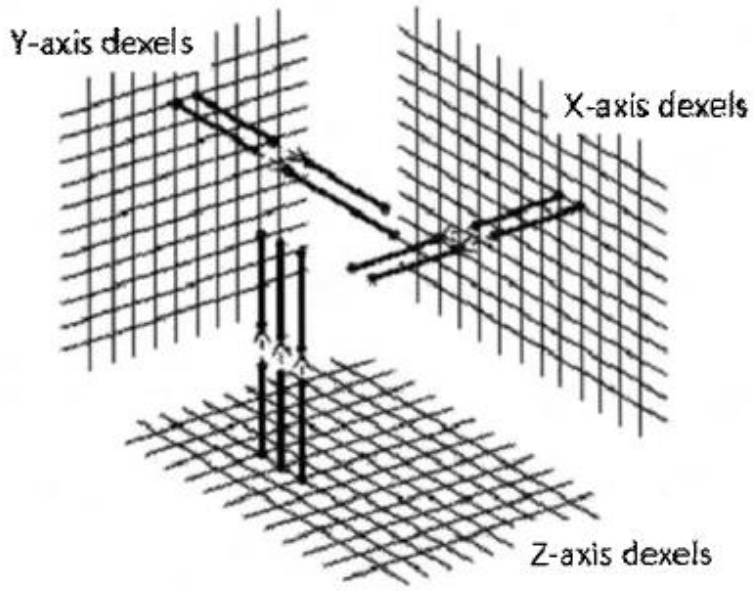


Fig. 5.3 A triple directions Dexel model

A 3-axis milling operation can be geometrically simulated by subtracting the cutter swept volume from the vertical segments of Z-map, see fig5.5 below. This subtraction is equivalent to an update of the top height of the Z-map segment if cutter swept volume passes over it and the bottom surface of the volume cuts the segment lower than the current height [77].

After the simulation, the remaining material on the target shape is estimated by computing the difference between height of segments of the Z-map model of the milling simulation result. Polygons in the target shape are regarded as machined if the height difference of Z-map segments at the polygon is sufficiently small [78]. Then the area of the machined polygons is accumulated. The cutting direction with the maximum machined area in the target shape is selected as the optimal cutting direction.

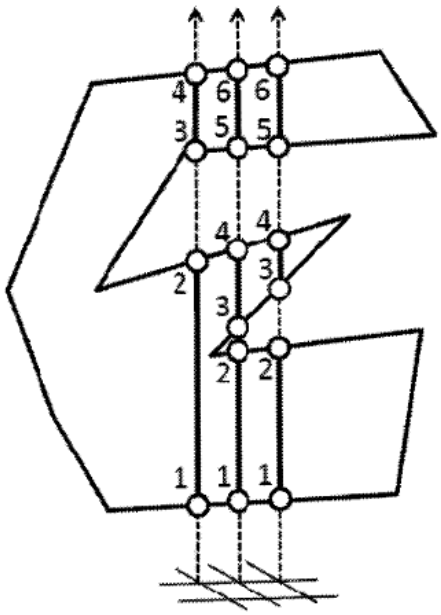


Fig. 5.4 Conversion from a polygonal model into a Dexel model

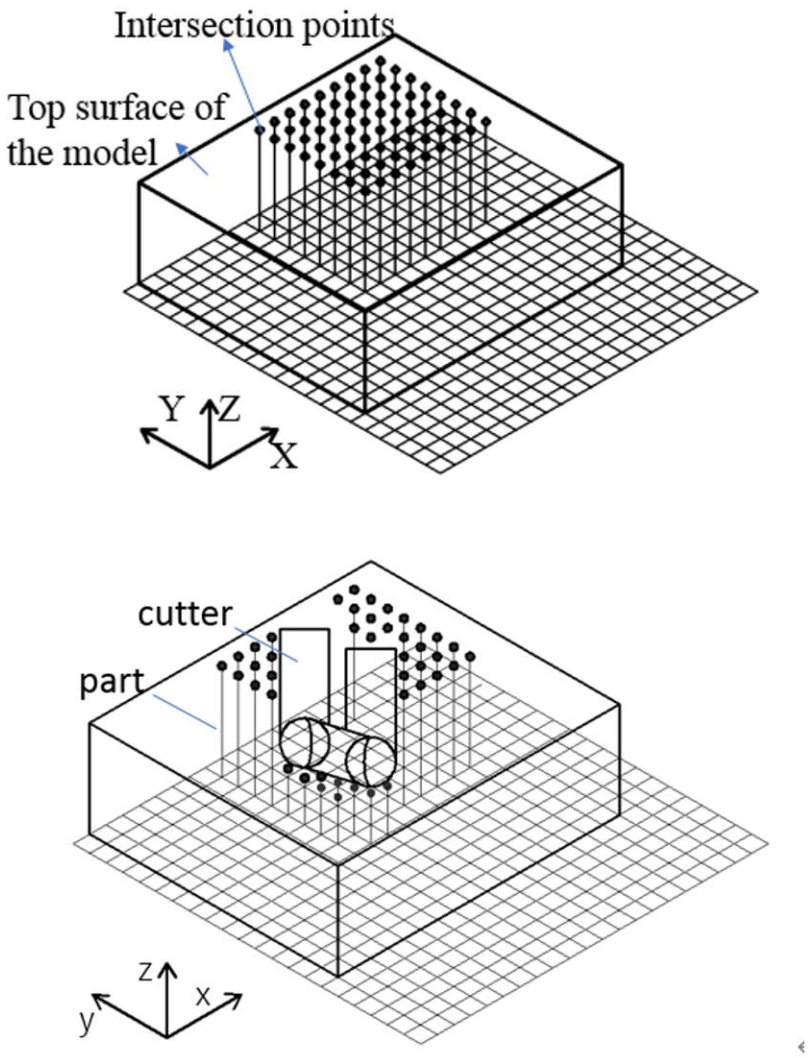


Fig. 5.5 A 3D analogy of Z-map and workpiece (upper), Z-map process of this example (below)

5.2.2 T-map structure

In 3D processing, Z-map and depth buffer are usually use for CWE 14). As the concept of 4D is an extension of 3D, it is a natural and effective idea to extend classic 3D algorithms to 4D space. The T-map algorithm mimics the Z-map algorithm and extends it conceptually along the T-coordinate axis. Z-map uses a 2D grid and emits rays along the Z-axis direction to represent the height of an object,. T-map, on the other hand, uses a 3D voxel model and emits rays along the T-axis,

representing the time that corresponding voxels exist in space, a microscopic representation is shown in Fig. 5.6.

Like Fig. 5.7, when the T-map ray intersects with the TOR in the spatio-temporal space, truncate T-map ray to T-map segment and calculate the length of the T-map segment. This length is referred to as the "first touch time," which represents the time when the material at the corresponding voxel position is removed. This process and structure are shown in Eq. (5.1) and Eq. (5.2).

$$(i,j,k) \rightarrow (x,y,z) \quad i,j,k \in Voxel(WP) \ \mathbb{R}^3 \qquad \text{Eq. (5.1)}$$

$$t = min\{t \mid t \in T(x,y,z) \cap B(TOR)\} \quad \mathbb{R}^4 \qquad \text{Eq. (5.2)}$$

$Voxel(WP)$*:3D Voxel of workpiece, i, j, k are index of voxel*

$T(x,y,z)$*: a T-map segment in* $\mathbb{R}^4$

$B(TOR)$*: Boundary of TOR*

Eq. (3) shows T-map segments extent from 3D voxel and compute the coordinate based on index of voxel. And Eq. (4) is a process of getting first touch t value.

This T-map structure is suitable for parallel computing to prevent extremely long computing time cause by the data explosion in the 4D space.

The basic steps of construction the T-map structure is represented below:

Input: The Tool-Occupied Regions (TOR) and certain T-map parameters related to the workpiece model, such as grid size, quantity, shape, etc.

Step 1 involves a preliminary process aimed at comparing the projection onto the XYZ hyperplane of a TOR with the range of T-map voxels. This process serves to preliminarily filter out tetrahedrons that fall outside the T-map range.

Step 2: Select a tetrahedron and retrieve all voxels within the range of the projection onto the XYZ hyperplane of the selected tetrahedron.

Step 3: Choose a voxel within the range and determine whether its T-map segment intersects with the selected tetrahedron. If an intersection occurs, record the time-value (T-value) of the intersection point on this selected grid. If another T-value is already stored in this grid, update it with the smaller value. Repeat Step 3 until all voxels within the range of the tetrahedron's projection are computed.

Step 4: If there are tetrahedrons that have not been computed, return to Step 2. If all tetrahedrons within the T-map range have been computed, output the result of

the T-map process.

Output: Provide the first-touch time of each T-map grid for CWE.

After these iterative processing steps of the entire T-map algorithm, the T-map structure can be constructed. This T-map structure is composed of 3D voxel model of the workpiece and T-map segments, which is shown in Fig. 5.8. Therefore, it is discrete in 3D space and continuous along T-axis. It can directly represent the CWE process or be converted into a mesh model using the 4D Marching Cube algorithm 16) to obtain surface shape.

Clearly, the T-map algorithm consists of two nested loop bodies. The primary focus of these loops is the determination of intersections. The flowchart depicting these two loop bodies is illustrated in Fig. 5.9. Similar to the scenario in 3D space, determining the intersection between a segment and a tetrahedron in 4D space can be simplified by assessing whether the intersection point lies within the tetrahedron.

In 3D space, there exists a straightforward method for determining whether a point lies within a triangular polygon or not. Initially, connect the point with the vertices of the polygon. Then, compute and compare the area of the polygon with the sum of the areas of the newly formed triangles resulting from these connections. If the sum is greater, the point lies outside the polygon. In 4D space, this comparison process can be extended to the volume between the tetrahedron and the sum of the volumes of the newly formed smaller tetrahedrons.

For easy understanding, I make a 2D analogy of T-map in Fig. 5.10.

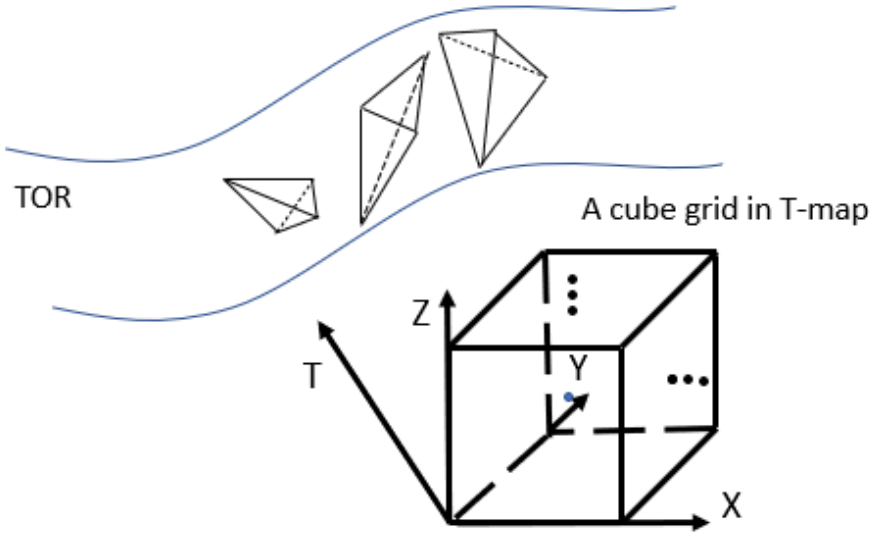


Fig. 5.6 Relation between voxel and TOR's tetrahedron

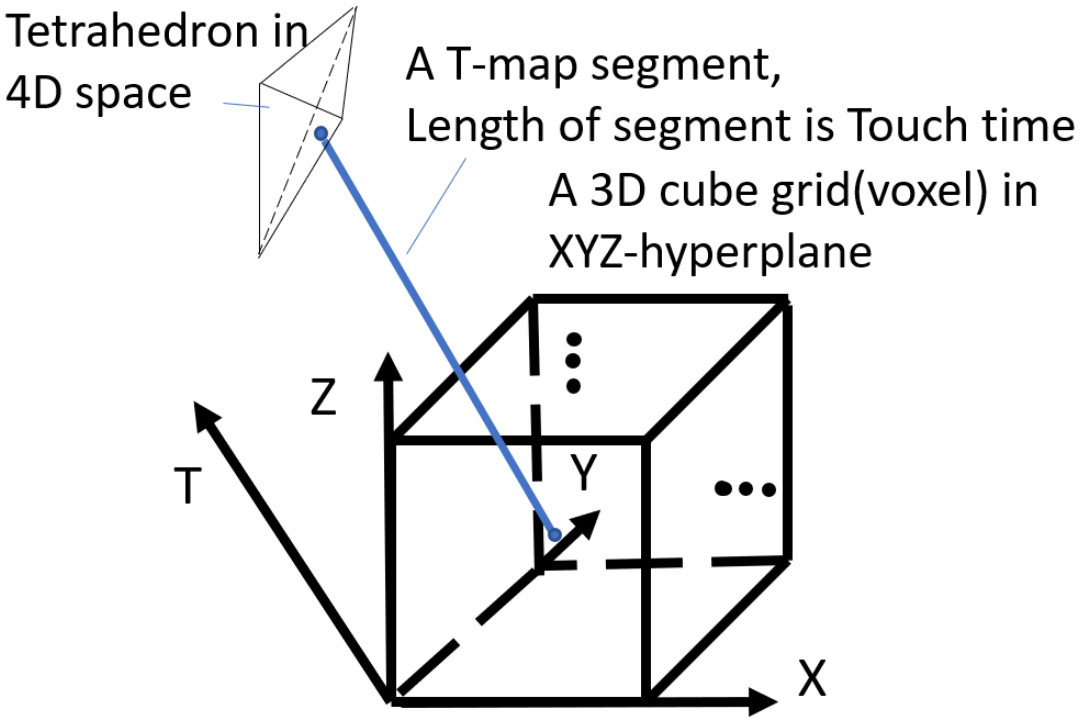


Fig. 5.7 A T-map ray emits from voxel.

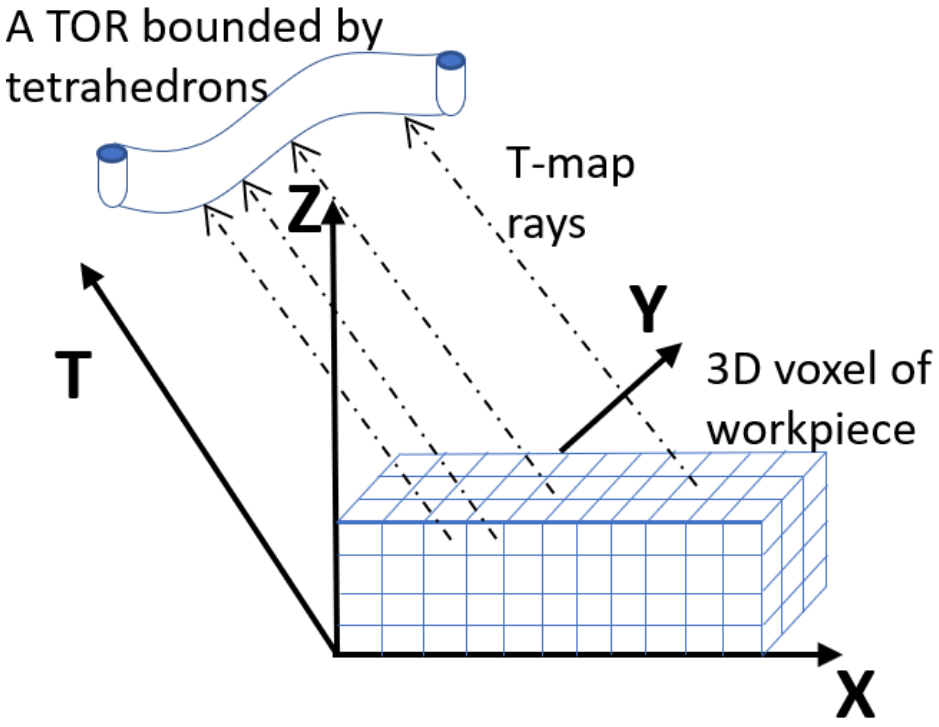


Fig. 5.8 T-map method analyze CWE

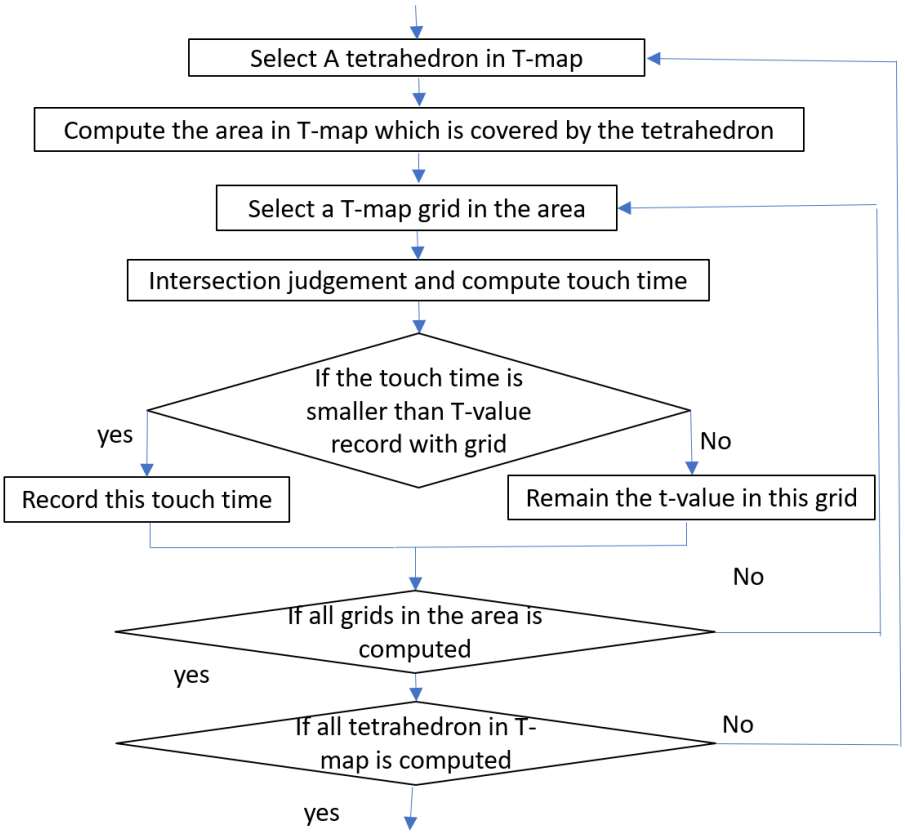


Fig. 5.9 Flowchart of T-map structure constructing

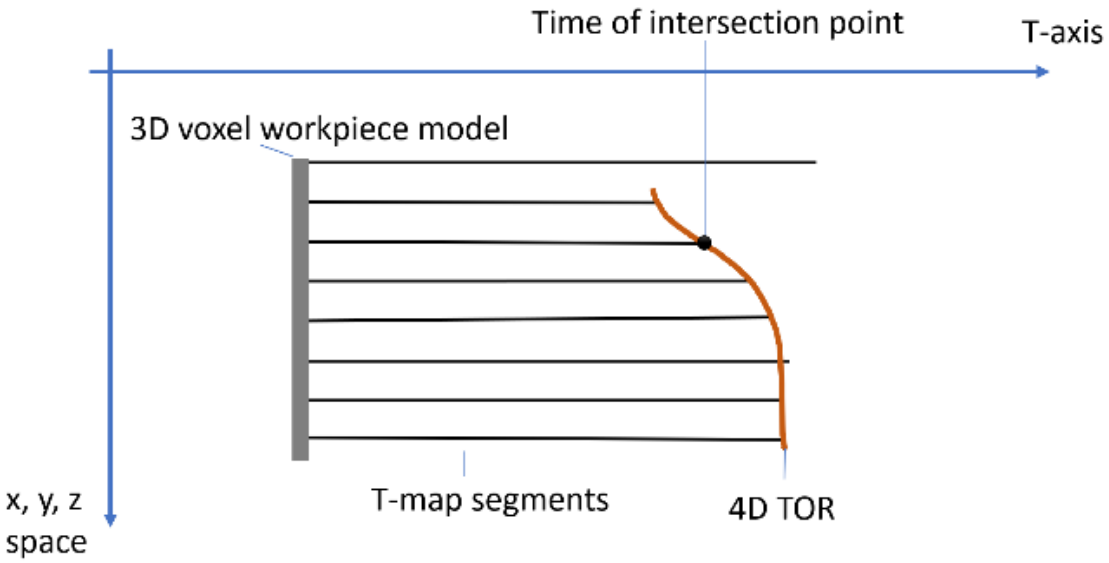


Fig. 5.10 2D analogy of T-map

5.3 Parallel strategy for T-map process

5.3.1 Embarrassingly parallel strategy

It is a comprehensible method in serial processing for 4D geometric model and it require a large number of iteration loop and inevitably takes a lot of processing time. To solve the T-map intersection judgement, the algorithm can be modified as follow.

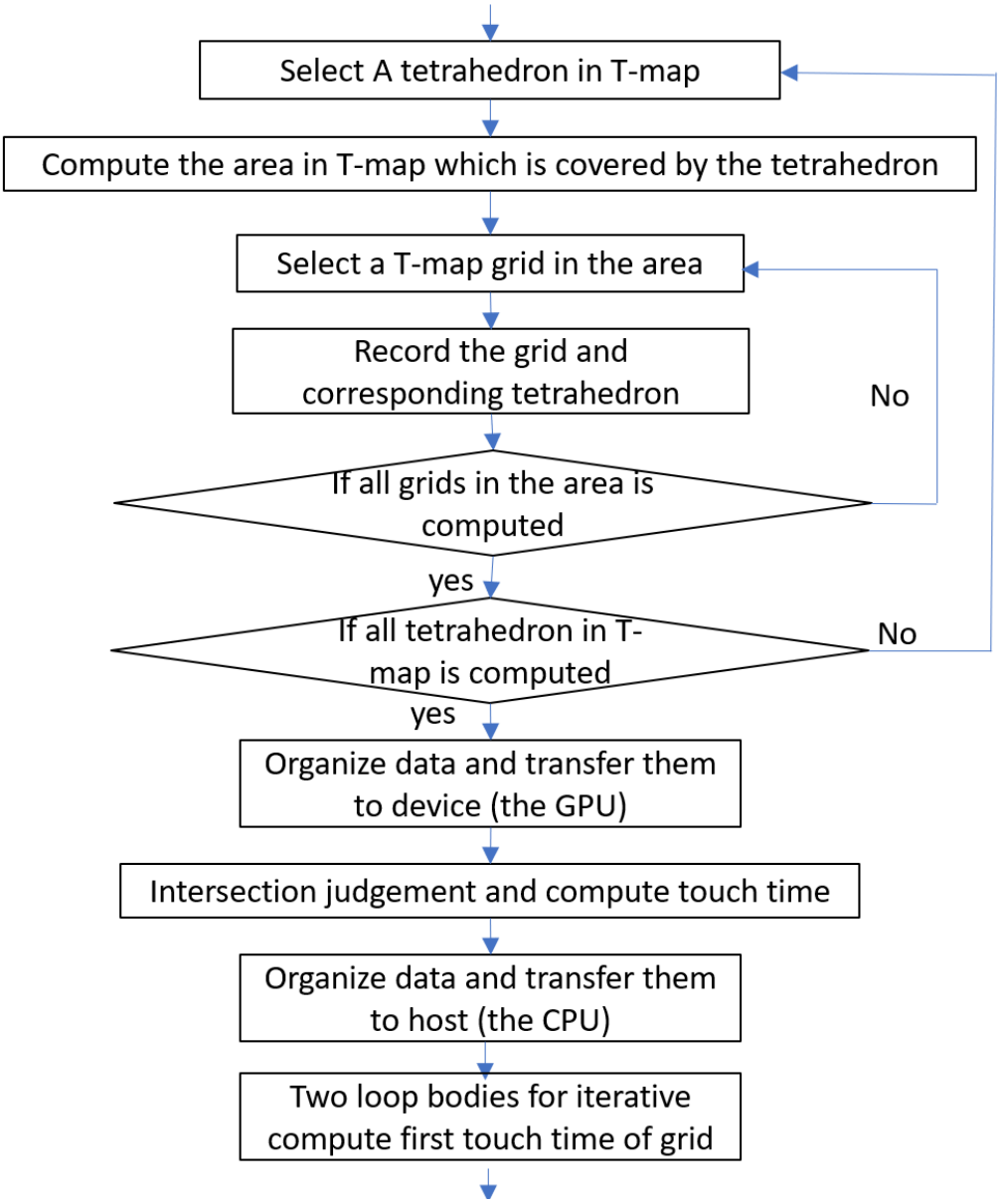


Fig. 5.11 T-map constructing flow with embarrassingly parallel processing

…

Step 2: Compute all covered cube grids of each tetrahedron that pass the preliminary process. Record these grids along with the corresponding tetrahedron index.

Step 3: Transfer data from host memory to the GPU's memory, then configure the thread network and initiate the kernel function. Conduct parallel processing for the computation of tetrahedrons' volumes, comparison, and determination of T-values. Sort the results and transfer the data back from the GPU's memory.

Step 4: Execute the two nested loop bodies and compare the returned results of each cube grid. Record the minimum T-value of a grid as its first-touch time.

…

This strategy follows an embarrassingly parallel approach [13], as depicted in the flowchart shown in fig. 5.11. Each cube grid covered by a tetrahedron is computed separately with the current tetrahedron. In essence, every thread calculates and records the intersection condition of a pair of tetrahedron and T-map segment. It's worth noting that the length of the line segment of the same grid may be recorded multiple times. Hence, it becomes necessary to record the minimum value of each T-map segment through an iterative method in step 4.

The method is straightforward to understand and implement into a program. It eliminates the need for multiple data transfers and starting the kernel function multiple times, making the algorithm stable and robust. However, a drawback is that it requires significant data preparation, occupying both host memory and device memory in the initial stage. Moreover, it also involves processing the output data serially thereafter. This may result in a certain loss of calculation speed or even trigger out-of-memory errors.

5.3.2 Dynamic parallel with atomic operation

Another parallel strategy involves designing the T-map process based on dynamic parallelism [12].

…

Step 2: Organize and transfer data to the GPU memory and initiate a parent kernel function. Subsequently, in Step 2.1 and 2.2, parallel processing will be carried out in each parent thread simultaneously by the parent kernel function.

Step 2.1: Select a tetrahedron that has passed the preliminary process, then retrieve all cube grids within the range of the projection onto the XYZ hyperplane of the selected tetrahedron.

Step 2.2: Establish a suitable thread network, then initiate a child kernel function within the kernel function of Step 2. The child kernel function handles intersection judgment and T-value computation between the selected tetrahedron and its covered cube grid respectively. Each T-value of this child thread is compared to the stored T-value in the corresponding grid, and it will be replaced if the stored value is larger than the value of the child thread.

Step 3: Transfer the final result to the host memory.

…

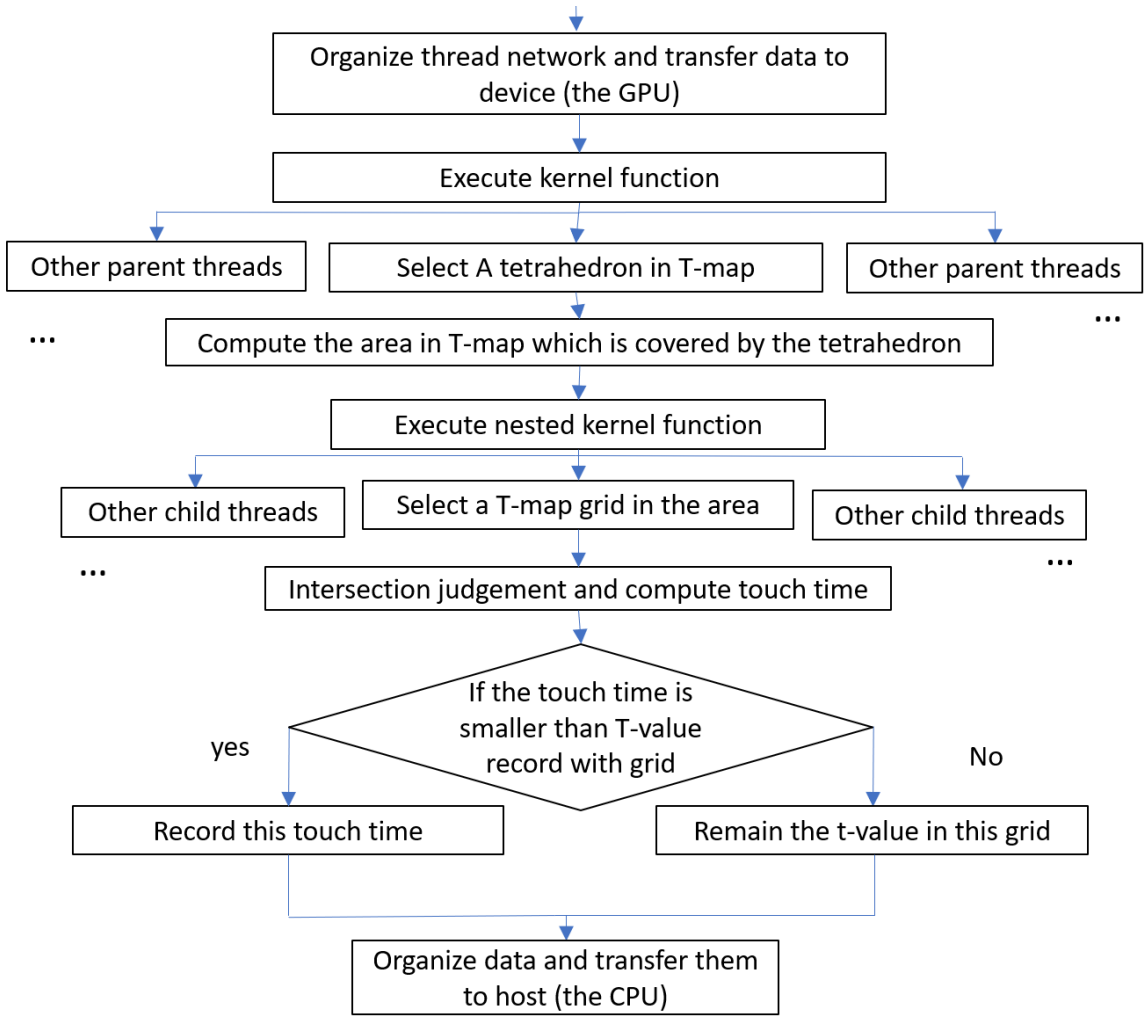


Fig. 5.12 T-map constructing flow with dynamic parallel processing

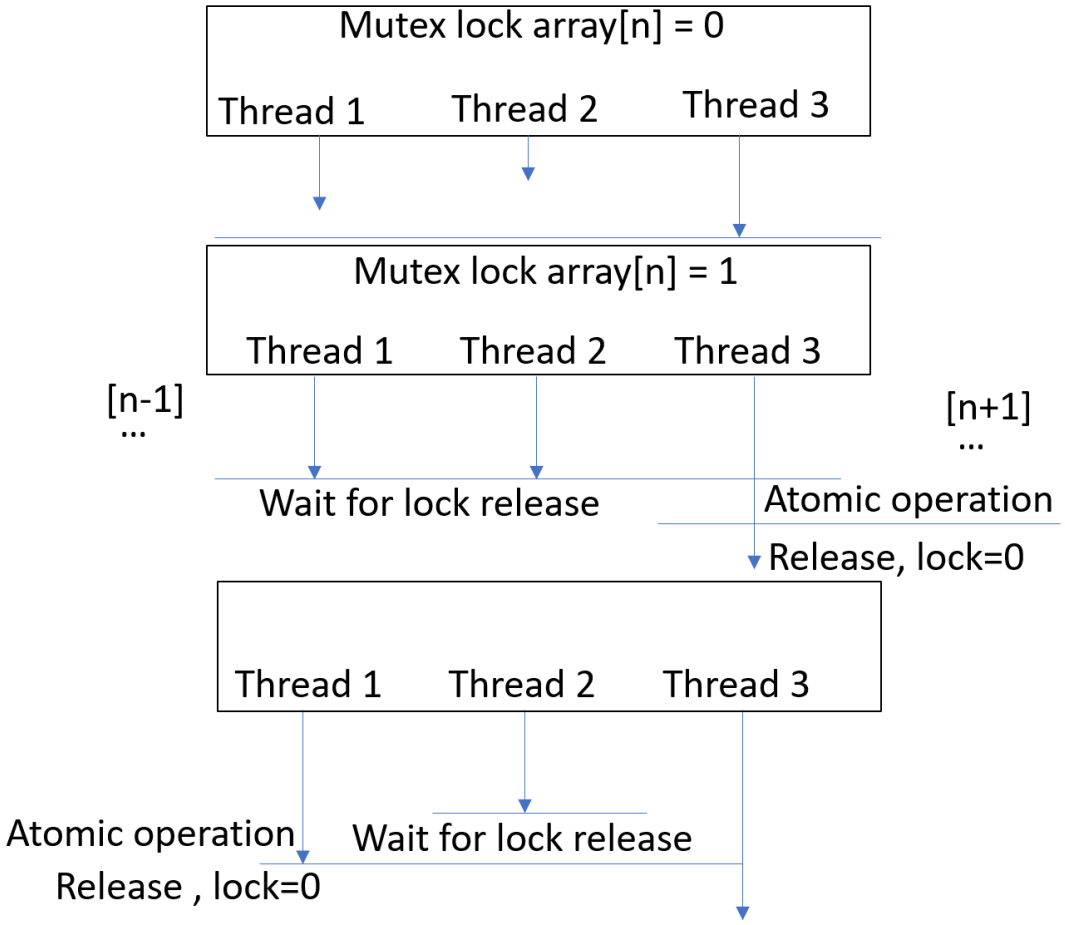


Fig. 5.13 Theory of mutex lock and atomic operation

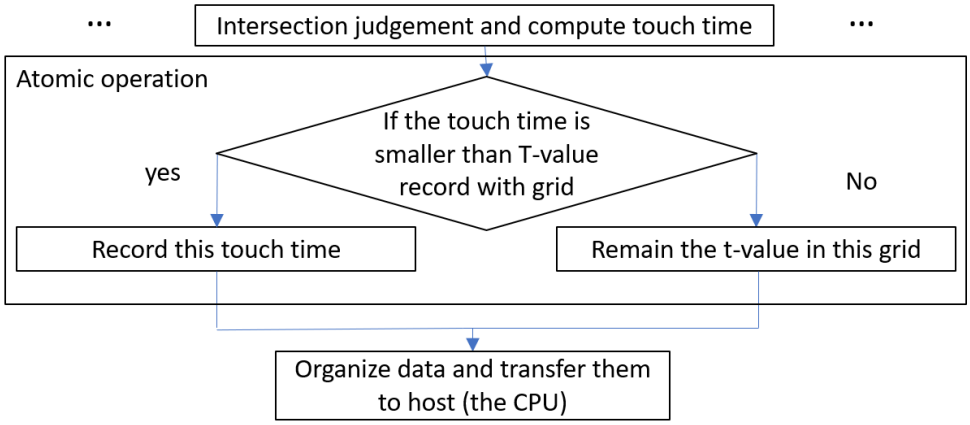


Fig. 5.14 Atomic operation part is the flow

This strategy leverages dynamic parallelism to transform the two nested loop bodies of the entire T-map processing into two nested kernel functions, as depicted

in the flowchart in fig. 5.12. Nearly the entire T-map processing process is parallelized. By significantly condensing the serial code, the running speed is greatly enhanced [66].

To achieve higher parallelism, the process of iterative comparison to find the minimum T-value is placed at the end of the child kernel function. However, this approach may lead to race conditions, potentially impacting the final result of each grid. To address this issue, atomic operations are introduced in the child threads to mitigate race conditions. Atomic operations enable a portion of parallel code to execute serially by thread sequence, ensuring that they are as inseparable as an atom [14].

There are various types of general atomic operations. In an attempt to mimic serial programming, all child threads of all parent threads execute serially through atomic operations. While this method may seem reliable, it significantly extends the calculation time due to the GPU's inferior serial processing capability compared to the CPU.

An alternative proposed modification to atomic operations involves the use of an array of mutex locks, as illustrated in the example shown in fig. 5.13. In the case of the T-map structure, first touch times belong to numerous cube grids. Hence, for each grid, the iteration to find the first touch time is independent. This mutex array is designed with the same length as the number of grids, allowing each cube grid to have a separate mutex lock. When applied in child threads, this ensures that the program is serial for one grid while parallel for all grids, as depicted in fig. 5.14. Although its speed may be slightly slower compared to atomic operations, it remains one of the most suitable parallel strategies.

5.4 Marching cube and T-map structure

5.4.1 Represent CWE shape

Figure 5.10 illustrates a 2D analogy of T-map result. In this figure, boundary of TOR, T-map segments and 3D voxel workpiece model bounded a closed area. The boundary can be regarded as iso-surface of 4D marching cube. By voxelization the 4D space, 4D marching cube method can be introduced to convert T-map structure to mesh model.

In section 3.4.1, the author introduce a modified 4D marching cube method proposed by Onosato et al. This modified 4D marching cube method uses two 3D voxels in different XYZ-hyperplane to consist a 4D voxel. This feature makes the preprocess of 4D marching cube easier and the marching cube process faster.

The 4D space is divided into many 3D XYZ-hyperplane based on a series of time steps, such as the fig 5.15.

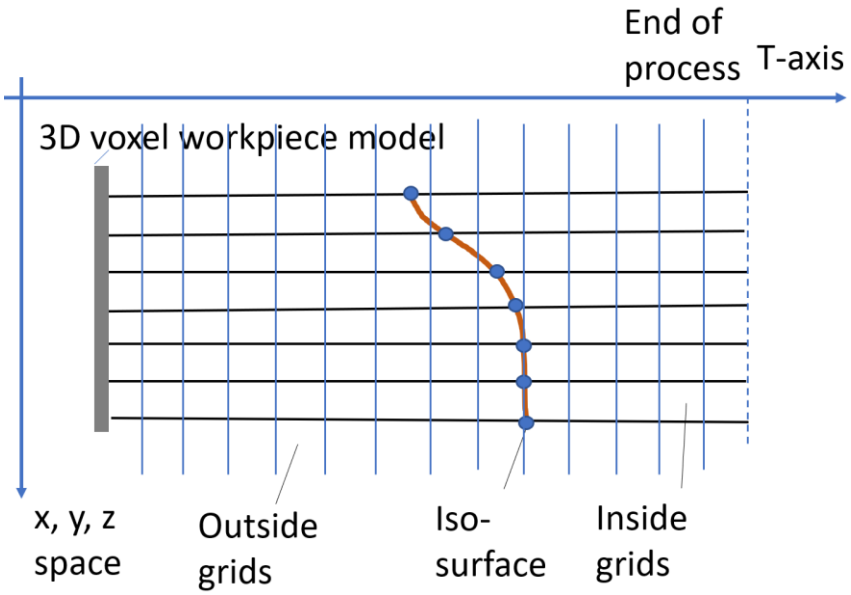


Fig. 5.15 2D analogy of T-map

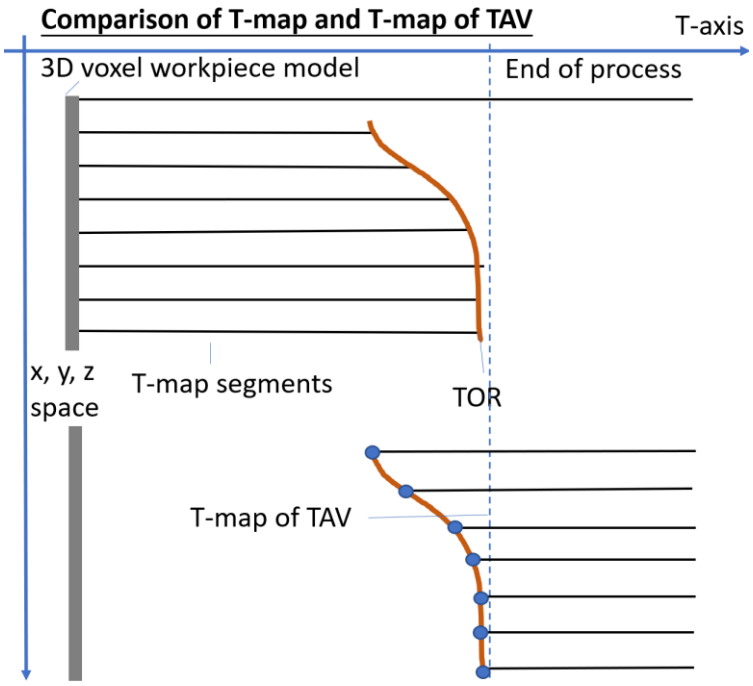


Fig. 5.16 Different parts in T-map method

5.4.2 Represent TAV shape

In section 3.5.4, an important 4D model to represent CWE process called as TAV was introduced. Since there is no direct method to generate TAV. the author try generating it in T-map study.

The basic idea is similar to generate mesh model of CWE from T-map structure. The difference is that it starts from the "first touch time point" where the T-map line segment intersects the boundary of the TOR and extends to the end time of the TOR. The space enclosed by the above parts can be understood as a TAV. An analogy and flow are shown in fig 5.16 and fig. 5.17.

When execute this process, a problem may appear. There may a cave in the TAV result, like fig. 5.18. The reason for this is that the tool displacement is too small and the tool has not completely moved out of the initial position. Since the T-map method can only record the intersection of the line segment and the TOR boundary. Therefore, the "TAV" generated by this method only represents the change history of the volume swept by the tool surface. In other words, the area swept inside the tool is not recorded, illustrated in fig. 5.19.

In this case, we need to pre-adjust the parameters of the tool base position and set them all to the inside of the TAV. The generated TAV can be used to CWE analysis, which is explained in fig. 5.20.

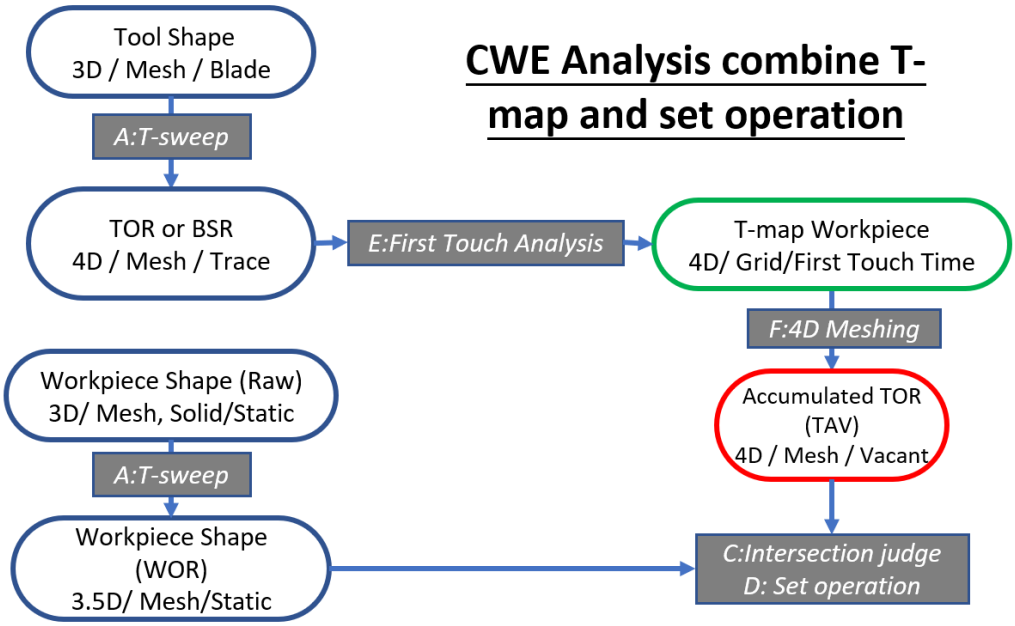


Fig. 5.17 CWE Analysis combine T-map and set operation

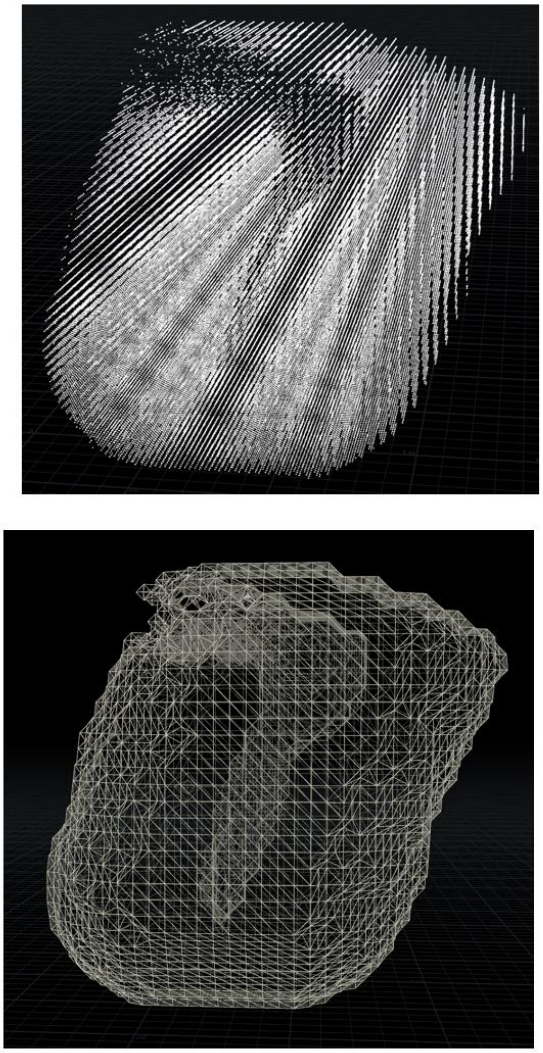


Fig. 5.18 A cave in T-map TAV result, projection

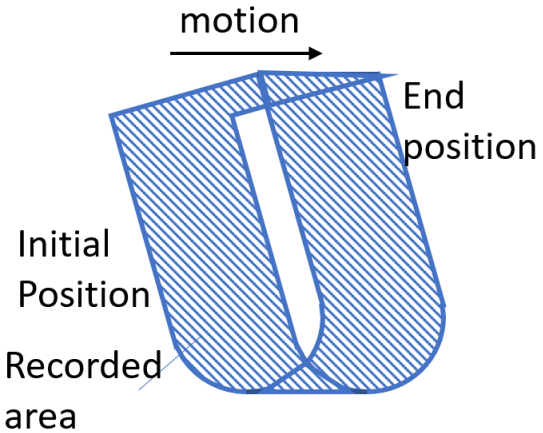


Fig. 5.19 An analogy of cave in T-map TAV

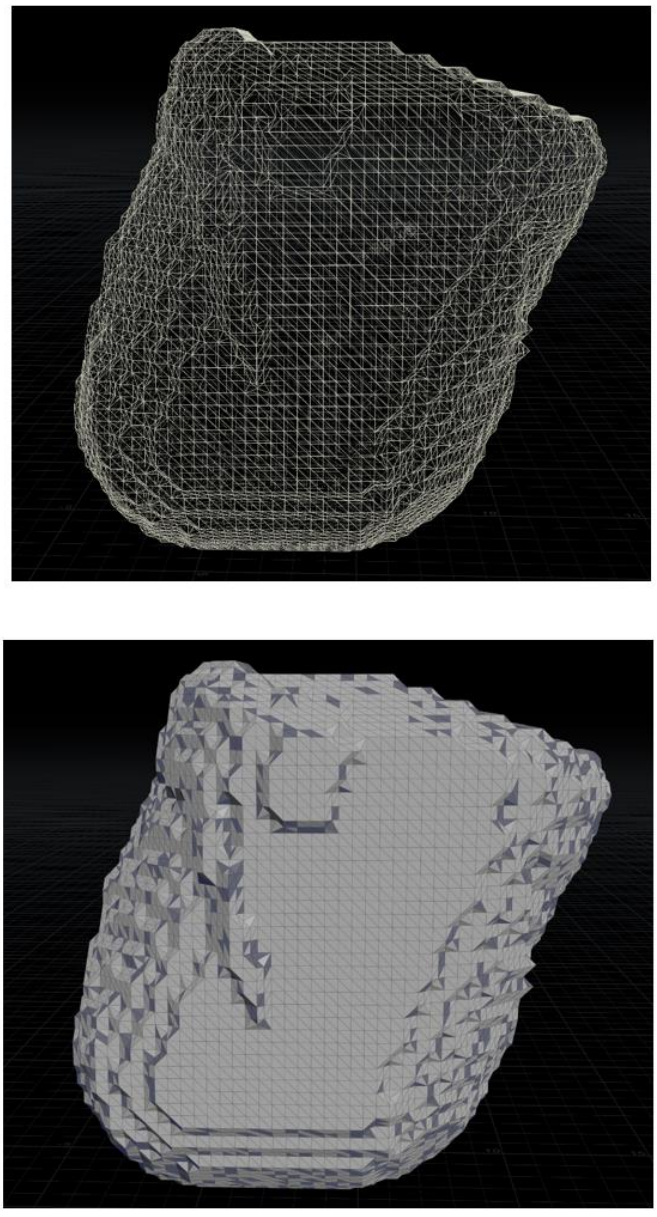
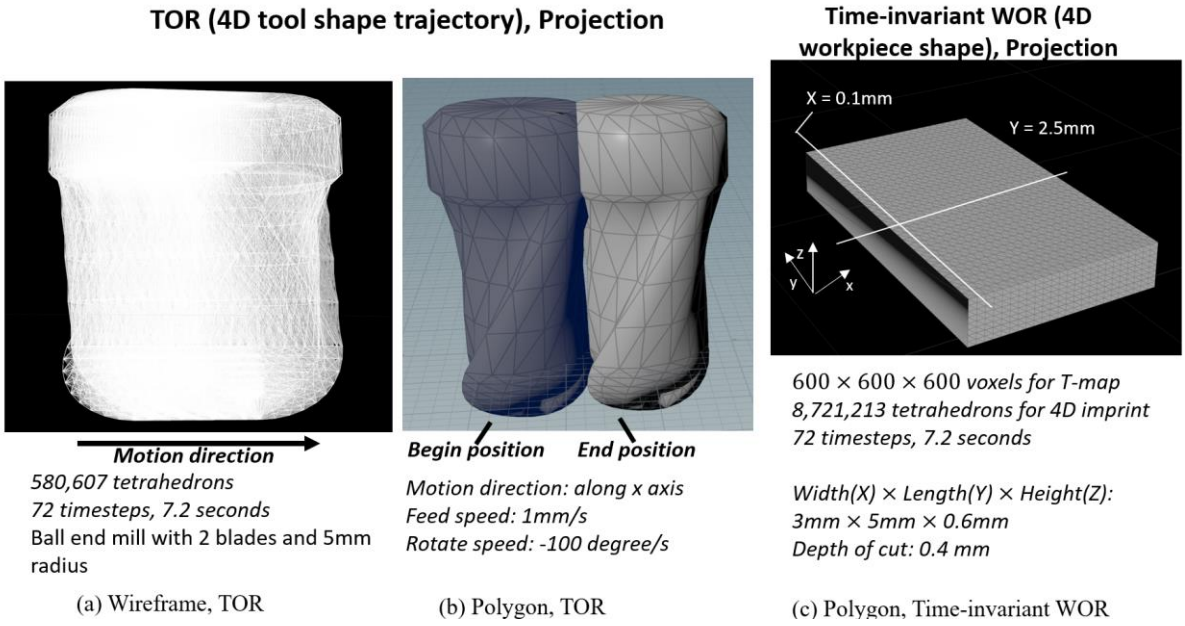


Fig. 5.20 T-map TAV result, projection

5.5 Result and discussion

According to Fig. 5.1, we implemented the T-map approach for CWE analysis. The Fig. 5.22 shows the T-map structure that meshed by the 4D marching cube method 7), the input models are illustrate in Fig. 5.21. The Schematic diagram of tool cutting workpiece is illustrated in Fig.20. Two figures on the left column of upper side are cutter and workpiece conditions (XYZ-hyperplane slices) when t is equal to 3s and 5s. Different colors show milled by different blade. Obviously, the XYZ-hyperplane slice can illustrate the shape deform and position relation of tool



(a) Wireframe, TOR (b) Polygon, TOR (c) Polygon, Time-invariant WOR

Fig. 5.21 Input models for analyzing

Table 5.1 Computing time consuming

Compute time and algorithms

Process type \ Grids	25,700	25.7e6	216e6
Serial program	0.032s	18.964s	NA
Parallel program	0.193s	10.593s	NA
Dynamic parallelism without atomic operation	1.080s	1.325s	3.757s
Dynamic parallelism with atomic operation	1.076s	1.382s	3.139s

and workpiece. Obviously, the XYZ-hyperplane slice can illustrate the shape deform and position relation of tool and workpiece. The nether figures are the YZT and XZT-hyperplane slice with X = 0.1mm (edge) and 2.5mm (center). With the increasing of T value, the material remaining and contact area can be observed.

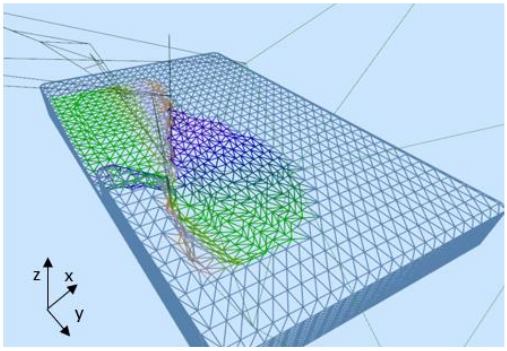
The PC with an Intel Core i9-10900K CPU, NVIDIA GeForce RTX 3080 GPU with 10 GB memory, and 32 GB main memory. And the time consume is illustrate in Table 5.1.

Table 5.1 provides the computing time under different conditions. It is observed that when the number of grids is not too high, algorithms utilizing parallel processing are evidently slower than the serial program. The authors attribute this to the necessity of data transmission between the host and the device before and after the kernel function. The preparation, execution, and subsequent operations of transmission consume a certain amount of computing time. Therefore, considering the computational cost, it is more appropriate to opt for a serial program when the number of grids is small. However, as the number of grids increases, the speed gap between the two approaches becomes increasingly apparent. Even the slower parallel strategy demonstrates a speed improvement of about 45% compared to the traditional serial code.

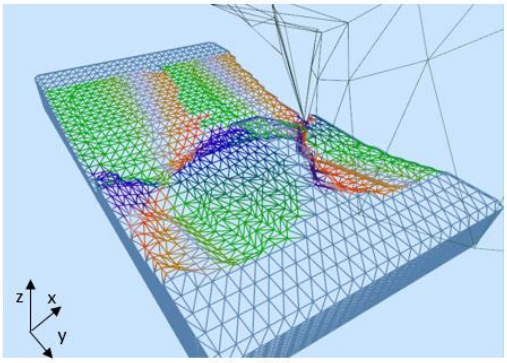
There is minimal difference between the results of dynamic parallelism with atomic lock and those without atomic lock. Based on a rough analysis of the data results from both approaches, the authors speculate that when the number of grids in the T-map is not too small, it becomes challenging for race conditions to occur. Additionally, for programs with atomic operations, there will not be an excessive number of iterations per grid.

In general, T-map is a relatively complete method. It can be used for parallel processing while controlling the amount of calculation and achieving the required accuracy. However, due to its discreteness in 3D space, it is difficult to find an efficient way to mesh it. Although the 4D marching cube method has a higher efficiency, it first requires a large amount of input data support and also generates a large number of unnecessary tetrahedrons. For the model in this example, the 4D marching cube will produce a mesh model composed of millions of tetrahedrons, which greatly complicates subsequent processing.

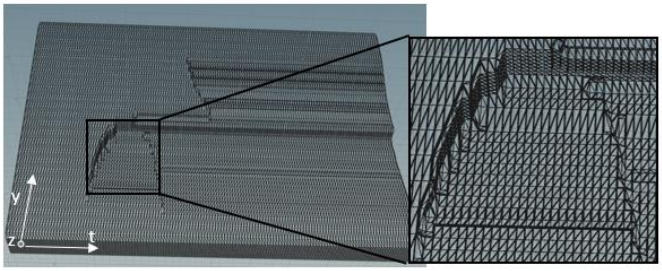
T-map



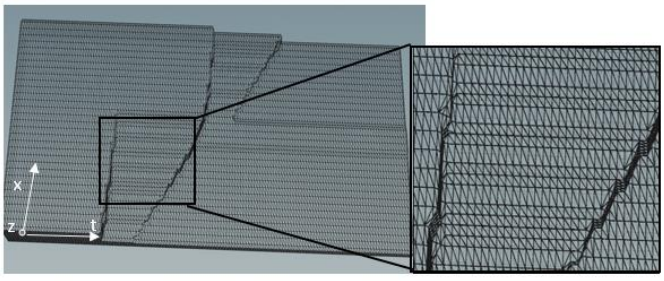
T = 3.0s



T = 5.0s



X = 0.1mm (Left side of model)



Y = 2.5mm (Mid of model)

Fig. 5.22 Result of CWE by T-map

5.6 Algorithm outline of 4D imprint method

The T-map structure emphasizes the characteristic which is continuous in time but discrete in 3D space. We tried using 4D marching cube to meshing the T-map structure. But it will produce too many redundant tetrahedrons and take a long time for postprocessing. Therefore, the 4D imprint method are developed to emphasizes the continuity of the 3D surface and simple structure for computing. But it gives up the continuity of the T- axis. The basic idea of 4D imprint is split 4D mesh by T-axis, process in 3D space, then connect them to 4D mesh again.

The algorithm extracts vertices data and relationships from a 3D mesh model of the workpiece. Specifically, the vertices are calculated with TOR's 3D temporal slices. Algorithm judge is point in or out of the shape of each TOR slice. Inside points will be moved to the exterior of the tool shape and their positions are recorded. This operation is iteratively executed until all sampling times have been traversed.

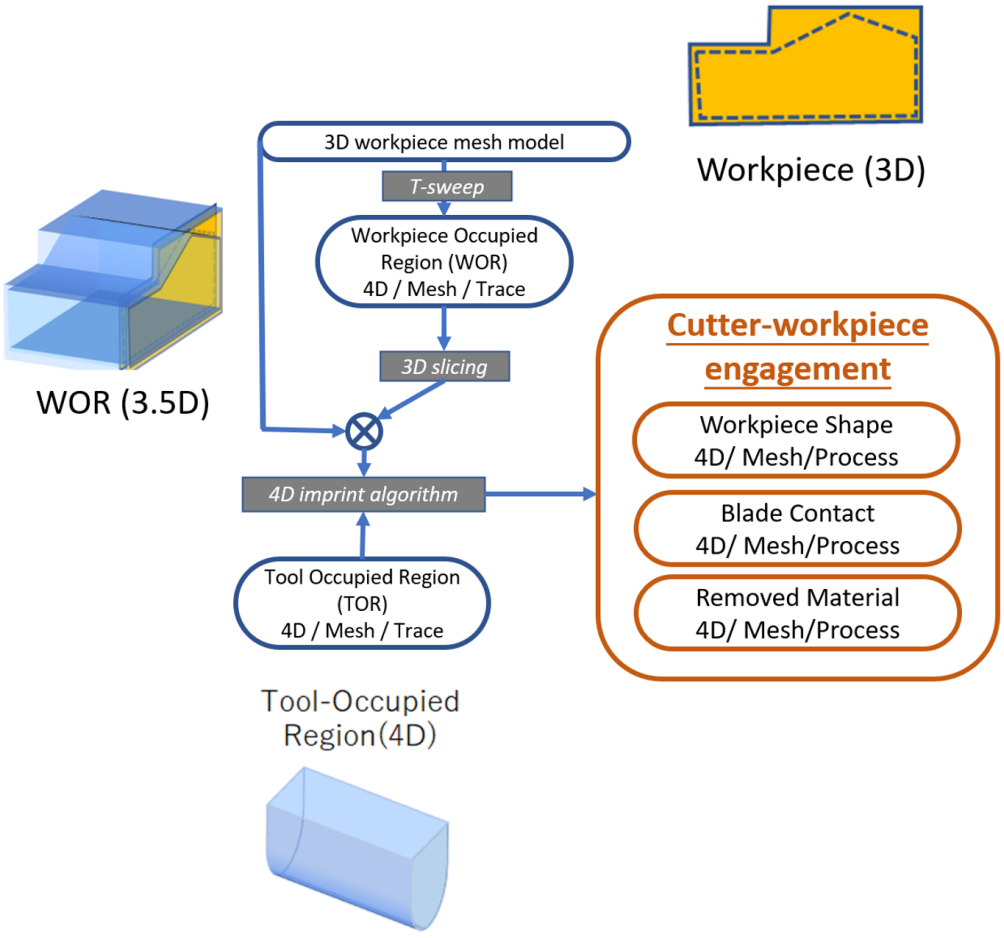


Fig. 5.23 Strategy of 4D imprint method

The vertices obtained at each time step are then restored to the 3D mesh and connected using Otomo's method to form a 4D mesh model. This 4D mesh model enables representation of the workpiece deformation during the CWE process. The algorithm is illustrated in fig. 5.23. Eq. (5.3), (5.4), (5.5), (5.6) are the mathematic indication of 4D imprint.

$$\left(P_t, R(TOR)\right) \rightarrow P_t' \ \ \mathbb{R}^3 \quad \text{Eq. (5.3)}$$

$$P_t' \rightarrow m_t^{3D}(V_t^{3D}, T_t^{3D}) \ \ \mathbb{R}^3 \quad \text{Eq. (5.4)}$$

$$M \in m_t^{3D} \ \ \mathbb{R}^4 \quad \text{Eq. (5.5)}$$

$$M \rightarrow \ m^{4D}(V^{4D}, T^{4D}) \ \ \mathbb{R}^4 \quad \text{Eq. (5.6)}$$

P_t: vertices at time t

$R(TOR)$: Region of TOR

P_t': vertices after displacing.

m_t^{3D}: 3D mesh at time t, V_t^{3D}: vertices, T_t^{3D}: Topology

M: meshes set

m^{4D}:4D mesh, V^{4D}: vertices, ,T^{4D}: Topology

Eq. (5.3) illustrates the displace of points, exemplified strategies are shown in Fig.5.23. Eq. (5.4) illustrates the remeshing process of points. Eq. (5.5) and (5.6) connect 3D mesh to 4D mesh. This process is illustrated in fig.5.24.

The basic steps of 4D imprint are represented below and fig. 5.24:

Input. The TOR and 3D workpiece shape.

Step1. Subdivide the TOR based on time steps. The results are a series of 3D models of dynamic tool shape.

Step2. Select the initial tool shape and process with the 3D workpiece shape. Change the position of vertices of workpiece and re-mesh it according to Eq.(6).

Step3. Repeat the process of Eq(6) by select different 3D models of tool shape along the time axis. The changed workpiece models are recorded, and changes are accumulated to next iterative.

Step4. Connect all 3D changed workpieces model to a 4D model. The connecting process realize by connecting same polygons in different time steps.

Output. The 4D model record changes of workpiece is the CWE analyzing result.

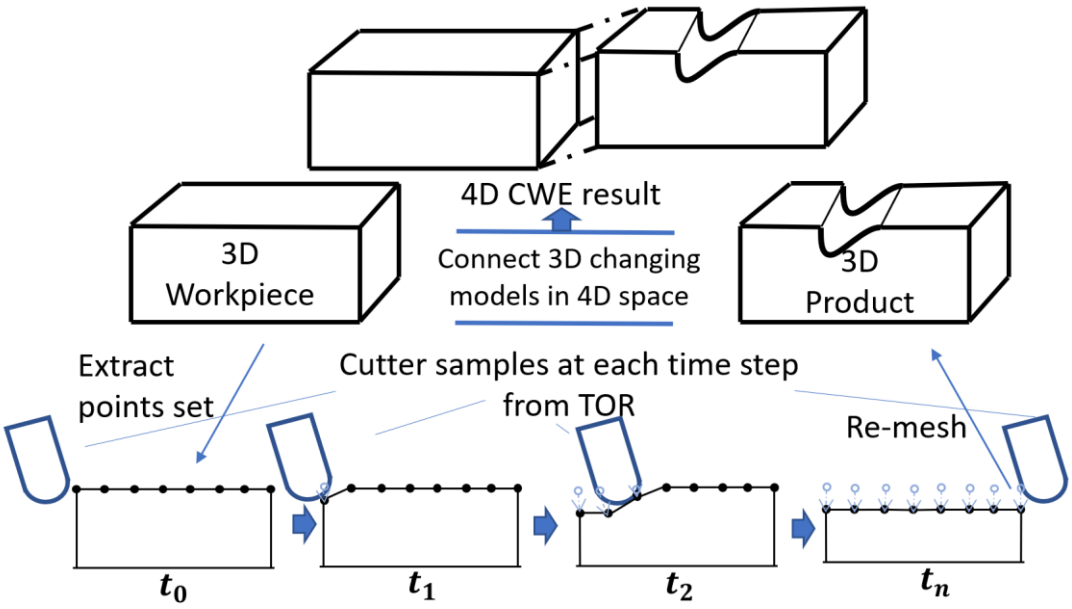
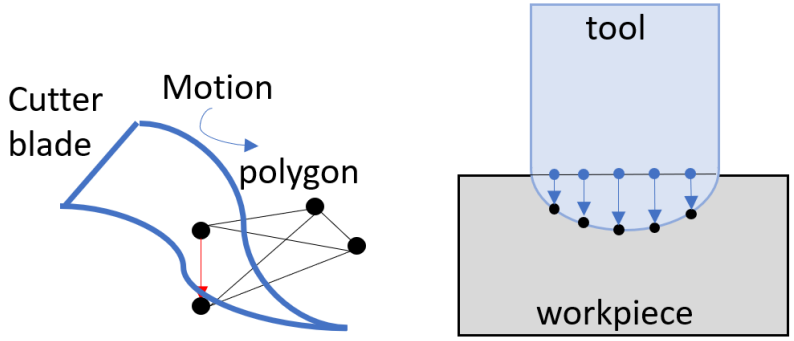
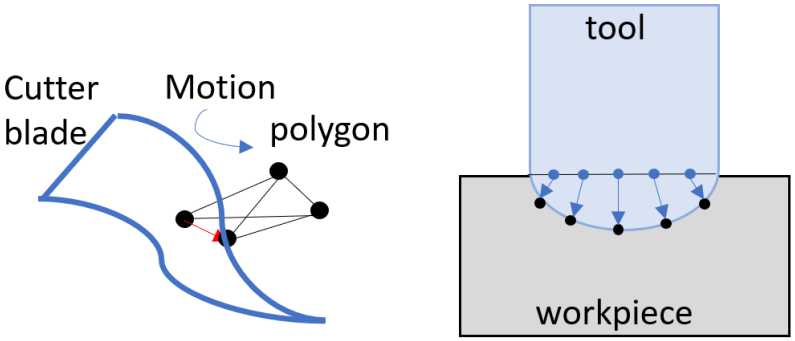


Fig. 5.24 Processing of 4D imprint



(a) Moving the penetrated point downwards.(microcosmic)

(b) Moving the penetrated point downwards.(macroscopic)



(c) Moving the penetrated point along cutter motion direction.(microcosmic)

(d) Moving the penetrated point along cutter motion direction.(macroscopic)

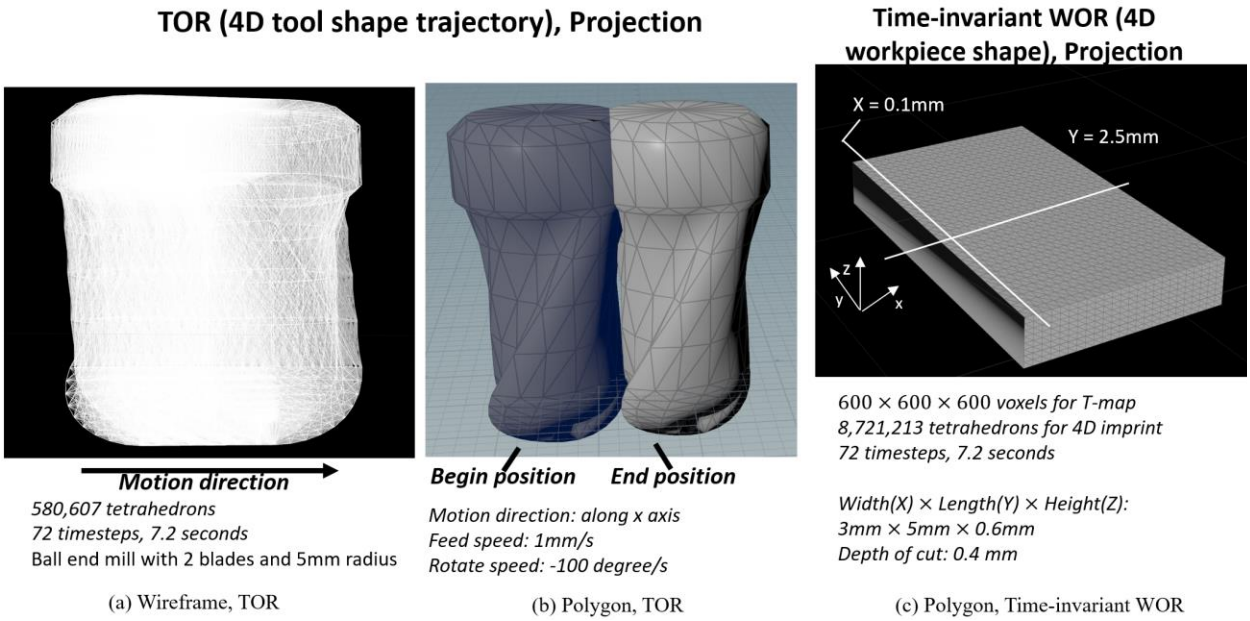
Fig. 5.25 Two strategies of points motion

5.7 Moving direction of touched point

Currently, two feasible strategies exist for moving points that exist inside a 3D slice of the TOR. The first method is a simple yet effective approach that involves moving the points directly downwards along the Z-axis until they reach the surface of the triangle mesh. The second method aims to simulate the cutting process, where points move along the cutter motion direction of the polygon mesh swept by the cutting tool to simulate material deformation caused by the tool. Both of two methods are illustrated in Fig. 5.25.

5.8 Result and discussion

After implementing the process in Fig. 5.24 and 5.25. There are slices of analyzing CWE by 4D imprint method in Fig. 5.26. The right column of upper part in fig.5.27 are XYZ-hyperplane slices with T= 3s and 5s. The nether figures are the YZT and XZT-hyperplane slice with X = 0.1mm (edge) and 2.5mm (center). Same slices with T-map method can compare the result between T-map and 4D imprint method. Because of the resolution and divided method of the input mesh model, there are zigzag and wrong normal cases in the result. the author think a more accurate 3D model with tiny polygons can reduce occurs of these problems. Also, you may notice that the YZ plane of the model is subdivided. This is done to increase



(a) Wireframe, TOR (b) Polygon, TOR (c) Polygon, Time-invariant WOR

Fig. 5.26 Input models for analyzing

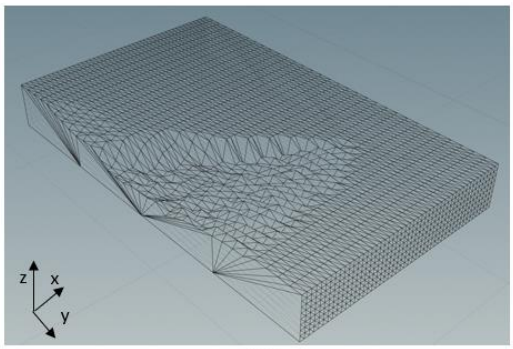
the resolution while also ensuring the accuracy of the recorded CWE.

Since the 4D imprint method, only changes the position (height) of the vertices in the model and does not change the topological information and the number of vertices in the model. Therefore, this method has some limitations. It does not take into account the material shavings removal, but only the deformation of the workpiece. This approach is similar to using a depth buffer to represent the 3D cutting process. The author thinks this method is really suitable for expressing deformation processing methods such as forging. Since the topologic structure remains, the author thinks the example of the material being cut into two parts is difficult to realize. For a cut where a large amount of material needs to be removed, it is possible, but the many unnecessary vertices may remain in the result model to represent it. This is a problem that the author hopes to solve by developing a vertex merging algorithm. So, the author considers optimizing the result represent of 4D imprint in future. In other hands, increase the time resolution of TOR can also achieve a result with better accuracy.

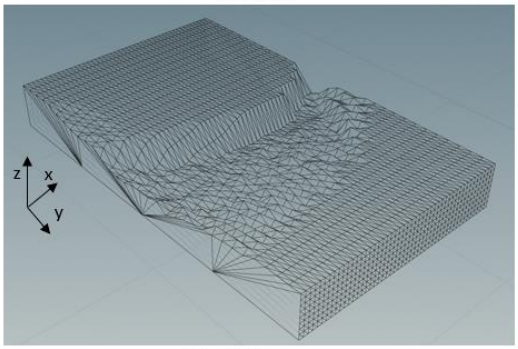
Processing results at different time steps will be connected in 4D space to become 4D CWE results. In this part, we use a connection method which similar to Otomo's method. Since the models at different time steps have the same topological structure, they can be connected. But the method will lose accuracy. Such a situation can be solved by reducing unnecessary vertices which mention in last paragraph and subdividing the time period.

4D imprint

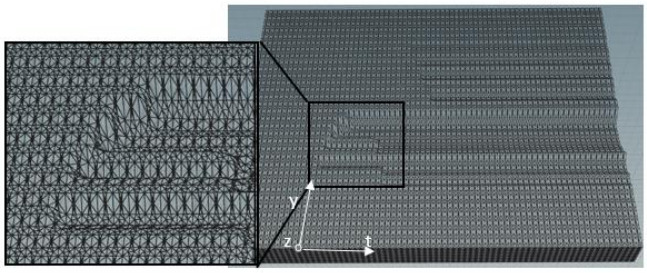
Computing Time: *380s*
Memory: *10.1Gb*



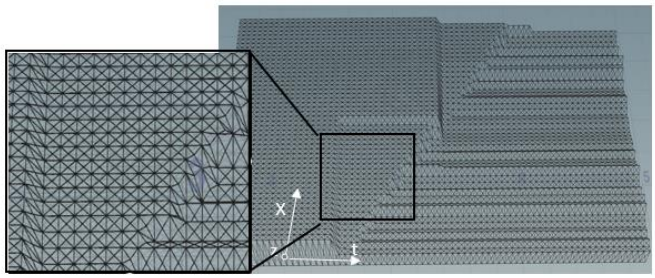
T = 3.0s



T = 5.0s



X = 0.1mm (Left side of model)



Y = 2.5mm (Mid of model)

Fig. 5.27 Result of CWE by 4D imprint

Chapter 6. Set operation method for CWE analysis

6.1 Algorithm outline of set operation method

The set operation results of the two 4D models were combined with the original tetrahedrons and subdivided tetrahedrons. These processes are expressed in Eq.(6.1), Eq.(6.2), Eq.(6.3), and Eq.(6.4). The original tetrahedrons are those that do not intersect with others but require classification inside or outside of another 4D model. The subdivided tetrahedrons are divided from the intersecting tetrahedrons, which are either inside or outside of another 4D model.

$$M_A \cap M_B = T_A^{in} \cup T_B^{in} \cup T_{Asub}^{in} \cup T_{Bsub}^{in} \quad Eq.(6.1)$$

$$M_A \cup M_B = T_A^{out} \cup T_B^{out} \cup T_{Asub}^{out} \cup T_{Bsub}^{out} \quad Eq.(6.2)$$

$$M_A - M_B = T_A^{out} \cup T_B^{in} \cup T_{Asub}^{out} \cup T_{Bsub}^{in} \quad Eq.(6.3)$$

$$M_B - M_A = T_A^{in} \cup T_B^{out} \cup T_{Asub}^{in} \cup T_{Bsub}^{out} \quad Eq.(6.4)$$

Step-1 Inside and outside determination

After the intersection judgment, the non-intersection of the two meshes is achieved. These non-intersected tetrahedrons must be categorized as inside or outside another mesh model to obtain parts 1 and 2 of the subtraction.

A straightforward approach was employed to extend a ray from a given point and tally the number of intersections between the ray and another 4D model. An odd count indicates that the point lies outside the model. Fig. 6.17 elucidates this process. Given that it's established that these tetrahedra do not intersect with others, the centroid of each tetrahedron serves as a suitable point for such judgments.

Step-2 Data organization for set operation

Before processing the set operation, information about intersections is organized.

The two tetrahedrons intersect in a single hyperline. The shape of the intersection region depends on the overlapping shapes of the 2D slices of the

two 3D tetrahedrons. This region has a polygonal shape and is shared by two intersecting tetrahedrons. To ensure that these shared faces generate the same new tetrahedrons in the set operation, they were triangulated using the Bowyer–Watson algorithm for Delaunay triangulation. This triangulation ensures the same pattern of the two shared faces by the features of Delaunay triangulation. These triangles are referred to as “crossed sections.”

A tetrahedron may intersect several tetrahedrons from other models, as shown in Fig. 6.10. Therefore, some data are integrated into a unit for the following set of operations. This unit, illustrated in Fig. 6.10, includes a tetrahedron base, all cross-sections intersecting with other tetrahedrons, and their directions.

Step-3 Advancing Front Technique

The Advancing Front Technique (AFT) [79] underwent modifications to decompose a tetrahedron into smaller tetrahedrons, dividing them into internal and external parts. This iterative method triangulates or tetrahedralizes the interior of the point cloud by continually generating triangles or tetrahedrons in a specified direction.

For each tetrahedral base, the AFT operates by projecting the tetrahedron onto a 3D space. Initially, the advancing fronts comprise all cross-sections in the tetrahedron base. Subsequently, new tetrahedra are extended from both sides of the crossed sections. An advancing front is linked to a vertex of the tetrahedron base or other cross-sections to form a new tetrahedron. In essence, all the vertices of the new tetrahedron are derived from the vertices of the tetrahedron base and the crossed sections. The primary consideration in finding a suitable connection point is to identify the vertex closest to the plane of the advancing front in the advancing direction, subject to a series of constraints, as depicted in Fig. 6.19. The three new faces of the tetrahedron are treated as new advancing fronts, and subsequently, the next-generation tetrahedron is extended until it contacts the face of the tetrahedron base or other advancing fronts.

The newly generated tetrahedrons are categorized based on their generation direction. If they are produced in the normal direction of the

advancing fronts, they are considered to exist outside another 4D model; conversely, they are deemed inside if generated in the opposite direction.

In Fig.6.24, an example of the AFT process of a tetrahedron base is illustrated. The red segments indicate active advancing fronts. Initially, the tetrahedron extends in the normal direction, generating the outer part of the tetrahedron base (shown in black). Then, the extension direction changes to the opposite of the cross-section's normal, producing the inner part in orange. This extensive process involves connecting the vertex of the cross-section with the selected cross-section to construct the nearest new tetrahedron. If no suitable vertex is found from the cross-section, a vertex of the original tetrahedron is utilized.

By employing the AFT, the subtraction of parts 3 and 4 can be achieved.

The CWE process can be represented by 4D mesh model set operation as eq(7.5):

$$M_{CWE} = M_{WOR} - M_{TAV} \quad Eq.(7.5)$$

6.2 Boundary intersection judgement between two 4D mesh model

6.2.1 Outline of boundary intersection algorithm

The initial step of 4D set operation is the boundary intersection algorithm of 4D mesh model. Since this part is independent of the set operation algorithm and has a complete, complex and independent algorithm, it is not listed in the steps of set operation, but is described separately in Section 6.2 as a preprocessing step of set operation. Matsunaga el al. from our lab developed the boundary intersection algorithm of 4D mesh model in 2017 [15]. However, since the algorithm was not fully completed in the paper of Matsunaga et al., the author improved and modified the algorithm. The modification mainly focuses on the following two points:

1. Modification of cross decision as a preprocessing of set operation.
2. Parallel processing algorithm of cross decision of complex models.

The boundary of 4D mesh model is consisted by tetrahedrons. Therefore, The boundary intersection algorithm focuses on the intersection cases between two

tetrahedrons from different 4D meshes and discuss cases case-by-case. So, let's start from the intersection of a pair of tetrahedrons in 4D space below.

Figure 6.1 is the processing flow of tetrahedrons intersection judgement. Through 5 decision steps, the cases intersection of tetrahedrons are classified to 3 kinds:

Case 1: Tetrahedrons intersection in same hyper-line.

Case 2: Tetrahedrons intersection in same hyperplane.

Case 3: Tetrahedrons not intersect.

Tetrahedron is a kind of 3D simplex. In 4D space, a tetrahedron must exist in a 3D hyperplane. So, the first decision judge two 3D hyperplanes ($Hp(T_i)$ and $Hp(T_i)$) which contain tetrahedrons (T_i and T_i)are parallel or not. If they are parallel, two hyperplanes will not intersect. Tetrahedrons are not intersect. The second decision judge two 3D hyperplanes are same or not. If they are same, two tetrahedrons in a same 3D hyperplane. Tetrahedrons intersection in same hyperplane is required to consider in decision 5. If two hyperplanes are different, two 3D

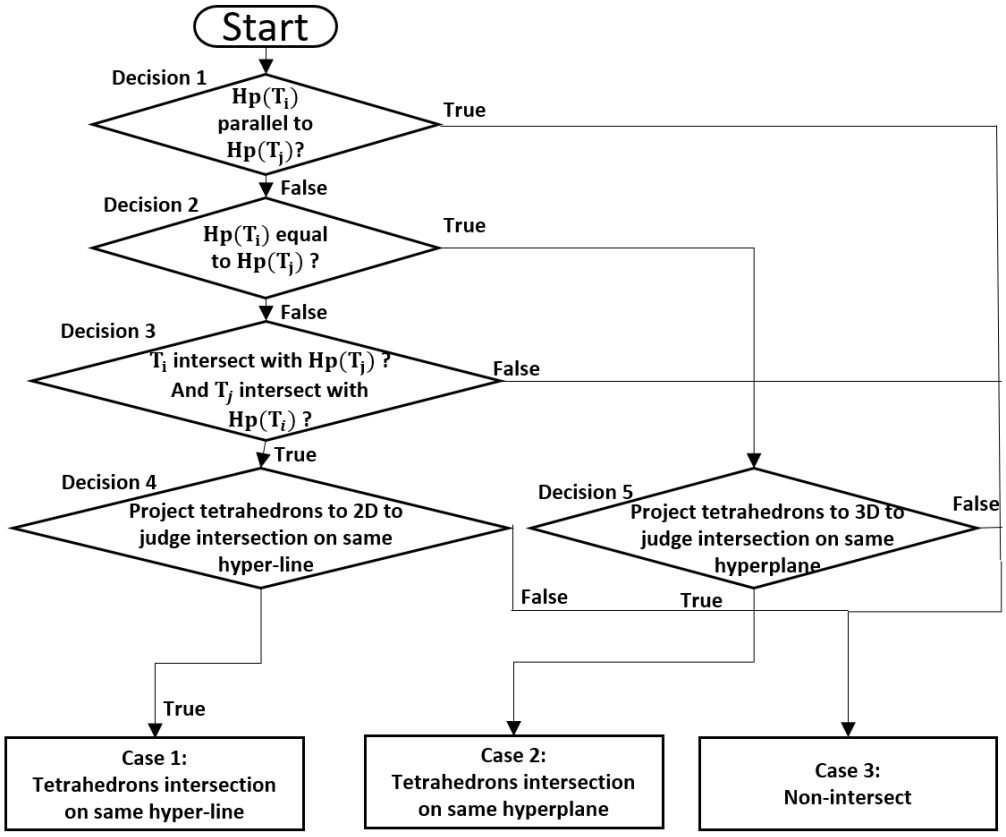


Fig. 6.1 Flow chart of tetrahedrons intersection judgement

hyperplanes must intersect on a 2D hyper-line. Decision 3 judge if the 2D hyper-line cross tetrahedrons. If the 2D hyper-line cross both of two tetrahedrons, the intersection judgement of tetrahedrons can execute in this 2D hyper-line. Algorithm judges hyper-line crossing tetrahedron by computing signed distances from vertices to the hyper-line. If there are both positive and negative distances, the tetrahedron and the hyper-line intersects. The process is fig. 6.2.

Repeat the above process until all tetrahedrons on the two 4D mesh models have been compared one by one, and the boundary intersection judgement is completed. Obviously, the time complexity of direct operation is $O(n^2)$. The algorithm itself is relatively complex, and the 4D mesh model of the general CWE process contains at least hundreds of thousands of tetrahedrons, which causes the boundary intersection judgement to take a long time.

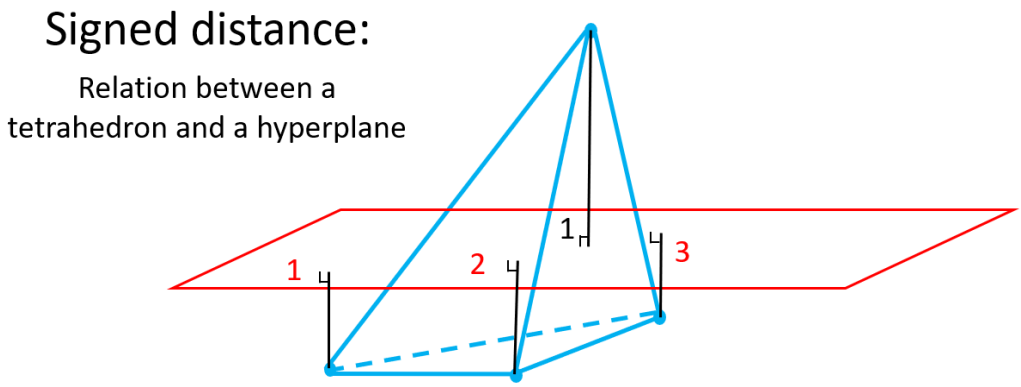


Fig. 6.2 Computing of signed distances

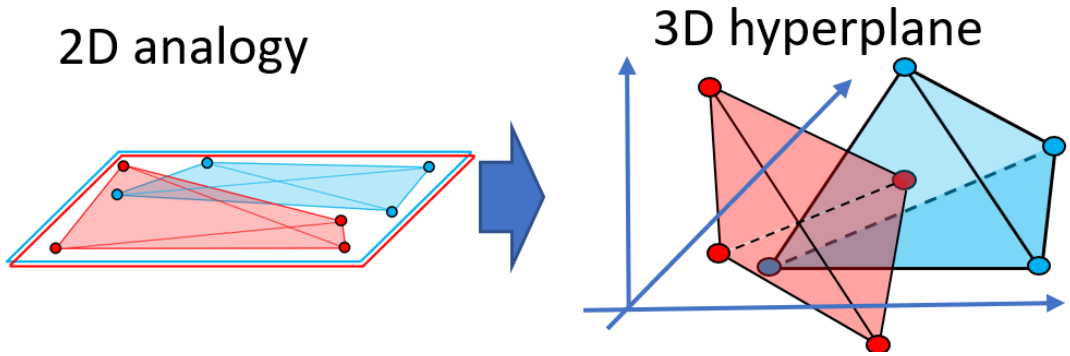


Fig. 6.3 Illustration of tetrahedrons intersect in a 3D hyperplane

6.2.2 Tetrahedrons intersect in a 3D hyperplane

In tetrahedron intersection algorithm, tetrahedrons intersection judgement in 3D

same hyperplane will be prioritized.

This is a case that intersect in $N-1$ dimensional space, like polygons overlap in 3D mesh model. This case illustrates in fig.6.3.

For complex CWE 4D mesh models, the intersection in a 3D hyperplane is very rare. And the 3D intersection judgement is now mature and will not be discussed in this article.

6.2.3 Tetrahedrons intersect on a 2D hyper-line

Tetrahedrons intersect on a 2D hyper-line is a common case of tetrahedrons intersection. Almost all intersection cases in a pair of complex 4D models is intersection on a 2D hyper-line.

This is a case that intersect in $N-2$ dimensional space, like polygons intersect on a segment of 3D mesh models. Therefore, 2D hyper-line is a term in this dissertation. Actually, it is a 2D plane. The hyper-line is a plane cut the tetrahedrons. The **cross section** is a 2D pattern on this 2D hyper-line. If two tetrahedrons intersect on the hyper-line, parts of 2D patterns of them will overlap. the author called this overlapped area is "**crossed section**". This case illustrates in fig.7.4.

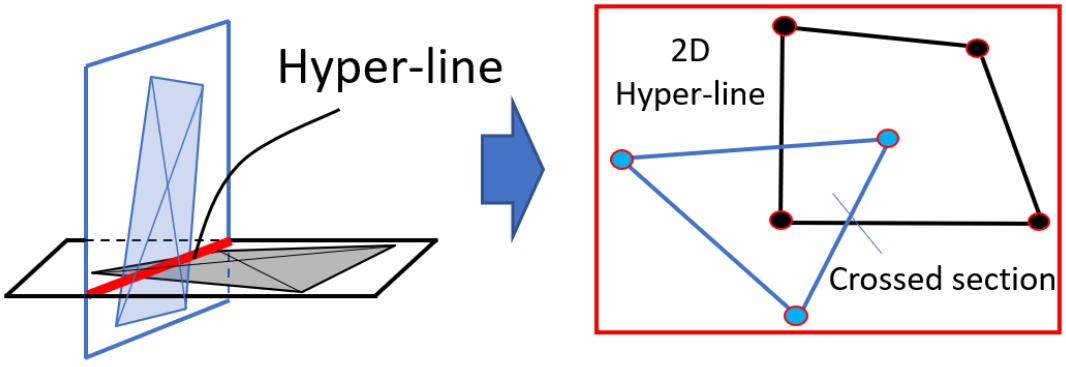


Fig. 6.4 Illustration of polygon intersection on a 2D hyper-line

6.2.4 Parallel strategies for accelerating the boundary intersection judgement algorithm

In section 6.2.1, the author points out that boundary intersection judgement algorithm developed by Matsunaga et al [15] require traversal all possible tetrahedrons pair of two 4D mesh model. For a 4D mesh model with hundreds of thousands or even millions of tetrahedrons, the times of judgments process will reach trillions. This will undoubtedly take a huge amount of time. Moreover, it is difficult to directly use Embarrassingly parallel to accelerate such a huge calculation. Because even if we can run so many threads concurrently, there are not enough SPs to handle these problems at the same time. This makes it need to be processed hundreds of millions of times of nested judgment statements, which is difficult to process in SP, sequentially in one SP.

Therefore, the author designed a series of parallel strategies for preprocess and process the boundary intersection judgment.

Generally, the input model of set operation that use for analyzing CWE is generate by Otomo's method, such as TOR and Time-invariant WOR. Therefore, the preprocesses is mainly based on some features of tetrahedrons in Otomo's prism.

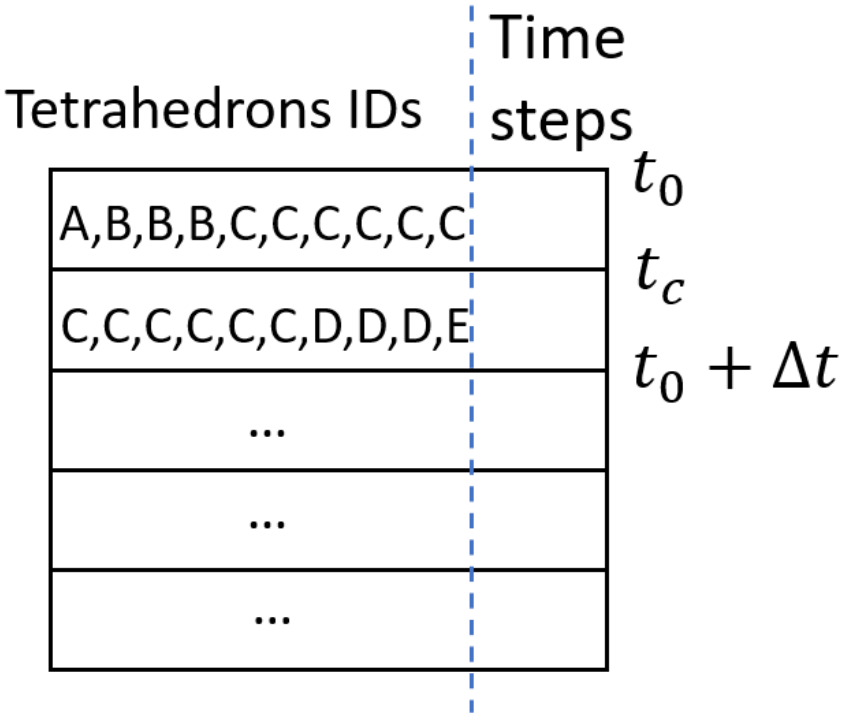


Fig. 6.5 Illustration of time steps with tetrahedrons types

According to section 3.4.3, there are 5 different kinds of tetrahedrons in Otomo's prism. A 4D mesh model generated by Otomo's method can be divided to several segments based on time step. In addition to kind C tetrahedron, A, B, D, E tetrahedrons must occupied entire time period in one time step. C across two complete time steps. This situation is illustrating in fig. 6.5.

The first preprocess traversal all time steps pair in two models and find the overlap time steps. According to the feature of tetrahedrons in Otomo's method, tetrahedrons, which inside of two overlap time steps from two model, must exist simultaneously in a period of time. In other words, they overlap on T-axis direction. Figure 6.6 is an example of overlapping time steps.

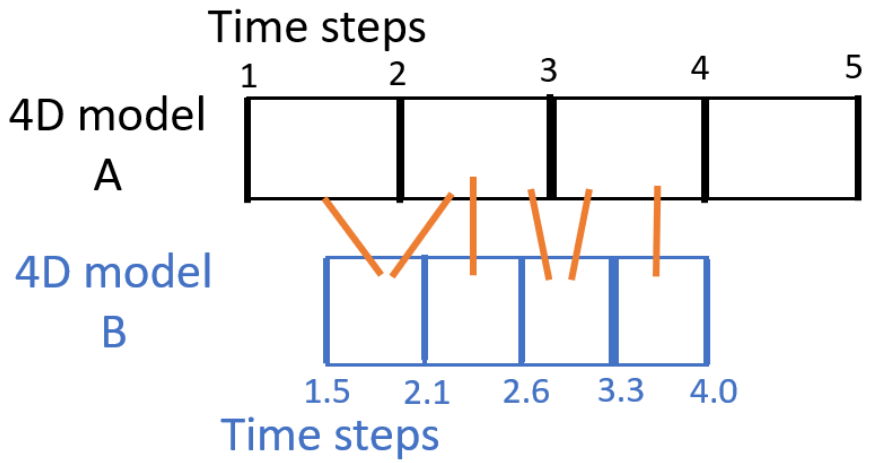


Fig. 6.6 A example of overlapping time steps

This kind of process is suitable for embarrassingly parallel. A comparing process is arranged to one thread. Only tens of thousands of threads are required to compare all time steps simultaneously. The algorithm flow is show in Fig. 6.7.

The second preprocess is a typical AABB-based method. Two tetrahedrons from two overlapping time step respectively must overlap on T-axis. Therefore, the second preprocess construct a 3D AABB to judge if these two tetrahedrons may intersect or not. Actually, the first and second preprocess is a complete 4D AABB method. But, the judgement of T-axis direction is faster by using feature of Otomo's method.

When we limit tetrahedrons number by time step. The number of tetrahedrons pairs can be handle by the capacity of GPU. Each tetrahedrons pairs is arranged to a thread. The flow of second preprocess is show in fig. 6.8.

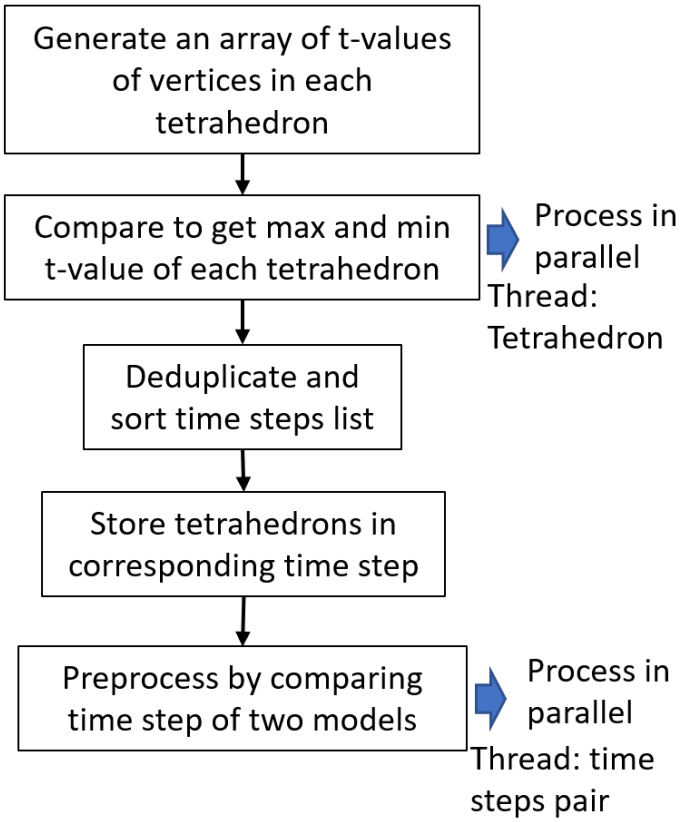


Fig. 6.7 Flow of time step overlapping checking

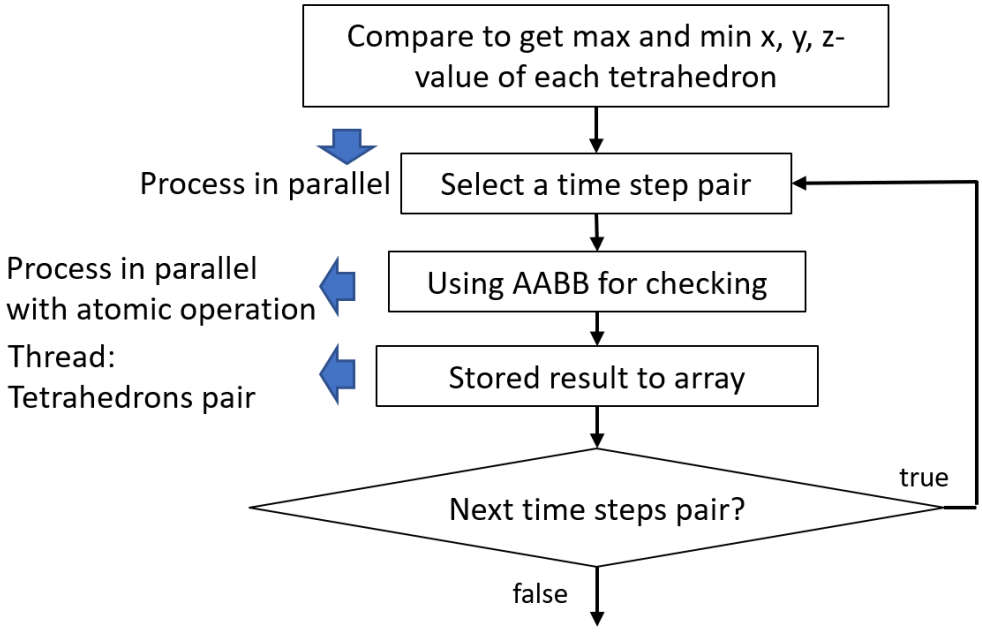


Fig. 6.8 Flow of AABB in 3D space

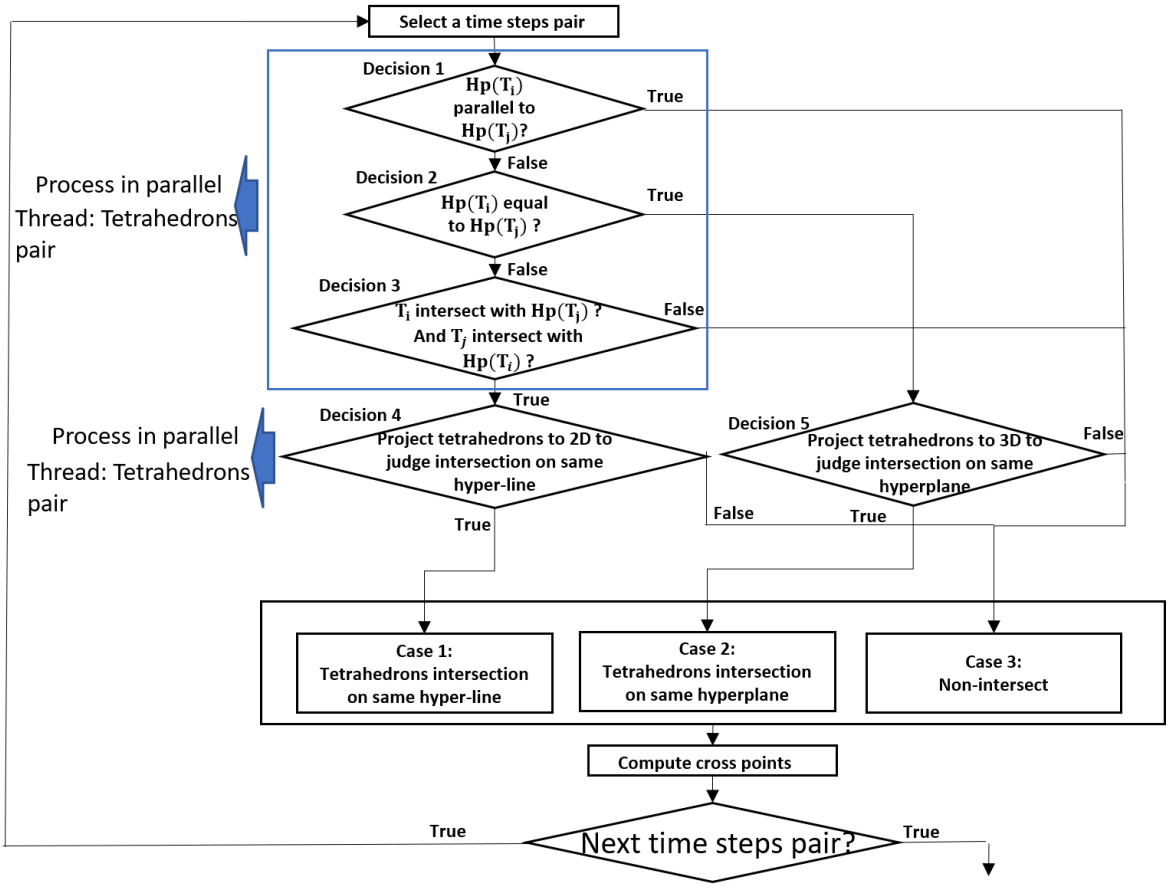


Fig. 6.9 Parallel flow of boundary intersection checking

Same to the second preprocess, The strategy of loop all time step pairs and process tetrahedrons pairs by parallel is also use in the main algorithm of boundary judgement intersection, like flow in fig. 6.9.

Theoretically, all parallel processing within the loop can be turned into child threads in dynamic parallel. The reason for not doing this is that the dynamic parallel strategy still requires launching too many threads at the same time.

This two-step preprocessing and parallel main program can effectively reduce the processing time. For the two models with millions of tetrahedrons mentioned above, the processing time of boundary intersection judgment will be reduced from dozens of hours to less than 15 minutes. This will reduce the time by more than 98%.

6.3 Structure of an intersected tetrahedrons

6.3.1 Crossed sections in intersected tetrahedron

As a tetrahedron in a 4D mesh model, when considering the boundary intersection of the 4D mesh model, this tetrahedron may intersect with multiple tetrahedrons from another 4D mesh model, such as fig.7.10. And because the 4D mesh model is a closed shape, each tetrahedron must have neighbors on any face. In other words, any face of a tetrahedron in the 4D mesh model must be completely shared with another tetrahedron. Therefore, a crossed section must have neighbors, and its edges must be shared by another crossed section. Therefore, a crossed section must have neighbors, and its edges must be shared by another crossed section. This other crossed section may be formed by the intersection of two other tetrahedrons, or by the intersection of the original tetrahedron and another tetrahedron. For the case where the original tetrahedron and another tetrahedron intersect, we can group this crossed section with the previous crossed section. This group is generally composed of multiple crossed sections.

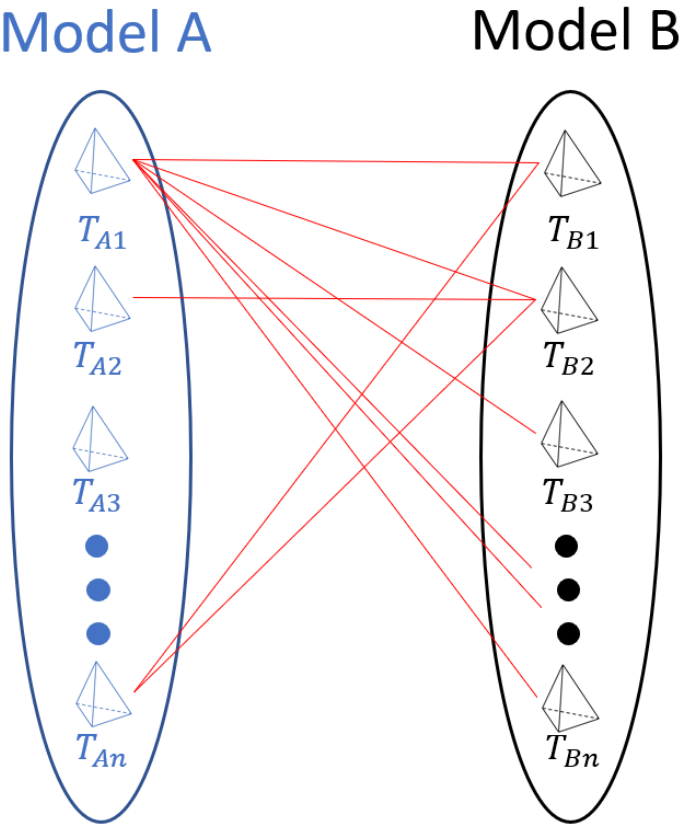


Fig. 7.10 Illustration of tetrahedrons intersection relation

These crossed sections and the tetrahedron in which they are grouped are understood as a basic unit in this study. There will be many such units in the intersection of 4D mesh models, and each unit is completely independent and different. There may be exactly the same cross section between two units, because this cross section is formed by the intersection of the tetrahedrons of these two units. However, the tetrahedron and crossed sections of any unit are completely unique. A unit is illustrated in fig. 6.11.

All the above discussions can also be based on the situation of 3D mesh model intersection and extended from there.

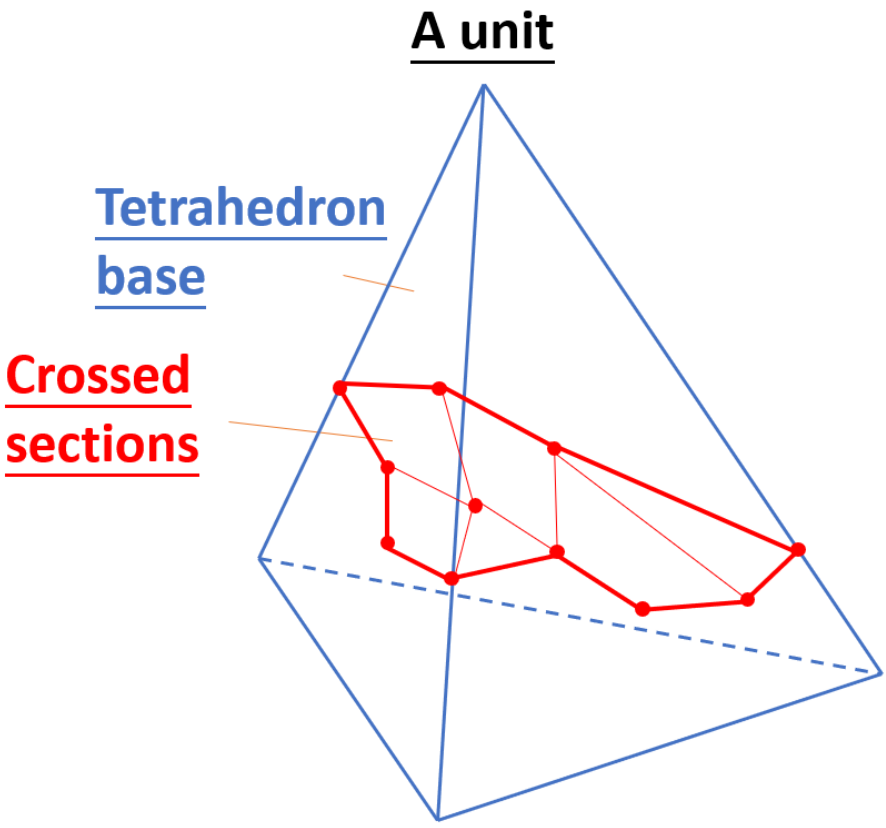


Fig. 6.11 Illustration of a unit of tetrahedron base with crossed sections

6.3.2 Subdivided crossed sections by Delaunay triangulation

Crossed section is a very important concept in this part of study. The situation of a crossed section is determined by the cross section of the two tetrahedrons on the 2D hyper-line. (Please strictly distinguish between crossed section and cross section. The cross section refers to the pattern of the tetrahedron on the 2D hyper-

line. The crossed section refers to the overlapping part of the cross sections of the two tetrahedrons.) Obviously, the cross section of a tetrahedron on a 2D hyper-line may have two patterns, triangle and quadrilateral. Therefore, the shape of the cross section may range from triangle to hexagon.

The many patterns of crossed sections will undoubtedly increase the complexity of the calculation. In order to simplify the subdivision process in Section 6.4, the author introduced Delaunay triangulation to triangulate the crossed sections.

In computational geometry, a Delaunay triangulation of a set of points in the plane divides their convex hull [80] into triangles such that none of the points lies inside the circumcircle of any triangle. This maximizes the size of the smallest angle in any of the triangles and tends to avoid creating sliver triangles.

Both potential triangulations that divide the quadrangle into two triangles meet the "Delaunay condition", indicating that the circumcircles of all triangles have void interiors.

In the study, the author used the Bowyer-Watson algorithm [81] to triangulate the vertices of crossed section. The step of algorithm is below:

Initial Triangulation: Start with a super-triangle that contains all the points to be inserted. This super-triangle is usually artificially constructed and large enough to encompass all the points.

Point Insertion: Insert each point one by one into the triangulation. For each point to be inserted, find all the triangles whose circumcircles contain the point. The circumcircle of a triangle is the circle that passes through all three vertices of the triangle.

Remove Affected Triangles: Remove all triangles whose circumcircles contain the inserted point. After removing these triangles, a polygonal hole is formed by the edges of the removed triangles.

Re-triangulate: Connect the inserted point to the vertices of the polygonal hole, forming new triangles. This ensures that all triangles formed after inserting the new point satisfy the Delaunay property.

Repeat the Above Steps: Repeat the insertion process for each point until all points have been processed.

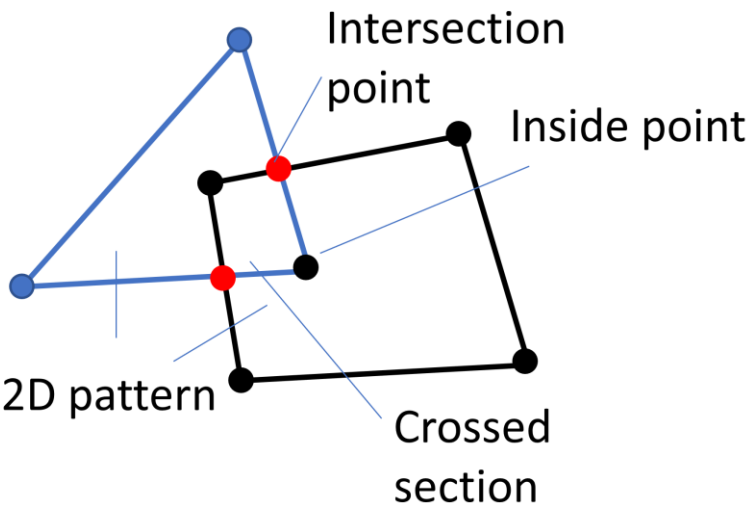


Fig. 6.12 Illustration of vertices of crossed sections

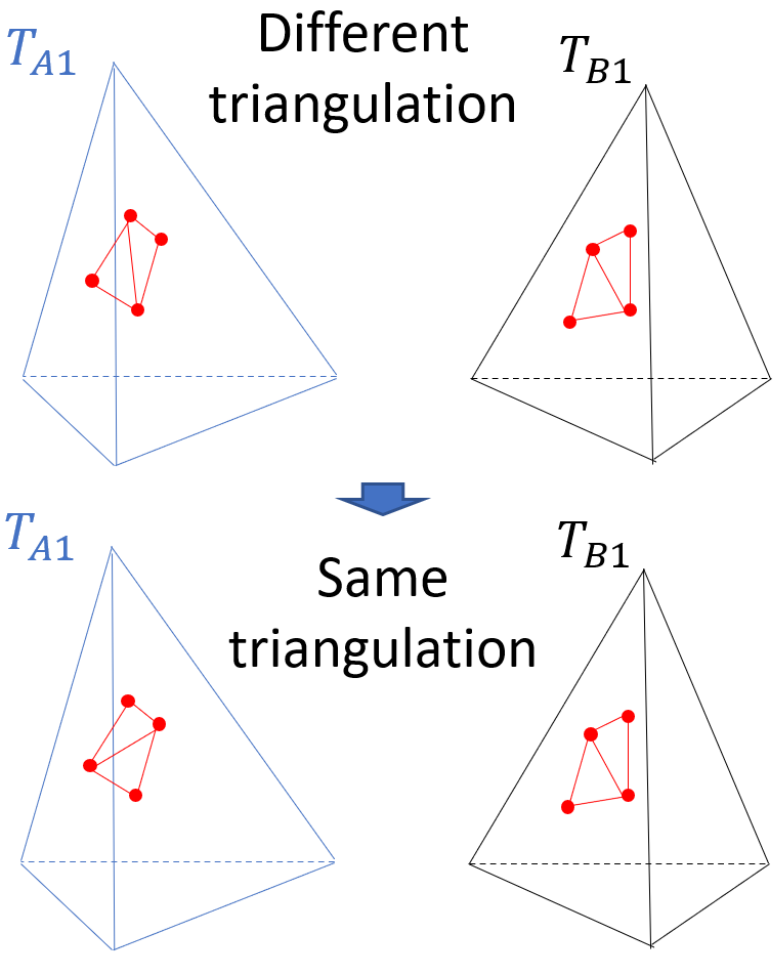


Fig. 6.13 A crossed section triangulation shared by two intersecting tetrahedrons

Remove the Super-triangle: Finally, remove all edges and vertices of the initial super-triangle from the triangulation, leaving only the triangulation of the actual data points.

The Bowyer-Watson algorithm has an average-case time complexity of O(n log n), where n is the number of points to be inserted. In the worst case, for very specific point configurations, the time complexity can be $O(n^2)$.

There are other two advantages to use Delaunay triangulation by Bowyer-Watson algorithm.

The first advantage is this method is used for triangulate a set of 2D points. There are two kinds of vertices in a crossed section. One kind of vertices is intersecting vertices. They are intersecting points of two 2D pattern of cross sections of two tetrahedron on the hyper-line. Another kind of vertices is the vertices of pattern inside of another pattern. These two kinds of vertices can bound a 2D convex hull. Since we don't know the sequence of these vertices, bound them to a 2D convex hull is difficult. The Delaunay triangulation by Bowyer-Watson algorithm is a good method for this case, cause it only require set of 2D points. The illustration is fig.6.12.

The Delaunay triangulation produced by the Bowyer-Watson algorithm is not always unique. The uniqueness of the Delaunay triangulation depends on the specific arrangement and placement of the input points. In some cases, particularly when the input point set has cocircular points (i.e., four or more points that are cocircular) or colinear points (i.e., three or more points that are colinear), there may be multiple legal Delaunay triangulations. But the vertices of crossed section rarely cocircular and colinear, this feature become the second advantage.

As we all know, the crossed section is shared by two tetrahedrons from different 4D mesh models, such as fig. 6.13. According to Matsunaga's paper, the subdivision result of this shared crossed section should same for guarantee the correctness of set operation. The Delaunay triangulation is same in the shared crossed sections just right.

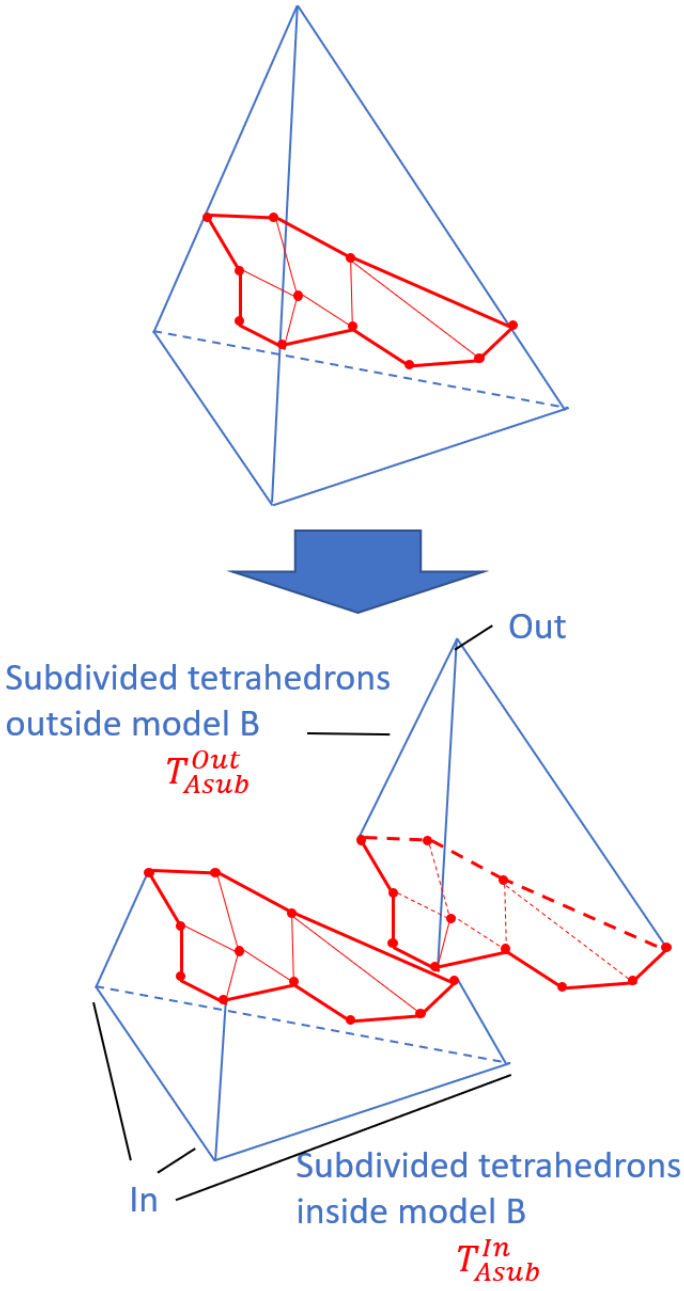


Fig. 6.14 Crossed sections cut tetrahedron base

6.3.3 In or out judgement of intersected tetrahedron's vertices

In a unit, the tetrahedron base must be completely divided into two or more independent regions by the crossed section. Let's turn our perspective to the macroscopic view. When two 4D mesh models intersect, the tetrahedron at the intersection will be cut off by the boundary of the other mesh model. Just like the

2D polygon of a 3D mesh model is cut off by 1D line segments, the 3D tetrahedron is also cut off by the 2D crossed section. So the segmented regions belong to tetrahedron are either inside another 4D mesh model or outside it. Fig. 6.14

In this study, the author uses in and out of vertices to represent the in and out of the region which they belong. Vertices of tetrahedron belong to different regions can be given "in" and "out" labels, obviously. The process of computing signed distances from vertices of tetrahedron to hyper-line which mention in 6.2.1 can also be used to give “in” and “out” label to vertices. “Out” corresponds to the positive, “In” corresponds to the negative.

6.3.4 Data organization and transfer

Until section 6.3.3, all the data and preprocess required for the set operation process has been prepared. Now the author will organize and store data for individual set operation steps.

Data is organize by array and store to a binary file. the author names the file .ztb file (Zonal tetrahedrons binary file). It saves the following data:

a) Tetrahedrons index of two 4D mesh models which intersection in a same 3D hyperplane. They are organized by pair of intersection.

b) Tetrahedrons index of two 4D mesh models which intersection on a 2D hyper-line. They are organized by pair of intersection.

c) Vertices coordinate of crossed sections belong to tetrahedrons base.

d) Triangulation of crossed sections belong to tetrahedrons base.

e) Vertices “in” and “out” label list of tetrahedrons base.

6.4 Classify the inside or outside original tetrahedrons

6.4.1 Dot product method

The inside or outside judgment original tetrahedrons is an independent process in 4D mesh models set operation process. When we remove the intersecting tetrahedrons, the remaining tetrahedrons are either inside or outside another 4D mesh model. The frame to execute inside or outside classify is a loop structure. All non-intersected tetrahedrons of a 4D mesh model are traversed to process another 4D model and judge.

In this study, three strategies to determine a original tetrahedron is inside and outside.

The first strategy is dot product method. It is a simple method to judge if a point is inside a shape or not. This method illustrates in fig. 6.15. Because all intersecting tetrahedrons are confirmed, these non-intersecting tetrahedrons must inside or outside another 4D mesh model entirely. Inside or outside relation of an arbitrary point of non-intersecting tetrahedron can represent the tetrahedron is inside or outside.

Traversal all tetrahedrons of another tetrahedron, and select an arbitrary point to compute the dot product of normal vector and the vector from a tetrahedron to this point, like eq(6.5)..

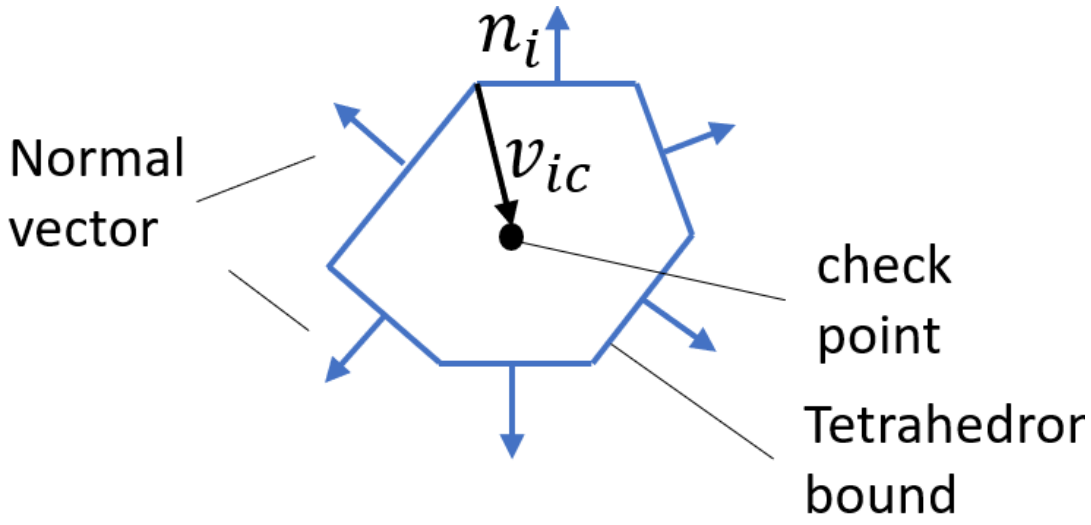


Fig. 6.15 Dot product method

$$n_i \cdot v_{ic} < 0 \quad Eq(6.5)$$

n_i is the normal vector of tetrahedron.

v_{ic} is vector from tetrahedron to selected point.

If all dot products are negative, the selected point is inside of another 4D mesh model. That indicate the selected tetrahedron is inside of another 4D mesh model. Otherwise, it means that the tetrahedron is outside another 4D mesh model.

This is a simple and efficient method. But its disadvantage is also obvious. It requires the enclosed 4D mesh model to be a convex hull, so it can only be applied to certain situations.

6.4.2 Adjacent label method

The second method is adjacent label method. It is an easy-to-understand method. For an inside tetrahedron, all its neighbors are either inside tetrahedrons or intersection tetrahedrons. Therefore, we only need to find a tetrahedron that is definitely inside or outside and gradually extend the inside or outside label until it touches the intersecting tetrahedron. Then we can find all inside or outside original tetrahedrons by traversing. This process is illustrated in fig. 6.16.

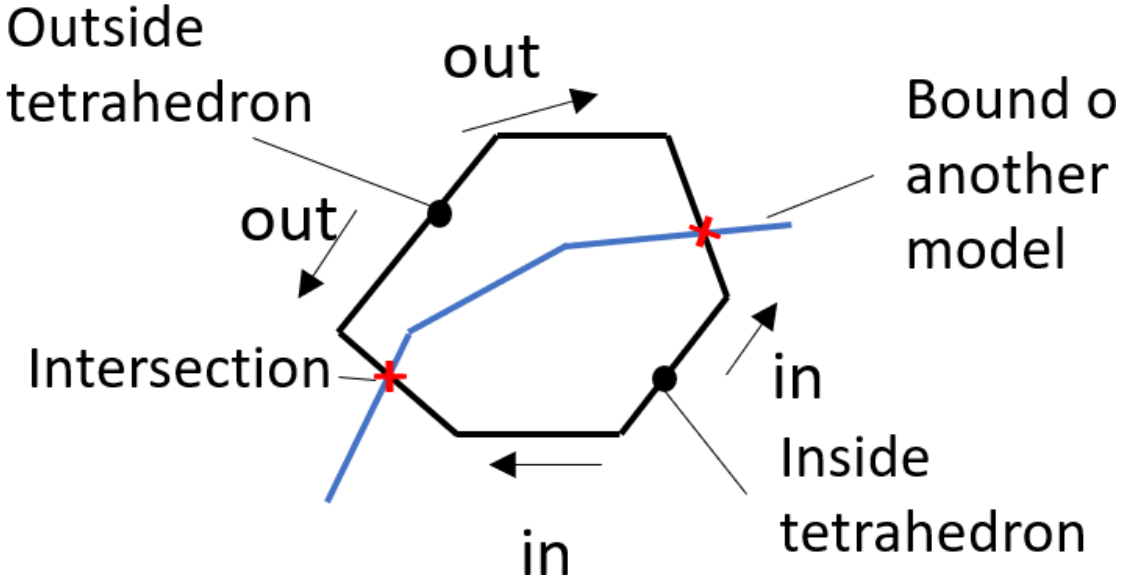


Fig. 6.16 Adjacent label method

This method has two fatal disadvantages. The first shortcoming is that it can only judge the intersection of two convex 4D mesh models. When there is a concave 4D mesh model, there may be multiple inside and outside areas separated by intersecting tetrahedrons. In this case, each area must have a starting point tetrahedron that determines the inside and outside. The second disadvantage is that it requires the input 4D mesh model data and boundary intersection data to be

extremely accurate. If even one intersecting tetrahedron is misjudged, it will have a great impact on the results of this method. For complex 4D mesh models, the correctness of the boundary intersection judgement cannot be perfectly guaranteed.

6.4.3 Ray method

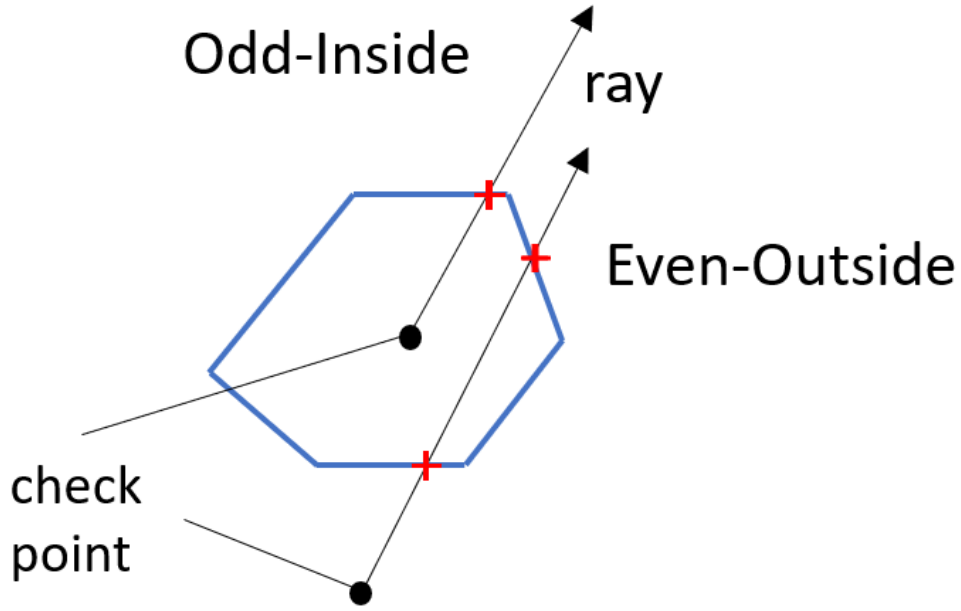


Fig. 6.17 Ray method

Ray method is the third method, and it is also the main method used in the author's program. Although it requires a huge amount of calculation, its accuracy is the highest. This method also requires selecting a point on the tetrahedron to be measured. A ray is emitted from this point to arbitrary direction. If the ray intersects the surface of another 4D mesh model at an odd number of points, it means that the tetrahedron is inside the other 4D mesh model, and vice versa. This process illustrates in fig. 6.17.

The time complexity of this method is also $O(n^2)$, due to the huge amount of data in 4D space. Even if the author assigns each selected point to a thread for parallel processing, it will take nearly 15 minutes to process a trillion times computation.

6.5 Subdivided the inside of intersected tetrahedron

by advancing front technique

6.5.1 Process outline

The AFT algorithm is a 2D point group triangulation algorithm like the Delaunay algorithm. It can also be upgraded in dimension like the Delaunay algorithm. In the author's algorithm, the author will use it to tetrahedralize the interior of a tetrahedron. The reason for choosing the AFT algorithm is that the AFT algorithm gradually generates new tetrahedrons, and a new tetrahedron must depend on a generated triangle (a face of a tetrahedron), so the new tetrahedron must inherit some characteristics of the source tetrahedron.

What the new tetrahedron needs to inherit is the inner and outer division of the source tetrahedron. According to section 6.3.4, we know that the input is the in and out labels of the vertices of the tetrahedron base in a unit. We need to convert it into the in and out labels of the newly generated tetrahedron. The AFT algorithm can effectively transfer this label. If the source tetrahedron has in and out labels, the newly generated tetrahedron must have the same label.

The basic of 2D AFT algorithms as follow[79]:

Initialize the Boundary: Start with the defined geometric domain and initialize the boundary of the domain. These boundaries consist of a series of edges (line segments), referred to as the front.

Select a Boundary Edge: Select an edge from the front. This edge will be used to generate a new triangle.

Find a New Point: Find a point opposite to the selected edge.

Form a New Triangle: Connect the point to the two endpoints of the selected edge to form a new triangle.

Update the Front: Remove the edge that has now become an internal edge of the triangle from the front and add the new edges of the triangle to the front.

Handle Intersections and Overlaps: Check and handle any intersections or overlaps between the new triangle and other parts of the front.

Repeat the Above Steps: Repeat the process of selecting a boundary edge, generating a new point, forming a new triangle, and updating the front until the entire domain is triangulated.

Complete the Triangulation: When there are no remaining edges in the front, the

triangulation of the entire domain is complete.

This process is illustrated in fig. 6.18. Front is shown by blue segments. They are stored in a list. It contains some initial segments and new generated segments which are not same to other front. Non front is shown by orange segments. When new generated segment same to front segment, the front segment will be deleted from front list.

Applying this method to a unit is illustrated in fig. 6.19. Because of tetrahedron is a 3D simplex, this method is a 3D AFT, but modified specifically to the study. The Initial fronts are crossed sections. New tetrahedrons extend from these crossed sections. There are two advancing directions, one is inside direction and the other is outside directions. New tetrahedrons generate along outside and inside directions are given a same label. Points that can find to connect to form a new tetrahedron are all vertices from crossed section and tetrahedron base. That means this method does not add new points to guarantee the quality of meshes, but easy. All new generate tetrahedron only within the tetrahedron base.

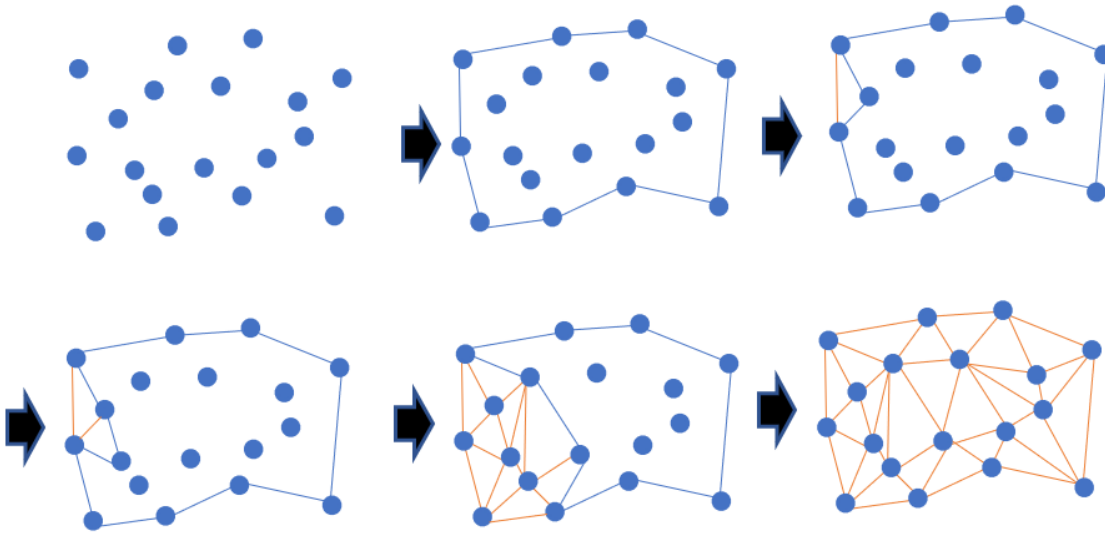


Fig. 6.18 2D analogy of AFT

6.5.2 Preprocess of AFT

There are several preprocess to start the AFT algorithm.

The first preprocess is mapping. According to section 3.1, a tetrahedron is a 3D simplex and exist in 3D space. Therefore, when we find the 3D hyperplane where it exists and use it as a 3D space, we can completely map a unit into this space, thus converting 4D into 3D. After this process, all 3D processing can be applied to the unit. The second preprocess is finding “on face vertices”. “On face vertices” refers to vertices of crossed section that position at the face of tetrahedron base. Because crossed sections cutting the entire tetrahedron base, find the contacting position of crossed sections and face of tetrahedron base is important. Then is the screening process. The unit that only a very small, crossed section in a tetrahedron is skipped to process. Most of part of the unit will be inside or outside another 4D mesh model. Ignoring these units is a strategy that reduces accuracy but increases calculation speed. Meanwhile, some error cases can be skipped. Skipped cases are shown in fig. 6.20.

Recording crossed sections belong to same 2D plane. A crossed section will not connect to any point on the same 2D plane as it to form a new tetrahedron.

Delaunay triangulation is performed again on some crossed sections. Because some vertices of crossed sections exist on the edges of other crossed sections, in order to ensure the smooth progress of AFT, we require that all points that can be connected should be vertices of its adjacent fronts. The process illustrates in fig. 6.21.

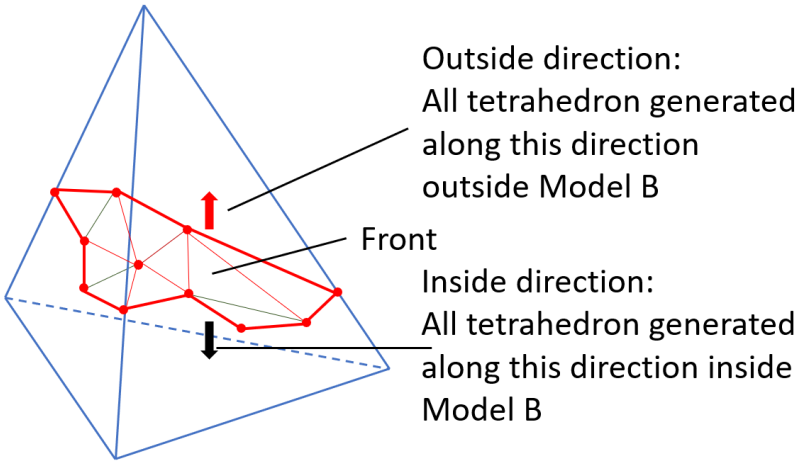
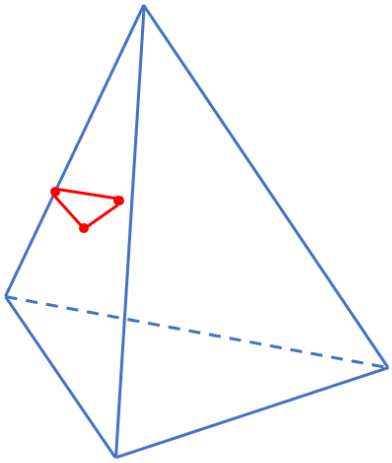
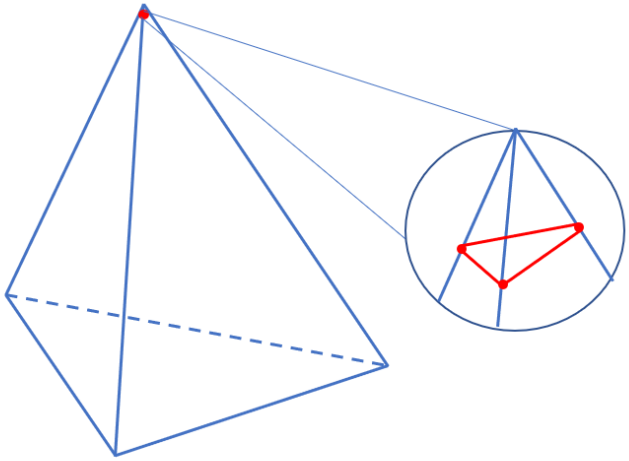


Fig. 6.19 AFT in a unit



(a) Slight intersection



(b) Only a very small crossed section

Fig. 6.20 Skipped cases in preprocess

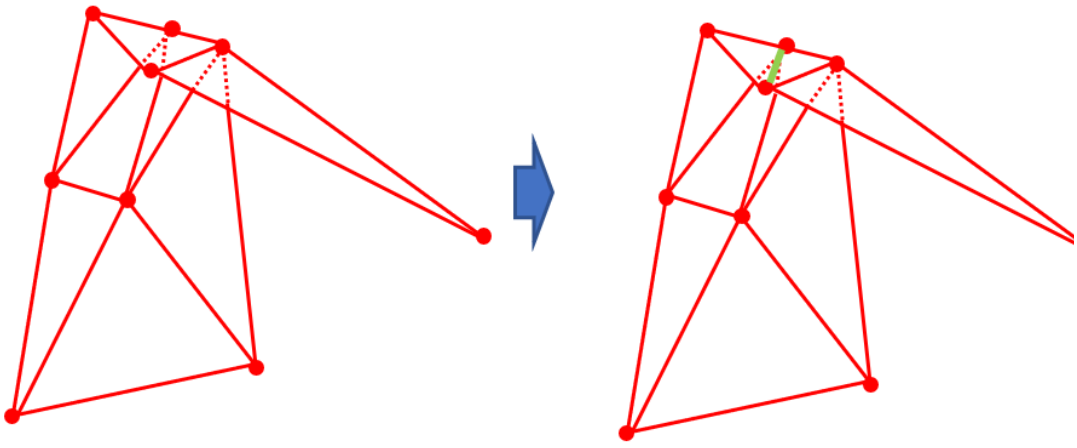


Fig. 6.21 Process Delaunay triangulation again

6.5.3 Give in or out label by different advancing direction

Distinguishing the inside and outside of the advancing direction is the core of

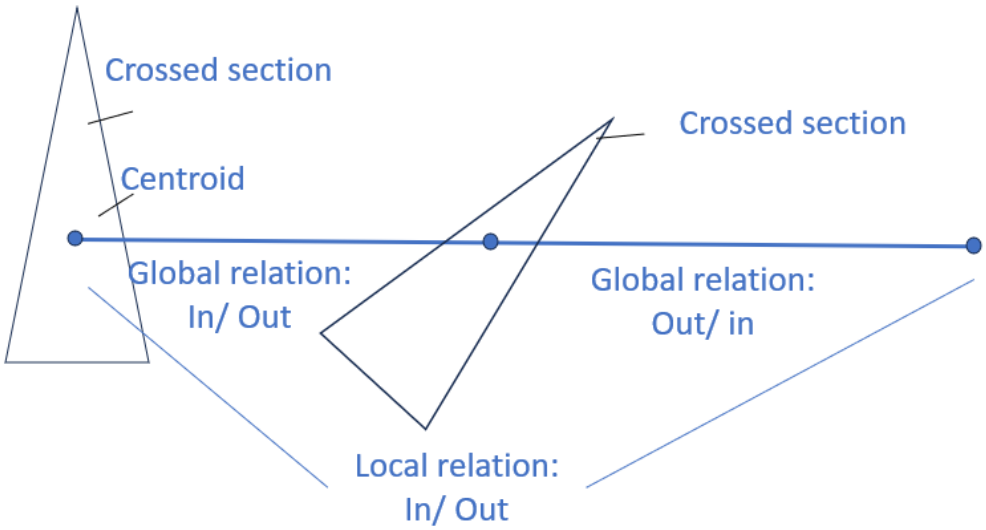


Fig. 6.22 Label achievement method

this modified AFT algorithm, and it is also the reason why the AFT algorithm is chosen for subdivision. That is, of the two sides of the crossed section, which one faces inward, and which faces outward. Although a 2D face does not have a direction in 4D space, here, a crossed section is given a normal vector. The normal vector of this crossed section is defined in the current unit and always points outward. According to section 6.3.4, all inside and outside information is bound to the vertices of the tetrahedron base. This information will reflect whether a vertex in a unit is on inside or outside direction of a crossed section. This is a local information, so we need to convert it into global information in a unit.

After trying many methods, the author adopted a global vertex inside-outside classified method based on the ray method that mention in 6.4.3. Connect lines from the four vertices to the centroid of arbitrary crossed section. The lines may pass through any number of crossed sections. Start recording from the crossed section which the centroid belongs. Starting from the inside and outside relation of the corresponding vertex and this crossed section. Every time it passes through a crossed section, the in and out labels change. In other words, if the segment through an odd number of crossed sections, the in and out labels will be opposite, if through an even number of crossed sections, the in and out labels will be the same. For a

point on such a line segment, each time it passes through a crossed section, its inside-outside relationship will change. This process illustrates in fig. 6.22 and fig. 6.23.

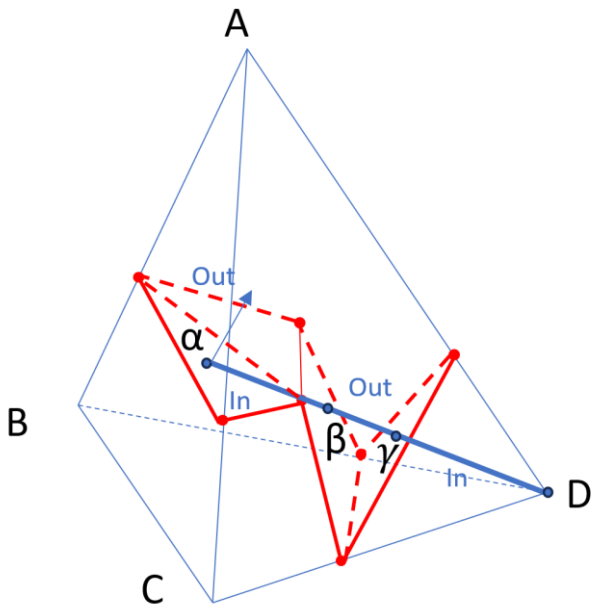


Fig. 6.23 An example of segments to judge in/out

6.5.4 Screen process of the fourth vertex for an advancing front

After confirm the initial front and the advancing direction, the main loop of the AFT can be executed. Each time of the loop execute finding the fourth vertex, generating new tetrahedron and confirm new advancing front.

This loop is shown as follows:

a) Screen 4th vertex from points of other fronts

i. Skip vertex in same 2D plane of advancing front

ii. Skip vertex which generated tetrahedrons has a dead face (a face that used two times, it already shared by two tetrahedrons)

iii. Skip vertex which generated tetrahedrons has a face is a triangle but not a confirmed front.

iv. Intersection test (with generated tetrahedrons)

v. Skip vertex on generated tetrahedrons face.

vi. Select the closest vertex.

b) Screen 4th vertex from 4 vertices of original tetrahedron

i. Skip vertex which generated tetrahedrons has a dead face (a face that used two times, it already shared by two tetrahedrons)

ii. Intersection test (with generated tetrahedrons)

iii. Skip vertex on generated tetrahedrons face.

c) Generate new tetrahedrons and new fronts.

d) Go to next front.

In a unit the loop process is executed two times, first time for outside direction, second time for inside direction. In other words, the main body of AFT is activated two times for different directions. A 2D example is illustrated in Fig. 6.24.

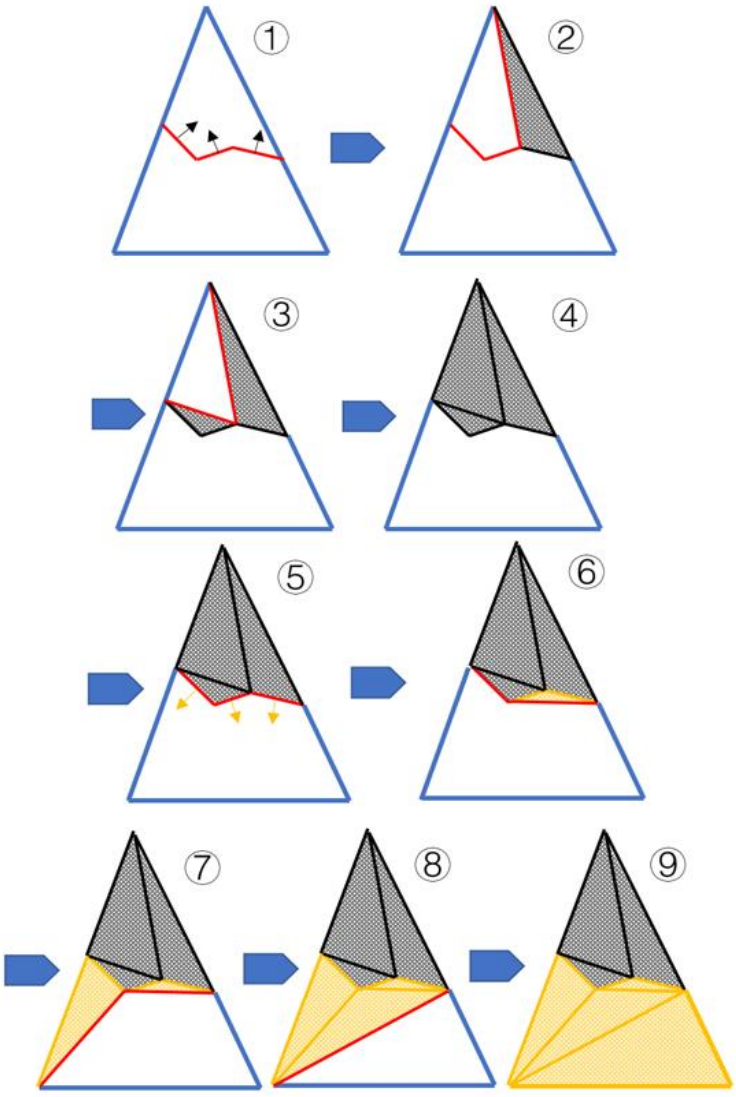


Fig. 6.24 An 2D analogy of modified AFT process in the study.

6.6 Result and discussion

Currently, there are still some bugs and problems with set operation algorithms for 4D mesh models with a large number of tetrahedra in spatio-temporal space. Generating TAV requires several tens of iterations of set operations, which currently takes several thousand hours to compute. Therefore, the author realizes the process of data in Fig. 5.27 and present the set operation results of models in fig. 6.25, which is the set operation result of subtraction between Time-invariant WOR and TOR, used to demonstrate the feasibility of the algorithm to represent CWE process, like fig.21. In the Fig.22, we can observe the intersection of cutter and workpiece from t=3s, 5s and x=0.1mm slices.

The TOR in these examples has 580,607 tetrahedrons and lasted 6.2 seconds by 72-timesteps. The cutter has two blades and 5mm radius. Feed speed is 0.1mm/s and rotation speed is 1.75 rad/s. In addition to the T-map method, which uses voxel model, the computed Time-invariant WOR has 2,907,071 tetrahedrons. The size of workpiece is width(X) ×length (Y) ×height (Z): 5mm×3mm×0.6mm. The estimated depth of cut is 0.4mm in this process. The configuration of computer is Win10, Intel Core i9-10900K CPU 32 GB memory and NVIDIA GeForce RTX 3080 GPU with 10 GB memory.

The generic method tries to achieve the TAV by executing set operation iteratively. It has high accuracy but need a very large computational load. Even use parallel processing, the transformation from a TOR to TAV that require tens of hours to complete the iterations.

Therefore, the data explosion of 4D models is an urgent problem that needs to be solved in current research. Due to the huge amount of calculation, we currently do not conduct a complete CWE analysis. But in this paper, we replace the subtraction process of Time-invariant WOR and TAV by the subtraction process of Time-invariant WOR and TOR. The XYZ-hyperplane slice of result in figure 22 can show the workpiece being cut at the corresponding time, but it cannot express the shape of the workpiece after the material is continuously removed. This simplified results in figure 22 can prove the correctness and validity of the Set operation method proposed in the paper. However, due to the lack of more complex results, it lacks convincingness in verifying the CWE analysis method proposed in the paper.

In addition, generating TAV by T-map is an easy solution of above problem. The

process has two time-consuming steps, T-map calculation and marching cube algorithm [17, 18]. These two steps only need several minutes to construct a TAV for the TOR. Although it is very fast, the resolution and accuracy of the generated model is lower than set operation's result.

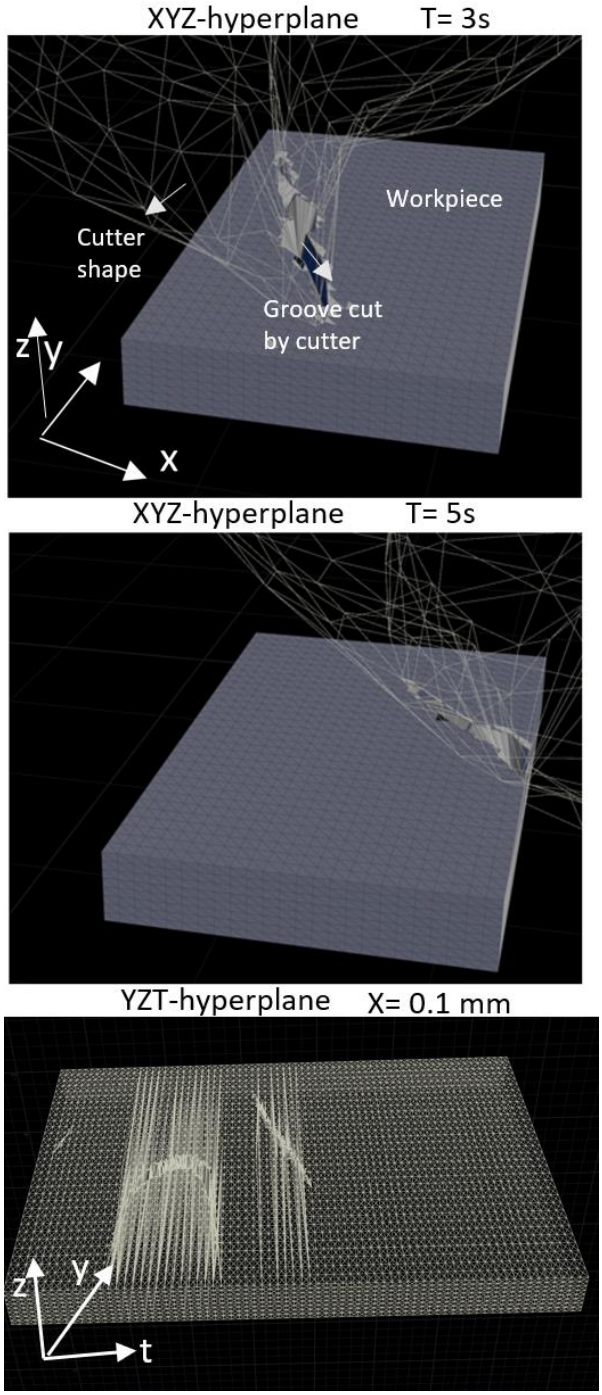


Fig. 6.25 Result of Preliminary Set operation, Time-invariant WOR-TOR

6.7 Set operation representation after hyperplane slicing

6.7.1 Process outline

At the end of 2023, the author tried to develop a method for set operation representation after hyperplane slicing. It tries to directly represent the set operation results of 4D mesh models. This method is based on the hyperplane slicing represent program the author developed in Houdini. Currently it can represent the subdivided 3D hyperplane slices of the inside and outside parts of the intersecting tetrahedron in Section 6.5.3.

Theoretically, it is not the real result of a 4D set operation, but a method that tries to display only the 3D slice of result of the set operation in corresponding the 3D hyperplane. The comparison of outline of two methods is illustrated in fig. 6.26.

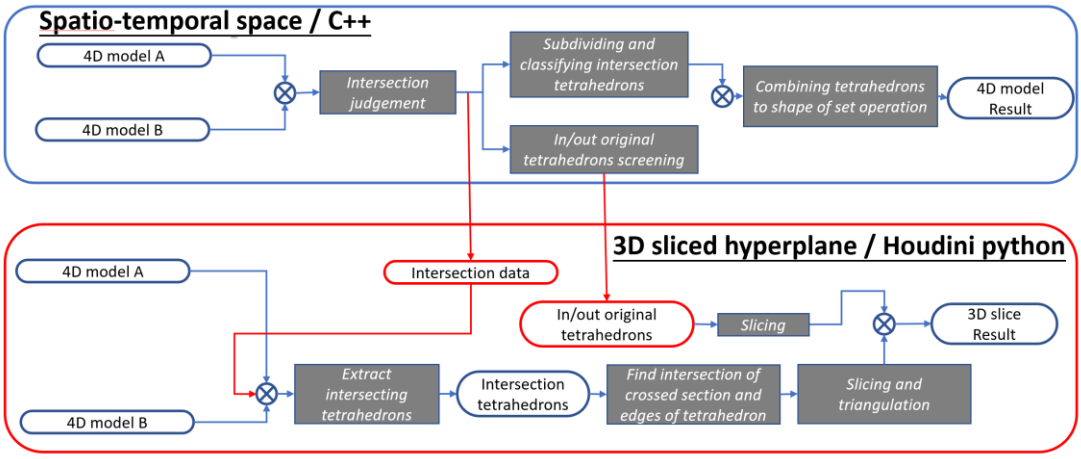
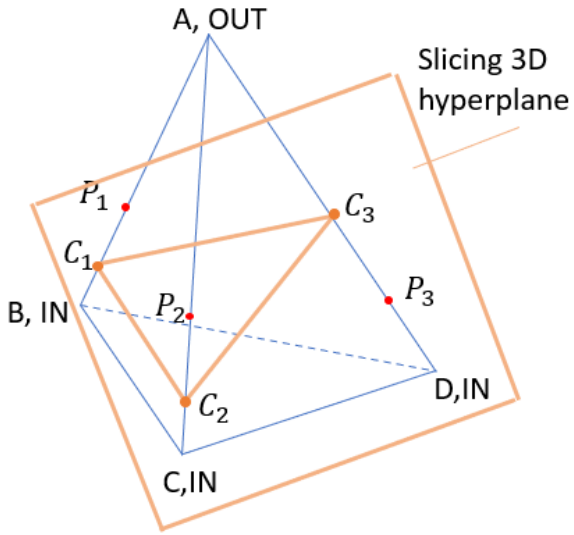


Fig. 6.26 The comparison of outline of two methods

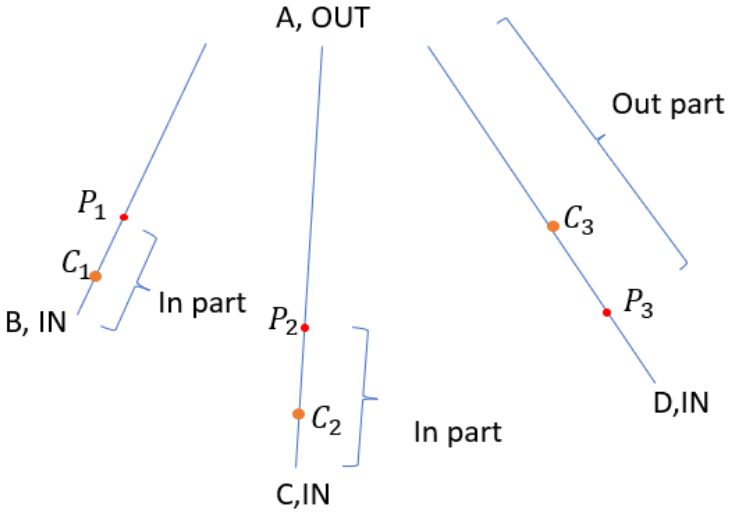
6.7.2 In or out judgement of sliced polygon's vertices

According to rule the section 7.5.3, edges of the tetrahedron base can be regards the judging segments. When we know the global inside and outside conditions of the vertex on the edge, as well as the intersection of the crossed section and the edge, we can determine the inside and outside conditions of the intersection of the 3D hyperplane slice and the edge, that is, the inside and outside conditions of the polygon vertex. This process is shown in fig. 6.27. p_1, p_2, p_3 are intersection points between crossed sections and edges. c_1, c_2, c_3 are vertices of polygon which is the

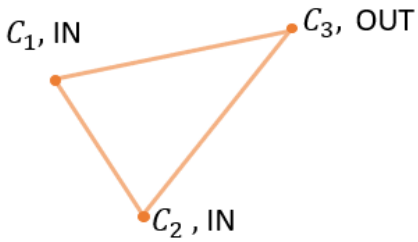
slice pattern on the hyperplane.



(a) Polygon in the slicing



(b) In/Out situation in each edge

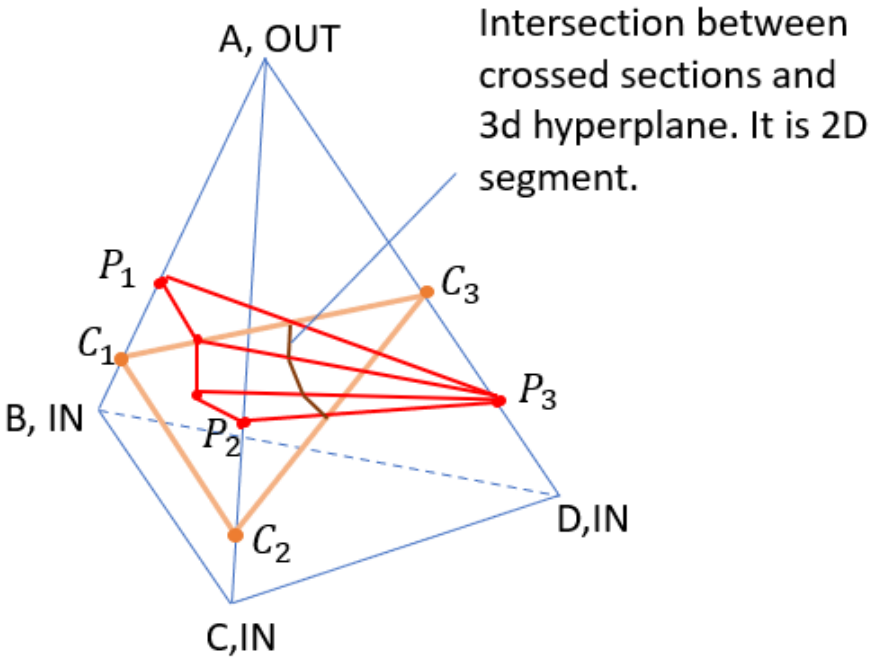


(c) Result of in/out polygon vertices

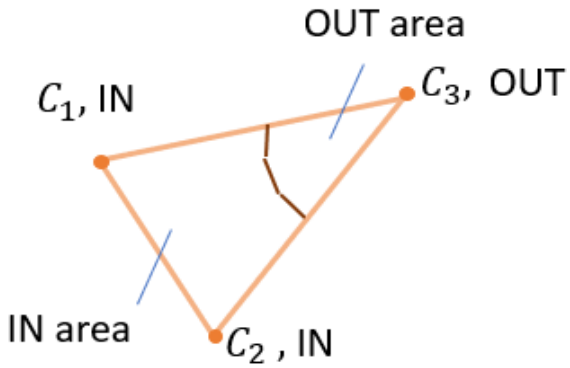
Fig. 6.27 In/Out judgement of sliced polygon's vertices

After that, find the intersection segments of the crossed section and the sliced polygon, and partition the sliced polygon according to it to get the inside and outside areas, such as fig.6.28.

There are two 600-cells in 4D space, radius are 2.0, centers are (0,0,0,0) and (1.5,0,0,0). fig. 6.29 is the projection. fig. 6.30 shows the inside and outside part of intersecting tetrahedrons at t=1s.



(a) Intersection between polygon and crossed section



(b) In/Out area

Fig. 6.28 Area partition of sliced polygon

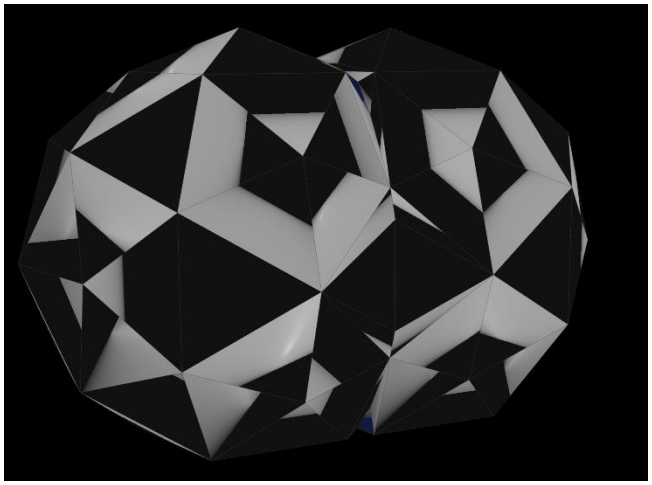
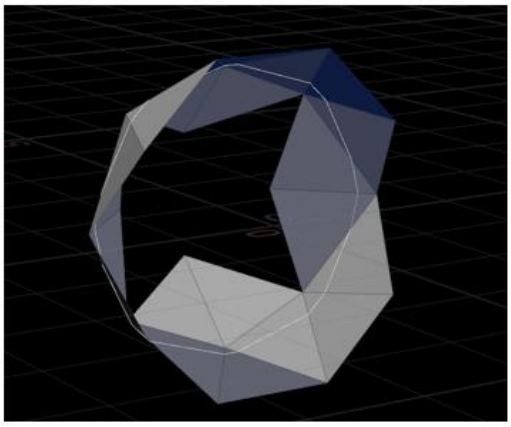
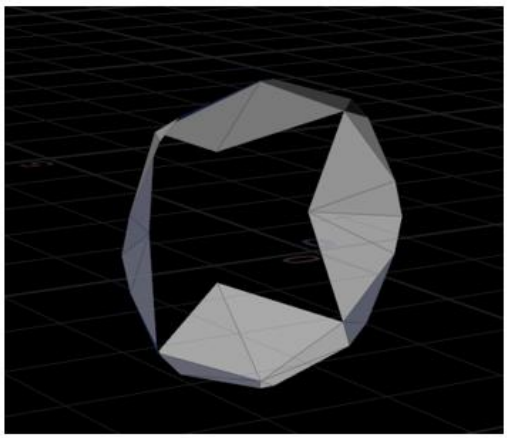
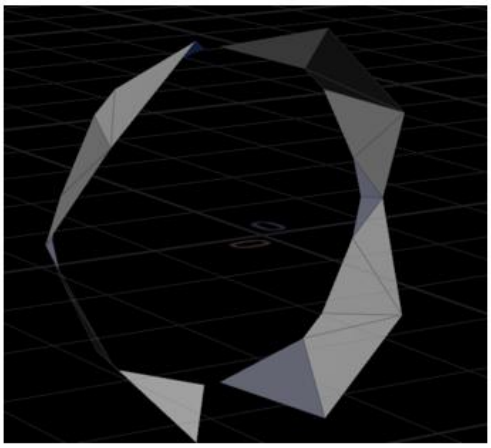


Fig. 6.29 Two 600-cell projection



(a) Intersecting tetrahedron slice





(b) Inside part (c) Outside part

Fig. 6.30 In/Out area slices of intersecting tetrahedron at t =1s

Chapter 7. Discussion and conclusion

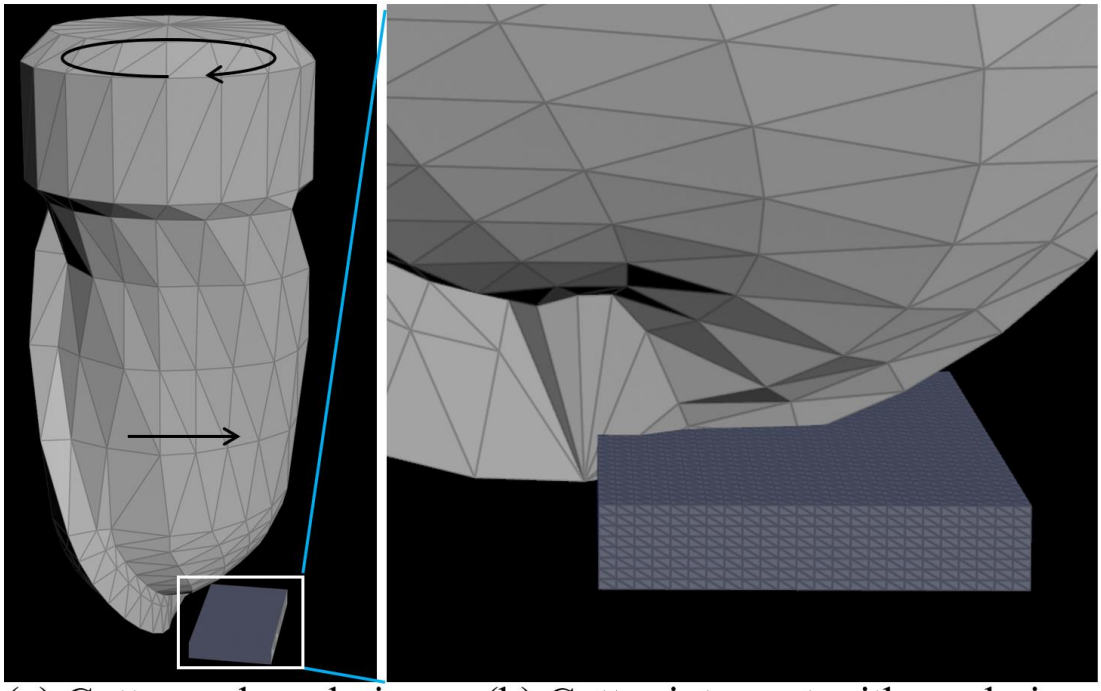
7.1 Quantitative analysis CWE process in spatio-temporal space

Table 7.1 lists the comparison of performance among these three methods and Figure 7.1 shows the input data in this study, parameter is same to fig 5.20 and 5.26. Meanwhile, GPU parallel processing is used to compute T-map structure. 20), 21), 22) The TOR in these three examples has 580,607 tetrahedrons and lasted 7.2 seconds by 72-timesteps. The cutter has two blades and 5mm radius. Feed speed is 0.1mm/s and rotation speed is 1.75 rad/s. In addition to the T-map method, which uses voxel model, the computed WOR has 2,907,071 tetrahedrons. The size of workpiece is width(X) ×length (Y) ×height (Z): 5mm×3mm×0.6mm. The estimated depth of cut is 0.4mm in this process. The configuration of computer is Win10, Intel Core i9-10900K CPU 32 GB memory and NIVDIA GeForce RTX 3080 GPU with 10 GB memory. The result illustrations are fig. 7.2 and 7.3 .

The method of using the T-map structure is the simplest of the three, but also the most limited. Because its accuracy depends on the resolution of voxels, higher accuracy means larger memory consumption. Since the T-map segments are straight lines, the T-map is only applicable to depict workpieces that are stationary in calculating process. Any movement of the workpiece needs to be transformed onto the motion trajectory of the tool, which increases the preparation time. Meanwhile, voxel model is suitable for parallel processing, that leads it can be executed very fast by GPGPU.

The 4D imprint method has the highest speed, but its accuracy is the lowest among these three methods. Compare to T-map method, the 4D imprint method is continuously in 3D space and no need to transfer motion of workpiece to the tool. The method does not generate new vertices, it only estimates the motion trajectories of points. Therefore, when there are not too many time steps, it still has good speed and memory consumption. These characteristics make the results of this method more suitable for user operation than for precise calculations. The preliminary set operation method has good generality for different 4D meshes and is continuous in all directions. However, this method requires significant computational resources.

Even obtaining the intermediate results shown in the paper still requires several hours of computation. When we consider cases which has more tetrahedrons such as WOR subtracts TAV, the time and memory consumption become even more significant. However, set operations are still the most accurate algorithm, suitable for reflecting the shape changes of the 4D model.



(a) Cutter and workpiece (b) Cutter intersect with workpiece

Fig. 7.1 Positions relation of cutter and workpiece

Table 7.1 Comparison among three methods

	Time	Memory	Quality
T-map (Resolution: 600x600x600)	Serial: 184 s Parallel: 3.13s	2.4 Gb	Higher resolution, higher quality
4D imprint	101s	785 Mb	Low
Set operation	211 min	1.5 Gb	High

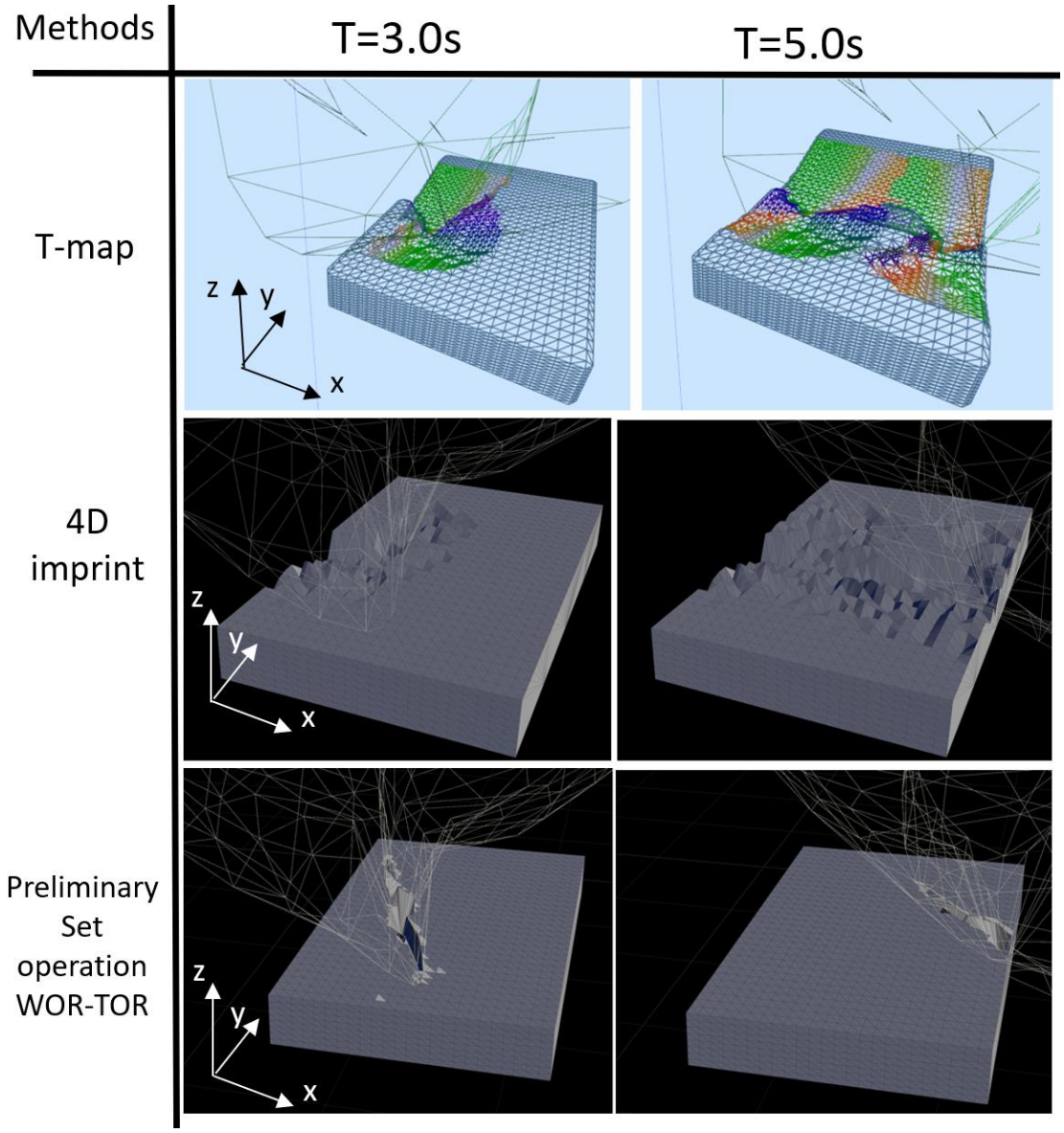


Fig. 7.2 Results of three preliminary methods, slices at t=3 and 5s

For 4D imprint and set operation method, the resolution of input 4D model is an important parameter of quality of CWE result. The resolution of 4D model which generated by Otomo's method is based on the time step. The shorter the time interval, the higher the resolution. However, in order to maintain the quality of the grid at high resolution, it is necessary to adjust the balance between spatial resolution and temporal resolution. In other words, the size of the three-dimensional grid needs to match the size of the fourth-dimensional grid. This often further increases the number of tetrahedrons and affects the efficiency of the calculation.

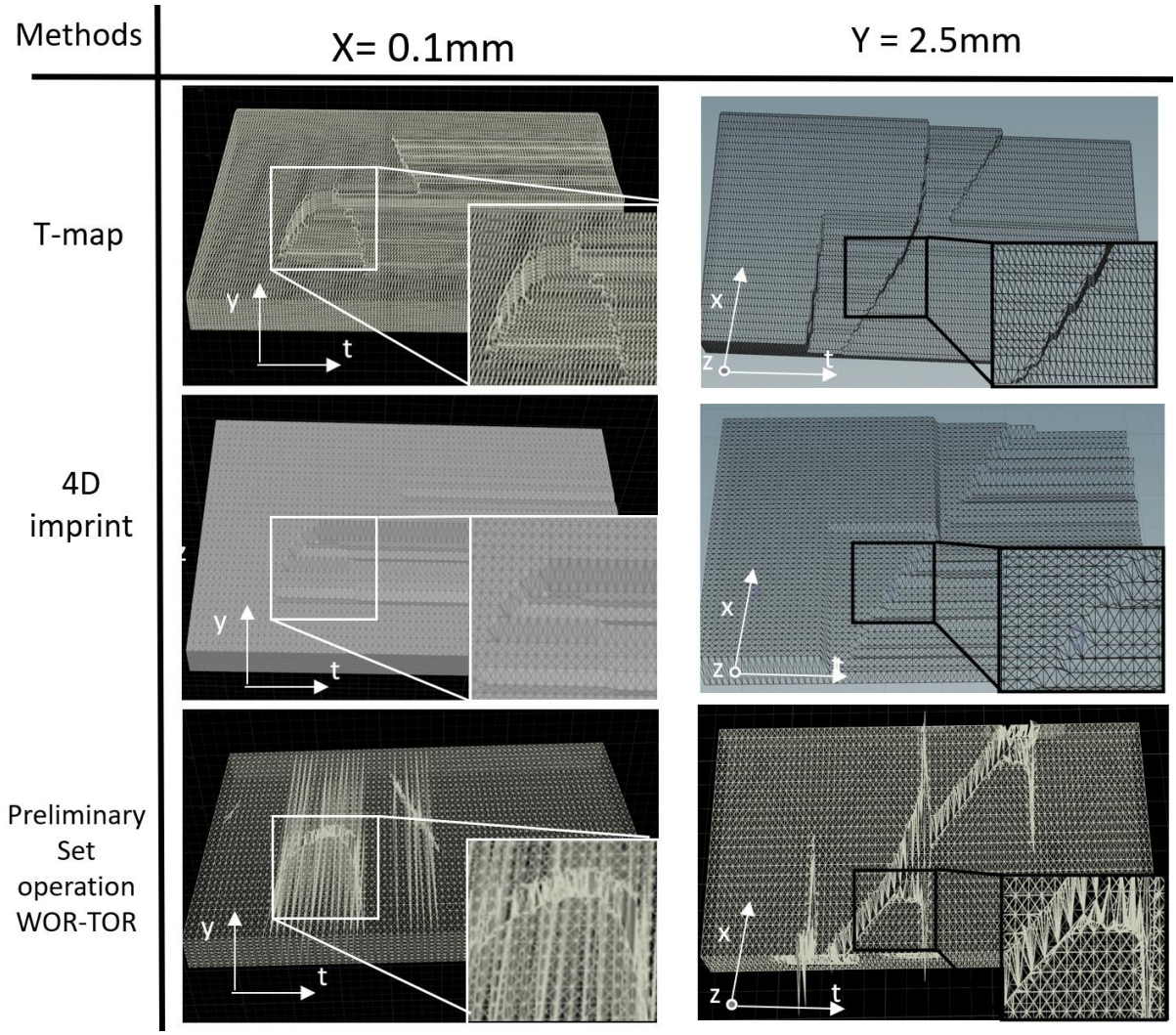


Fig. 7.3 Results of three preliminary methods, slices at x=0.01mm and y=2.5mm

7.2 Potential application of the research

I firmly believe in several potential applications of this research. The two most important points are:

1. AI-based post-processing. Due to its integrated nature, 4D data is very suitable for deep learning samples. It covers more comprehensive and extensive data than traditional 3D data sets. We believe that it is superior in both integrity and comprehensiveness. The results of the training will be used in various emerging CAD and CAM processes such as path planning and processing analysis.

2. Visualization of the processing process. For human users, the benefit of 4D models is that they can realize the visualization of CWE from any viewing angle

and viewing method. We can express the data we need through 3D hyperplane slices and projections in different directions and angles, and display other graphic parameters that are not easy to notice. At the same time, the generation speed of slices and projections of 4D models is significantly faster than the extrapolation and interpolation algorithms based on 3D motion models. Because 4D models are continuous and have been generated in advance.

7.3 Conclusion

This dissertation introduces the basic idea and features of conducting CWE analysis in spatio-temporal space.

The dissertation tries to propose a plan to construct a 4D CWE system. It starts with traditional CWE process method and foundation of 4D geometry. An entire system frame of CWE analysis based on 4D geometric models was established.

Additionally, the dissertation mainly presents and implements three different methods that utilize 4D geometric models for CWE analysis.

In chapter 5, two local continuous methods of CWE analysis in spatio-temporal space are proposed. T-map method emphasizes the continuous in T-axis and discrete in XYZ- axes. 4D imprint method emphasizes the continuous in XYZ-axes and discrete in T-axis. Both of two methods are developed from the exist methodologies in traditional 3D model processing.

In chapter 6, a generic method of CWE analysis based on 4D mesh models' set operation is proposed. The method can analyze CWE and avoid most of disadvantage of methods in last chapter. However, this method is limited by the huge amount of calculation, which makes it difficult to completed generate TAV. A preliminary implementation of set operation method is executed for proving correct.

For situations where continuity is not a concern, the T-map method can provide accuracy results by implementing 4D marching cube algorithm after the T-map method. If the user needs to quickly observe the CWE results, then T-map result with voxel model and 4D imprint method would be good choices. In most cases, set operation method can be implemented and achieve a precise result. But the method will take a long time to finish computation.

7.4 Future work

The following are some future works:

1. Through the improvement of the algorithm, realizing the generation of TAV, and completing the general set operation algorithm finally. Meanwhile, parallelize the algorithm.
2. Optimize algorithms for faster speed and higher accuracy.
3. Implement approaches in the actual manufacturing process, such as in the field of process planning, deep learn for the manufacturing and constructing machining cases library.
4. In future, the application field of 4D models set operation algorithm will be expanded. We will try representing the deformation process of real objects and apply it to robotics, medicine and so on.

Reference

[1] Onosato, M. , et al. (2007)., A study on geometric modeling methods for constructing cyber fields, Proceedings of the Japan Society for Precision Engineering Autumn Conference 2007

[2] Onosato, M. , et al. (2008). Research on 4D shape modeling for cyber field (the 1st Report). Proceedings of the Japan Society for Precision Engineering Spring Conference 2008

[3] Kawagishi, R, et al. (2008). Research on 4D shape modeling for cyber field (the 2nd Report). Proceedings of the Japan Society for Precision Engineering Spring Conference 2008

[4] Kawagishi, R., (2010) Development of A Four-Dimensional Mesh Modeling System for Cyber Field

[5] Bhaniramka,P., et a., 2000, “Isosurfacing in higher dimensions," Proc. VIS2000,, pp. 267-273. doi: 10.1109/VISUAL.2000.885704

[6] Onosato, M., et al. (2014). Weaving a Four-dimensional Mesh Model from a Series of Three-dimensional Voxel Models. Computer-Aided Design and Applications, 11, 649-658. doi: 10.1080/16864360.2014.914383

[7] Cameron, S. (1990). Collision Detection by Four-Dimensional Intersection Testing. IEEE Transactions on Robotics and Automation, 6(3), 291-302. doi: 10.1109/70.56661

[8] Eisemann, E., et al. (2008). Single-Pass GPU Solid Voxelization for Real-Time Application. In Proceedings of Graphics Interface 2008 (pp. 73-80). ISBN: 9781568814230. doi: 10.5555/1375714.1375728

[9] Black, Don V & Gopi, M. & Wessel, F. & Pajarola, Renato & Kuester, Falko. (2012). Visualizing Flat Spacetime: Viewing Optical versus Special Relativistic Effects. American Journal of Physics - AMER J PHYS. 75. doi:10.1119/1.2730838.

[11] Onosato, M., Kawagishi, R., Kato, K., Date, H., & Tanaka, F. (2010). Four dimensional mesh modeling for spatio-temporal object representation. In Proceedings of the Asian Conference on Design and Digital Engineering (pp. 579-589). Jeju, Korea.

[12] Otomo, I., Onosato, M., & Tanaka, F. (2014). Direct Construction of a Four-Dimensional Mesh Model from Three-Dimensional Object with Continuous

Rigid Body Movement. Journal of Computational Design and Engineering, 1(2), 96-102.doi: 10.7315/JCDE.2014.010

[13] Yabuki, Y., et al. Neighboring tetrahedron search algorithm based on 4D geometry in 4D extended Ball-Pivoting Algorithm. In Proceedings of the JSPE Spring Conference (pp. 9-10).

[14] Kaisa, K., Onosato, M., & Tanaka, F. Collision Avoidance Method for Moving Objects by Shape Deformation Using 4D Mesh Model. Proceedings of the Japan Society for Precision Engineering Spring Conference 2012 (pp. 543-544).

[15] Matsunaga, D., et al. (2017). Boundary intersection determination and intersection shape derivation for 4-dimensional mesh models. Proceedings of the Japan Society for Precision Engineering Spring Conference 2017.

[16] Kameyama, H., et al. (2014). Using a Four-Dimensional Mesh Model to Represent a Tool Motion Trajectory in Five-Axis Machining. International Journal of Automation Technology, 8(3), 437-444. doi: 10.20965/ijat.2014.p0437

[17] Kiita, K., Onosato, M., & Tanaka, F. (2013). Representation of tsunami propagation phenomenon using 4D mesh model. In Proceedings of the 2013 Hokkaido Branch Academic Conference of JSPE (pp. 67-68).

[18] Zhang, T., et al. (2021). Using four-dimensional geometric models for representing dynamic machining process based on parallel processing by GPGPU. In Proceedings of the International Conference on Leading Edge Manufacturing in 21st century (pp. 60-65).doi: 10.1299/jsmelem.2021.10.058-053

[19] Inui, Masatomo & Zhang, Tong & Umezu, Nobuyuki. (2020). Milling Simulation-Based Method to Evaluate Manufacturability of Machine Parts. doi: 10.1115/DETC2020-22124.

[20] Moriwaki, T., Sugimura, N., & Wang, L. (1993). Development of an Integrated CAD/CAE System for Machine Tool Design. Integrated Manufacturing Systems of JSPE, 81-94.

[21] Ma, H., Liu, W., Zhou, X., Niu, Q., & Kong, C. (2021). High efficiency calculation of cutter-workpiece engagement in five-axis milling using distance fields and envelope theory. Journal of Manufacturing Processes, 68, 1430-1447.

doi: 10.1016/j.jmapro.2021.06.055

[22] Gong, X., Feng, H. Y. (2015). Cutter-workpiece engagement determination for general milling using triangle mesh modeling. Journal of Computational Design and Engineering, 3, 151-160. doi: 10.1016/j.jcde.2015.12.001

[23] Shao, Z., Guo, R., Li, J., Peng, J. (2011). Accurate Modeling Method for Generalized Tool Swept Volume in 5-axis NC Machining Simulation. Journal of Software, 6(10), 2056-2063.

[24] Gupta, S. K., Saini, S. K., Spranklin, B. W., Yao, Z. (2005). Geometric algorithms for computing cutter engagement functions in 2.5D milling operations. Computer-Aided Design, 37, 1469-1480. doi: 10.1016/j.cad.2005.03.001

[25] Aras, E., Yip-Hoi, D. (2008). Geometric Modeling of Cutter/Workpiece Engagements in Three-Axis Milling Using Polyhedral Representations. Journal of Computing and Information Science in Engineering - JCISE, 8. doi:10.1115/1.2960490

[26] Zhang, T., Yabuki, Y., Onosato, M., & Tanaka, F. (2021). Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (1st report) – overview of research framework and some preliminary results –. Proceedings of the Japan Society for Precision Engineering Spring Conference, Online. doi: 10.11522/pscjspe.2021S.0_366

[27] Hanson, A. J., & Heng, P. A. (1991). Visualizing the fourth dimension using geometry and light. In Proceedings of Visualization '91 (pp. 321-328). San Diego, CA, USA. doi: 10.1109/VISUAL.1991.175821.

[28] Zhang, T., Onosato, M., & Tanaka, F. (2021). Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (2nd report) – tool-workpiece engagement analysis and acceleration based on GPGPU –. Proceedings of the Japan Society for Precision Engineering Autumn Conference, Online. doi: 10.11522/pscjspe.2021A.0_66

[29] SyBridge. (n.d.). 3-axis vs 5-axis CNC machining. Retrieved from https://sybridge.com/3-axis-vs-5-axis-cnc-machining/

[30] Apro, K. (2009). Secrets of 5-axis machining. Industrial Press. ISBN 9780831133757. OCLC 1008856747.

[31] Modern Machine Shop. (n.d.). An overview of 3+2 machining. Retrieved from https://www.mmsonline.com/articles/an-overview-of-3-2-machining

[32] Yip-Hoi, Derek & Huang, Xuemei. (2006). Cutter/Workpiece Engagement Feature Extraction from Solid Models for End Milling. Journal of Manufacturing Science and Engineering-transactions of The Asme - J MANUF SCI ENG. doi: 128. 10.1115/1.1948395.

[33] Larue. A., Altintas. Y., (2005).Simulation of flank milling processes,.. International Journal of Machine Tools and Manufacture, Volume 45, Issues 4–5, 2005, Pages 549-559,ISSN 0890-6955.

[34] Yip-Hoi, D., & Huang, X. (2006). Cutter/Workpiece Engagement Feature Extraction from Solid Models for End Milling. Journal of Manufacturing Science and Engineering, 128(1), 249. doi: 10.1115/1.1948395

[35] Ferry, W., & Yip-Hoi, D. (2008). Cutter-Workpiece Engagement Calculations by Parallel Slicing for 5-axis Flank Milling of Jet Engine Impellers. Journal of Manufacturing Science and Engineering, 130(5), 051011. doi: 10.1115/1.2927449

[36] Wang, W. P., & Wang, K. K. (1986). Geometric modeling for swept volume of moving solids. Computer Graphics and Applications, 6, 8–17. doi: 10.1109/MCG.1986.276586

[37] Chiou, C.-J., & Lee, Y.-S. (1999). A shape-generating approach for multi-axis machining G-buffer models. Computer-Aided Design, 31, 761–776. doi: 10.1016/S0010-4485(99)00069-X

[38] Chiou, C.-J., & Lee, Y.-S. (2002). Swept surface determination for five-axis numerical control machining. The International Journal of Machine Tools and Manufacture, 42, 1497–1507. doi: 10.1016/S0890-6955(02)00110-4

[39] Du, S., Surmann, T., Webber, O., & Weinert, K. (2005). Formulating swept profiles for five-axis tool motions. The International Journal of Machine Tools and Manufacture, 45, 849–861. doi: 10.1016/j.ijmachtools.2004.11.006

[40] Sheltami, K., Bedi, S., & Ismail, F. (1998). Swept volumes of toroidal cutters using generating curves. The International Journal of Machine Tools and Manufacture, 38, 855–870. doi: 10.1016/S0890-6955(97)00053-9

[41] Roth, D., Bedi, S., Ismail, F., & Mann, S. (2001). Surface swept by a toroidal cutter during 5-axis machining. Computer-Aided Design, 33, 57–63. doi:

10.1016/S0010-4485(00)00063-4

[42] Aras, E. (2009). Generating cutter swept envelopes in five-axis milling by two parameter families of spheres. Computer-Aided Design, 41, 95–105. doi: 10.1016/j.cad.2009.01.004

[43] Gong, H., & Wang, N. (2009). Analytical calculation of the envelope surface for generic milling tools directly from CL-data based on the moving frame method. Computer-Aided Design, 41, 848–855. doi: 10.1016/j.cad.2009.05.004

[44] Lee, S. W., & Nestler, A. (2011). Complete swept volume generation, Part I: swept volume of a piecewise C1-continuous cutter at five-axis milling via Gauss map. Computer-Aided Design, 43, 427–441.

[45] Lee, S. W., & Nestler, A. (2011). Complete swept volume generation, Part II: NC simulation of self-penetration via comprehensive analysis of envelope profiles. Computer-Aided Design, 43, 442–456.

[46] Blackmore, D., Leu, M. C., & Wang, L. P. (1997). The sweep-envelope differential equation algorithm and its application to NC machining verification. Computer-Aided Design, 29, 629–637. doi: 10.1016/S0010-4485(96)00101-7

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

[48] Chung, Y. C., Park, J. W., Shin, H., & Choi, B. K. (1998). Modeling the surface swept by a generalized cutter for NC verification. Computer-Aided Design, 30, 587–594. doi: 10.1016/S0010-4485(97)00033-X

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

[50] Jerard, R. B., Hussaini, S. Z., Drysdale, R. L., & Schaudt, B. (1989). Approximate methods for simulation and verification of numerically controlled machining programs. The Visual Computer, 5, 329–348. doi: 10.1007/BF01999101

[51] Park, J. W., Shin, Y. H., & Chung, Y. C. (2005). Hybrid cutting simulation via

discrete vector model. Computer-Aided Design, 37, 419–430. doi: 10.1016/j.cad.2004.07.003

[52] Lee, S. W., & Nestler, A. (2012). Virtual workpiece: workpiece representation for material removal process. The International Journal of Advanced Manufacturing Technology, 58, 443–463. doi: 10.1007/s00170-011-3431-2

[53] Zhang, T., Onosato, M., & Tanaka, F. (2023). Continuous representation of machining process using 4-dimensional geometric models - Cutter-workpiece engagement analysis and processing surface estimation in Spatio-Temporal space. LEM&P, Rutgers University, New Jersey, USA.

[54] Hanson, A. J., & Heng, P. A. (1992). Four-dimensional views of 3D scalar fields. In Proceedings Visualization '92 (pp. 84–91). Boston, MA, USA. doi: 10.1109/VISUAL.1992.235222.

[55] Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. Computer Graphics, 21(4), 163–169. doi: 10.1145/37402.37422

[56] Shen, H.-W. (2011). Time-Dependent Isosurface Extraction. In D. Charles et al. (Eds.), The visualization handbook (pp. 57–67). Academic Press.

[57] Fausto Bernardini et al.: “The ball-pivoting algorithm for sutface reconstruction”, IEEE Transactions on Visualization and Computer Graphics, Vol.5, No.4, pp.349-359, 1999. doi: 10.1109/2945.817351.

[58] Si, H. (n.d.). TetGen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator [Internet]. Berlin (Germany): Weierstrass Institute for Applied Analysis and Stochastics, Research group of Numerical Mathematics and Scientific Computing.

[59] Zhang, T., Onosato, M., & Tanaka, F. (2024). Continuous Representation of Machining Process using 4-dimensional Geometric Models: Cutter-workpiece engagement analysis and processing surface estimation in Spatio-Temporal space. IJAT, 18(4), 2024. doi: 10.20965/ijat.2024.p0463

[60] Nakamoto, K., et al. (2008). Development of a Virtual Machining Simulator using Voxel Model. Journal of the Japan Society for Precision Engineering, 74(12), 1308-1312. doi: 10.2493/jjspe.74.1308

[61] Kaneko, J. (2013). Industrial Applications of GPU/Parallel Processing Technology (Keynote Speech). Proceedings of the Japan Society for Precision

Engineering Annual Conference, 2013(0), 789-790.

[62] Inui, M., Chen, Q., & Umezu, N. (2022). Visualization of 3+2 Axis Machining Result by Combining Multiple Z-map Models. Computer-Aided Design & Applications, 19, 825-837.

[63] Aristek system. (n.d.). Most Frequently Used 3D Modeling Software in 2021. Retrieved from https://aristeksystems.com/blog/3d-modeling-soft-2021/#houdini

[64] SideFX. (n.d.). Houdini HOM Documentation. Retrieved from https://www.sidefx.com/docs/houdini/hom/index.html

[65] J. Fung, F. Tang and S. Mann, "Mediated reality using computer graphics hardware for computer vision," Proceedings. Sixth International Symposium on Wearable Computers,, Seattle, WA, USA, 2002, pp. 83-89, doi: 10.1109/ISWC.2002.1167222.

[66] Cheng, J., et al. (2014). Professional CUDA C Programming (1st. ed.). Wrox Press Ltd., GBR.

[67] Shane, C. (2012). CUDA Programming: A Developer's Guide to Parallel Computing with GPUs (1st. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[68] Duane, S., & Mete, Y. (2015). CUDA for Engineers: An Introduction to High-Performance Parallel Computing (1st. ed.). Addison-Wesley Professional.

[69] NVIDIA Corporation. (n.d.). CUDA Toolkit. Retrieved from https://developer.nvidia.com/cuda-toolkit

[70] Inui, M. (2014). Introduction to GPU Parallel Graphics Processing: Introduction and Utilization of CUDA and OpenGL. Gijutsu-Hyoron-Sha. (Software Design Plus). ISBN: 978-4-7741-6304-8. https://ndlsearch.ndl.go.jp/books/R100000002-I025187396

[71] ScienceDirect. (n.d.). Dynamic Parallelism. Retrieved from https://www.sciencedirect.com/topics/computerscience/dynamicparallelism

[72] Huang, W., & Inui, M. (2010). An algorithm for determining the optimal cutting direction in rough milling of impellers. In Proceedings of the 2010 International Symposium on Flexible Automation.

[73] Inui, M., et al. (2020). Milling Simulation-Based Method to Evaluate Manufacturability of Machine Parts. doi: 10.1115/DETC2020-22124.

[74] Inui, M., & Ohta, A. (2007). Using a GPU to Accelerate Die and Mold Fabrication. IEEE Computer Graphics and Applications, 27(1), 82-88.

[75] Zhang, T., Onosato, M., & Tanaka, F. (Year). Using four-dimensional geometric models for representing dynamic machining process based on parallel processing by GPGPU. In Proceedings of the 10th International Conference on Leading Edge Manufacturing in 21st Century (LEM21), Kitakyushu, Japan. doi: 10.1299/jsmelem.2021.10.058-053

[76] Gao, Xinrui & Zhang, Shusheng & Hou, Zengxuan. (2007). Three Direction DEXEL Model of Polyhedrons and Its Application. Proceedings - Third International Conference on Natural Computation, ICNC 2007. 5. 145-149. doi: 10.1109/ICNC.2007.777.

[77] Akenine-Möller, T., Haines, E., & Hoffman, N. (1999). Real-Time Rendering. Natick, MA: A K Peters, Ltd.

[78] Inui, M. (2003). Fast inverse offset computation using polygon rendering hardware. Computer-Aided Design, 35, 191-201. doi: 10.1016/S0010-4485(02)00052-0

[79] Fleischmann, P., & Selberherr, S. (1997). Three-dimensional Delaunay mesh generation using a modified advancing front approach. In Proceedings of IMR97 (pp. 267-278).

[80] Shewchuk, J. (2002). Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. Proceedings of the 11th International Meshing Roundtable.

[81] Bowyer, A. (1981). Computing Dirichlet tessellations. The Computer Journal, 24(2), 162–166. doi: 10.1093/comjnl/24.2.162

Research Achievement

2024.7 Tong Zhang, Masahiko Onosato, Fumiki Tanaka. “Continuous Representation of Machining Process using 4-dimensional Geometric Models: Cutter-workpiece engagement analysis and processing surface estimation in Spatio-Temporal space ” IJAT Vol.18 No.4, 2024.

2023.6 Zhang Tong, Onosato Masahiko, and Tanaka Fumiki . Continuous representation of machining process using 4-dimensional geometric models - Cutter-workpiece engagement analysis and processing surface estimation in Spatio-Temporal space – LEM&P, Rutgers University, New Jersey, USA

2023.3 Zhang Tong, Onosato Masahiko, and Tanaka Fumiki . Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (4th report) – Cutter-workpiece engagement analysis based on set operation between 4D mesh models – . JSPE 2023 Spring Conference, Tokyo, Japan

2022.3 Zhang Tong, Onosato Masahiko, and Tanaka Fumiki . Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (3rd report) – Analysis of boundary intersection in 4D mesh modeling and its parallel processing with GPU – . JSPE 2022 Spring Conference, Tokyo Institute of Technology (Online), Tokyo, Japan

2021.11 Zhang Tong, Onosato Masahiko, and Tanaka Fumiki . Using four-dimensional geometric models for representing dynamic machining process based on parallel processing by GPGPU. The 10th International Conference on Leading Edge Manufacturing in 21st Century (LEM21), Kitakyushu, Japan

2021.9 Tong Zhang, Masahiko Onosato,and Fumiki Tanaka. “Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (2nd report) – tool-workpiece engagement analysis and acceleration based on GPGPU –” The Japan Society for Precision Engineering Autumn Conference, Online

2021.3 Tong Zhang, Yuga Yabuki, Masahiko Onosato,and Fumiki Tanaka. “Comprehensive representation of machining process in spatio-temporal space based on four-dimensional geometric models (1st report) – overview of research framework and some preliminary results –” The Japan Society for Precision Engineering Spring Conference, Online

2020 Inui, Masatomo & Zhang, Tong & Umezu, Nobuyuki. (2020). Milling Simulation-Based Method to Evaluate Manufacturability of Machine Parts. 10.1115/DETC2020-22124.

2019.7 Tong Zhang, Masatomo Inui, Nobuyuki Umezu. “Milling simulation-based approach for determining optimal milling direction of ejector pin” i3CDE 2019 (International Congress and Conferences on Computational Design and Engineering), Penang, Malaysia.

2019.3 Tong Zhang, Masatomo Inui, Nobuyuki Umezu. “Automatic determination of optimal cutting direction of ejector pin” The Japan Society for Precision Engineering Spring Conference, Tokyo, Japan

Acknowledgements

I would like to extend my deepest gratitude to my advisor, Professor Masahiko Onosato, for his unwavering support throughout my doctoral program. His expertise and dedication have been instrumental in shaping my research journey. Professor Onosato not only provided invaluable guidance on research methodology but also offered critical insights that helped me refine my thesis. His consistent encouragement and constructive feedback were crucial in helping me navigate the complexities of academic writing. His mentorship has had a profound impact on my growth as a researcher and as an individual.

I am also profoundly thankful to my co-advisor, Professor Fumiki Tanaka, for his insightful guidance on my research. His expertise in the field provided a fresh perspective, helping me to broaden my understanding and approach my work from different angles.

Also, I would like to special thanks to Professor Satoshi Kanai for his valuable advice on this dissertation.

In addition, I want to express my heartfelt thanks to Professor Masatomo Inui, who was my advisor during my master's program in Ibaraki University. Professor Inui laid the foundation for my academic career by providing me with a solid grounding in the basics of research and nurturing my academic qualities.

My gratitude also extends to my lab colleagues, whose camaraderie and support made my doctoral journey all the more enriching. In particular, I would like to acknowledge Mr. Yabuki and Mr. Egusa, who were not only excellent collaborators but also good friends.

Furthermore, I am grateful to Hokkaido University for providing the DX fellowship, which offered essential financial support during my doctoral studies. This fellowship allowed me to focus on my research without the added stress of financial constraints.

Finally, I am deeply indebted to my parents for their relentless support throughout my academic journey. They have always been my pillars of strength, providing me with everything I needed to succeed. Their unwavering belief in my abilities and their constant encouragement have been my greatest motivators. Their sacrifices and unconditional love have made all my achievements possible, and I cannot thank them enough for everything they have done for me.