# FEATURE-BASED AUTOMATED TOOL PATH PLANNING FOR DISCRETE GEOMETRY

by

Fei Shen

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Mechanical Engineering

Charlotte

2022

Approved by:

_____
Dr. Joshua Tarbutton

_____
Dr. Edward Morse

_____
Dr. Angela Allen

_____
Dr. Harish Cherukuri

_____
Dr. Yuri Godin

# ABSTRACT

FEI SHEN. FEATURE-BASED AUTOMATED TOOL PATH PLANNING FOR DISCRETE GEOMETRY. (Under the direction of DR. JOSHUA TARBUTTON)

Computer Numerical Control (CNC) machining is a critical manufacturing technology used in effectively all modern products. Any improvement in efficiency or automation that reduces the cost of CNC machining is of tremendous value to the manufacturing industry. One of the most time-consuming steps in CNC machining, especially in a high-mix low-volume scenario, such as prototyping, is the current tool path planning workflow. The current industrial state of Computer-Aided Manufacturing (CAM) tools used to generate toolpaths requires highly trained CNC programmers. Typically, programmers manually select the features to be machined, the tools to use for each feature, the specific tool paths topology, and the feeds and speeds.

The research community has placed a lot of focus on the automation of the tool path planning process, in order to reduce the significant effort required to generate toolpaths. Researchers have developed novel feature recognition techniques, automated tool path generation methods, and tool selection algorithms. However, these methods all come with certain caveats and limitations. Some only work on analytic geometries. While others only work on certain feature types.

This dissertation introduces a feature based automated tool path planning system with a focus on implementing robust and generalized algorithms that work on arbitrary geometries with the full range of features based on discrete geometry. Support for discrete geometry is valuable because in many situations , only discrete geometry is available, such as in models generated from 3D scanning systems. Specifically, a robust region segmentation technique is developed to simplify machining feature recognition from discrete geometry. Once the features are recognized, an automated optimal cutter set selection approach aimed at a minimum machining time is proposed

to improve the machining efficiency for arbitrary features. In addition, an automated deburring tool path planning method is introduced to eliminate the manual edge deburring and specifically to work with 3D discrete geometry. With the robust and automated algorithms as a solid foundation, a fully automated tool path planning system with limited human interactions is built and demonstrated on a series of parts with complex intersecting features. The net result is a complete 3D CAM process that goes from geometry to G-code in less than 10 minutes.

# DEDICATION

To my wife, Tong, my parents, and my parents-in-law for their consistent support.

# ACKNOWLEDGEMENTS

First and foremost, I would like to extend my deepest appreciation to my dissertation chair and academic advisor, Dr. Joshua Tarbutton for his consistent help throughout my journey at UNC Charlotte. Thank you for always giving me the greatest freedom to find my research interest and explore the unknown. Dr. Tarbutton's passion for research has been a constant motivation to push me forward.

I would particularly like to thank Dr. Edward Morse for serving as the co-chair of my dissertation committee. I appreciate his insight and guidance, which was extremely crucial in shaping the methodology section of this research.

I sincerely thank my other committee members, Dr. Angela Allen, Dr. Harish Cherukuri, and Dr. Yuri Godin. Without their advice and suggestions, this dissertation could not have been improved this much.

Also, I am deeply grateful to my colleagues at Toolpath Labs, who also made great contributions to this dissertation. I am thankful for my mentor, Dr. Justin Gray, for his constant encouragement and guidance on my journey. I would like to thank the CEO of Toolpath Labs, Mr. Andy Powell, for his support on the experiments of the dissertation and his flexibility with my schedule. Also, I would like to thank Mr. Abdullah Zafar and Mr. Cameron Bieganek, for their close collaboration on the important research topics of feature recognition and tool path generation.

I am thankful to my former colleagues, Dr. Martin Koerdel and Dr. Valeri Golovlev for offering me the internship opportunity at Siemens AG. Working in a leading industrial manufacturing company in the early career has been an unforgettable experience, which has significantly improved my problem-solving and programming skills.

Moreover, I feel lucky to have Luis Vega, Mohammadrafi Marandi, and Christoph Kossack, the three exceptional PhD students, as my lab mates. I really enjoyed the time we spent in the office by helping each other in both academic and personal life. I also owe much thanks to my dear friends, Dr. Kang Ni, Yue Peng, and Dr. Yuanye

Liu who cared about my progress and made this journey full of fun.

Finally, I would like to express my deepest gratitude to my parents and in-laws for their tremendous understanding and encouragement in the past few years. Special thanks to my wife, Tong. Without her commitment, it would be impossible for me to complete my study.

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

AAG Attributed Adjacency Graph

APT Automatically Programmed Tool

ARAG Attributed Region Adjacency Graph

BRep Boundary Representation

CAD Computer-Aided Design

CAM Computer-Aided Manufacturing

CAPP Computer-Aided Process Planning

CC Cutter Contact

CL Cutter Location

CNC Computer Numerical Control

CSG Constructive Solid Geometry

FR Feature Recognition

GD&T Geometric Dimensioning and Tolerancing

GPGPU General-Purpose Graphics Processing Unit

IPM In-Process Model

MIC Maximum Inscribed Circle

RP Rapid Prototyping

# CHAPTER 1: INTRODUCTION

Computer Numerical Control (CNC) refers to the automated control of machine tools based on the specific coded instructions sent to the controller of the machine tool. Based on the received programs, CNC machine tools can perform specific tasks much more efficiently and more precisely than manually controlled machines or processes. Hence, manufacturing costs can be reduced, production efficiency can be improved, and work safety can be enhanced by using CNC machines. The manufacturing industries have been transformed considerably since the introduction of the first CNC machines in the 1940s and 1950s. Today's CNC machines are able to produce parts in tight tolerances, and the cutting tools can cut a large load of raw material at a fast speed.

The early programs for controlling CNC machines were manually written by machinists. The programming process was inefficient, error-prone, and skill-dependent. The trial-and-error approach was commonly used in the machining process. The traditional machinists usually spent years on training and practicing to be capable of operating and programming CNC machines efficiently. The introduction of Computer-Aided Design (CAD) software packages eased the difficulty in designing parts with complex geometries, but on the other hand it added extra challenges to machinists who had to program the CNC machines to produce these complicated parts. Thus, CAM software tools were developed to increase the productivity and boost the efficiency of the CNC programming process. In today's machine shops, a machinist may act as a machine programmer/process planner, machine operator, or setup operator. The machine programmer or process planner is responsible for analyzing the CAD design, defining the datum and the fixture, selecting machining

operations and cutting tools, utilizing CAM systems to generate tool paths and define the associated cutting speed and feed rate, and estimating the production time and machining cost. The generated tool paths and other machining information are then converted by the CAM systems into a machine understandable program called G-code, which is a set of instructions that can control the actions of CNC machines. In a typical workflow, the setup operator loads the G-code into a CNC machine and sets up the tools. The machine operator subsequently loads the raw material and runs the G-code on the machine to transform the raw material into a product with a designed shape. Modern CAM packages can generate sophisticated tool paths for 5-axis CNC machines to produce complicated and high-quality parts as well as provide high-speed tool paths that help manufacture parts faster than ever. However, planning the machining process still relies on the experience and knowledge of the process planners. Two process planners may each propose unique manufacturing plans for the same machining task, and this adds extra inconsistency to the sophisticated job. The inefficiency and variability in process planning increase the resources and time costs of the enterprise.

Computer-Aided Process Planning (CAPP) was presented in the 1960s to build a bridge between CAD and CAM with an aim to improve the efficiency of the process planning. CAPP is a technique that leverages the modern computers to aid process planners in making decisions on the manufacturing plans in a systematic way. To date, a great deal of research and investment has been dedicated to developing CAPP systems and even automated CAPP systems. However, few CAPP systems can provide significant process planning solutions to the manufacturing industry. The automated CAPP system still remains in its infant stage, and the enormous complexity is part of the reason for its slow development.

Other than its high value in a production environment, an automated tool path planning system can also contribute to the application of CNC in Rapid Prototyping

(RP). RP is a method for converting 3D designs into their physical prototypes without intermediate processes. Because of its capability to construct physical models rapidly, RP has been widely used to significantly reduce the time required for the product development cycle. Currently, additive processes dominate the RP market. In the low-end market, desktop 3D printers have become more commonplace because of their acceptable price and user-friendly characteristics. Compared with CNC machines, such as a CNC mill or a CNC lathe, desktop 3D printers do not require the user to know how to generate a tool path. A user of a 3D printer usually just needs to upload the file that contains the design part and push a button to start. The printer can generate a layer-based tool path automatically and print the part accordingly. In the high-end market, several metal printing processes, such as direct metal laser sintering, electron beam melting, and laser engineering net shaping, have been developed to produce metal parts. The motivation behind the development of these processes is to create truly functional metal parts, although these commercial metal printing machines can cost as much as a few hundred thousand dollars. However, the already existing much cheaper CNC machines that can create a metal part from raw material have attracted little attention in the RP field. One important reason is that the tool path planning process requires years of experience and extensive user interactions. An automated tool path planning system that can significantly contribute to the realization of a push-button CNC machining process would make it possible to open a door for the application of CNC machines in RP.

The goal of this dissertation is to develop a feature-based automated tool path planning system that can generate G-code automatically for machining parts with a minimum amount of user interactions. The automated tool path planning system is mainly composed of four modules: input module, feature module, cutter module, and automation module, as shown in Figure 1.1.

The input module contains the information received by the system from the user's

end, which includes the part model, the stock dimensions, the material for the stock, the machining direction, and the user's cutter library representing the available cutters. The system accepts discrete geometry represented as point cloud or triangle mesh. Support for discrete geometry is valuable because, in many situations, discrete geometry is the only available format. In the modern world of 3D printing, discrete geometry is the primary geometric format that is distributed, and it is desirable to develop CNC tool path planning that can machine parts which otherwise might be 3D printed. In addition, 3D scan data might be produced to replicate a part where the discrete geometry may be the only data available. Finally, some modern CAD systems, such as OpenSCAD and TinkerCAD, only provide discrete geometry. The feature module takes the discrete part model as the input and recognizes the machining features from the model. The automation module is the core module of the system that executes all the tasks related to machining automation. The module is able to select the optimal set of cutters from the cutter library and generate tool paths for each machining operation including the facing, roughing, finishing, and deburring operations. The facing operation is developed to remove the extra material on top of the part along the machining direction. The rouging and the finishing operations clear the material enclosed by the geometry of each machining features. After all the recognized features are machined, the deburring operation is employed to deburr the undesired boundary edges that might reduce product life, decrease dimensional accuracy, and introduce difficulties into assembly. To improve the efficiency of the system, a sequencing function is added to optimize the sequences of the operations. In the final step, the tool paths in each operation along with the cutter information are exported as a G-code program.

Compared with the state-of-the-art CAM packages, the developed tool path planning system requires many fewer user interactions. In this system, the user does not need to plan the operations for machining the part model and organize the sequence

of the operations. The optimal cutters are also selected by the system automatically whereas the cutters are defined by the user in the existing CAM tools, which requires years of experience in machining. Based on this system, there is a great potential that a push-button desktop-desk CNC machine can be implemented so that an inexperienced user can use the machine to prototype metal parts easily.

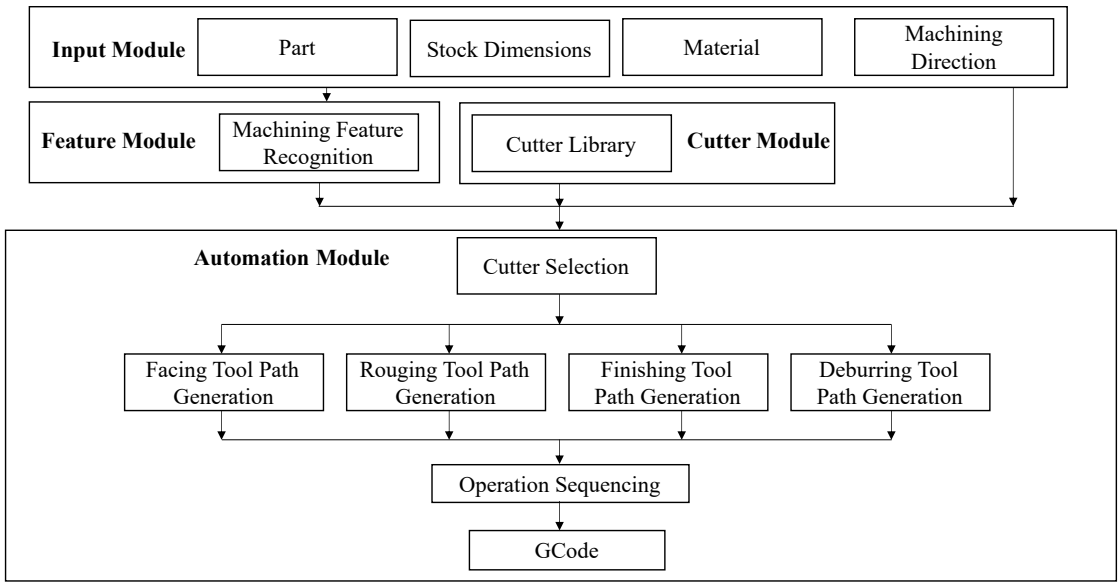


Figure 1.1: Workflow of the feature based automated tool path planning system.

Specifically, the objective of this research is to implement robust and generalized algorithms that work on arbitrary geometries with the full range of features and use discrete geometry as it is the lowest common geometric denominator.

Three key contributions of this dissertation are summarized as follows:

1. Improved robustness for discrete geometry region segmentation with three steps, enabling better feature recognition:

(a) Primary shapes region segmentation to detect primitives within discrete geometry;

(b) Merging shape regions to handle low resolution and noisy discrete geometry; and

(c) Final island region segmentation to handle freeform surfaces alongside regular

surfaces within discrete geometry.

2. Introduced a unique optimal cutter set selection approach for minimum cutting time for arbitrary feature machining.

3. Created a novel method for automated detection and generation of the deburring tool paths to finish the feasible boundaries.

As a secondary contribution, a hybrid approach for machining feature recognition that can handle complex intersecting features is developed to recognize STEP-NC machining features in close collaboration with Abdullah Zafar.

The remaining parts of the dissertation are arranged as follows. Chapter 2 introduces the background of this research. Chapter 3 presents the new region segmentation approach along with a hybrid feature recognition method to recognize machining features from discrete geometry. Chapter 4 introduces the automated milling tool path planning method with an optimal set of cutters. Chapter 5 presents the automated feasible boundary detection and deburring tool path generation approach. Chapter 6 provides and discusses the experimental results that validate the proposed tool path planning methodology. Chapter 7, the last chapter, summarizes the major contributions of this dissertation and discusses the opportunities for future work.

# CHAPTER 2: BACKGROUND

This chapter presents the related background information about the tool path planning, CAPP, machining feature recognition, and geometry representation. First, the traditional milling tool path planning methods are introduced, followed by the technologies that try to automate the tool path generation by using CAPP and machining feature recognition. Finally, the analytic and discrete geometry representations are reviewed, the benefits and disadvantages of these approaches are compared, and the tool path planning approaches on discrete geometry are also analyzed.

## 2.1 Traditional Tool Path Planning Methods

The task of CNC machining is to drive the cutter to engage with the raw material, which is also called the workpiece, so that the extra material is removed from the workpiece and the desired shape of the designed part is formed. An operation or tool path typically aims to cut a particular machining feature, such as a hole, slot, edge round, chamfer, pocket, planar face, step, profile, or freeform surface. These machining features are usually machined by specific cutting tools. For example, drill bits and reamers are used to drill and clean a hole while slot cutters can be used to cut a slot. Other features are typically machined by using milling cutters based on the milling tool paths. The tool path strategies for making holes, slots, edge rounds, and chamfers are already well-defined in modern CAM packages and are relatively easy to use compared with the milling tool paths. The relevant concepts in milling tool path planning are reviewed in addition to the traditional approaches for generating milling tool paths.

The goal of any milling tool path planning method is to generate accurate lines

and curves called tool paths, as shown in Figure 2.1, so that the cutting tool can move in contact with the workpiece along the specific trajectories formed by the tool paths to produce the desired part without gouges or collisions, as shown in Figure 2.2. The Cutter Contact (CC) points on the tool paths are the instantaneous contact points between the tool and the workpiece. In the ideal scenario, all points on the surface of the designed part should be CC points so that any geometric errors in the manufactured part are minimized. However, this is not possible in real-life scenarios because the machining time would be increased tremendously. Hence, the CC points must be discretely separated from each other as well as from the resulting tool paths. The separation between tool paths causes geometric inaccuracies for non-planar surfaces in the manufactured part. The unmachined region is called the scallop or cusp, and the upper limit on the height of the scallop is called the scallop height allowance, as shown in Figure 2.3. The scallop height allowance is used to describe the geometric inaccuracies caused by the separation of tool paths. The Cutter Location (CL) point is the location of the tool in space. For flat end mills, it is the center of the bottom of the tool; for ball end mills, it is the center of the ball at the end of the tool. The CL is used by the CNC machine as a reference location to drive the tool along the tool path. The CL data usually is calculated based on CC data, which provides the necessary geometric information for CNC machining. A gouge occurs when the cutter removes excess material from the workpiece at the CC point while a collision happens when the cutter collides with other sections of the part, machine, or fixtures. A valid tool path should be free of gouges and collisions.

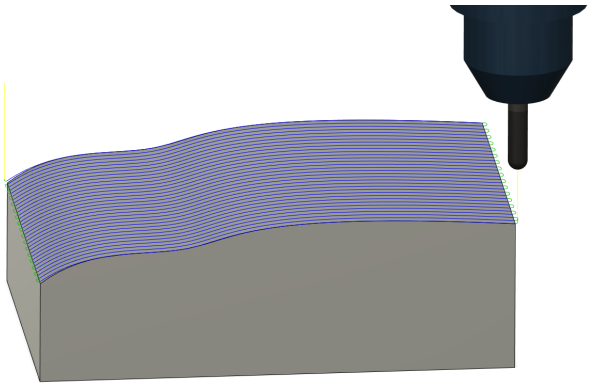

Figure 2.1: Designed part and tool path.

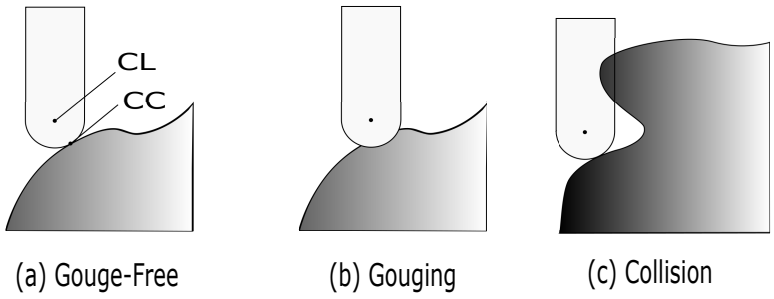


Figure 2.2: Gouging and collision.

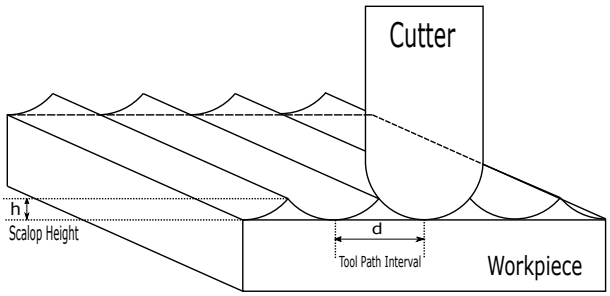


Figure 2.3: Geometric inaccuracies in tool path.

Traditionally, the milling tool path is generated either (a) with the help of curves or surfaces defined in the object domain of the designed part or (b) by specifying curves in the parametric domain of the designed part. Most modern tool path planning strategies are based on three well-established algorithms: iso-planar [2–4], iso-parametric [5–8], and iso-scallop [9–11]. The iso-planar and iso-parametric methods

fall under the categories (a) and (b), respectively, while the iso-scallop is an extension of the first two methods.

### 2.1.1 Iso-planar Milling Tool Path Planning

One of earliest CNC programming languages, the Automatically Programmed Tool (APT) approach can be considered a generalization of the iso-planar. APT, as shown in Figure 2.4, generates CL data by driving the tool along curves that are the intersections of a user-defined surface, called the drive surface, and the part surface [2]. Gouging is avoided by predefining a check surface representing the boundary of the part that must not be gouged. The drawback of the APT approach is that all CL data require several iterations, which may not converge, to search for a point at which the tool is tangent to the surface of the part. The computational efficiency was improved by the iso-planar methods, as shown in Figure 2.5, in which tool paths are generated by intersecting the surface with a series of infinite parallel planes. The distance between the parallel planes is determined by the requirement of the scallop height. The iso-planar method is simple and robust, and it is used extensively in commercial CAM packages to generate Cartesian or constant z-level tool paths. However, a computational, expensive surface-plane intersection is needed in the iso-planer method. In addition, the overall length of the tool paths is very large; hence, the iso-planar method is machining inefficient. The tool path is referred to as the direct parallel tool path when the cutting planes are horizontal [12] whereas it is called the contour parallel tool path when the cutting planes are vertical.

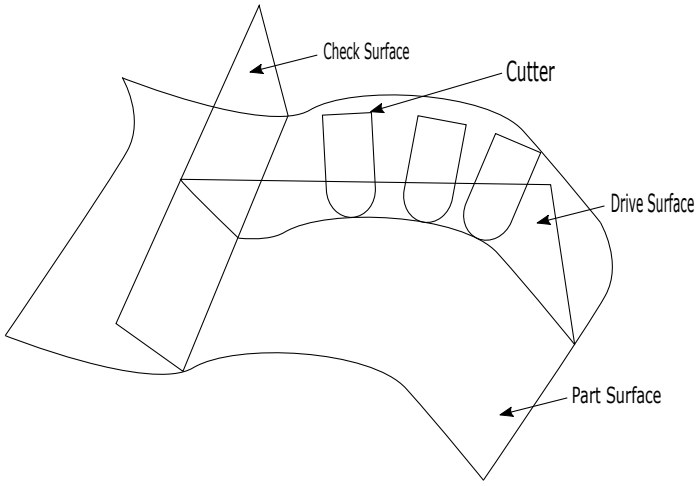


Figure 2.4: APT approach for CNC milling tool path generation.

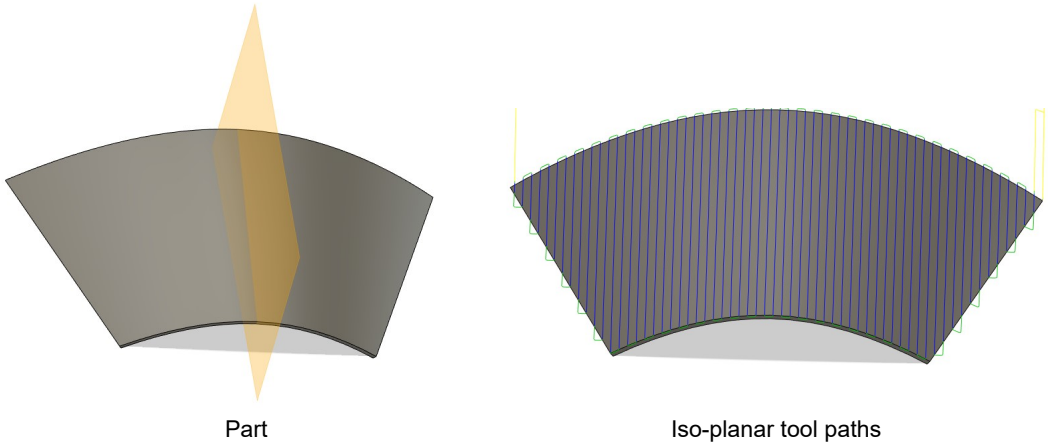


Figure 2.5: Iso-planar approach for CNC milling tool path generation.

### 2.1.2 Iso-parametric Milling Tool Path Planning

The iso-parametric tool path, as shown in Figure 2.6, is planned by mapping the parametric surface, S(u,v), onto Euclidean space and then keeping one of the parameters, either u or v, constant while incrementing another parameter. Compared to the iso-planar method, costly surface-surface intersection computations were avoided. However, a main drawback of the iso-parametric method is that a constant step in the parametric domain results in varying the scallop-height distribution on the machined surface. This method is also not suitable for generating the tool path for a compound and trimmed surfaces because gouging may occur in these surfaces. When

the underlying free-form surfaces are extended and combined, it is necessary to repair the surface to ensure error-free tool paths because the boundary of the tool path generated by iso-parametric tool paths is no longer confined [13].

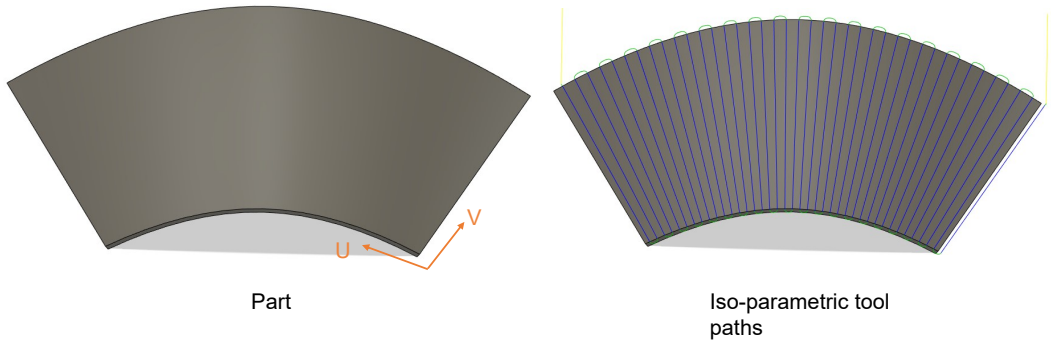


Figure 2.6: Iso-parametric approach for CNC milling tool path generation.

### 2.1.3 Iso-scallop Milling Tool Path Planning

Both the iso-parametric and the iso-planar methods can produce uneven scallop heights in the parts with non-planar surfaces. To meet the tolerance requirement over the entire surface, the step distance between two adjacent tool paths must be kept conservative, which leads to unnecessary tool paths in some areas and eventually increases the machining time. The iso-scallop method, as shown in Figure 2.7, is proposed in [9] to overcome the issue of non-optimal machining in these two methods. This method is an extension of the iso-planar and iso-parametric methods. The side steps between the cutter location points in two adjacent tool paths are adjusted so that the resulting scallop height is constant. The iso-scallop method can produce the shortest tool path among these generated by the three common methods for planning tool paths [9]. However, surface offsetting is involved in the iso-scallop method, and it is complex and computationally expensive.

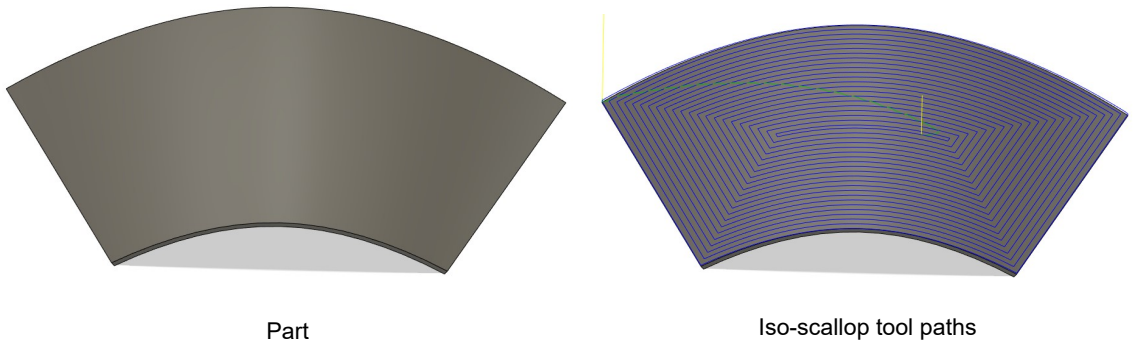


Figure 2.7: Iso-scallop approach for CNC milling tool path generation.

Although several tool path planning strategies for different machining features are available in modern CAM packages, the tool path for machining a product, represented in a solid model designed in a CAD system, still cannot be generated without human intervention. The current practice of CAM packages still requires extensive interaction by the user. To generate tool paths for a designed part, CAM users must select the tool path planning strategy with appropriate parameters and define a cutting tool for each machining feature. The traditional tool path planning process is time-consuming and prone to have errors. The process planners could spend hours to generate the tool path, but the actual machining time is only a few minutes. To improve production efficiency and automate the design-to-manufacturing process, a new software tool is necessary to improve the integration of CAD and CAM modules.

## 2.2 Computer-aided Process Planning

Process planning is an essential manufacturing activity because a detailed plan must be developed that specifies exactly how to manufacture the designed parts in the CAD model with the assistance of CAM packages and planning the tool path is a vitally important part of the activity. CAPP refers to the use of a computer to aid the planning process, and it can be viewed as a communication agent between the CAD and CAM systems. Since Niebel [14] first described the CAPP concept in 1965, considerable research effort has been devoted to CAPP.

Generally, there are two types of approaches for CAPP, and they are variant and generative [15]. The variant process planning approach is based on the rule that similar parts require similar plans. This approach requires a human operator to classify a part, provide information about the part, retrieve a similar process plan from a database, and make the necessary modifications. This approach is advantageous from the maintenance perspective, but its main drawback is that the quality of the process plan still depends on the knowledge of a process planner. Compared with the variant approach, the generative approach requires less human intervention in planning the process. This approach generates new process plans based on decision logic, process knowledge, manufacturing rules, and input about the features and attributes of the part. Due to the distinguishing feature of requiring little interaction with the user, the generative CAPP approach has been studied more frequently than the variant approach in recent research.

To date, CAPP systems have been extensively researched [16–34]; however, most of this research has focused on the connections between part models and the manufacturing plans based on feature technology. The generation of automatic tool paths has been little studied. Liang et al. [35] proposed an automated tool path planning system, although the system only deals with the rough machining of planar features. Hou and Faddis [36] described an automated tool path planning method in an integrated CAD/CAPP/CAM based on machining features. In the system, the machining features serve as the agent that transfers geometric information about the design part from the CAPP system, called FBMach, to the CAM software, UG (now Siemens NX) for the creation of tool paths and the NC program. However, the capability of feature recognition technology in FBMach is limited. The system cannot recognize all the machining features, especially intersecting features. In addition, the $2\frac{1}{2}D$ pocket feature only can have one bottom surface so that the tool path can be generated for the feature. Xu et al. [37] introduced an automated tool path

planning approach integrating the Structuralized Machining Process (SMP) with the CAD/CAPP/CAM system. In this research, SMP is defined as a multi-level structure comprised of sequential machining operations. An agent-based system was developed to take SMP and the associated in-process models as inputs and export operation controlled templates that can map each machining operation into CAM systems to generate the tool path automatically. Although this approach works for many parts with several machining features, it still faces difficulty in generating the tool path when the model of the part contains freeform surfaces.

Compared with the significant advancements in the CAD and CAM technologies, the development of CAPP has been relatively slow. To date, few commercial CAPP systems exist that can provide a disruptive solution in process planning for the manufacturing industry. Part of the reason is its complex nature. As surveyed by Alwswasi [16], a complete CAPP system would require various input information for the user, including the machining process and process capabilities, a feature model, Geometric Dimensioning and Tolerancing (GD&T), stock materials, and the requirements for the surface finishes. And the expected output of the system involves the selection of processes, the operation sequence, cutting tools, cutting conditions, selection of jigs and fixtures, tool path, and the estimated time and cost to manufacture the part. The current CAPP systems are not capable of handling the sophisticated input information and producing all the expected output automatically.

## 2.3 Machining Feature Recognition

Feature Recognition (FR) is an activity that interprets a part model composed of low-level geometric elements (e.g., points, edges, and faces) into high-level entities (e.g., holes, slots, and pockets) to support the downstream engineering tasks, such as manufacturability analysis [38], optimization [39], design validation [40], and manufacturing planning [41–43]. FR plays a central role in CAPP for integrating the CAD system with the CAM system. The major reason for its importance is that almost

all CAPP systems are based on features or require features as the input data.

### 2.3.1 Definition of Machining Feature

Depending on the application areas, the word "feature" has different definitions. For example, in CAD modeling, the surface features are described as a set of primitives defined by their topological and geometric parameters [44]. In FR, the features are specified according to their topological patterns and geometric characteristics, such as inner/outer boundary loops, convexity/concavity edges, and surface/volume feature [45]. The manufacturing features are the most widely identified feature in FR, which represents a series of geometric entities that can be machined by downstream manufacturing processes, such as machining [46, 47], sheet metal stamping [48], forging and casting [49], and molding [50]. The machining feature is the most studied feature among the manufacturing features; it also is the focus of this research. One of the extensively accepted definitions of a manufacturing feature is specified in STEP-NC [30,31], which is the application of the ISO Standard for the Exchange of Product Model Data (STEP) to numerical controlled machines. In STEP-NC, the manufacturing features are classified into regions and $2\frac{1}{2}D$ manufacturing features. The region feature is the same as a freeform feature, while the $2\frac{1}{2}D$ manufacturing features are composed of the $2\frac{1}{2}D$ machining features as well as their replications. The STEP-NC machining features are defined in STEP AP 224 [31], which is a STEP standard to describe the information in need for making a manufacturing plan for the machining features. In the standard, the $2\frac{1}{2}D$ manufacturing features can be classified into four categories: machining features, transition features, replicate features, and compound features (see Figure 2.8). Machining features are defined as the features that are used in $2\frac{1}{2}D$ machining, which is distinguished by the feature that the cutter moves mostly in the XY plane, while it only moves along the Z axis to a certain depth to remove a layer of material. The $2\frac{1}{2}D$ machining features include planar face, pocket, slot, step, hole, profile, rounded end, boss, thread, spherical cap, toolpath feature, and com-

pound features. Some examples of the $2\frac{1}{2}D$ features specified in the STEP-NC are shown in Figure 2.9. A transition feature including the edge round and the chamfer is a feature located at the border of two features. A replicate feature is an assembly of a few similar features, such as an array of holes. Meanwhile, a compound feature is a combination of two or more machining features, such as a countersink hole or a counterbore hole.

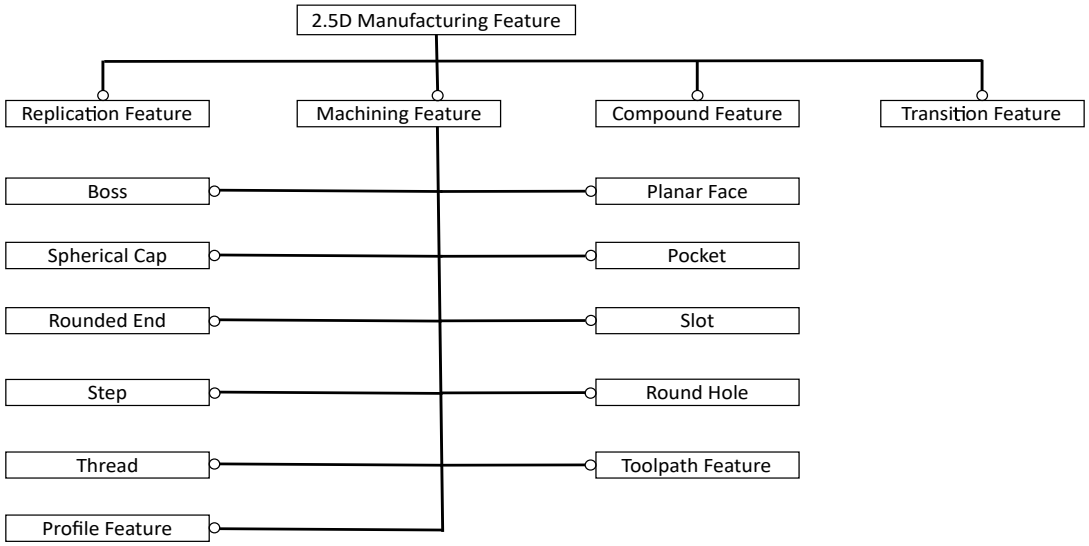


Figure 2.8: $2\frac{1}{2}D$ manufacturing features in STEP-NC.

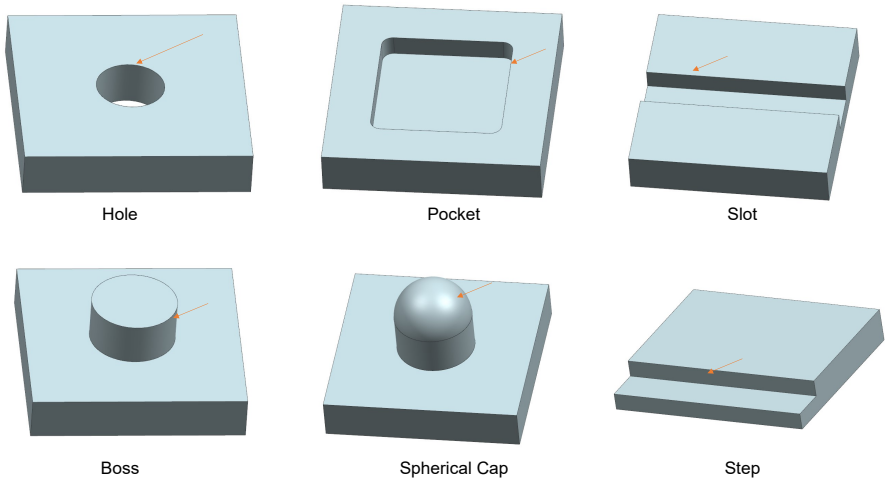


Figure 2.9: Examples of machining features in STEP-NC.

### 2.3.2 Feature Recognition Methods

The research on FR has been active for more than two decades, and many different approaches have been introduced. The main problem in FR systems is the identification of the implicit patterns from a solid model represented by explicit geometric entities. In most developed approaches, it is handled by matching templates, rules, or graphs that act as the expected geometrical and topological properties of the machining features. Although numerous approaches have been introduced for FR, the five most prominent approaches are the graph-based method [51–53], volumetric decomposition method [46, 54–57], rule-based method [58], hint-based method [59, 60], and artificial neural network method [61, 62]. In addition, hybrid approaches [63–68] combine the benefits of each single method. Among all these methods, the graph-based approach is the most widely employed method and thus will be reviewed in detail as follows.

The graph-based FR involves the search for the patterns of face-edge in the input solid model. In this approach, each face and edge of the solid model are represented by a node and an arc, respectively. Each node is assigned with some attributes, such as the surface finish of a face and the geometric type of the face, whereas each arc is attached with several other properties, such as the convexity and concavity of the corresponding edges. Usually, a subgraph is decomposed from the complete graph, and then the graph matching approach is used to determine if the pattern of the subgraph matches that of a template feature. Figure 2.10 shows the process of graph-based FR for a slot feature. In the figure, the subgraph with three node faces (7, 8, and 9) connected by two concave edges is the same as the pattern in the template graph of the slot feature with two wall faces connected by a floor face. Joshi and Chang [51] presented the original graph-based FR method and they introduced the concept of Attributed Adjacency Graph (AAG), which, as stated above, forms the basic principle of this type of FR. The initial AAG is limited to recognizing

features from a negative polyhedral solid model. Subsequent studies improved the performance of graph-based approach by assigning more geometric attributes to the nodes and arcs, such as perpendicularity, coaxiality, and parallelism [52, 53].

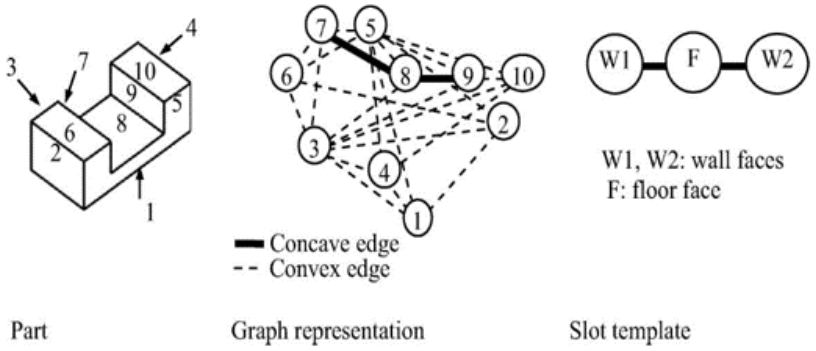


Figure 2.10: Graph-based FR for a slot feature [1].

Although graph-based methods demonstrate a high level of competence in recognizing independent features that do not change their face topology and other features through interaction, these methods have a major drawback in lacking the capability to handle arbitrary feature interactions. In the cases of the feature interactions, the basic topology of the features is changed as many vital faces are either deleted or partially trimmed or decomposed into many faces with different shapes. An example of a feature interaction that prevents the graph-based method from recognizing features properly is illustrated in Figure 2.11. In this case, a large slot interacts with two smaller slots, and it is expected to be between faces 1 and 7 with face 4 as the floor. However, the concave edge connection between faces 1 and 7 is deleted due to its interactions with the other smaller slots. The FR system fails to recognize the large slot feature because the AAG does not match the template of slot feature.

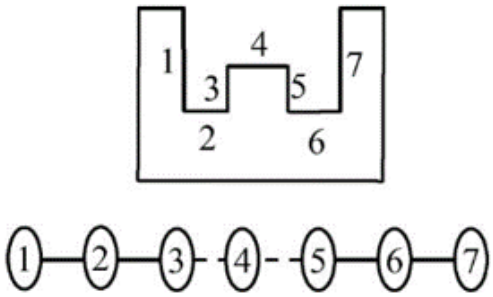


Figure 2.11: Interactions between three slot features and the corresponding AAG [1].

The graph-based approach is believed to be the most used FR method among the commercial FR systems. For instance, Siemens NX has a CAPP module based on the FR technique, called Feature Based Machining. First, the machining features specified in STEP AP 224 are recognized in the system, and then machining operations are assigned to these features based on their characteristics. However, the system has difficulty dealing with feature interactions. As shown in Figure 2.12, the FR technique in Siemens NX can detect the isolated holes and the top slot properly. However, the slot in the center was detected as two separate slots, and the pocket feature cannot even be recognized at all because the graph of pocket is interrupted when the pocket interacts with the slots on the top and in the center.

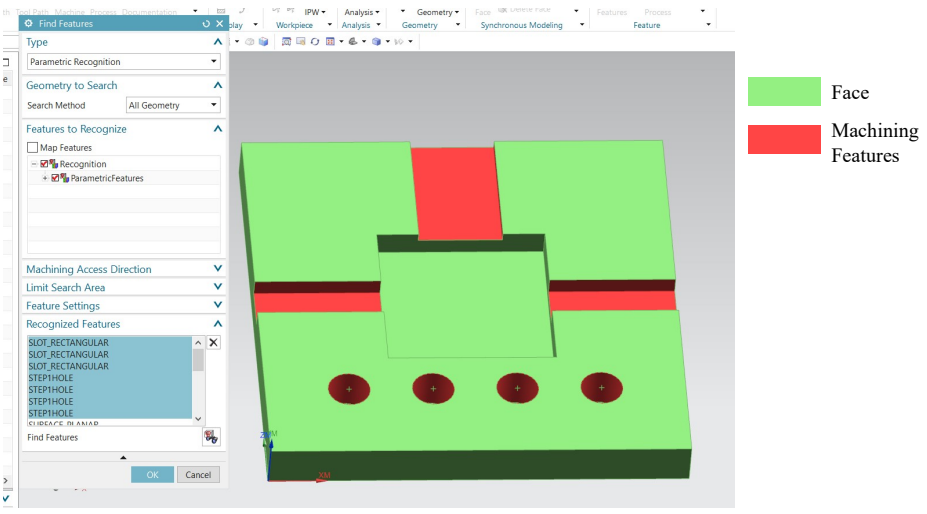


Figure 2.12: An example part for the FR technique in Siemens NX.

Although these methods are promising, each method has its own disadvantages and benefits. Today's FR techniques are still not able to handle arbitrary intersecting features.

## 2.4 Analytic Geometry and Discrete Geometry

Geometry representation is a method for representing the geometry of an object in computers. Basically, there are two types of geometry: analytic and discrete geometries. In analytic geometry, the objects are represented by either a set of surfaces and curves or a combination of primitives defined by mathematical equations. In contrast, the discrete geometry describes the shape of the objects using discrete data.

### 2.4.1 Analytic Geometry

Constructive Solid Geometry (CSG) and Boundary Representation (B-rep or BRep) are two typical analytic geometries. These geometry representations are the 3D solid modeling techniques in the geometric modeling kernel, which is the core component of the modern CAD/CAM packages. As shown in Figure 2.13, the geometric modeling kernels currently on the market include Parasolid, ACIS, APM Engine, Shape Manager, Open CASCADE, Convergence Geometric Modeler, C3D, and Granite, among which Parasolid and ACIS dominate the kernel market. Most advanced CAD/CAM packages were developed based on these kernels. Siemens NX, Solid Edge, SolidWorks, MasterCAM, and GibbsCAM are built based on Parasolid. AutoCAD, Inventor, and Fusion 360 used ShapeManager. CGM is the kernel of CATIA.

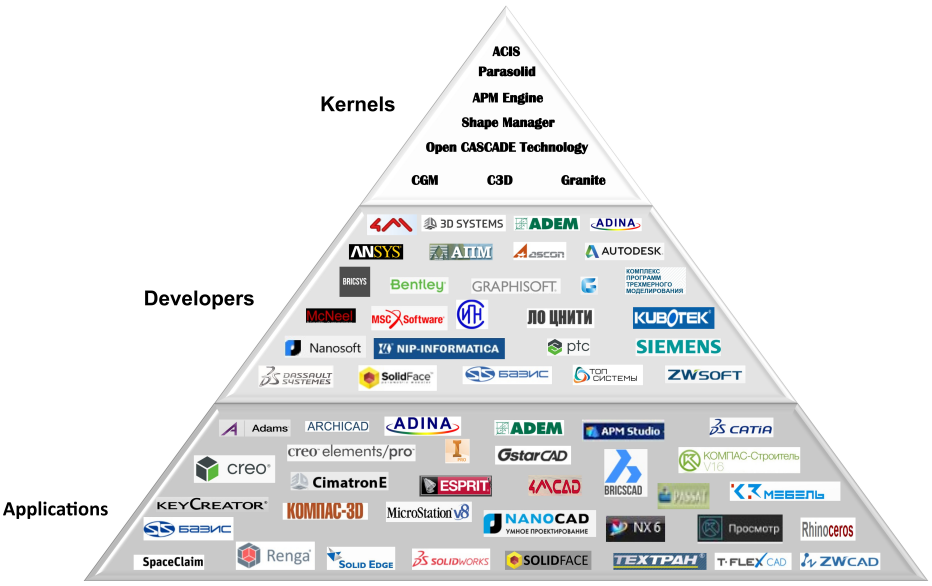


Figure 2.13: Geometric modeling kernels.

The CSG technique describes a solid model using Boolean operations (i.e., intersection, union, and difference) and geometric transformations of solid primitives, such as blocks, triangular prisms, cylinders, spheres, cones, and torus. The topology and geometry information are stored implicitly and must be retrieved from the primitives when required. The tree structure is used in the CSG approach to organize the relationship between the solid primitives. Figure 2.14 shows the CSG structure of a part model in which the model on top of the tree is represented by a Boolean union of a block and a cylinder. The major benefit of the CSG structure is that the features can be arranged simply based on the order of model construction and deconstruction. Nevertheless, the CSG approach can only describe limited types of part geometry, and its implicit representation and non-uniqueness limit its application in FR. Usually, CSG is combined with BRep for solid modelling in CAD/CAM packages. In comparison, BRep explicitly represents a solid model through a collection of connected surface elements, and it uses low-level information to describe the geometry and topology of the solid model in terms of its boundary. The geometric entities that describe the topology include faces, edges, and vertices, whereas the geometry of the

solid is described by surfaces, curves, and points. Compared with the CSG, the BRep is more flexible, and it has a much more extensive set of operations.

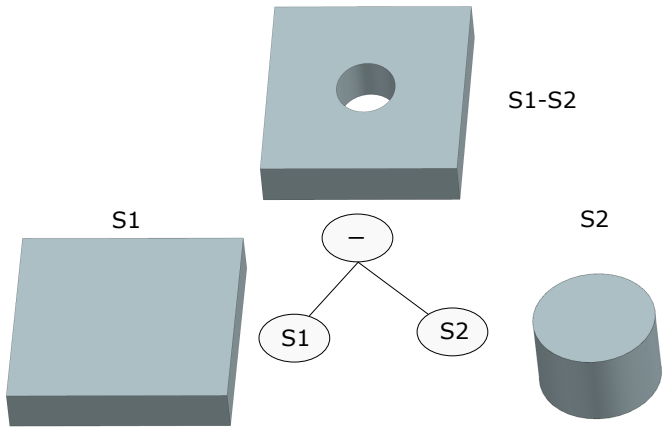


Figure 2.14: The CSG structure of a solid model.

### 2.4.2 Discrete Geometry

Four common types of discrete geometry exist: point cloud, depth map, volumetric representation, and polygonal mesh. The point cloud represents the part geometry by unorganized discrete points on its surface. In comparison, when representing an object by a depth map, a uniform grid is created based on the dimensions of the object and each pixel in the grid contains the depth information of the part along a specific direction. Specifically, the polygonal mesh can be considered a special type of BRep, where all the faces of the part model are defined by planes. As shown in Figure 2.15, the model is called triangle mesh when all the planes in the polygonal mesh are triangles. The triangle mesh (a "tessellated" surface) is used extensively in additive manufacturing because of the simplicity of its data format and its capability to interchange data between CAD/CAM systems. Most modern CAD packages have the function to convert a solid model into a triangle mesh. Unlike all other geometry representations, the volumetric approach describes objects by the volumetric data instead of a surface. In this approach, the space constrained by the dimensions of the object is subdivided into several small elements, called voxels. Each voxel defines a

specific volume of the part. It can also be assigned with additional attributes, such as the material density and color to represent the various states of the part at the location.

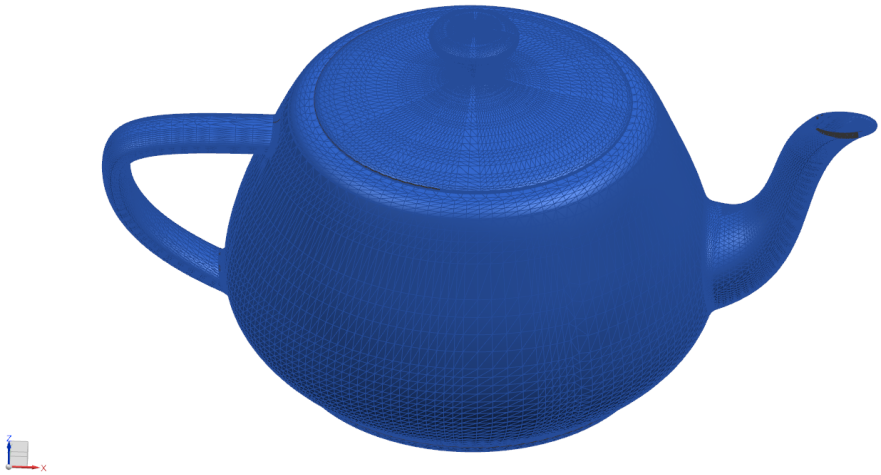

Figure 2.15: Triangle mesh of Utah Teapot.

### 2.4.3 Tool Path Planning on Discrete Geometry

Compared with the analytic geometry, the discrete geometry has much more flexibility in representing parts with complex geometry; its geometric information is also much simpler although the precision of the discrete geometry is limited. In addition, one way in which the discrete geometry is superior to the analytic geometry when used in CNC machining is its computational efficiency in gouging detection, where the gouging between the cutting tool and the workpiece is detected during the machining process [69]. And hence the triangle mesh is widely used in CAM software for gouging detection.

With the development of 3D scanning techniques in metrology and the 3D printing techniques in rapid prototyping, the discrete geometry representation is becoming even more popular than ever in the manufacturing industry. Because of its high flexibility in representing complicated geometry, the discrete geometry has been ex-

tensively applied in machining freeform and compound surfaces. Numerous tool path planning methods [69–73] have been developed for machining discrete geometry in the past two decades. Kim and Yang [69] presented an iso-planar approach to generate 3-axis CNC tool paths from triangle meshes. To keep a constant scallop height, the CL surface is maintained according to the defined deformation vectors. Sun et al. [70] introduced a new iso-parametric method for generating tools path for cutting freeform surfaces based on triangle mesh. In their approach, a parameterization procedure is applied to the triangle meshes to get the parameters of the surface. The tool paths are subsequently generated based on the parameter lines obtained from the surface. Xu, Sun and Wang [72] developed a method to generate contour parallel tool paths by offsetting the curves formed by the boundary of the triangle mesh. To handle the self-intersections problem in offsetting the boundary curve, the mesh flattening technique is utilized.

With the advent of General-Purpose Graphics Processing Unit (GPGPU), the computing power of modern computers has been greatly enhanced, and several studies have explored the utilization of GPGPU in automated tool path planning based on discrete geometry [74–77]. Tarbutton et al. [74] proposed a voxel-based tool path generation method under the parallel processing framework. The part was first voxelized into a voxel model, and the tool path was then generated by updating the Z height in each CL point in the template tool path to a minimum value that the tool is not gouging with the surface of the part. Ray casting was employed to detect the part surface. Konobrytskyi et al. [75] developed an approach for generating the tool path for 5-axis CNC machining based on highly parallel discrete volumetric geometry representation. A two-level tree volume representation, which was well-constructed to match GPGPU, was proposed to be the underlying geometry of the part. The 5-axis CNC tool path was generated by successively performing the 3D contour offsetting algorithm the entire surface of the part is covered.

Tool selection plays an important role in the reduction manufacturing time of CNC machining. A GPU-accelerated automatic generalized cutter selection for finishing freeform surfaces based on depth map was presented by Balabokhin et al. [76]. In this approach, the tool performance was determined as a function of the area that can be milled by the tool in the given cutter location point below the tolerance surface and the tool feed rate was evaluated at each surface point for each tool. The tool with the best performance was selected as the optimal tool for each surface point. The GPU was used to perform the most time-consuming calculations in the proposed algorithm in parallel, which are building the depth of cutter maps, building "finished area" maps, and building the surface finishing performance maps.

In summary, although several methods were developed to generate tool paths for discrete geometry, the feature-based automated tool path planning for discrete geometry has been little studied.

# CHAPTER 3: AUTOMATED MACHINING FEATURE RECOGNITION

This chapter introduces a new method for region segmentation on triangle meshes. It also describes a hybrid method of graph-based and volume decomposition approaches for recognizing the STEP-NC machining features from the segmented regions.

## 3.1 Region Segmentation

Unlike recognizing machining features from analytic geometries which are composed of a limited collection of connected surfaces, the feature recognition from discrete geometries handles massive discrete data. As shown in Figure 3.1, the parts in this study are represented by triangle meshes consisting of a large number of triangles. To facilitate the recognition of machining features from the triangle meshes, the region segmentation is an essential step to reduce the complexity of the problem by merging the triangles into different regions.

In this study, a new face-based robust mesh segmentation method involving three steps is introduced to segment the mesh into several regions representing different surface shapes. In the first step, a primary shapes region segmentation approach is utilized to detect primitives with the triangulated mesh. With the presence of noise in the mesh, a primitive may be segmented into several regions. Thus, an additional step is subsequently taken to merge the regions with similar surface parameters. Finally, a final islands region segmentation is applied to merge the isolated regions into freeform surface regions.

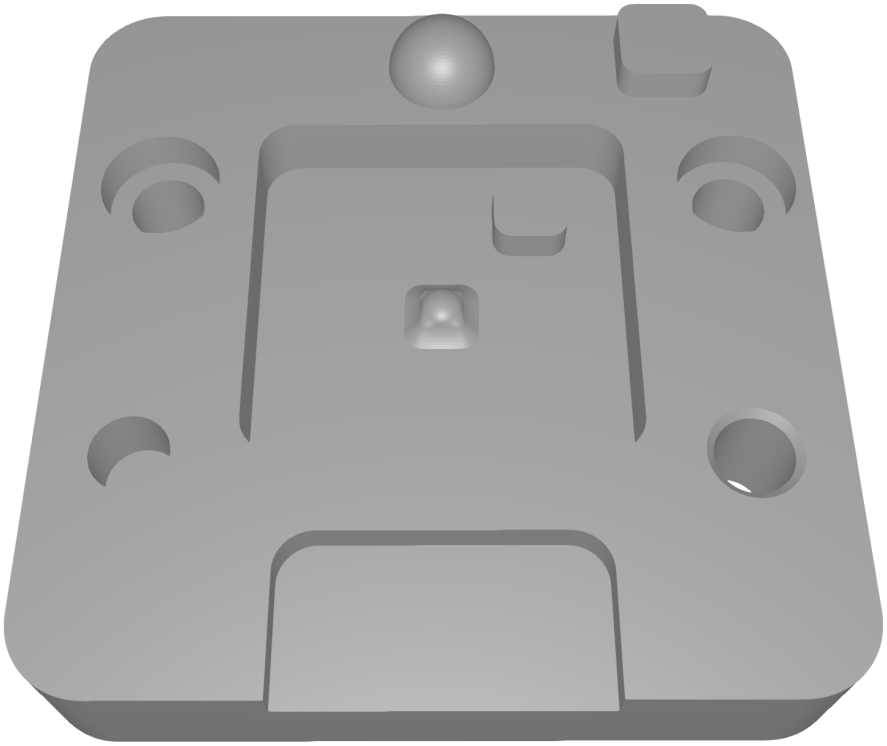

Figure 3.1: An example part represented by a triangle mesh.

### 3.1.1 Primary Shapes Region Segmentation

For most meshes from the mechanical parts, the underlying shapes for the surface regions are primitives. The segmentation of primitives can significantly reduce the number of elements be handled in the following step and thus the machining feature recognition from triangle meshes can be simplified considerably. Therefore, a region segmentation algorithm is developed in this work to segment the meshes into different regions, called primary shape regions. Each region represents a primitive shape, i.e., a plane, a cylinder, a sphere, a cone, or a torus.

Initially, an adjacency graph is built for the target triangle mesh. As shown in Figure 3.2(a) and (b), each triangle has three edges and each edge are shared by two triangles in a manifold mesh. For each triangle, the triangles that share a common edge with it are its adjacent triangles. To create the adjacency graph, the adjacency

information is first detected and saved for each individual triangle in the mesh. The adjacency information for all the triangles in the mesh then forms the adjacency graph as shown in Figure 3.2(c). The adjacency graph facilitates the understanding of the mesh topology for a computer algorithm. Starting from any single triangle in a manifold mesh and moving its adjacent triangles, all the triangles can be gradually visited. As shown in Figure 3.2(b), the mesh of a cube contains 2 triangles on each of its 6 faces for a total of 12 triangles. For example, if the beginning triangle is the triangle $T_1$, its neighboring triangles, $T_2$, $T_3$, and $T_4$ can be discovered. Then moving to the next triangle $T_2$, its adjacent triangles $T_5$ and the other triangle can be also visited. By performing the operation successively for each triangle in the mesh, all 12 triangles can be detected.

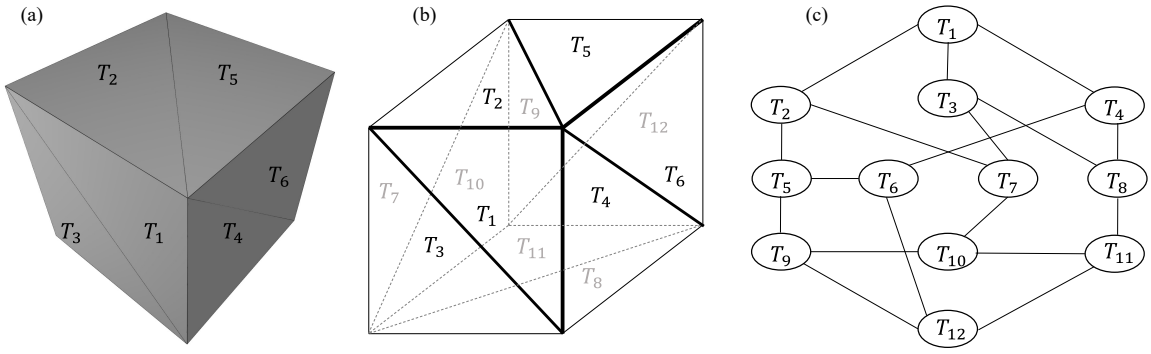


Figure 3.2: Triangle mesh of a cube and its adjacency graph: (a), triangle mesh; (b), boundaries and triangles within the mesh; (c), adjacency graph of the triangles.

Based on the adjacency graph, a region segmentation algorithm is gradually applied to detect the primary shape regions. The planar regions are first segmented as the plane is the most common shape in mechanical parts. When a bounded plane is composed of a few connected triangles, all the triangles on the plane should have the same surface normal. This intrinsic property of the plane is employed to segment the planar regions within a mesh.

**Algorithm 3.1** Region Growing for Planar Region Segmentation

**Input :** $mesh$ is the triangle mesh

**Input :** $graph$ is the adjacency graph of the triangles in the mesh

**Input :** $\alpha_t$ is the angle threshold for grouping the triangles into planes

1: $regions = []$
2: **for** $i \in 1 : n_{tris}(mesh)$ **do**
3: $seed\_tri = mesh.tris[i]$ ▷ get a triangle within the mesh
4: **if** $!visited(seed\_tri)$ **then**
5: $region = []$, $queue = []$, $\overrightarrow{n_{average}} = \overrightarrow{n_{seed\_tri}}$
6: add $seed\_tri$ to $region$, add $seed\_tri$ to $queue$
7: **while** $!is\_empty(queue)$ **do**
8: $query\_tri = pop(queue)$
9: $neighbors = get\_adjacent\_tris(query\_tri, graph)$
10: **for** $neighbor\_tri \in neighbors$ **do**
11: $\beta = get\_angle(\overrightarrow{n_{neighbor\_tri}}, \overrightarrow{n_{average}})$
12: **if** $\beta < \alpha_t$ and $!visited(neighbor\_tri)$ **then**
13: add $neighbor\_tri$ to $region$, add $neighbor\_tri$ to $queue$
14: $\overrightarrow{n_{average}} = get\_average\_normal(region)$
15: **end if**
16: **end for**
17: **end while**
18: **end**
19: add $region$ to $regions$
20: **end if**
21: **end for**
22: **return** $regions$

The region growing algorithm shown in Algorithm 3.1 is executed to group the

triangles into the planar regions. Starting from a seed triangle, its adjacent triangles are identified based on the adjacency graph and the corresponding geometric information from these triangles is extracted. For each of the adjacent triangles, the angle between its face normal and the average face normal of the region is computed. If the angle is within a threshold and the triangle has not been visited yet, then the adjacent triangle is added to the queue data structure. Initially, the average face normal is equal to the face normal of the seed triangle. Once a new triangle is added to the planar region, the average face normal is updated as the average value of the face normals of all the triangles in the region. This process is repeated for the next triangles in the queue until the queue is empty. By the end of the operation, a planar region is segmented from the mesh. If there are still triangles that have not been visited, the whole operation is performed again until all the triangles are visited. Based on the region growing algorithm, the triangle mesh is decomposed into multiple planar regions and the parameters of the planes are also obtained by fitting a plane to the points in each planar region. A planar region can have only a single triangle or multiple triangles.

In the next stage, the cylindrical regions are segmented by using a similar region growing algorithm. The Gaussian image approach [78] is utilized to determine if the neighboring facets or regions are on the same cylinders. For an arbitrary cylinder, its Gaussian image is a great circle on the Gaussian sphere. As the cylinder in the triangle mesh is composed of triangles, the face normals of the triangles on a cylindrical region form a part of the great circle on the Gaussian sphere. Before starting the segmentation of cylindrical regions, a new adjacency graph is created based on the adjacency relationships between the different planar regions. After the triangles in the mesh are merged into planar regions, new boundaries form for each of the planar regions. Each region boundary is formed by multiple triangle edges. Two planar regions are labeled as adjacent regions if they share at least one common edge

along their boundaries. After the adjacency graph is created for the planar regions, a similar region growing algorithm is applied to segment the cylindrical regions. Taking a seed region to start, its neighboring regions are found. For each tripod of the three regions including the seed region itself, a cylindrical region test is applied. The three planar regions are on the same cylinder if two conditions can be fulfilled: first, taking one triangle on each region into account, there is a constrained plane passing thru the origin of Gauss sphere such that the distances of the three face normals of the triangles to the constrained plane are below a threshold value; second, the difference between the radius of the circle fitted on the 3 points defined by the face normals and the radius of the Gauss sphere is below a threshold value. Once three neighboring regions are determined to be on the same cylinder, they are merged into a cylindrical region, and the parameters of the cylinder are then computed by fitting a cylinder to the points in the region. The region growing algorithm keeps growing and merging additional adjacent regions into the same cylinder if the surface normal of the additional region is perpendicular to the axis of the cylinder and the projected distance from the center of the cylinder to any of the points in the region on the axis is equal to the radius of the cylinder.

The conical, spherical, and toroidal shapes are subsequently detected by following similar segmentation approaches, as previously stated. In the segmentation of each primitive shape from the mesh, a primitive shape is first fitted to the adjacent regions and the region growing algorithm is used successively to merge the regions on the same primitive shapes. Based on the primary shapes region segmentation algorithm, the triangle mesh can be segmented into primary regions. Figure 3.3 shows the result of detecting the primary shapes from the mesh example illustrated in Figure 3.1. The triangle mesh is tessellated from a CAD model that contains several planar shapes, a few cylindrical shapes, a spherical shape, a conical shape, and a freeform shape. Among these shapes, the planar, cylindrical, spherical, and conical shapes are

segmented correctly from the mesh into different primary regions while the freeform shape in the middle of mesh is decomposed into several primary regions, which can be merged by the method introduced in the following section into a single surface region.

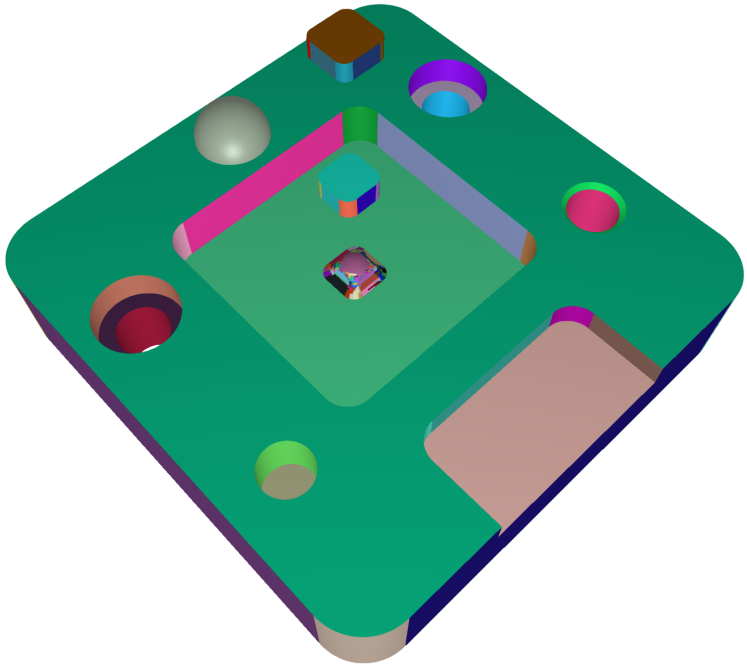

Figure 3.3: Segmentation result of the example triangle mesh after applying the primary shapes region segmentation algorithm.

### 3.1.2 Merging Primary Shape Regions

Most existing segmentation approaches are sensitive to noise. The triangle mesh is usually assumed to be noiseless in these methods. However, in the manufacturing industry, the triangle mesh can be obtained from different sources. For example, the mesh can be generated from machining simulation in CAM software. The mesh can also be 3D scanned data from a manufactured part. In these cases, the resolution of the discrete geometry may be low and even worse, the data may contain noise. Thus, the primary shapes detection approach may fail on these cases. As shown in Figure 3.4(a), the cylinder in a low resolution mesh is segmented as three independent

cylindrical regions in the process of the primary shapes region segmentation.

**Algorithm 3.2** Merging Primary Shape Regions

```
    Input : regions is the primary shape regions
 1: graph = build_adjacent_graph(regions)
 2: merged_regions = []
 3: for region ∈ regions do
 4:     a_regions = get_adjacent_regions(region)
 5:     for a_region ∈ a_regions do
 6:         if is_mergeable(region, a_region) then
 7:             mergeable_regions = region_growing(a_region, graph)
 8:             merged_region = merge_regions(mergeable_regions)
 9:             add merged_region to merged_regions
10:         end if
11:     end for
12: end for
13: return merged_regions
```

To improve the robustness of the primary region segmentation algorithm, an additional step is taken to merge the neighboring regions that are on the same shape after detecting the primary shapes. As shown in Algorithm 3.2, an adjacency graph is first built based on the boundaries of the regions. If two neighboring regions with the same shape type have a similar parameter, then the two regions can be merged into a single primary shape region. For example, in the case of a cylinder, two cylindrical regions can be merged into a single region if the angle between their axes and the difference between their radii are both within the defined threshold. Once two mergeable regions are detected, similar to the detection of primary shapes, a region growing algorithm is utilized to grow the adjacent region and identify all the adjacent regions with a similar shape property until no new region can be added. Finally, the mergeable regions are combined into a single primary shape region.

The segmentation result of the mesh example after merging the primary shape regions is shown in Figure 3.4(b). The three cylindrical regions detected in the procedure of primary shapes region segmentation are merged into a single cylindrical region.

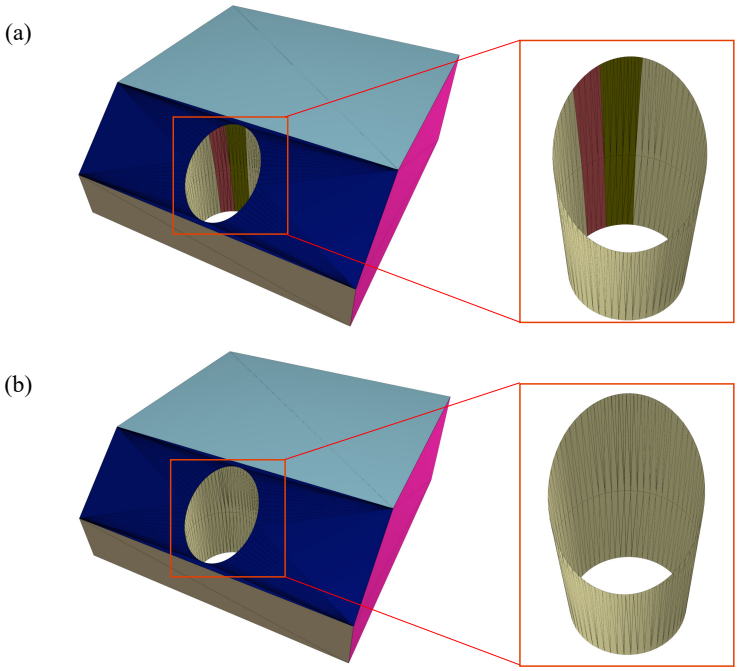


Figure 3.4: Comparison between the results of primary shapes region segmentation and the two steps region segmentation methods on a triangle mesh with a low resolution: (a), primary shapes region segmentation; (b), two steps region segmentation.

### 3.1.3 Final Islands Region Segmentation

With the algorithms for detecting and merging primary shape regions, the primary shapes within the triangle mesh can be identified. However, some mechanical parts are designed to contain freeform shapes. As the mesh is decomposed into multiple primitive shapes after the primary shapes segmentation and the freeform shapes cannot be described by any single primitive shape, the freeform shapes are detected as a few primary shape regions. As shown in Figure 3.5(a), the planes in a mesh example are all detected correctly as the planar regions. Nevertheless, the freeform shape in the middle of the mesh is segmented into several primary regions instead of a whole region.

**Algorithm 3.3** Final Islands Region Segmentation

**Input :** $regions$ is the primary shape regions

```
1: freeform_regions = []
2: graph = build_adjacent_graph(regions)
3: i_regions = get_regions_with_inner_boundary(regions)
4: for region ∈ i_regions do
5:     boundaries = get_inner_boundaries(region)
6:     for boundary ∈ boundaries do
7:         c_regions = get_connected_regions(graph, boundary, region)
8:         for t_region ∈ c_regions do
9:             f_regions = get_connected_regions(graph, region)
10:            add f_regions to c_regions
11:        end for
12:        freeform_region = merge_regions(c_regions)
13:        add freeform_region to freeform_regions
14:    end for
15: end for
16: return freeform_regions
```

To detect the freeform shapes within a discrete geometry, a final islands region segmentation algorithm, as described in Algorithm 3.3, is applied after the primary shapes segmentation. As the recognition of general freeform shapes that can intersect with primary shapes or other freeform shapes is very complex, the freeform shapes are assumed to be isolated and located on top of a single primary shape to reduce the complexity of this problem. In the beginning of the algorithm, a new adjacency graph is created for the regions in the mesh after the primary shape regions are detected. The primary shape regions that contain inner boundaries are then detected. A region can have multiple boundaries that are categorized as inner and outer boundaries. A

boundary is classified as an outer boundary if it can enclose all the projected points when projecting the points on the region and the boundary onto a plane along the surface normal of the region (for planar region) or the axis of the region (for other types of regions). Otherwise, the boundary is labeled as an inner boundary. As shown in Figure 3.5 (a), the red dotted curve is the outer boundary of the green planar region and the black solid curve is the inner boundary of the region. In the subsequent step, the freeform shapes are detected based on the inner boundaries. For each inner boundary of these primary regions, the regions connected to the inner boundary are recognized. If the number of connected regions is more than one, these connected regions are probably on a freeform shape. Once these regions are detected, their adjacent regions are successively identified. The process is repeated until no new adjacent region can be found. All the detected regions are eventually merged into a single region and marked as a freeform shape region. As shown in Figure 3.5 (b), the freeform shape is detected successfully from the mesh example based on the final islands region segmentation algorithm.

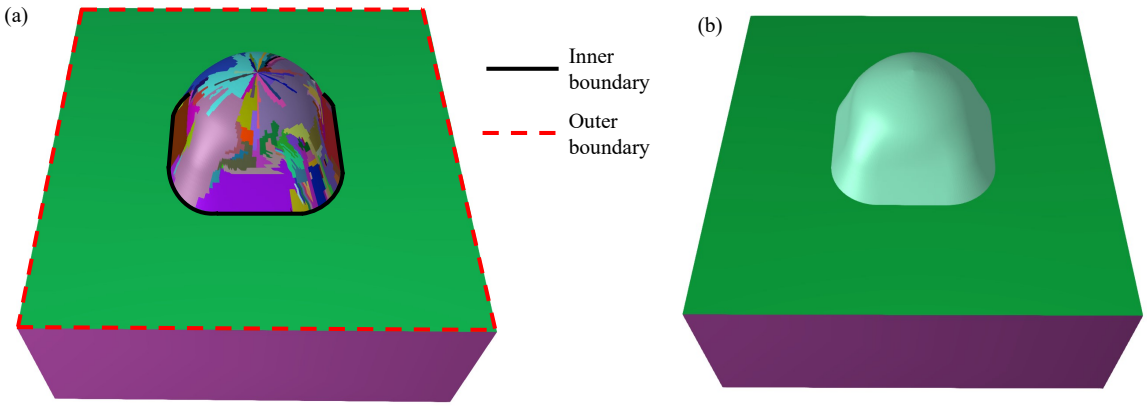


Figure 3.5: Comparison between the segmentation results of an example mesh before and after applying the final islands region segmentation: (a), before; (b), after.

In summary, a new region segmentation algorithm with three steps is developed in this dissertation to detect primitive and isolated freeform shapes within triangle meshes. As shown in Figure 3.6, the algorithm can decompose a triangle mesh into

different surface regions. Each region either represents a freeform or a primitive shape.

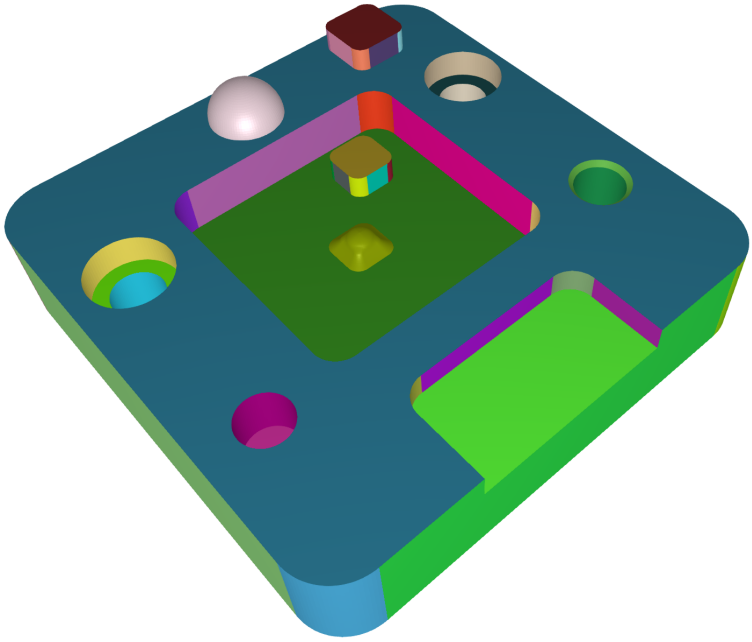

Figure 3.6: Final segmentation result of the example part after applying the three steps region segmentation algorithm.

## 3.2 Machining Feature Recognition

The features have various definitions based on their areas of application. As the goal of this work is to automate the tool path planning process for CNC machining, the machining features defined in STEP-NC are the target features to be recognized from the triangle meshes.

As reviewed in the background chapter, graph-based, rule-based, hint-based, volume decomposition, and artificial neural networks are the five most used method for feature recognition. However, each method has its own benefits and drawbacks. In this study, a hybrid feature recognition approach combining the advantages of the graph-based and the volume decomposition methods is utilized to recognize the STEP-NC machining features from the detected surface regions. The details of the machining feature recognition method are described in the following sections.

### 3.2.1 Graph-based Approach

One of the major drawbacks of the traditional graph-based approach is its failure to guarantee that the recognized features are all machinable [1]. To resolve this problem, Xu et al. [79] proposed a new graph-based feature classification for recognizing STEP-NC machining features from triangle meshes. The triangle mesh was first segmented into surface regions, and the machining features are subsequently recognized from the surface regions. In their graph-based approach, a new set of graphs was defined for STEP-NC machining features to ensure manufacturability. Based on these graphs and some geometric hints of the inner/outer boundaries and concave/convex edges, the STEP-NC features are recognized. This approach is used in this study to identify the simple machining features with limited interactions with other machining features.

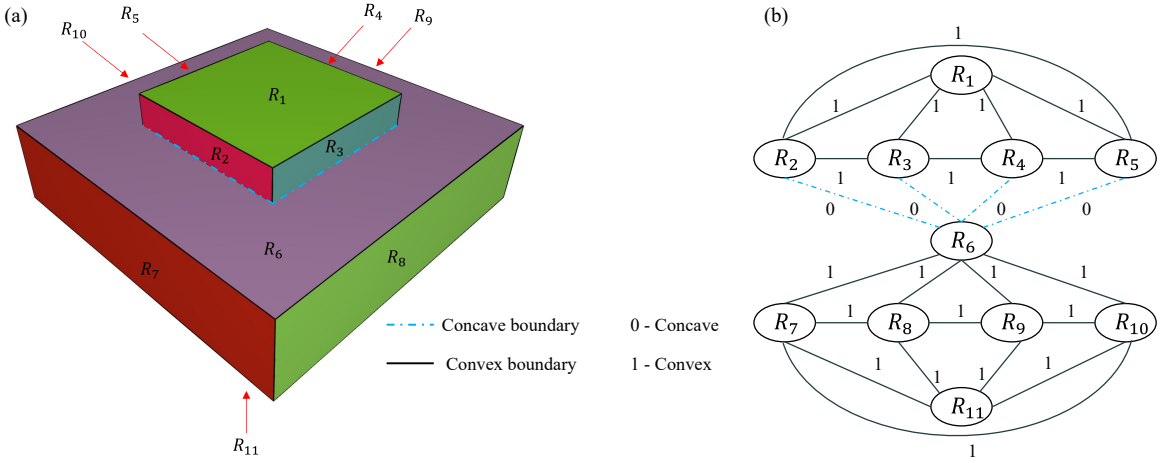


Figure 3.7: An example part with 11 surface regions after segmentation and the ARAG of the surface regions: (a), the surface regions of the part; (b), ARAG.

In this initial step, an attributed region adjacency graph (ARAG) is created based on the adjacency relationships between the detected surface regions. The ARAG is composed of a set of node and edges. Each node in the ARAG represents a surface region while each edge indicates that the two regions connected by the edge are adjacent. In addition, each edge is assigned with an attribute showing whether the two regions share a convex triangle edge or not. As shown in Figure 3.8, an edge is

concave when the angle between the normals of two faces sharing the edge is less than 180°while the edge is convex if the angle is larger than 180°. In a triangle mesh, an edge can be shared by two triangles. The convexity test on a triangle edge is executed by utilizing the vertices and face normal information of two neighboring triangles that share the edge. As shown in Figure 3.9, triangles A and B share the edge $P_cP_d$. To test whether the edge $P_cP_d$ is concave or convex, Equation 3.1 is utilized. Figure 3.9 shows the ARAG of a part example with 11 surface regions after segmentation. Among the adjacent regions, regions $R_2$, $R_3$, $R_4$, and $R_5$ share a concave triangle edge with the region $R_6$. Thus, a value of zero is assigned to the edges connecting the nodes representing these regions. Other adjacent regions all share a convex triangle edge and, hence, a value of one is assigned to these edges in the ARAG.

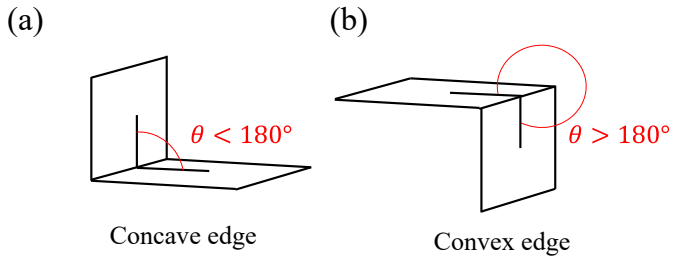


Figure 3.8: Concave and convex edges: (a), concave edge; (b)convex edge.

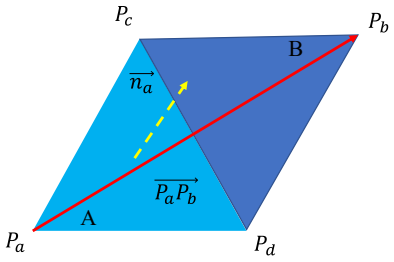


Figure 3.9: Convexity test on a triangle edge.

$$\begin{cases} \overrightarrow{P_aP_b} \cdot \overrightarrow{n_a} <= 0, & \text{convex edge,} \\ \overrightarrow{P_aP_b} \cdot \overrightarrow{n_a} > 0, & \text{concave edge,} \end{cases} \tag{3.1}$$

where $\overrightarrow{n_a}$ is the face normal of the triangle A, $P_a$ and $P_b$ are the vertices of triangles

A and B, respectively, and $\overrightarrow{P_aP_b}$ is the vector from vertex $P_a$ to vertex $P_b$.

In the pure graph-based approaches, the complete graph of the part is decomposed into sub-graphs and the features are recognized by matching the pattern of each sub-graph with the template. However, the graph may be difficult to decompose and analyze when there are thousands of nodes and edges in the graph. In the STEP-NC machining features, some geometric "hints" can be used to infer the locations of machining features so that the sub-graph of the features can be more easily extracted. The holes and closed pockets in STEP-NC are usually connected with a region's inner boundaries that only have convex edges. Bosses are located at a region's inner boundaries with only concave edges. The profile features are connected to the outer boundaries of the regions. As shown in 3.10, region $R_6$ has both an inner and an outer boundary, and all the edges on the inner boundary are concave. Hence, there might exist a boss feature composed of the region $R_6$'s adjacent regions that are connected with the inner boundary.

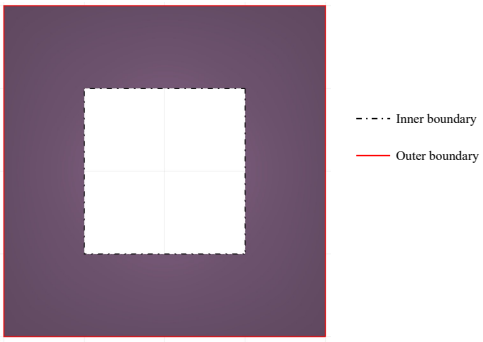


Figure 3.10: Inner and outer boundaries of a surface region.

After obtaining the regions connected with the inner or outer boundaries, the graph for the adjacent regions is created and compared with the template graph of feature. Once a feature is recognized, its related regions are labeled as extracted and will not be considered when detecting other machining features. As shown in Figure 3.11, the ARAG between regions $R_2$, $R_3$, $R_4$, and $R_5$ matches the pattern of a boss feature, and thus they are extracted and marked as a boss.

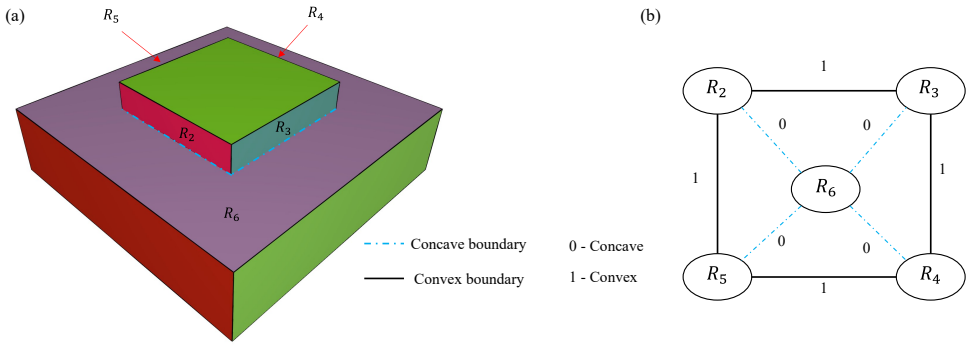


Figure 3.11: The boss feature and its ARAG: (a), regions related to the boss feature; (b), ARAG of the boss feature.

By using the graph-based approach, the isolated STEP-NC machining features can be successfully recognized from the surface regions. As shown in Figure 3.12, the feature recognition algorithm detects all the machining features from the triangle mesh example, including 4 faces, 2 bosses, 3 through holes, 3 blind holes, 1 pocket, 1 open pocket, 1 countersink, 1 region, 1 spherical cap, and 1 profile. In addition to the independent features, the combination of a blind hole with a flat bottom and a through hole with the same axis and a smaller radius with the blind hole is labeled as a counterbore hole. The combination of a countersink and its coaxial hole is marked as a countersink hole.

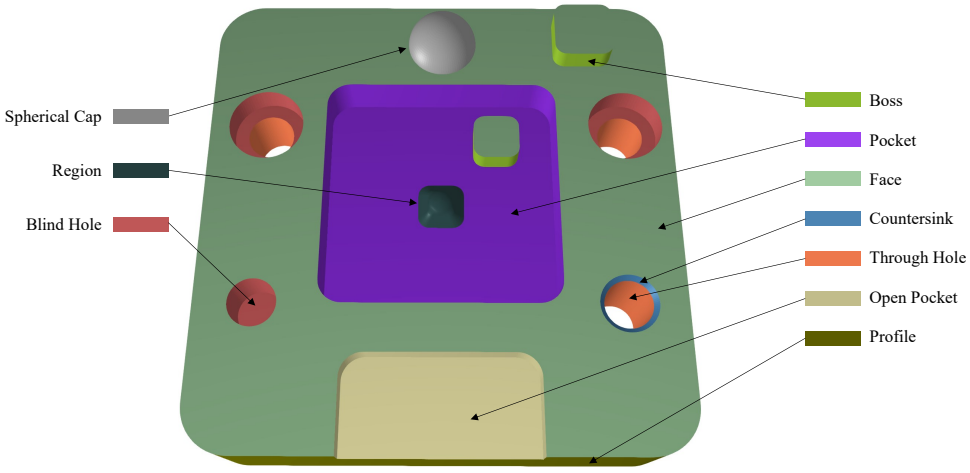


Figure 3.12: Result of machining feature recognition from the example part.

### 3.2.2 Volume Decomposition

Although the graph-based approach can extract all the isolated machining features, it relies heavily on the adjacency graph and the hints of "inner/outer" boundaries. The pure graph-based approach fails to identify some features in the mesh where the inner/outer boundary loops are not well formed. Figure 3.13(a) shows the segmentation results of an example part. In this part, the profile of the pocket intersects with multiple regions including two fillet regions and a face region, so, there is no inner boundary in a single region to locate the pocket. Thus, the graph-based approach fails to recognize the pocket feature from the part. In this study, a convex hull volume decomposition approach [46] is employed to identify the machining features from unrecognized regions after the graph approach is applied. In the first step of the volume decomposition approach, a convex hull is computed for the whole mesh and then the volume defined by the mesh is subtracted from the convex hull. This process is repeated on the resultant volumes until the net volume is equal to zero. Once a negative volume is extracted, the unrecognized regions corresponding to this volume are merged and identified as a feature. After applying the volume decomposition, the pocket is successfully recognized from the example part, as shown in Figure 3.13(b).

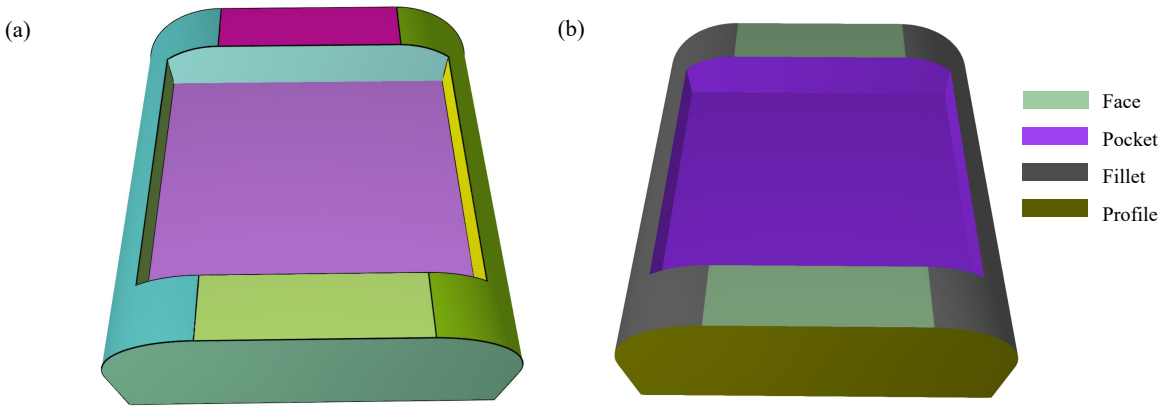


Figure 3.13: An example part without inner boundary: (a), segmentation result; (b), feature recognition result.

### 3.2.3 Hybrid Approach for Feature Interactions

Feature interactions are the cases that multiple machining features intersect with each other so that the geometrical shapes of the original features are altered. The ability to handle feature interactions is an important performance indicator for a feature recognition system. As stated in the preceding sections, the graph-based approach is sensitive to interacting features; hence, a hybrid approach is utilized to handle feature interactions. After the graph-based approach is employed to identify the isolated features without intersecting with other features, the volume decomposition method is applied to extract the volume defined by the intersecting features. Finally the graph-based approach is used again to decompose the volume into multiple features. Figure 3.14(a) shows an example part with feature interactions. The pocket in the middle of part intersects with three slots so that the inner boundary on the top face is broken and the graph for the pocket is destroyed. Similar to the example part shown in Chapter 2, which was tested on the NX software, the graph-based approach cannot recognize the pocket feature. However, with the hybrid approach, the slots and the pocket are successfully extracted from the part. The result of the feature recognition is shown in Figure 3.14(b).

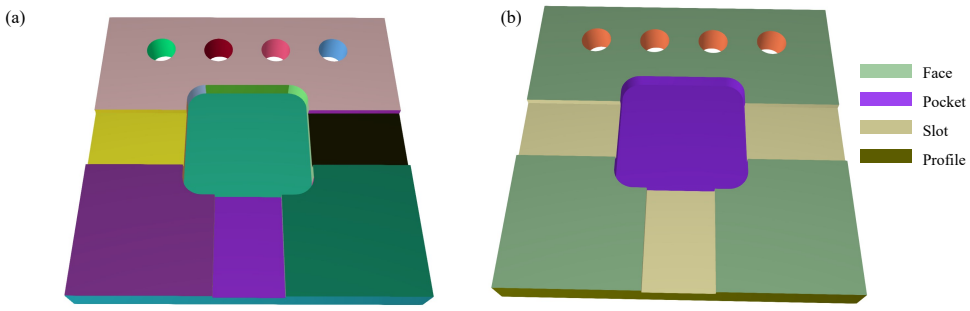


Figure 3.14: An example part with feature interactions: (a), segmentation result; (b) feature recognition result.

## 3.3 Case Study

Several real mechanical parts with different machining features were tested to validate the effectiveness of the proposed method. Figures 3.15 through 3.18 show the results of applying the proposed feature recognition method to four mechanical parts with a triangle mesh as the underlying geometry representation. In the first part, 29 machining features were identified correctly from the part, including 11 bosses, 1 closed pocket, 13 faces, 3 holes, and 1 profile. In the second part, 13 machining features with 4 through holes, 4 blind holes, 2 curved faces, 2 planar faces, and 1 profile were recognized. In the third part, a total of 101 features were recognized: 87 faces, 5 through holes, 6 blind holes, 1 through pocket, 1 boss, and 1 profile. In the last part, 26 machining features were recognized, including 4 faces, 10 through holes, 9 through pockets, 1 boss, and 2 profiles.

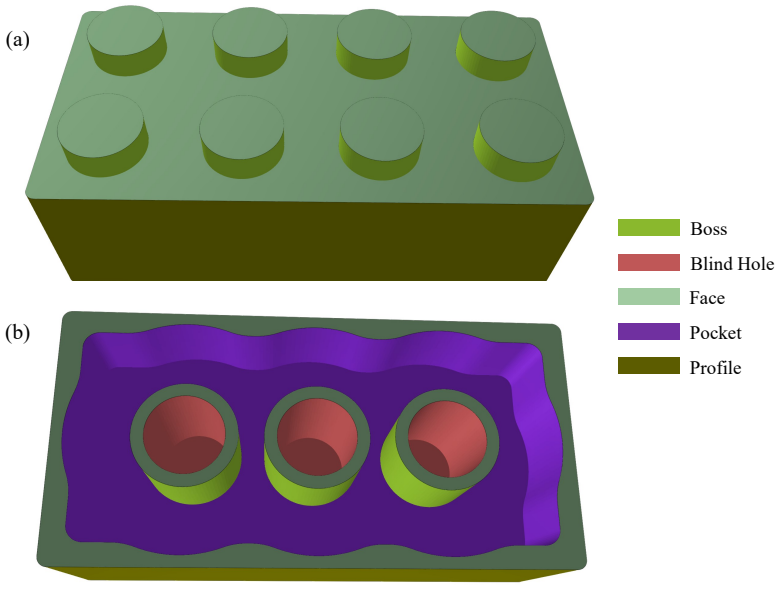


Figure 3.15: Result of machining feature recognition from example part 1 with different views: (a), top view; (b), bottom view.

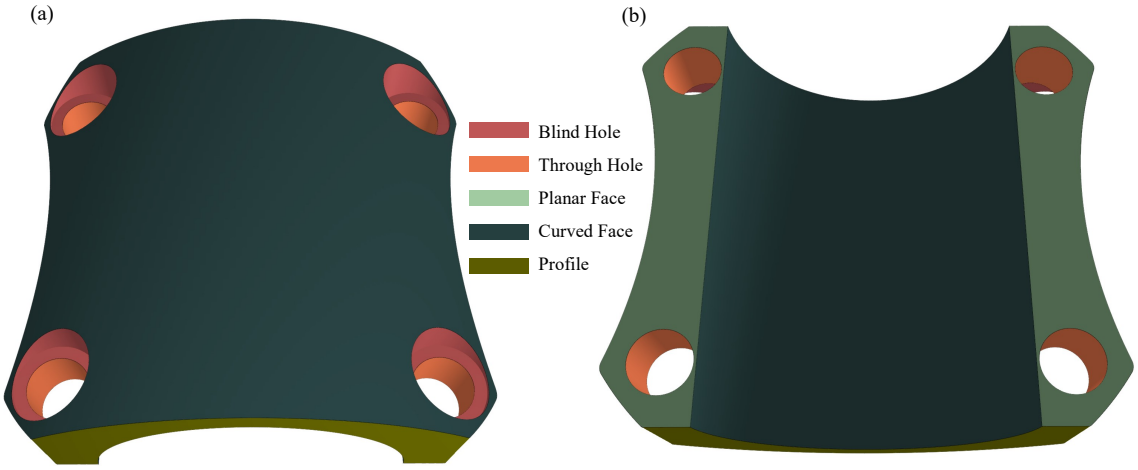


Figure 3.16: Result of machining feature recognition from example part 2 with different views: (a), top view; (b), bottom view.

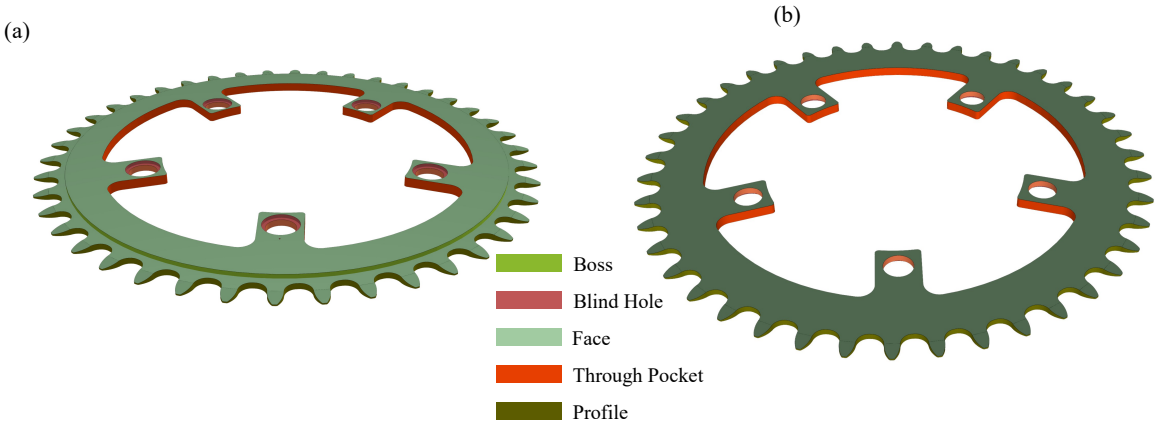


Figure 3.17: Result of machining feature recognition from example part 3 with different views: (a), top view; (b), bottom view.

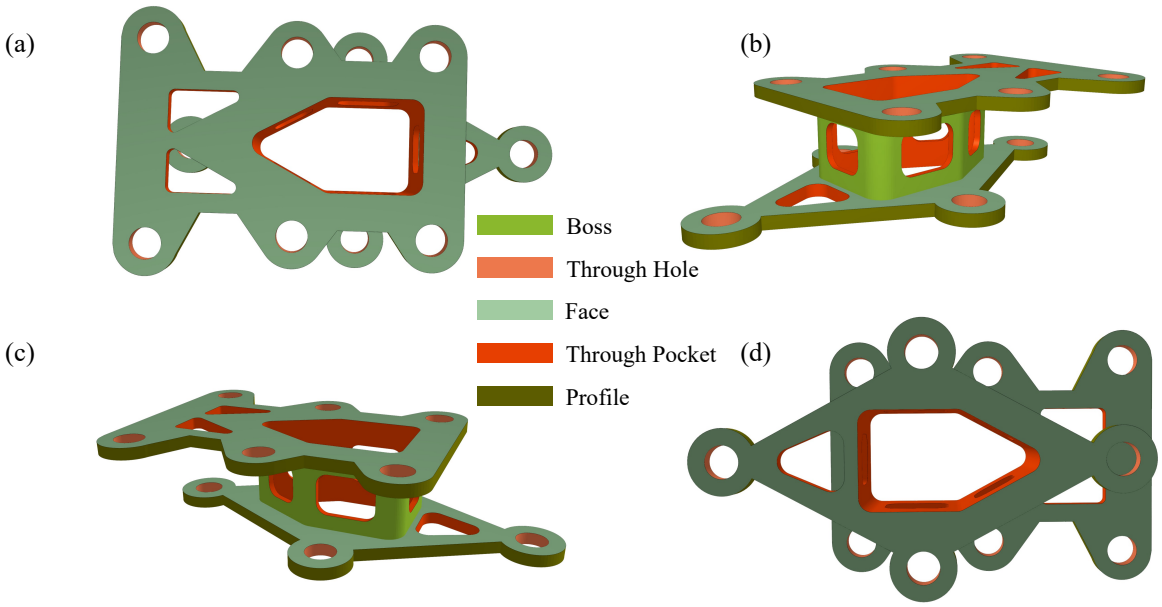


Figure 3.18: Result of machining feature recognition from example part 4 with different views: (a), top view; (b), side view 1; (c), side view 2; (d), bottom view.

# CHAPTER 4: AUTOMATED MILLING TOOL PATH GENERATION

Generally, the task of machining mechanical parts is to remove the extra material outside the volume bounded by the geometry of the part designs from their raw stock material. The machining process is usually arranged in two stages: roughing and finishing operations. In the roughing operation, most of the material is cut from the stock with a large cutter. A smaller cutter is subsequently utilized in the finishing operation to machine the uncut and the corner regions that the large cutter cannot reach to ensure the accuracy of the final production. This chapter introduces the automated tool path planning strategies for the roughing operations. In addition, the method for generating the tool paths for the facing operation, which is another common machining operation, is described.

## 4.1 Roughing Tool Path Generation

The roughing operation is designed to clear a large quantity of material from the stock efficiently. In the roughing operation of the traditional CAM packages, the part is usually cut layer by layer from the top to the bottom along a user-defined cutting direction. For each layer, tool paths are generated for a selected cutting tool to clear the material within defined boundaries on that layer. However, this approach cannot guarantee all the features being machined to states that are ready for the finishing operation because different features maintain different tool accessibility. Although a small cutter can be used in roughing to remove most of the material from all the machining features, the total machining time would be increased considerably. In this study, a different strategy is developed to rough the part in a feature-based tool path planning system. The roughing tool paths are generated independently for each

machining feature and then sequenced to machine the whole part.

### 4.1.1 In-process Model

The initial step in the roughing tool path planning is to identify the volume of material to be removed. This task is completed by comparing the difference between the current state of the workpiece with the target state of the workpiece. The current workpiece can be defined by the stock or the workpiece after the previous machining operations. To keep tracking the state of the workpiece, an in-process model (IPM) is an essential feature for an automated tool path planning system. Basically, the IPM can be created by different types of geometry representation, such as the BRep, triangle mesh, depth map, and voxel model. Compared to the other representations, the depth map approach has advantages in terms of editing and scalability. As the developed tool path planning system needs to update the IPM frequently after each machining operation, the depth map method is thus employed as the IPM.

As shown in Figure 4.1, the initial IPM is created from the stock. The stock can be specified by the dimensions in X, Y, and Z directions or a mesh. When the dimensions are provided, a mesh is created for the stock based on the dimensions. Then the mesh is projected onto the user-selected base plane to generate the initial IPM. After each machining operation, the IPM is updated based on the tool paths in the operation.

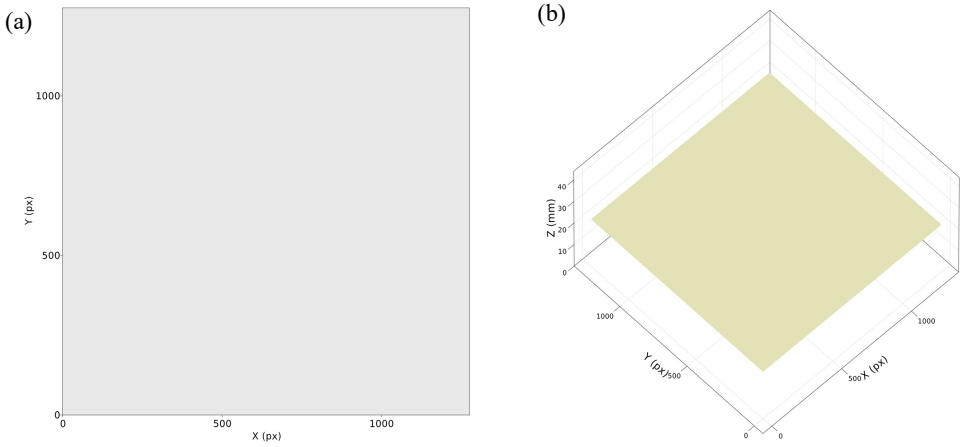


Figure 4.1: The initial IPM: (a), 2D plot; (b), 3D plot.

### 4.1.2 Roughing Tool Path Generation for a Single Feature

When roughing each machining feature, the volume to be removed, called the removal volume, is first extracted. The removal volume is computed as the Boolean difference between the current workpiece and the volume defined by the target feature. To obtain the removal volume, a depth map is also generated for the target feature by projecting the mesh region in the feature to the IPM. The pocket feature in a part exampl and its depth map representation are shown in Figures 4.2 and 4.3, respectively.

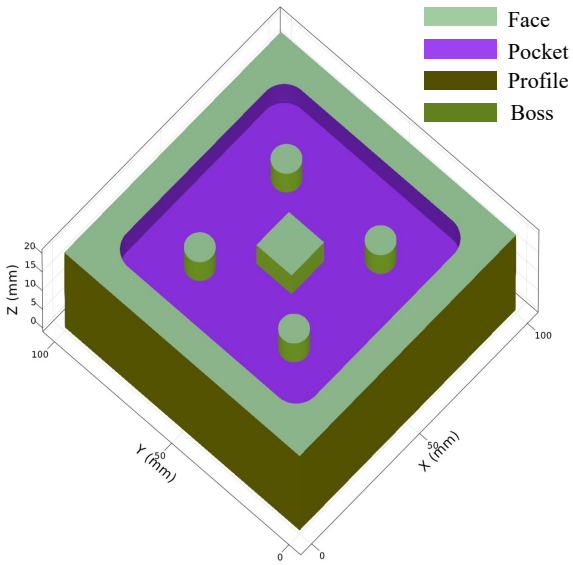


Figure 4.2: An example part with a pocket feature.

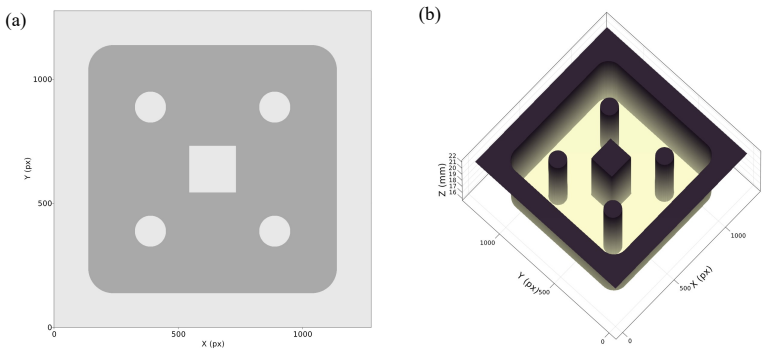


Figure 4.3: The depth map of the feature: (a), 2D plot; (b), 3D plot.

The removal volume represented by a depth map is obtained by subtracting the feature depth map from the IPM. Figure 4.4 shows the depth map of the removal volume for the target pocket feature to be machined in the part example. As shown in the figure, only the volume bounded within the pocket feature is to be removed in the roughing operation. In the product that requires a tight tolerance, a small amount of material is usually intentionally left on the boundary and the bottom of the feature, where the critical requirements of dimensional accuracy present. The material is then machined in the finishing operation so that a good surface finish can be reached at the critical locations. Considering the small amount of material, the removal depth map is offset by a small value to represent the actual volume to be removed.

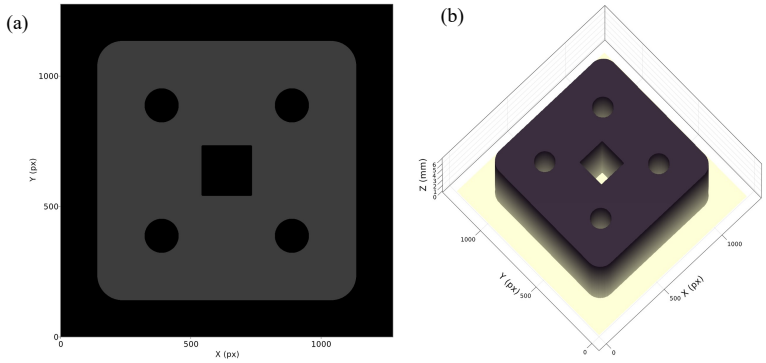


Figure 4.4: The depth map of the removal volume: (a), 2D plot; (b), 3D plot.

After the removal volume is constructed, the tool paths are successively generated to cut the volume. Basically, the direction-parallel and contour-parallel are two common types of tool path planning strategies for the roughing operation. In the direction-parallel paths, the cutting tool moves along the line segments in a defined direction backward and forward. Meanwhile, in the contour-parallel paths, the cutter is driven along the curves at constant distances from the boundaries. For a roughing operation, determining which type of tool path planning strategy to apply significantly depends on the efficiency of the strategy. The direction-parallel approach can generate efficient tool paths for machining simple geometries, such as a rectangle

pocket, because the number of tool retractions resulting from tool inaccessibility is much lower than the number of retractions for the complex geometries, such as a pocket with a complex profile and multiple inner bosses. In comparison, the contour-parallel tool paths are more efficient for complex geometry because a smaller number of tool retractions are required. To be able to generate efficient tool paths for the parts with various geometrical shapes, the contour-parallel tool path strategy is utilized in this study to generate the roughing tool paths.

The start and end depths are first detected from the removal depth map as its maximum and minimum values. The number of layers is then computed based on the depth of cut in each layer, which is related to the properties of the selected cutter. The tool paths are subsequently generated for each layer based on the boundaries of the removal depth map at that layer. An image processing technique is applied to detect the boundaries from the removal depth map. The depth map is converted into a binary image that has only 0 and 1 values and the boundaries are detected as the contours along the region on the binary image with 1 value. As shown in Figure 4.5, the outer and inner boundaries are detected from the removal depth map of the pocket feature example.

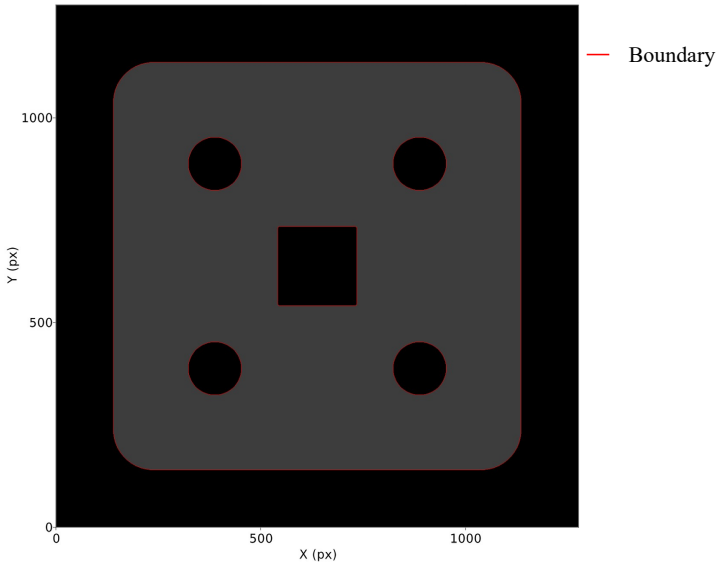


Figure 4.5: Contours detected from the removal depth map.

After the boundaries of the pocket for the layer are detected, the contour parallel strategy is employed to generate the roughing tool path. The boundaries are converted to a set of polylines and the contour-parallel tool paths are generated by repeatedly offsetting the boundaries until no new polylines can be formed by offsetting. The initial offset distance is equal to the cutter radius to avoid cutting the boundaries. The offset distance in the successive iterations is equal to a constant value, called step-over length. As shown in Figure 4.6, the offset polylines define the contour-parallel paths for machining the material on the layer.

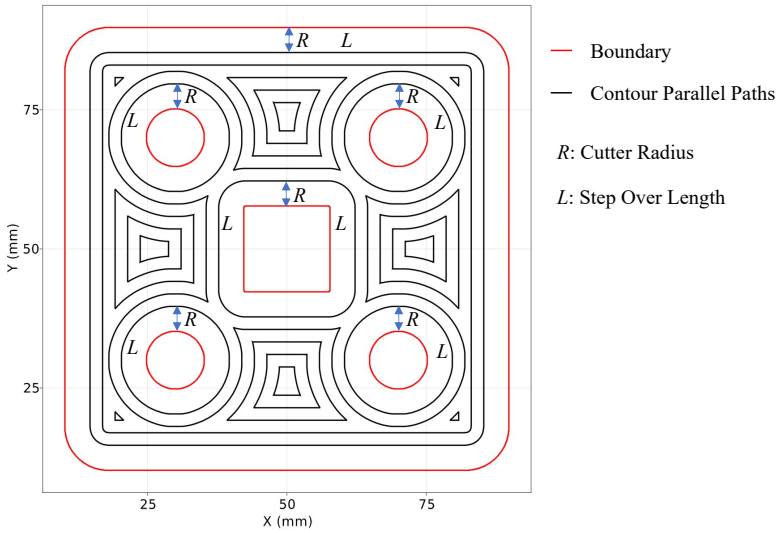


Figure 4.6: Boundaries and the contours parallel paths created from the boundaries.

Before removing the material from the stock, the cutting tool needs to get down to the required depth. When machining the external features, such as an open pocket, the cutter can plunge into a position where no material exists. However, when machining closed pockets and slots, the method to enter the workpiece at the starting depth needs to be considered. Plunging, pre-drilled hole, linear ramping, and circular ramping are four common methods of entry. The pre-drilled hole technique requires drilling holes at the starting points. It is usually not preferred as the additional drilling process may increase the total machining time. In linear ramping, the cutter is simultaneously fed in the axial direction and in one radial direction. In the circu-

lar ramping, also called helical interpolation, the cutter moves along a circle while gradually increasing the depth. The circular ramping is preferable to the other three methods because this approach typically has less radial engagement on the cutting tool with the cutting forces distributed across the three different axes. However, not all of the geometry has enough space for the circular ramping. In these circumstances, other entry methods are used. A gouging detection technique is employed to select the best entry method. The circular ramping is selected if the gouging is not detected. As shown in Figure 4.7, the circular ramping paths are generated to enter the pocket at the defined height. The contour-parallel paths in each layer are linked by the nearest neighbor path search. Gouging detection is also considered when linking the paths. The contour paths are linked by a rapid path if gouging is detected. Otherwise, a transition path is generated to link the contour paths.

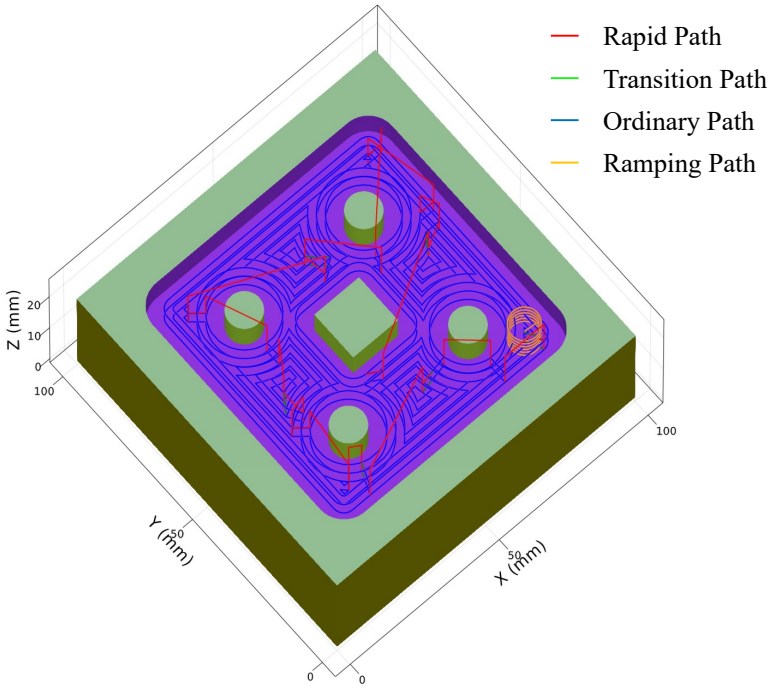


Figure 4.7: Roughing tool path for the pocket feature in the example part.

### 4.1.3 Roughing Tool Path Generation for Profile Features

As shown in Figure 4.8, a profile feature defines the outer profile of the part. Compared to the closed pocket feature, the boundary of the profile feature need to be pre-processed so that the contour-parallel tool paths can be properly generated.

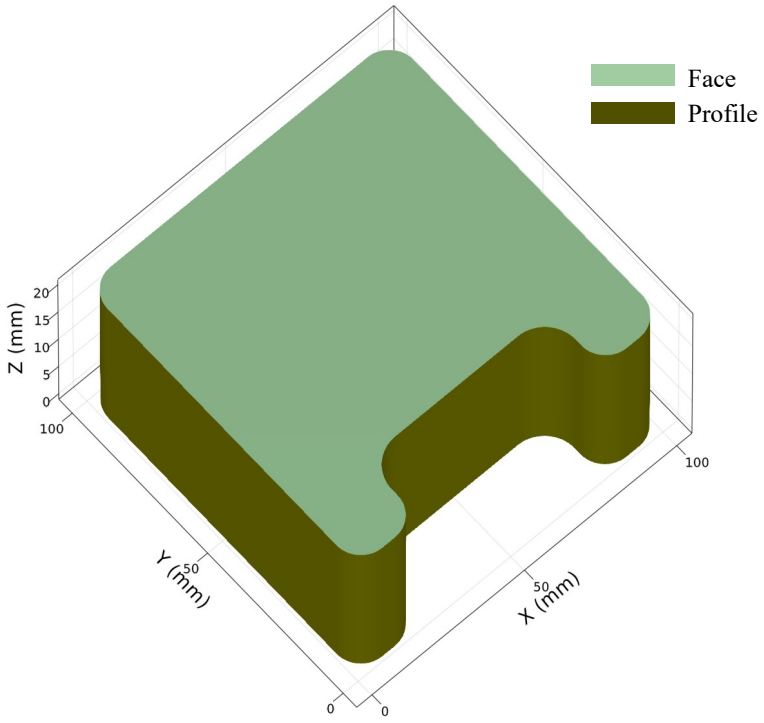


Figure 4.8: An example part with a profile feature.

As shown in Figure 4.9, the removal volume of the profile feature includes two boundaries. The inner boundary is defined by the profile whereas the outer boundary is defined by the stock boundary. These two boundaries cannot be used to generate the contour-parallel tool paths directly. The initial offset distance is equal to the cutter radius so that gouging is avoided for the boundaries of the closed pocket. However, the cutter plunges into the workpiece directly if the stock boundary is offset by the cutter radius. Even worse, the contour-parallel tool path cannot be generated if the closest distance between the stock and the profile boundaries is less than the cutter diameter because there is no space to create new offset polylines from the boundaries.

To resolve this issue, a virtual boundary is created by offsetting the stock boundary by a distance $L$, as written in Equation 4.1. As shown in Figure 4.9, a virtual boundary is created for the profile feature example.

$$L = \begin{cases} 2 * R_{cutter} - L_{step\_over}, & \text{D} >= L_{step\_over} \\ 2 * R_{cutter} - \frac{D}{2}, & \text{D} < L_{step\_over} \end{cases} \tag{4.1}$$

where $R_{cutter}$ is the cutter radius, $L_{step\_over}$ is the step over length, and $D$ is the closest distance between the stock and the profile boundaries.

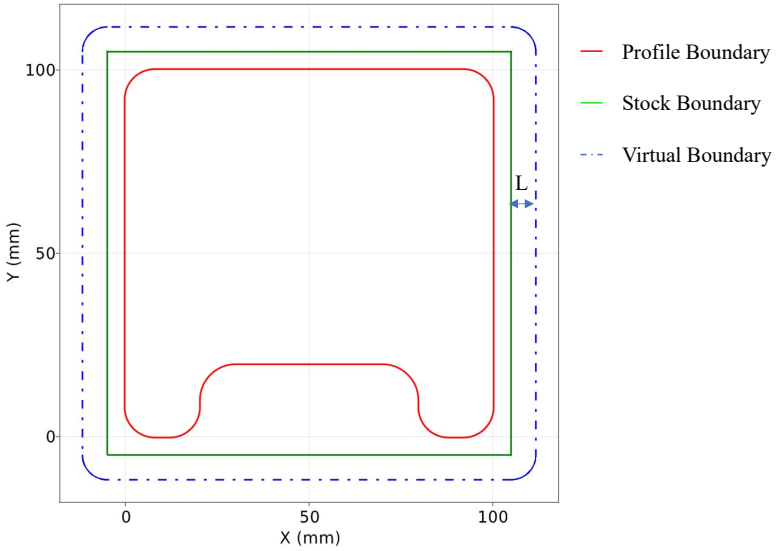


Figure 4.9: The profile, stock, and virtual boundaries.

The contour-parallel tool paths are subsequently generated based on the virtual and profile boundaries for each layer, as shown in Figure 4.10.

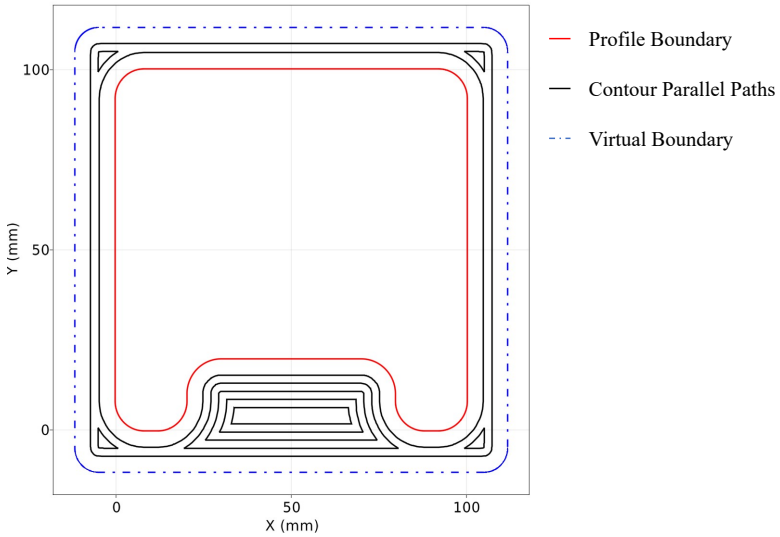


Figure 4.10: Target boundary, its associated features, and the generated tool path to deburr the boundary.

The contour-parallel paths are then linked together by rapid and transition paths to generate the tool path for machining the profile feature as shown in Figure 4.11. Compared to the closed pocket feature, the ramping path is not needed for the profile feature because the cutter can move to the required depth from outside the stock.

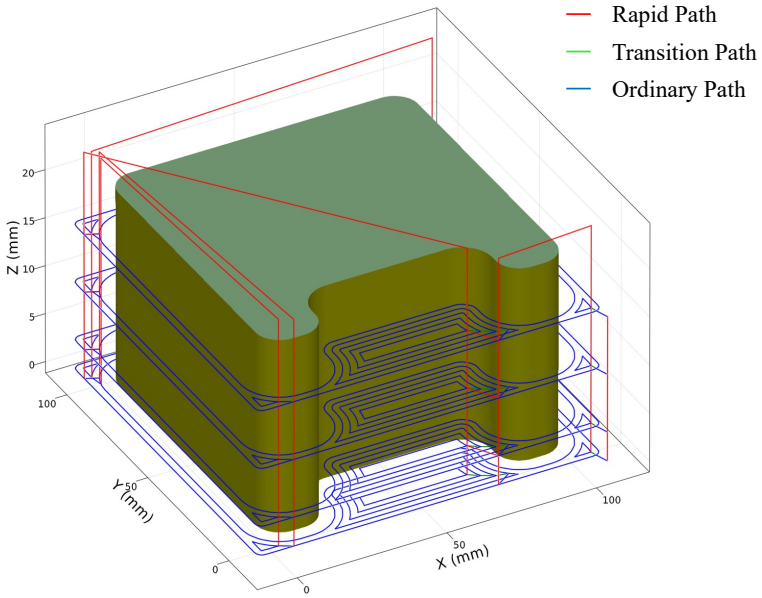


Figure 4.11: Roughing tool path for the profile feature in the example part.

## 4.2 Automated Tool Path Generation with Multiple Cutters

Conventionally, only a single cutter is used in the roughing operation. When machining the parts with narrow bottlenecks, a large cutter can only cut a portion of the material from the feature. As an example, for a part with a pocket feature shown in Figure 4.12, the distances between inner rectangle boundaries are only 7 mm. Therefore, to clear the material between the inner boundaries, the diameter of the cutter must be reduced to be smaller than 7 mm. However, the machining efficiency would significantly decrease as the length of the tool paths increases greatly for a smaller cutter. Roughing with multiple cutters is an attractive option to improve the machining efficiency. A large cutter can be used to remove most of the materials from the region with simple geometrical shapes. A smaller cutter can then be applied to clear the more complicated portions like sharp corners and narrow bottlenecks, by taking advantage of its greater degree of accessibility. However, for parts with complex geometrical shapes, it is rather difficult to determine the optimal tool set with minimum machining time.

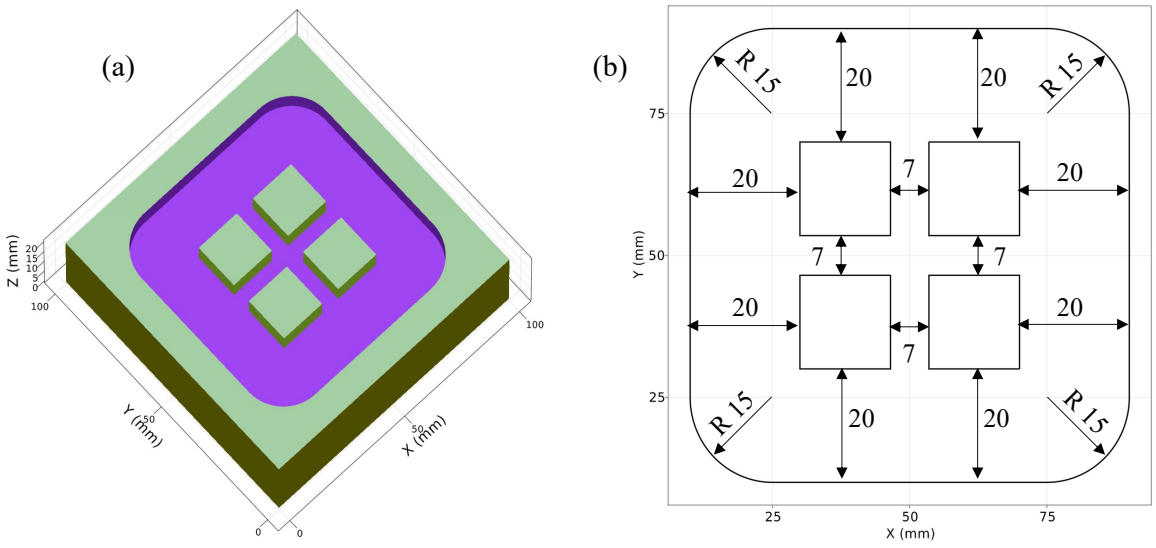


Figure 4.12: Example part with a pocket feature: (a), machining features; (b), sketch.

### 4.2.1 Optimal Cutter Set Selection

A graph-based approach is introduced in this study to select the optimal set of cutters for roughing. As shown in Figure 4.13, a directed graph is built for the cutter selection. Each node of the graph represents an IPM, and each edge is the total machining cost and the required machining operation to transform the IPM from one state to another state. $IPM_0$ is the initial IPM, which is defined by the dimensions of the stock. $IPM_n$ is the final IPM. The diameters of the available cutters in the library are sorted in a descending order. The largest cutter is used to the generate the tool path for the first operation $O_1$ to transform the stock from $IPM_0$ to $IPM_1$. The first operation and its machining cost form the edge $IPM_0$ to $IPM_1$ on the graph. A smaller cutter in the library is selected for the subsequent operation. The operations and their machining costs are added to the edges of the graph between the IPMs. The process is repeated until all the materials in the target feature are machined or the smallest cutter in the library is selected. The nodes of the graph are determined after all the $IPM$s are obtained. Additional machining operations are then added to transform the IPM with a small number to the IPM with a larger number by using the last cutter to create the latter IPM. These operations and their machining costs are added as the additional edges in the graph. Once the graph is constructed, the shortest path from the first node to the last node on the directed graph is searched. The shortest path provides the optimal set of cutters that can manufacture the feature with a minimum amount of machining cost. The details of the graph based optimal cutter set selection are elaborated in the following sections.

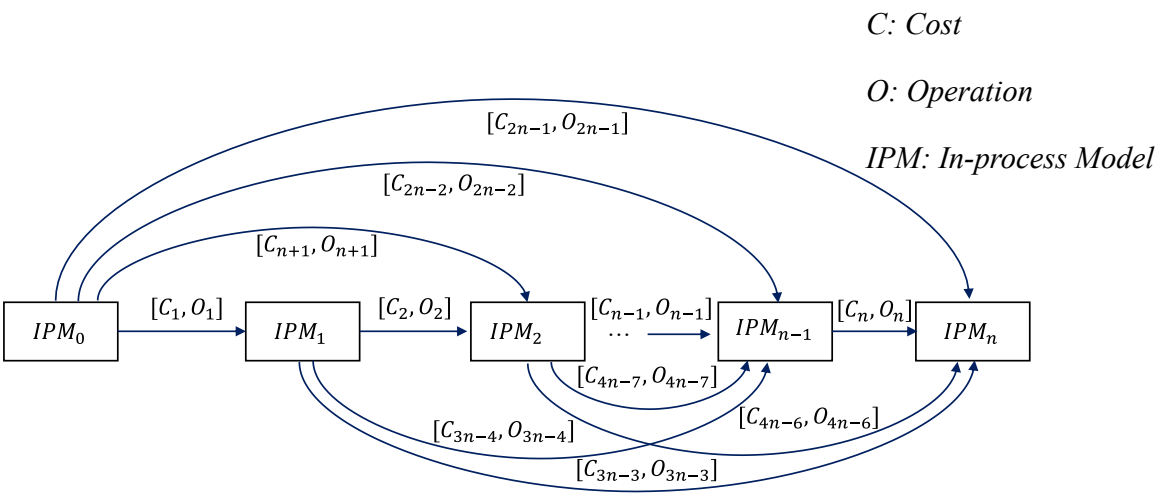


Figure 4.13: Directed graph for the selection optimal cutter set.

### 4.2.2 Machining Cost Estimation

The machining cost in each operation is estimated by the summation of the time moving along the paths in the operation and the tool change time. This study considers four types of tool paths for a machining operation: rapid, entry, transition, and ordinal paths. Rapid path is used to move the cutting tool rapidly from one position to another position in the air without gouging the part. The entry path, including linear ramping, circular ramping, and plunging paths, is employed to enter the workpiece at the desired depth. The transition path is utilized to connect different paths smoothly. The ordinary paths are all other paths beyond the three types of paths indicated. The transition and ordinary paths are assigned with the same feed rate while the entry and rapid paths may have different feed rates than that of the transition and ordinary paths because the tool engagement between the cutter and the workpiece differs in these paths. For example, rapid paths do not cut the workpiece so they can be assigned a higher feedrate than the heavy cutting ordinary path.Considering the different feed rates for different types of tool path and the time change time, the total machining time for each operation can be computed

using Equation 4.2.

$$t_{total} = L_r * f_r + L_e * f_e + L_o * f_o + t_{change}, \tag{4.2}$$

where $l_r$, $l_e$, and $l_o$ are the lengths of the rapid, entry, and ordinal paths, respectively, $f_r$, $f_e$, and $f_o$ are the feed rates defined for the rapid, entry, and ordinal paths, respectively, and $t_{change}$ is the time cost for the tool change.

### 4.2.3 Maximum Cutter Selection

Generally, the maximum cutter in the graph starts from the maximum available cutter in the cutter library. However, the maximum cutter can be selected differently when the target machining feature is a pocket feature because not all the cutters can fit inside the pocket without gouging the boundaries. The maximum inscribed circle (MIC) approach is utilized to determine the maximum cutter for the graph. Traditionally, MIC is defined as the maximum circle that can be totally enclosed by a closed curve. MIC is widely used in many fields. For example, in metrology, the out of roundness is quantified as the maximum deviation between the measured data points and the MIC [80]. When additional curves are included inside the closed curve, the MIC is extended to be the maximum circle that completely fits inside the outer curve without intersecting any of the inner curves in more than one point. This study uses an image processing technique to calculate the MIC that can be fit into a pocket. In a discrete geometry of the image representation, the MIC can be formulated as a max-min problem as written in Equation 4.3. In the initial step, the outer boundary defined by the profile of the pocket and the inner boundaries defined by the bosses inside the pocket are detected from the depth map of the pocket feature. For each point inside the region enclosed by the outer profile, the minimum distance between the point and all points on the boundaries are then computed. The computation of the minimum value among the distances between the point and all the boundary

points is expensive. Thus, a K-dimensional(KD) tree data structure is applied to accelerate the computation of the minimum distance. The diameter of the MIC is obtained from the maximum value among all minimum distances.

$$R_{MIC} = \max_{\forall [x_C, y_C] \in \Omega} ( \min_{\forall [x,y] \in S} \sqrt{(x_C - x)^2 + (y_C - y)^2} ), \tag{4.3}$$

where $R_{MIC}$ is the radius of the MIC, $[x_C, y_C]$ is the center of the MIC, $S$ is the set of points on the outer and inner boundaries, and $\Omega$ is the set of all the points enclosed by the outer boundary.

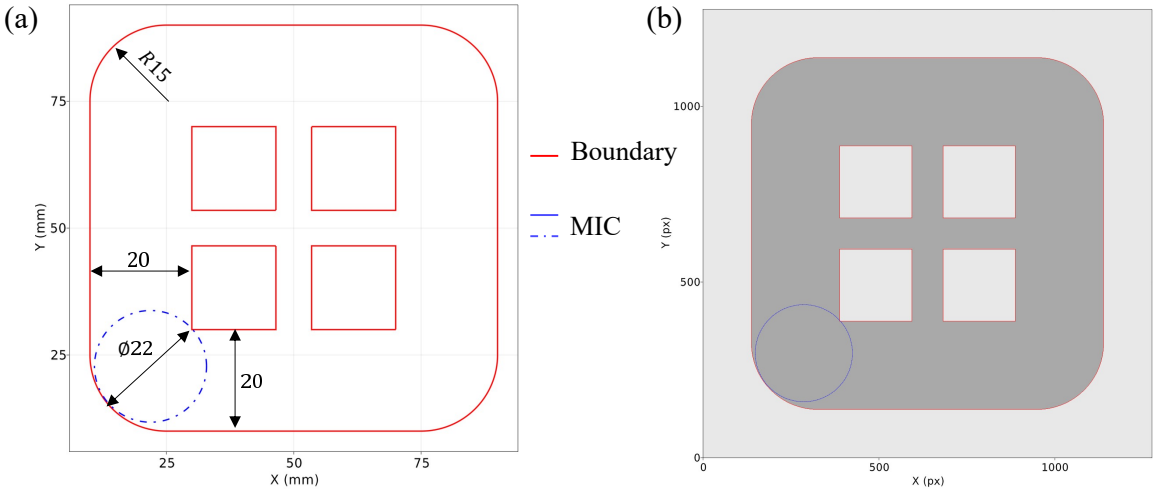


Figure 4.14: Maximum inscribed circle (MIC) for a pocket feature: (a), outer and inner boundaries and MIC; (b), detected contours, and MIC on the depth map.

Based on the MIC approach, the complexity of the graph is reduced as the depth of the graph is shortened. As shown in Figure 4.14, the diameter of the MIC for the pocket feature in the part example is around 22.0 mm. Any cutter with a diameter less than 22.0 mm can fit inside the pocket to remove a portion of the material. Based on the example of the cutter library shown in Figure 4.15, all the cutters in the library can be used as the initial cutter of the graph. However, in real machining, the method for entering the workpiece without gouging also needs to be considered. When machining a pocket feature, the circular ramping, which create a hole inside

the pocket, is the preferred method for entry. The diameter of the circular ramping is usually assigned to 95% of the cutter diameter and thus the cutter with a diameter of 10.0 mm is selected as the maximum cutter.


Figure 4.15: An example cutter library.

### 4.2.4 Generation of IPMs

The algorithm complexity of the proposed graph approach is $O(n^2)$ and most of the computation time occurs in creating and updating the IPMs. Thus, an IPM that is efficient for editing is essential for the success of the algorithm. The depth map approach is used to represent the IPM in this study because of its high efficiency in creation and editing. The initial IPM, $IPM_0$, is created based on the dimensions of the stock, as shown in Figure 4.16.

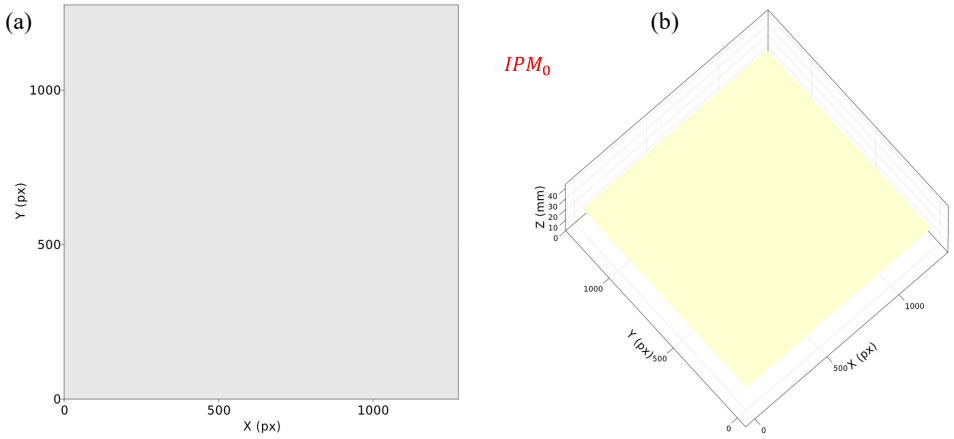


Figure 4.16: Initial IPM ($IPM_0$) represented by a depth map: (a), 2D plot; (b), 3D plot.

The other IPMs in the graph are created by updating the current IPM based on the tool paths generated in associated operations. For the pocket feature example, the first roughing operation is produced by using the first cutter that has a diameter

of 10.0 mm. After the tool paths for the first operation are generated, the cutting process of the tool paths are simulated based on the $IPM_0$ to create $IPM_1$, as shown in Figure 4.17.

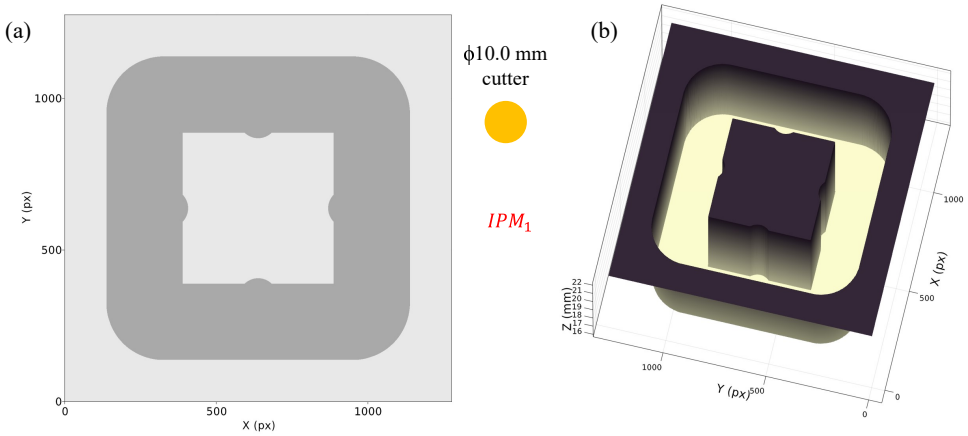


Figure 4.17: $IPM_1$ represented by a depth map: (a), 2D plot; (b), 3D plot.

After an IPM is created, it is compared with the feature depth map to check if all the material in the feature is removed. If not, a smaller cutter is employed to generate tool paths for the next operation. Figure 4.17 shows that the material in between the rectangular boundaries remains uncut. Thus, a cutter with a diameter of 7.5 mm is selected to produce the next operation and a new IPM, $IPM_2$, is created based on the tool paths in the operation.

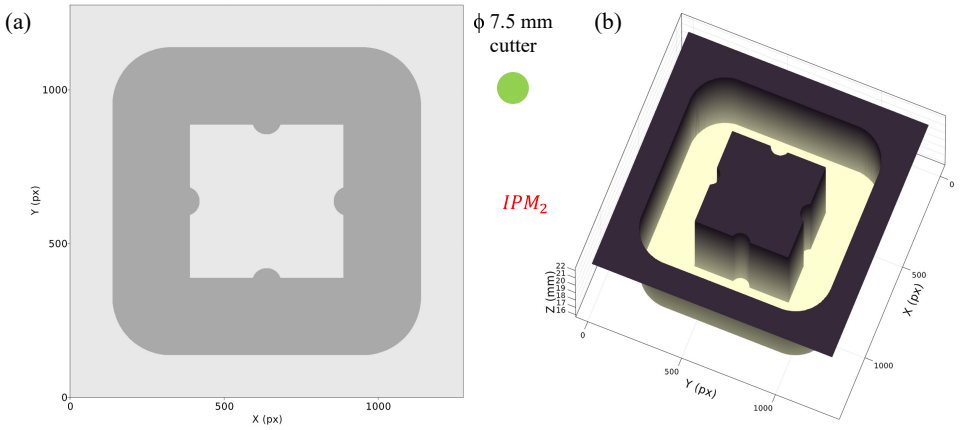


Figure 4.18: $IPM_2$ represented by a depth map: (a), 2D plot; (b), 3D plot.

As shown in Figure 4.18, the 7.5 mm cutter only clears a small portion of material

from the regions between the rectangular boundaries. Therefore, an additional cutter is required to machine these narrow regions and the next smaller cutter is the library, the cutter with a diameter of 5.0 mm is selected as the cutter used in the next operation.

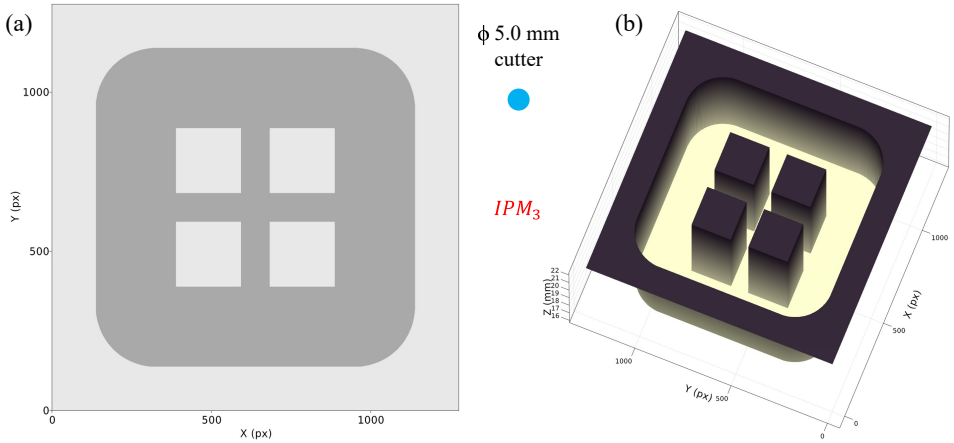


Figure 4.19: $IPM_3$ represented by a depth map: (a), 2D plot; (b), 3D plot.

The tool paths in the operation associated with the 5.0 mm cutter form a new IPM, $IPM_3$, as shown in Figure 4.18. Compared with the feature depth map, all the material in the pocket is removed. And hence the machining operation with the 5.0 mm cutter is the final operation. With a combination of the 10.0 mm, 7.5 mm, and 5.0 mm cutters, the pocket feature can be machined to a desired shape. However, this set of cutters and operations may not be the optimal set for cutting the pocket with a minimum amount of machining cost. Thus, a graph is built with the three cutters to compute the optimal set of cutters.

### 4.2.5 Roughing Tool Path Generation

In each roughing operation, the tool paths are generated based on the target removal volume and the selected cutter by using the method introduced in the preceding section. Figure 4.20 shows the removal volume and the generated tool paths for transforming the initial IPM, $IPM_0$, to $IPM_1$. The removal volume is extracted by subtracting $IPM_1$ from $IPM_0$, and then the tool path with a 10.0 mm cutter is

generated according to the developed roughing path planning strategy.

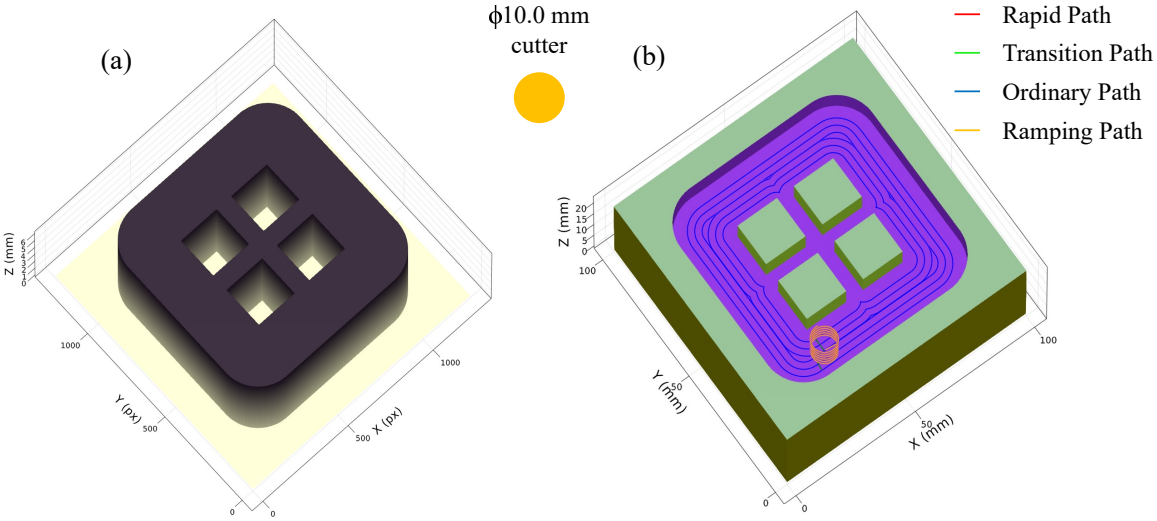


Figure 4.20: Tool path generation from the removal volume: (a), removal volume; (b), generated tool path.

The tool paths for successively using the 10.0 mm, 7.5 mm, and 5.0 mm cutters to cut the pocket feature are shown in Figure 4.21. The 10.0 mm cutter moves around the region outside of the narrow region between the inner boundaries to remove most of the material in the pocket. The 7.5 mm cutter jumps between the corner regions to cut a portion of the material in the narrow region. Finally, the 5.0 mm cutter advances to the narrow region to clear all remaining material in the pocket.

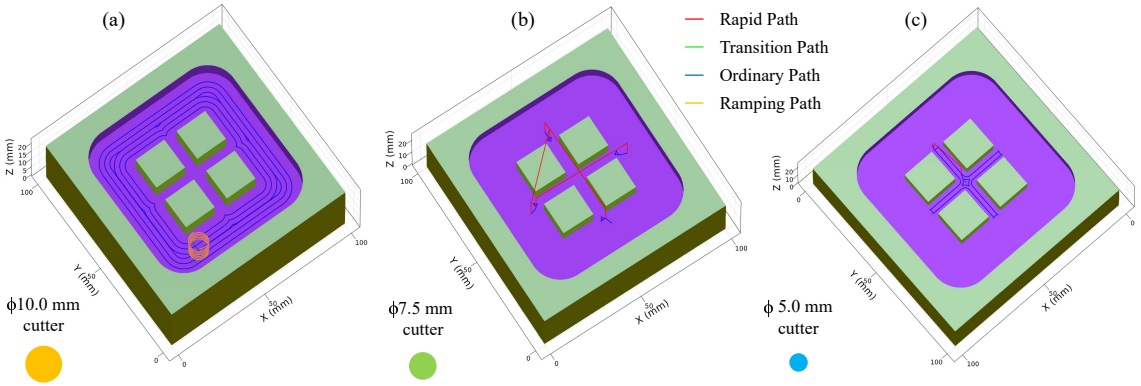


Figure 4.21: Tool paths for three cutters to cut the pocket feature: (a), tool paths for 10.0 mm cutter; (b), tool paths for 7.5 mm cutter; (c), tool paths for 5.0 mm cutter.

However, this set of cutters is not necessarily the optimal set of cutters. For

example, the 7.5 mm cutter may be skipped because the region machined by the 7.5 mm cutter is also accessible to the 5.0 mm cutter. Using a single 5.0 mm cutter can remove all the material in the pocket. Nevertheless, the total machining cost may be higher than the set of three cutters. The graph-based approach is utilized to evaluate each combination of cutters and select the optimal set. After all the IPMs and operations are generated, the IPMs are added to the graph as the nodes. The operations and their machining costs are added to the graph as the edges. In addition, the extra edges of the graph are also added by generating the operations able to transform the IPM with a smaller number to a larger number. For example, the 7.5 mm cutter can transform the IPM from $IPM_0$ to $IPM_2$. Thus, an operation, as shown in Figure 4.22, is created for the 7.5 mm cutter based on $IPM_0$ and $IPM_2$. In the operation, the removal volume is first built according to the difference between $IPM_0$ and $IPM_2$, and the tool paths are subsequently generated based on the removal volume. The operation and its machining cost are added as an extra edge of the graph. The other operations are generated using a similar method.

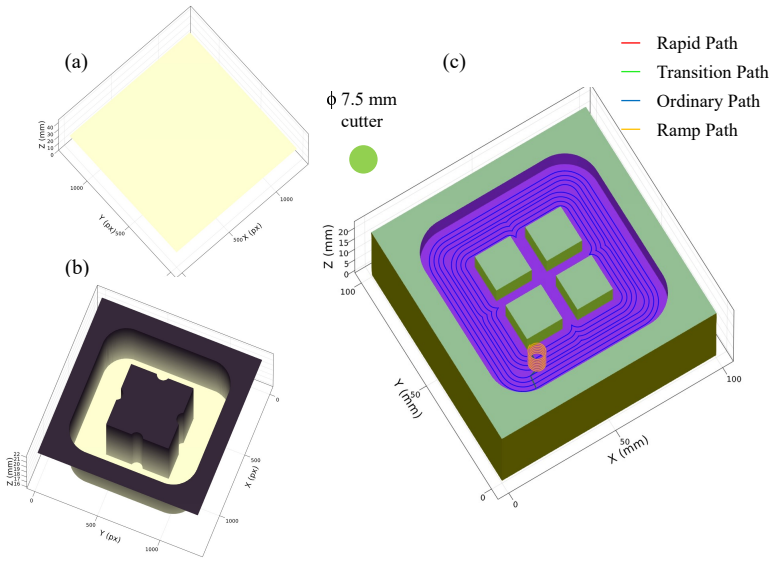


Figure 4.22: Tool paths for the operation to transform the workpiece from $IPM_0$ to $IPM_2$ : (a), $IPM_0$ ; (b), $IPM_2$ ; (c), the generated tool path.

### 4.2.6 Volume Expansion

The roughing operation might create regions with sharp corners. These sharp corners may not be accessible to a cutter in the subsequent operation if the contour-parallel path strategy is applied directly to generate the tool path to cut the removal volume. Consequently, uncut regions are formed after the subsequent operation. As demonstrated in Figure 4.23(a), when moving along the generated contour-parallel paths, the 5.0 mm cutter cannot access the sharp corner region in the removal volume left by the previous operation with a 7.5 mm cutter. Thus, uncut regions, as shown in Figure 4.23(b), are produced after the operation.

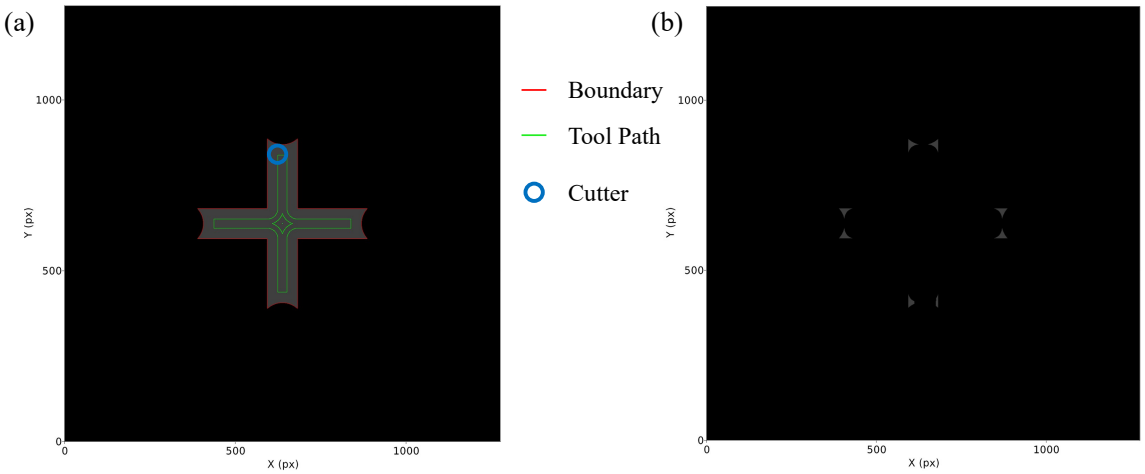


Figure 4.23: Uncut region because of the sharp corners : (a), removal volume with sharp corners and contour-parallel tool paths for a 5.0 mm cutter; (b) uncut regions left by the tool paths.

To reduce the uncut regions, the region expansion approach proposed by Zhou, Zheng and Chen [81] is used to expand the removal volume so that new contour-parallel tool paths can be generated to cover the sharp corner. The hard edges and the soft edges are first detected from the removal depth map. The hard edges are the edges of the boundaries of the feature whereas the soft edges are the edges along the boundary of the current removal volume that do not belong to the hard edges. The soft edges are then offset by the cutter radius, and the hard edges are offset by

the cutter diameter, as shown in Figure 4.24. After the edges are expanded, the new boundary contours are detected from the removal depth map and they are used to generate the tool path.

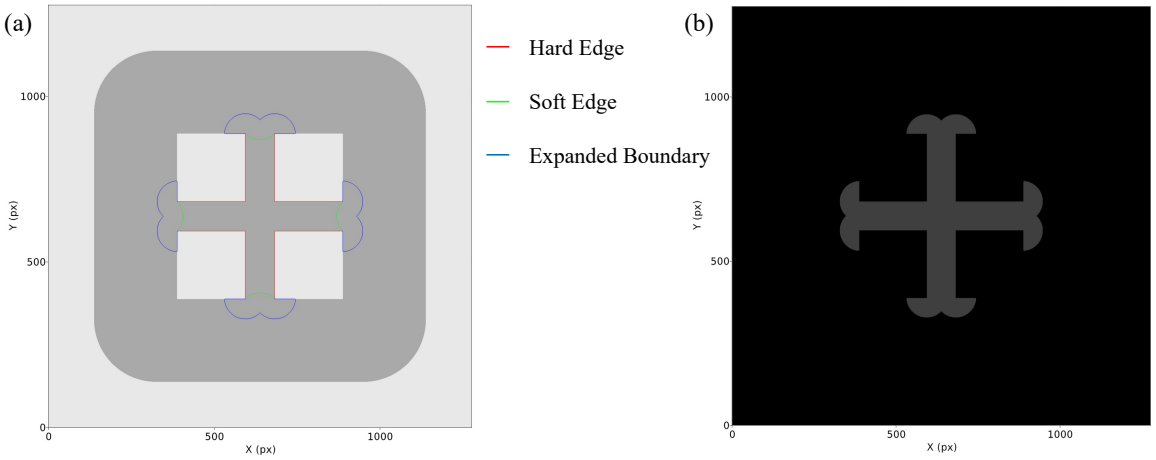


Figure 4.24: Region expansion based on soft and hard edges: (a), the soft edges, hard edges, and expanded boundary; (b), expanded removal volume.

The comparisons of the tool paths using and not using the expansion strategy are shown in Figure 4.25. The simulation result indicates that the expansion method can remove the material in the corner region so that the part quality is improved.

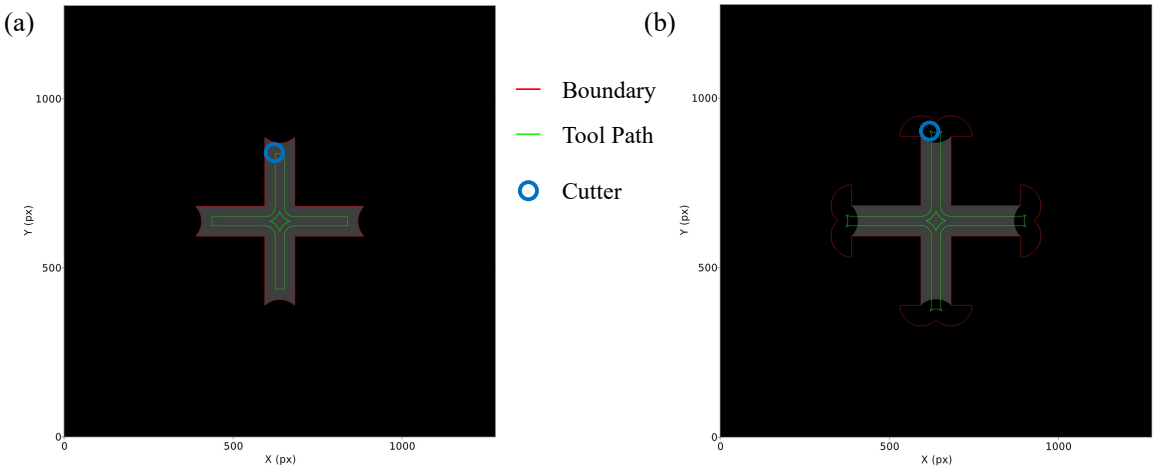


Figure 4.25: Comparisons of the boundaries and tool paths on the depth map of the removal volume before and after region expansion: (a), before; (b), after.

### 4.2.7 Cutter Set Selection and Optimization

After all the nodes and edges of the graph are obtained, the final step is to build the directed graph and find the shortest path between the first and the last nodes in the graph, which represents the minimum cost to transform the workpiece from $IPM_0$ to $IPM_n$. Dijkstra's graph approach [82] is employed in this study to find the shortest path. As shown in Figure 4.26, the shortest path between the $IPM_0$ and $IPM_3$ is the combination of the path connecting $IPM_0$ and $IPM_1$ and the path connecting $IPM_1$ and $IPM_3$, which requires a total machining cost of 3.47 minutes to machine the pocket feature.

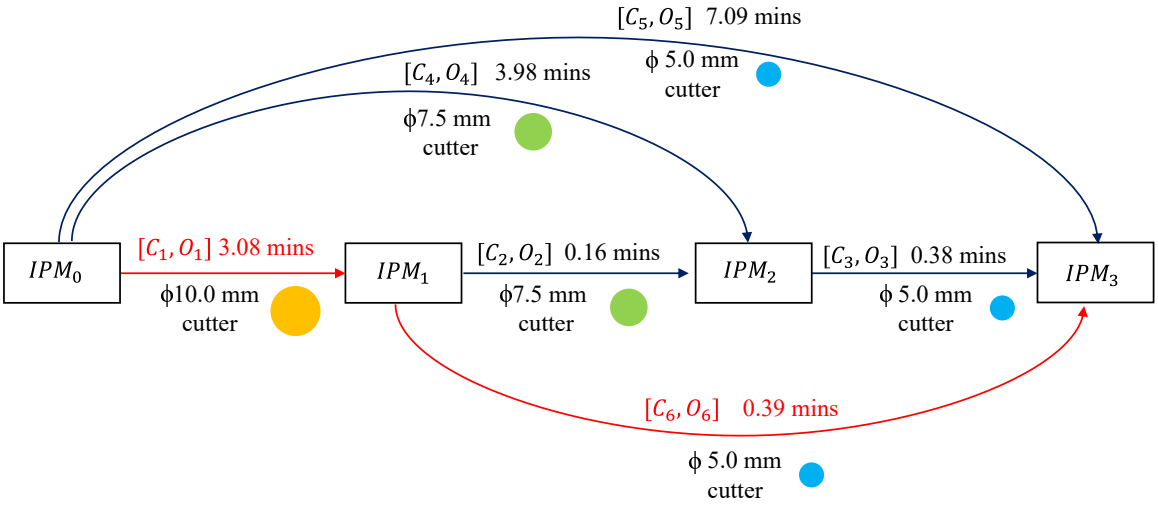


Figure 4.26: Directed graph for the example part.

The shortest path in the graph determines the optimal cutter set and the associated operations to machining the target features. The tool paths of the selected operations for cutting the pocket example are shown in Figure 4.27. In the machining process, the 10.0 mm cutter removes most of the material in the pocket while the narrow regions between the rectangles are moved by the 5.0 mm cutter.

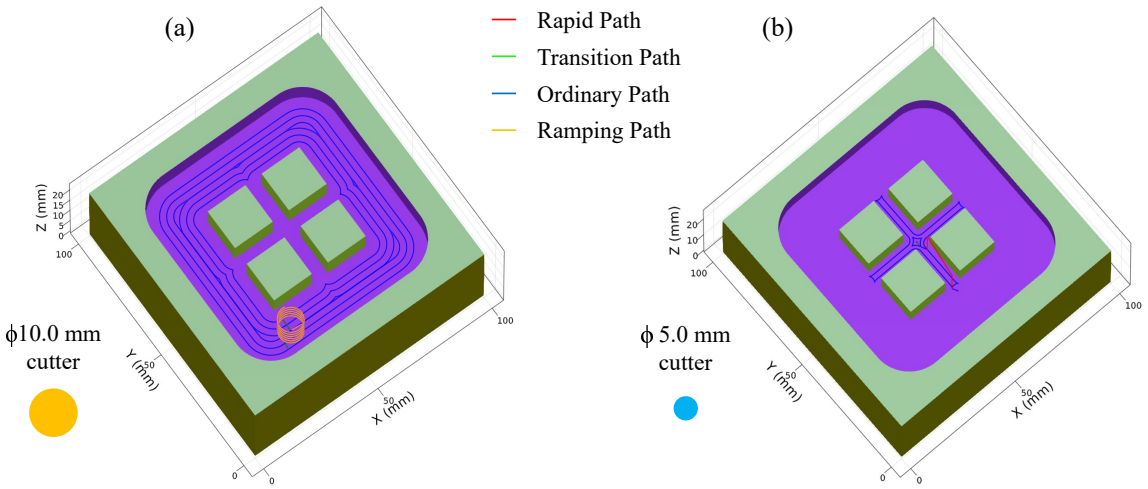


Figure 4.27: Selected cutters and the associated tool paths: (a), tool path for the 10.0 mm diameter cutter; (b), tool path for the 5.0 mm diameter cutter.

## 4.3 Facing Tool Path Generation

When manufacturing a part on a CNC machine, it is essential to ensure that the reference plane is as flat as possible as the accuracy of the reference plane largely affects the accuracy of the machined part. To meet the accuracy requirement, stocks with tight tolerances can be used directly. However, the price increases with the tolerance of the stock and is much higher than the regular raw stock. Alternatively, most machinists would start with a regular raw stock that is larger than the part to be manufactured and would use a facing operation to remove the extra material so that a flat reference plane is created on top of the stock. A facing operation is required to be as efficient as possible because there is no significant feature on top of the stock. Thus, the larger available flat end mill or fly cutter and the most efficient tool path strategy are usually employed for the facing operation. The zig-zag tool path strategy, which is a variant of the direction-parallel strategy, is efficient for machining the simple geometry; and, thus, is used to generate facing tool paths. As shown in Figure 4.28, the facing tool paths are generated by detecting the size of material above the part along the cutting direction and then producing a zig-zag path pattern to cut the material layer by layer. In each layer, the cutting tool moves along a selected

direction (usually X or Y) until all the material in the direction is machined and then retracts to the next starting point to move along a new path in the same direction. The pattern is repeated until all the material on the layer is machined.

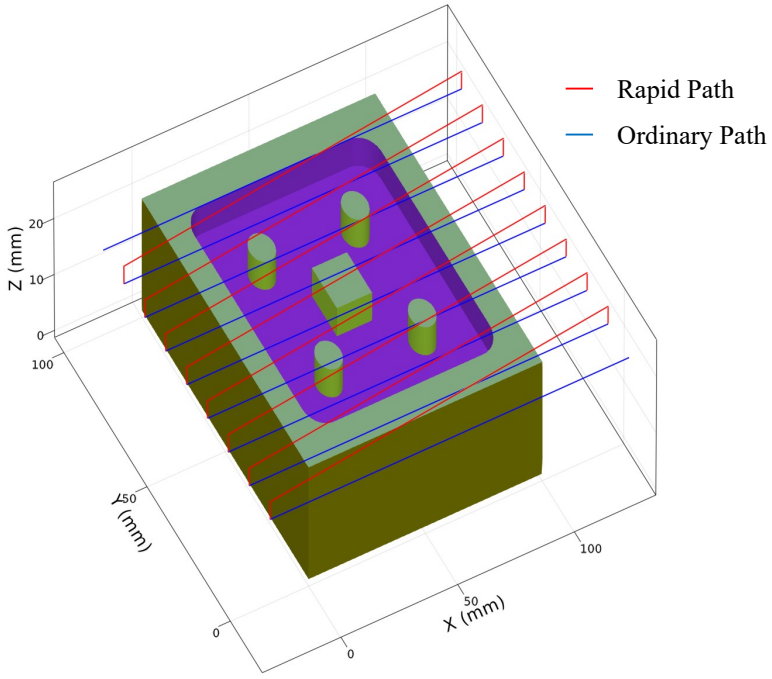


Figure 4.28: Target boundary, its associated features, and the generated tool path to deburr the boundary.

# CHAPTER 5: AUTOMATED DEBURRING TOOL PATH GENERATION

Burr, also known as material overhang on the edge of a metal part, is formed at the end of most machining processes, such as turning, milling and drilling. The mechanism of burr formation has been extensively studied [83–85]. According to Sofronas, the major contributor to burr formation is the plastic deformation flow during the cutting process [83]. A burr has several negative properties, including reduced product life, decreased dimensional accuracy, and the introduction of difficulties into the assembly [86]. It can even cause injury to the operator. Thus, deburring, or the removal of burr, is usually an unavoidable procedure in the product manufacturing process of metal parts.

Currently, many deburring operations are still performed manually. Such manual operations tend to be time-consuming, and the operations may lead to inconsistent material removal and surface finish. Numerous deburring and edge-finishing processes have been developed to replace manual deburring [85–87]. Gillespie [85] suggested classifying these processes into four categories, including mechanical, thermal, chemical, and electrical deburring processes. Among these processes, the thermal, chemical, and electrical deburring methods may introduce unintended damages to functional surfaces of the work parts while limiting burr removal capability at the micro level [86]. These deburring processes become even more complicated when large burrs are required to be removed from intersecting holes. The most frequently used mechanical deburring processes include brush deburring, bonded abrasive deburring, abrasive jet deburring, barrel deburring, centrifugal barrel finishing, robotic deburring [88–92], and CNC deburring [93–95].

Recent research efforts have increasingly been devoted to robotic deburring [88–92].

The major advantages of the robotic systems are the ability to replicate the motions accurately; manipulate heavier, higher-powered tools for a faster finishing process; and accurately perform in hazardous, noisy, and ergonomically unsuitable situations [86]. However, robotic deburring faces two main challenges: the accurate registration of the workpiece to its CAD model and fast collision-free trajectory generation [94]. It is critical to locate the workpiece in the robot's coordinate system through registration so that the burr can accurately be removed. Song and Song [90] proposed a robotic deburring process by modifying the tool path generated from the CAM system. The deburring tool path was first extracted from the G-code generated from the CAM system. An iterative closest point was successively implemented to match the tool path with the contour of the actual workpiece. In the last step, the tool path was modified to compensate for the errors in the position/orientation. Impedance control was applied to avoid excessive force in the deburring process. However, the quality of the deburred workpiece was not reported.

CNC deburring has been known as a high-quality, low-labor cost, and a high production rate approach. Several methods have been proposed to generate tool paths for deburring drilled holes on CNC machines. Sato et al. [93] introduced an analytical method to generate the tool paths automatically to remove burrs resulting from the drilling of a circular pipe. The differential geometry of the cylinder-to-pipe intersection curve was derived analytically. The intersection curve was then approximated using a set of line segments with an equivalent length that are connected by the points sub-sampled on the curve. The actual cutter locations on the tool path were derived based on the normals of the points at the inner pipe surface and the normals of the same points on the hole. However, this method only works for drilled intersections that are perpendicular to each other. To resolve the limitation, Schützer et al. [95] developed an approach to generate tool paths for deburring complex-shaped drilling intersections with ball-end cutters. However, the intersection curve was manually

extracted from the CAD model and converted to the point cloud.

Although various methods were developed to deburr drilled holes on CNC in the research community, the automated approach to detect and generate tool paths for deburring arbitrary boundaries regardless of the machining operations has rarely been studied. State-of-the-art CAD/CAM packages are versatile enough to design and analyze complex parts as well as generate tool paths for various machining operations, such as turning, milling, drilling, and grinding. However, the functionality to generate the deburring tool path automatically for arbitrary design parts is still rarely available. The users are required to select which edges to deburr. In a fully automated tool path planning system, the human intervention should be avoid as much as possible. In this research, a machining feature based technique is introduced to generate deburring tool path automatically for removing undesired burrs on 3-axis CNC machines.

To automate the deburring process, understanding the types of burrs and the formation of these burrs is an essential starting point. Various process parameters can affect the formation and shapes of burrs, such as the cutter geometry, the part geometry, the workpiece material, and the cutting conditions. Depending on different operations and applications, the burrs were also described differently. There is still no universally acceptable description or classification for machining burrs. Gillespie and Blotter [96] were among the first to propose a classification for machining burrs. They classified the machining burrs into four categories: Poisson burr, roll-over burr, tear burr, and cut-off burr. In the turning and drilling operations, the Possion burr and roll-over burr might be formed on the edge of the workpiece [96]. Chern [97] identified five types of burrs formed in the face milling: knife-type, wave-type, curl-type, edge breakout, and secondary burr. The first three types of deburrs are the primary burrs described as the roll-over burr produced on the tool exit edge. The edge breakout was found when the metal removal rate was very high. The secondary burr is the small burr remaining on the edge of the machined part after optimizing the cutting

condition and tool geometry.

Although different types of burrs can be formed in different machining operations, fortunately, they are generally located along the edges where the cutter enters or exits the workpiece. Thus, automatic detection of the boundary edges of the part is a fundamental procedure for any automated deburring system. As shown in Figure 5.1, the system starts by detecting the feasible boundaries in a selected cutting direction. The deburring tool paths are then generated to remove the burrs and create a chamfer with a constant width by using a ball end cutter on each feasible boundary. To avoid removing excessive material from the part, an additional step is added to detect gouging and, if detected in the tool path locations, modify the original tool path. The details of each step are elaborated in the following sections.

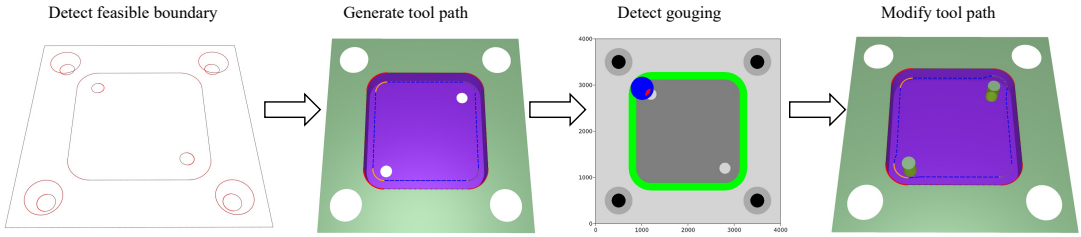


Figure 5.1: Workflow of the feature-based automated deburring tool path generation system.

## 5.1 Detection of Feasible Boundaries

A boundary edge is the outer border of a shape or a machining feature. In a machining feature with a triangle mesh as the underlying geometry representation, each edge of the triangles can be isolated or shared by two triangles. A boundary edge is defined as an isolated edge that only exists in a single triangle within the machining feature while a boundary is a set of connected boundary edges. To be feasible for deburring from a selected cutting direction, a boundary must fulfill two conditions: be visible from the direction and not be concave. Thus, in the procedure for detecting feasible boundaries, these two conditions are validated for every boundary of the machining features. Figure 5.2 shows a part with multiple machining features, including

four blind holes, four through holes, two bosses, two pockets, a profile, and two face features. Among these features, the blind holes, the through holes, the bosses, the profile, one pocket, and one face are visible from +Z direction while the remaining pocket and face are not visible as they are occluded by other features. Therefore, the feasible boundaries only exist in the boundaries of the visible features.

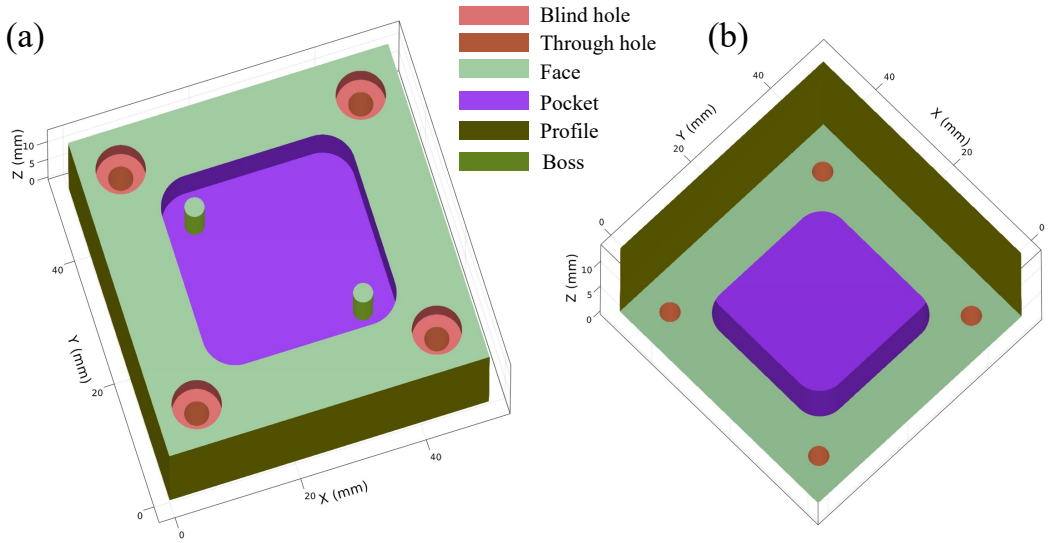


Figure 5.2: A part with multiple machining features: (a), top view; (b), bottom view.

The pseudo code of the algorithm for detecting the feasible boundaries is described in Algorithm 5.1. In the initial step, all the available boundaries are extracted from the machining features. The points that are visible from the cutting direction are then detected. In this research, a ray tracing algorithm is implemented to detect the visible points. Next, the points on each boundary are compared with the visible points. If all the points of a boundary are a subset of the visible points, then the boundary is a candidate boundary for deburring. Meanwhile, a feasible boundary cannot have concave edges as they are not reachable for the cutters. Thus, the convexity test discussed in Chapter 3 is applied on each candidate boundary. A candidate boundary is selected as a feasible boundary if all the edges on the boundary are convex. The machining features associated with the boundary are also identified to assist the

computation of the deburring tool path in the following step.

**Algorithm 5.1** Feasible Boundaries Detection

**Input :** $features$ is the array of machining features
**Input :** $part$ is the geometry of the part
**Input :** $dir$ is the cutting direction

1: $f\_boundaries = [], f\_features = []$
2: $boundaries = get\_boundary(features)$ ▷ get the boundaries of the features
3: $visible\_points = get\_visible\_points(part, dir)$
4: **for** $boundary \in boundaries$ **do**
5: $points = get\_points(boundary)$ ▷ get the points on the boundary
6: **if** $points \in visible\_points$ && $is\_convex(boundary)$ **then**
7: $push(f\_boundaries, boundary)$
8: $l\_features = get\_linked\_features(boundary)$
9: $push(f\_features, l\_features)$
10: **end if**
11: **end for**
12: **return** $(f\_boundaries, f\_features)$

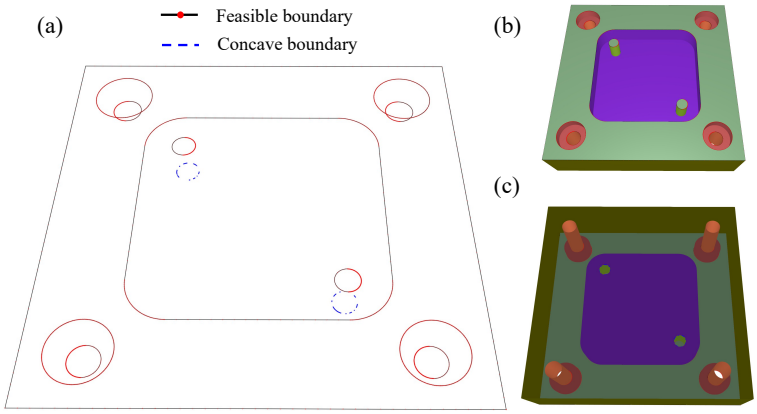


Figure 5.3: Detection of feasible boundaries and their associated features: (a), feasible and concave boundaries; (b), top view of the boundaries and their associated features; (c), bottom view of the boundaries and their associated features

As shown in Figure 5.3, the boundary between the profile and the top face, the boundary between the pocket and the top face, the four boundaries between the blind holes and the top face, the four boundaries between the through holes and the blind hole, and the two boundaries between the bosses and their neighboring faces are detected as feasible boundaries that can be cut from the +Z direction. The

two boundaries between the bosses and the pocket are not recognized as feasible boundaries because they are concave boundaries.

## 5.2 Deburring Tool Path Generation with Ball End Cutter

After all the feasible boundaries are detected, the next procedure is to generate tool paths to deburr each boundary. The goal of deburring is to reduce the size of the burrs on the boundary by cutting a small amount of material from the boundary. New boundaries are formed after deburring. However, the deburring process is also a cutting operation, meaning it may create new burrs, called secondary burrs, along the new boundaries. The process parameters and cutting conditions to reduce the size of the secondary burr have been little studied. Sato et al. [93] and Abele et al. [94] proposed that the penetration direction for deburring should be along the normal vector that bisects the edge vectors at the two adjacent surfaces sharing the target edge so that the resultant secondary burrs between the main bore and cutting directions, as well as the side direction and the cutting direction can both be minimized. Uniform chamfers along the edges of the resultant boundary are usually desired for aesthetic purposes.

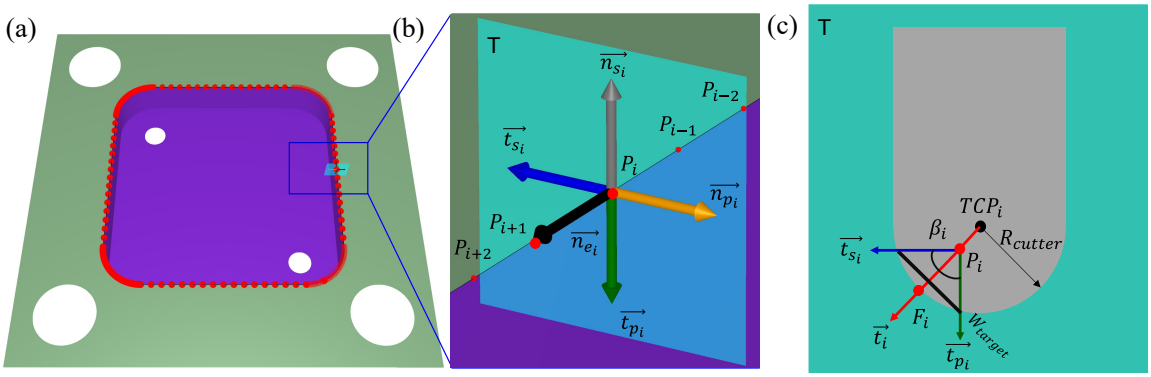


Figure 5.4: Modelling the tool center location for deburring boundary by using ball cutter: (a), machining features and the boundary between them to be deburred; (b), unit tangent vectors at the point $P_i$; (c), computation of the tool center location at $TCP_i$.

As shown in Figure 5.4(a), a boundary is formed by multiple edges connected

by two points. The tool locations for a ball end cutter to create chamfers with a constant width along each boundary edge are computed for each boundary point and then connected together as the tool path for deburring. The process for computing the tool locations is described as follows.

Taking edge $e_i$ as an edge on the target boundary, the unit direction vector $\overrightarrow{n_{e_i}}$ along the edge $e_i$ can be computed as shown in Equation 5.1.

$$\overrightarrow{n_{e_i}} = \frac{\overrightarrow{P_i P_{i+1}}}{|\overrightarrow{P_i P_{i+1}}|}, \tag{5.1}$$

where $P_i$ and $P_{i+1}$ are the start and end points of the edge $e_i$, respectively.

In a part composed of multiple machining features, a boundary edge is shared by two features. The feature that has a larger average angles between the normals of the faces on the boundary and the cutting direction is defined as the primary feature whereas the other feature sharing the boundary is the secondary feature. For example, the target boundary shown in Figure 5.4(a) is shared by a pocket and a face feature. The pocket is the primary feature for the boundary while the face is the secondary feature.

To compute the cutter locations, the tangent normals between the boundary edge and the associated features need to be derived, as shown in Figure 5.4(b). The primary tangent normal $\overrightarrow{t_{p_i}}$ between the primary feature $p_i$ and the edge $e_i$ can be written as shown in Equation 5.2.

$$\overrightarrow{t_{p_i}} = \overrightarrow{n_{p_i}} \times \overrightarrow{n_{e_i}}, \tag{5.2}$$

where $\overrightarrow{n_{p_i}}$ are the normal direction of the boundary point $P_i$ on the primary feature.

The secondary tangent normal $\overrightarrow{t_{s_i}}$ between the edge $e_i$ and the secondary feature

$s_i$ follows a similar pattern, as shown in Equation 5.3.

$$\overrightarrow{t_{s_i}} = \overrightarrow{n_{e_i}} \times \overrightarrow{n_{s_i}}, \tag{5.3}$$

where $\overrightarrow{n_{s_i}}$ are the normal direction of the point $P_i$ on the secondary feature.

As shown in Figure 5.4(c), to create a chamfer with a constant width along the boundary, the cutter center location is set at a point along the direction that bisects the primary and secondary tangent normals, as given by Equation 5.4.

$$\overrightarrow{t_i} = \frac{\overrightarrow{t_{p_i}} + \overrightarrow{t_{s_i}}}{2} \tag{5.4}$$

As the target chamfer width is usually much smaller relative to the radius of the ball cutter, the chord error caused by the curvature of the cutter can be neglected and the resultant chamfer can be treated as a straight line. Thus, the distance between the boundary point $P_i$ and its farthest point $F_i$ on the resulting chamfer, $|\overrightarrow{P_iF_i}|$, called the offset distance, can be approximated using Equation 5.5.

$$|\overrightarrow{P_iF_i}| = \frac{W_{target}}{2 * tan(\frac{\beta_i}{2})}, \tag{5.5}$$

where $W_{target}$ is the target chamfer width and $\beta_i$ is the edge angle.

The edge angle $\beta_i$ is the angle between the neighboring surfaces at the edge $e_i$. It is derived in Equation 5.6.

$$\beta_i = acos(\frac{\overrightarrow{t_{p_i}} \cdot \overrightarrow{t_{s_i}}}{|\overrightarrow{t_{p_i}}| \cdot |\overrightarrow{t_{s_i}}|}) \tag{5.6}$$

Combining Equations 5.4 and 5.5, the location of the tool center point is given by Equation 5.7.

$$\overrightarrow{TCP_i} = \overrightarrow{P_i} + (|\overrightarrow{P_iF_i}| - R_{cutter})\frac{\overrightarrow{t_i}}{|\overrightarrow{t_i}|}, \tag{5.7}$$

where $R_{cutter}$ is the radius of the ball cutter.

Finally, the cutter location, which is defined at the tip of the cutter, is computed to create the tool path. The method to compute the cutter location is shown in Equation 5.8.

$$\overrightarrow{CL_i} = \overrightarrow{TCP_i} - R_{cutter}\overrightarrow{n_{cutting}}, \tag{5.8}$$

where $\overrightarrow{n_{cutting}}$ is the defined cutting direction.

The above process is applied on all the points on the boundary to obtain the cutter locations for deburring each edge and the final tool path is a polyline connected by these cutter locations. The generated tool path for deburring the target boundary between the pocket and the face is shown in Figure 5.5.

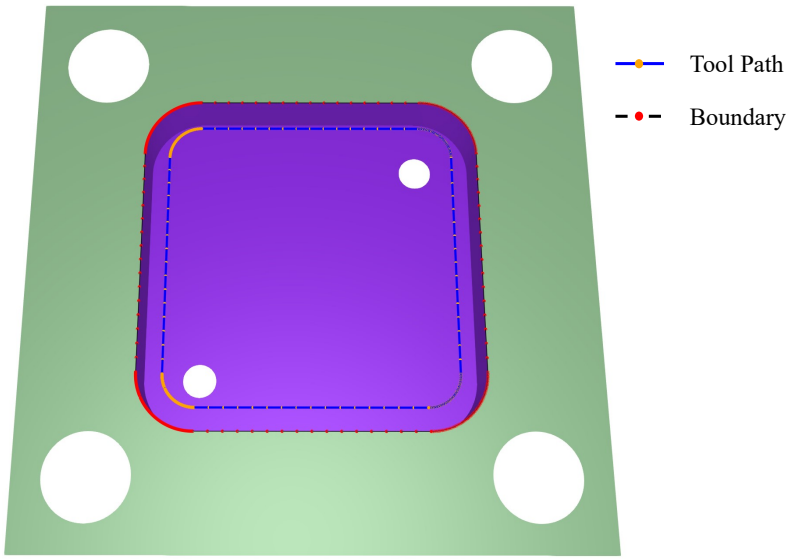


Figure 5.5: Target boundary, its associated features, and the generated tool path to deburr the boundary.

## 5.3 Estimation of Boundary Point Normals

As presented in the previous section, the normals of the boundary point at the primary and secondary features are required to compute the cutter locations. In analytic geometry, the point normals can be obtained directly from the equation of the surface. However, in discrete geometry like the triangle mesh, as shown in Figure

5.6(a), no surface equation is available to compute the point normals. In this research, a cutter shadowed approach is developed to estimate the normals for the boundary points.

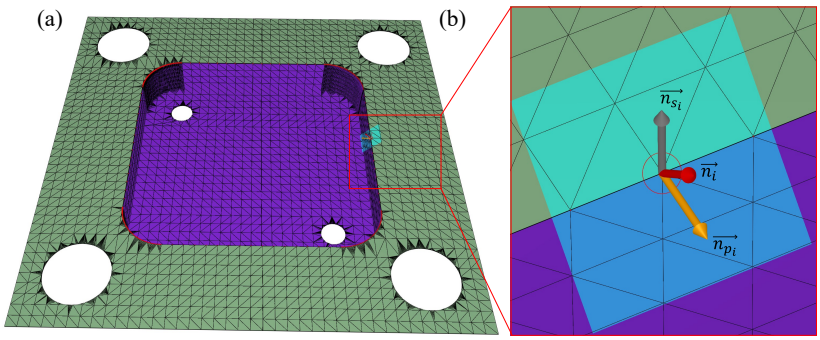


Figure 5.6: Estimation of normals of boundary points: (a), mesh and boundary; (b), initial normal and estimated point normals at the $P_i$.

In the first step, the initial normal is computed by averaging the normal of the triangles that share the boundary point. The triangles on the neighboring features are then projected onto the plane formed by the boundary point and the initial normal vector. At the same time, a circle with a radius of the chamfer width and the center at the boundary point is created. The intersecting areas of each projected triangle with the circle is subsequently computed. The adjusted point normal is equal to the area averaging of the face normals. The normal of the point at the primary feature is computed using Equation 5.9. Similarly, the normal of the point at the secondary feature can be computed based on Equation 5.10. As shown in Figure 5.6(b), the normals of the boundary point $P_i$ at the primary and secondary features are computed based on the proposed approach.

$$\overrightarrow{n_{p_i}} = \frac{\sum_{j=1}^{N} A_{p_{f_j}} \overrightarrow{n_{p_{f_j}}}}{\sum_{j=1}^{N} A_{p_{f_j}}}, \tag{5.9}$$

where N is the number of projected triangles on the primary feature $p_i$ that intersect

with the circle, $A_{p_{f_j}}$ is the area of projected triangle $p_{f_j}$, and $\overrightarrow{n_{p_{f_j}}}$ is the face normal of projected triangle $p_{f_j}$.

$$\overrightarrow{n_{s_i}} = \frac{\sum_{j=1}^{N} A_{s_{f_j}} \overrightarrow{n_{s_{f_j}}}}{\sum_{j=1}^{N} A_{s_{f_j}}}, \tag{5.10}$$

where N is the number of projected triangles on the secondary feature $s_i$ that intersect with the circle, $A_{s_{f_j}}$ is the area of projected triangle $s_{f_j}$, and $\overrightarrow{n_{s_{f_j}}}$ is the face normal of projected triangle $s_{f_j}$.

## 5.4 Gouging Detection

Gouging refers to a situation when the cutter removes excess material from the workpiece in a cutter location point. The initially generated tool path cannot guarantee a gouging-free process as the engagement between the cutter and the workpiece is not considered in the tool path generation process. A depth map approach is employed to detect gouging in each cutter location. As shown in Figure 5.7(a), a depth map is generated for the part geometry by projecting the geometry onto a plane, called the base plane, which is oriented at the cutting direction. The geometry of the cutter is also represented as a depth map by projecting the cutter geometry onto the plane at the bottom of the cutter and oriented to its axis. An example of a cutter depth map is shown in Figure 5.7(b). The region to be removed by the cutter during the deburring process, which is safe to be cut is labeled as the safe zone on the part's depth map. The safe zone shown in Figure 5.7(c) is created by projecting the target removal region onto the base plane.

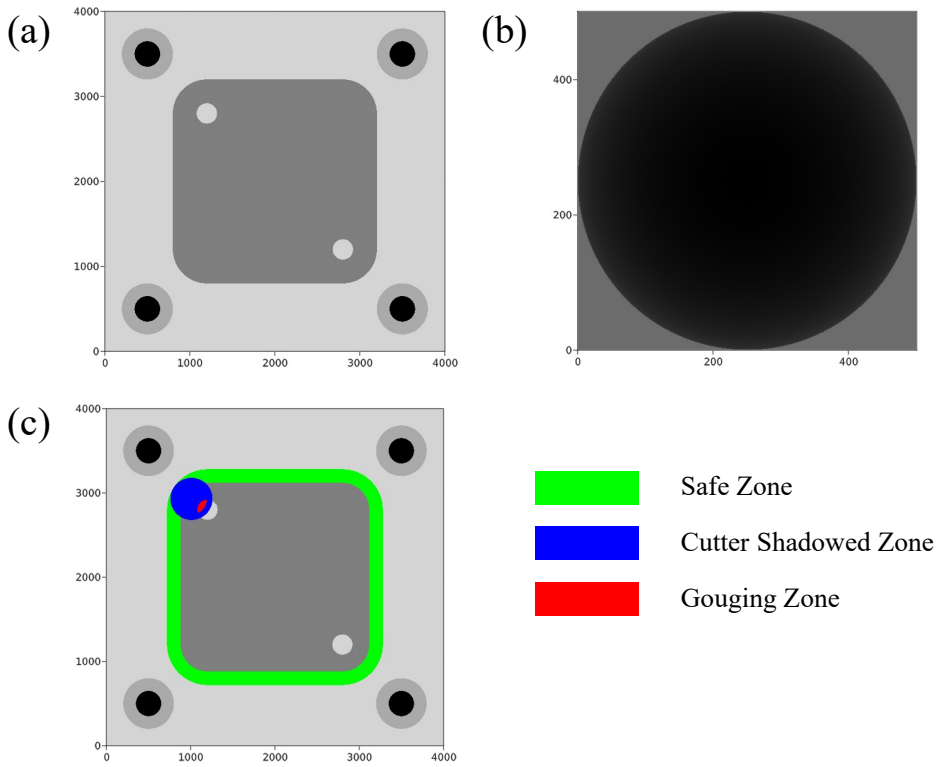


Figure 5.7: Gouging detection based on depth map: (a), depth map of the part; (b), depth map of the ball cutter; (c), safe zone, cutter shadowed zone, and gouging zone on the part depth map.

**Algorithm 5.2** Gouging Detection

**Input :** $CL[x, y]_i$ is the cutter location
**Input :** $depth_i$ is the depth of the cutter at the cutter location
**Input :** $part_{dm}$ is the depth map of the part
**Input :** $cutter_{dm}$ is the depth map of the cutter
**Input :** $img_{sz}$ is the binary image of the safe zone

$h, w = size(cutter_{dm})$
$c_x = ceil(h/2), c_y = ceil(w/2)$ ▷ $c_x$, $c_y$ is the cutter center
$x, y = CL[x, y]_i$
**for** $dx = 1 : h, dy = 1 : w$ **do**
    **if** $img_{sz}[x + dx - c_x, y + dy - c_y] == true$ **then**
        **continue**
    **end if**
    **if** $part_{dm}[x + dx - c_x, y + dy - c_y] > cutter_{dm}[dx, dy] + depth_i$ **then**
        **return** $true$
    **end if**
**end for**
**return** $false$

After the depth maps and safe zone are built, Algorithm 5.2 is applied to each cutter location to detect gouging. If the depth at the cutter location is less than the depth of the part at the same location, the cutter is gouging the workpiece. Figure 5.7(c) shows a gouging location where the ball end cutter gouges the boss when deburring the boundary between the pocket and the top face.

## 5.5 Tool Path Modification

It is essential to ensure non-gouging at every position of the cutter; otherwise, the cutter will remove excessive material from the part. In this research, a tool path modification algorithm (i.e., Algorithm 5.3) is developed to adjust the cutter location once gouging is detected. Based on the depth map approach for gouging detection, the algorithm loops over each cutter location in the tool path and checks if the cutter is gouging with the part at the location. If a gouging location is detected, the cutter is moved closer to the boundary by one pixel width and gouging detection is checked once again at the new location; this process continues until there is no gouging. The cutter is then adjusted to the new location without gouging. The tool path after modification is shown in Figure 5.8(a) and (b). Compared with the original tool path shown in Figures 5.5 and 5.8(c), the cutter locations in the two corner regions close to the bosses are lifted up to avoid gouging while the other cutter locations remain unchanged.

**Algorithm 5.3** Tool Path Modification

**Input :** $CLs$ is the cutter locations in the tool path

**Input :** $Vs$ is the offset vectors $\overrightarrow{PF}$

**Input :** $dir$ is the cutting direction

**Input :** $part_{dm}$ is the depth map of the part

**Input :** $cutter_{dm}$ is the depth map of the cutter

**Input :** $img_{sz}$ is the binary image of the safe zone

**Input :** $resolution$ is the length of the pixel in the depth maps

1: $n = length(CLs)$ is the number of cutter locations
2: **for** $i = 1 : n$ **do**
3: $CL = CLs[i]$, $V = Vs[i]$
4: $CL[x, y], depth = project(CL, dir, base\_plane)$ ▷ project onto the base plane
5: $V_H, distance = project(V, dir, base\_plane)$ ▷ project onto the base plane
6: **if** $is\_gouging(CL[x, y], depth, part_{dm}, cutter_{dm}, img_{sz})$ ▷ gouging detection **then**
7: $n\_steps = distance/resolution$
8: **for** $j = 1 : n\_steps$ **do**
9: $CL_{new}[x, y] = CL[x, y] - V_H * j * resolution$
10: $depth = get\_depth(CL_{new}[x, y], part_{dm}, cutter_{dm}, img_{sz})$
11: **if** $is\_gouging(CL_{new}[x, y], depth, part_{dm}, cutter_{dm}, img_{sz})$ **then**
12: $CLs[i] = get\_location(CL_{new}[x, y], depth)$
13: **break**
14: **end if**
15: **end for**
16: **else**
17: **continue**
18: **end if**
19: **end for**
20: **return** $CLs$

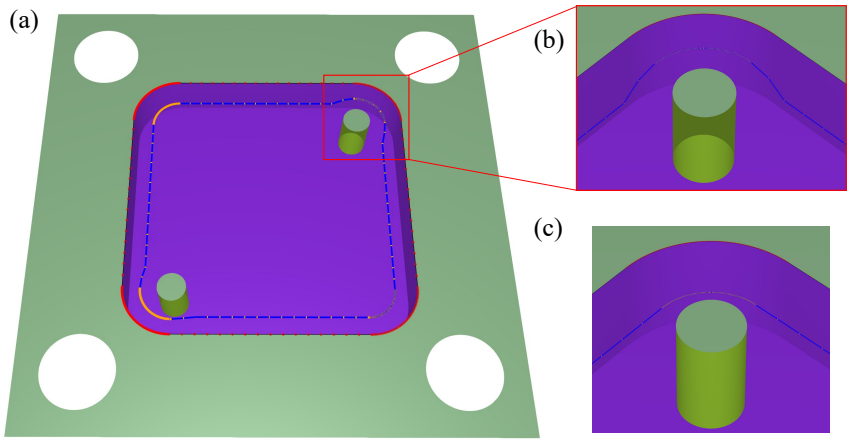


Figure 5.8: Tool path after modification: (a), the modified tool path; (b), close-up view of the modified tool path; (c), close-up view of the tool path before modification.

# CHAPTER 6: EXPERIMENTAL RESULTS

Chapters 4 and 5 introduced the automated tool path planning methods for generating tool paths for the milling and deburring operations based on the machining feature recognition technique developed in Chapter 3. This chapter presents the experimental results of the methodologies when used to manufacture two parts with multiple machining features.

## 6.1 Equipment

A Tormach 770MX vertical CNC mill was used to machine the test parts to validate the generated tool paths. The tool paths were post-processed into G-code to run the CNC machine.

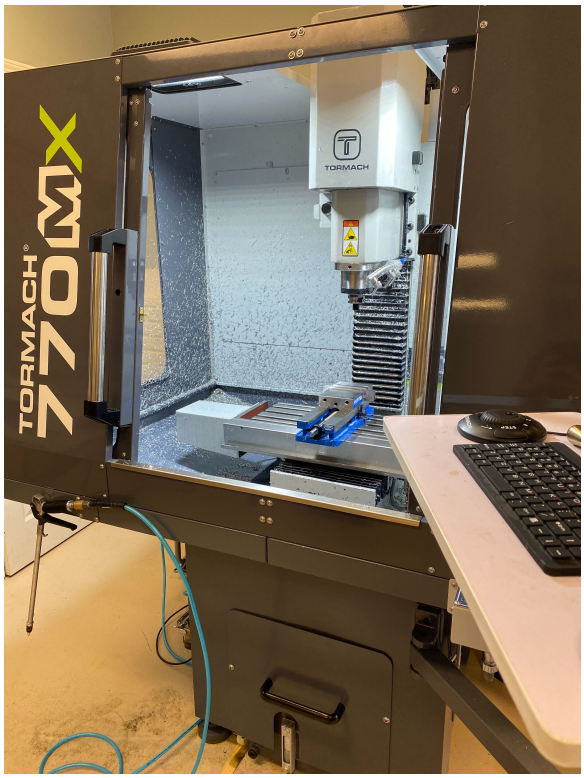


Figure 6.1: Tormach770MX CNC Mill.

As shown in Table 6.1, the CNC machine is equipped with six flat end mills with different diameters ranging from 1.59 to 9.53 mm and as well as a ball end mill with a diameter of 3.18 mm. When generating the tool paths for milling the first test part, the cutters were automatically selected from the available set of flat end mills. In addition, the machine does not have an automatic tool changer. The time for the manual tool change was set to 30 seconds when estimating the machining cost in each operation of the graph. The ball end mill was used for deburring the second test part.

Table 6.1: Tool library of the Tormach CNC machine.

| Cutter | Diameter |
|---|---|
| flat end mill | 9.53 mm |
| flat end mill | 7.94 mm |
| flat end mill | 6.35 mm |
| flat end mill | 4.76 mm |
| flat end mill | 3.18 mm |
| flat end mill | 1.59 mm |
| ball end mill | 3.18 mm |

## 6.2 Milling Operations

The method for automatically generating the milling tool paths was tested on an example of a part modelled as a 2X4 Lego brick. The part has multiple machining features including 11 bosses, 1 profile, 1 pocket, and 13 faces, as shown in Figure 6.2. These features have different accessibility for a cutter to machine from a selected machining direction. When the +Z direction is defined as the cutting direction, only the 8 bosses, 9 faces, and the profile on the top of the part are accessible whereas the pocket, the profile, the other 3 bosses, and the other 4 faces on the bottom of the part can only be cut completely from the -Z direction. In this research, the tool paths were generated from both the +Z and -Z directions to demonstrate the ability

of the proposed method to generate tool paths from an arbitrary user selected cutting direction.

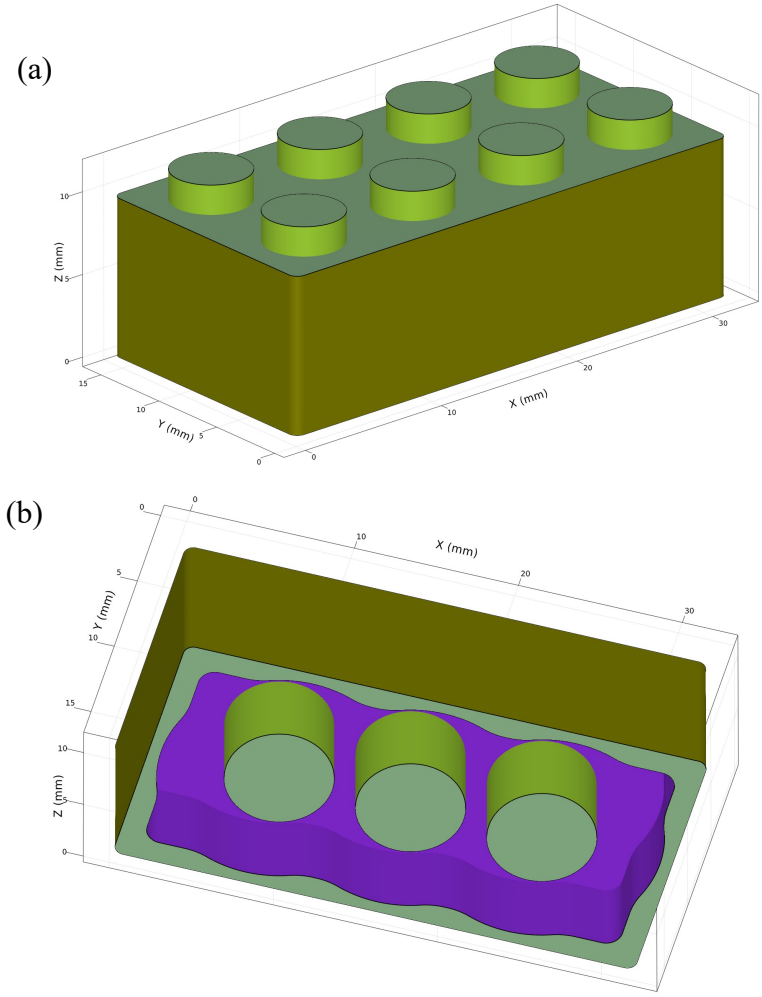


Figure 6.2: An example part for testing automated tool path planning methodology for the milling operations: (a), top view; (b), bottom view.

The tool paths for cutting the Lego part from the +Z direction are shown in Figures 6.3, 6.4, 6.5, and 6.6. These tool paths are generated for four operations. The first operation is a facing operation. The second operation is created to remove the material for the profile feature. The third and fourth operations are produced to clear the large face feature with four bosses inside the face. The tool paths for cutting the Lego part from the -Z direction are shown in Figures 6.8, 6.8, and 6.9. Three operations are created for machining the part. The same as the machining operations

from the +Z direction, the first operation is a facing operation, the second operation is to clear the profile feature, and the third operation is to cut the material inside the pocket feature. The details of the operations are described as follows.

To machine the part from the +Z direction, a facing operation was selected as the first operation because the size of the stock material is slightly larger than the part. The subsequent operation cannot proceed if the extra material on top of the stock is not removed. As shown in 6.3, the Zig-Zag tool paths is generated for the facing operation. The largest flat end mill in the tool library, which has a diameter of 9.53 mm, was automatically selected as the cutter for the operation.

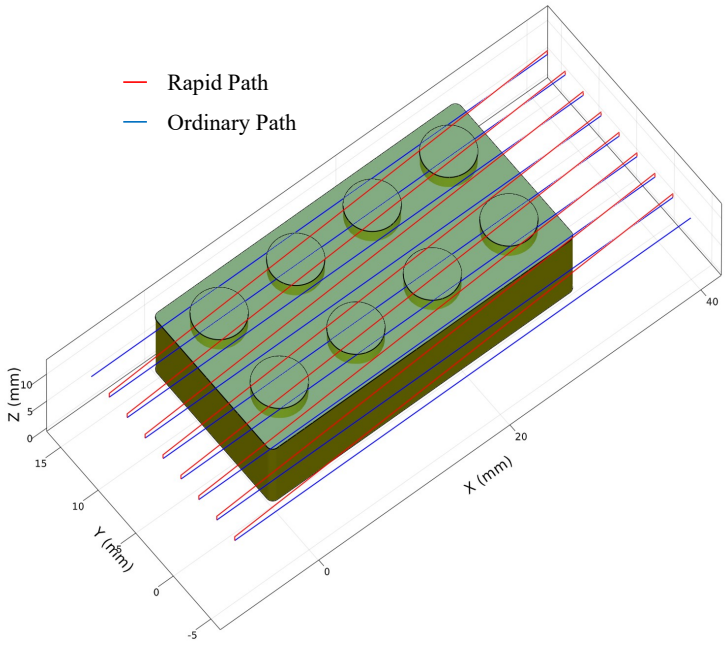


Figure 6.3: Tool paths for the first operation when machining from the +Z direction of the test part.

A non-zero removal volume was subsequently detected for the profile feature. Thus, the tool paths, as shown in Figure 6.4, were generated to remove the material outside the profile. Only the 9.53 mm diameter cutter is used. As all the edges of the profile were convex, the largest cutter was sufficient to clear all the material.

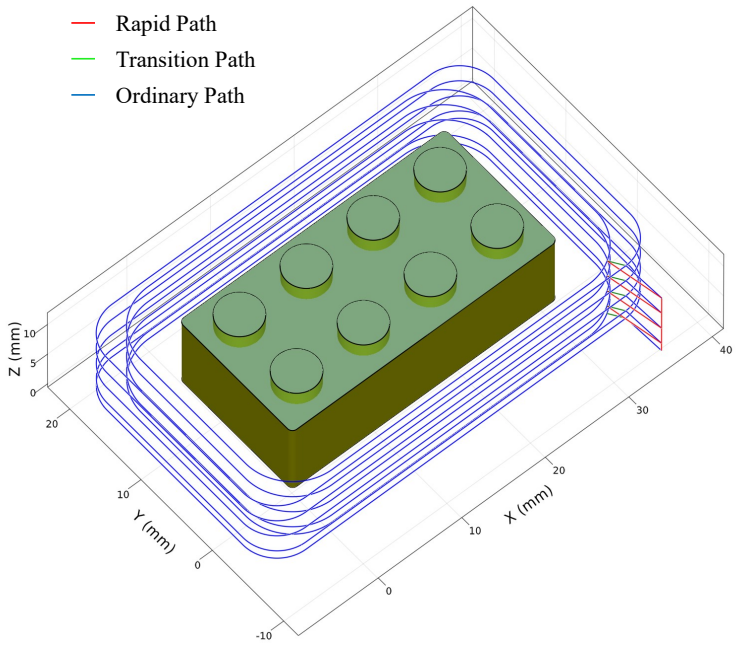


Figure 6.4: Tool paths for the second operation when machining from the +Z direction of the test part.

Ultimately, the largest flat end mill with a diameter of 9.53 mm and the smallest flat end mill with a diameter of 1.59 mm were selected by the graph approach as the optimal set of cutters to machine the face feature. The large cutter was used to cut the material outside the bosses and the small cutter was selected to the clear the material between the bosses. The tool paths for the large and the small cutters are depicted in Figures 6.5 and 6.6, respectively.

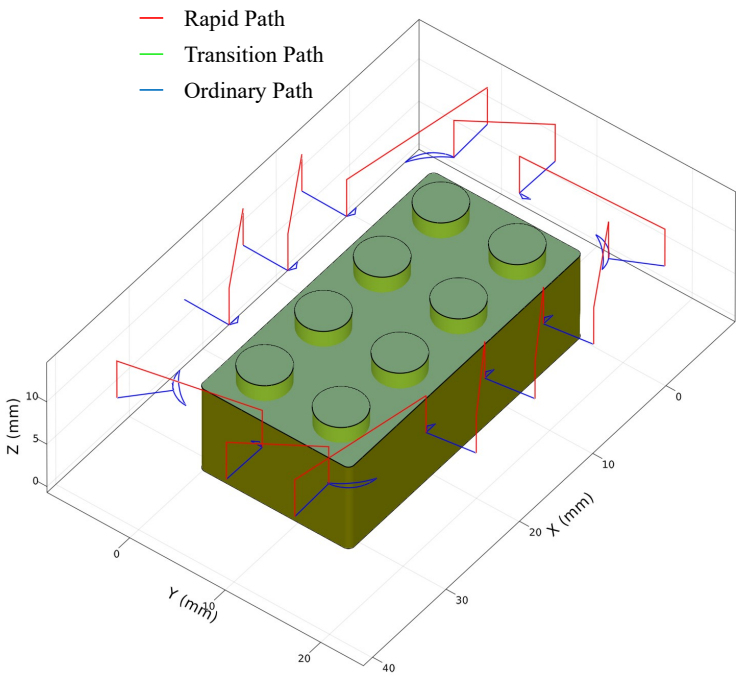


Figure 6.5: Tool paths for the third operation when machining from the +Z direction of the test part.

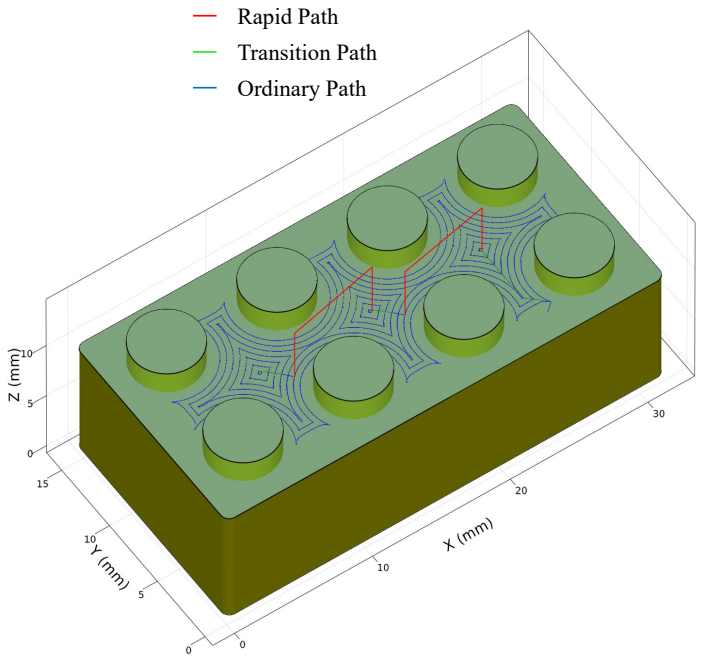


Figure 6.6: Tool paths for the fourth operation when machining from the +Z direction of the test part.

When machining the part from the -Z direction, three operations were determined

as the set of operations that can remove all material outside the volume bounded by the part design from the stock material. Similarly, the first operation was a facing operation that cleared the extra material on top of the part while the second operation cut the material outside the profile feature; the tool paths for the first two operations are shown in Figures 6.7 and 6.8, respectively. In the final operation, only the smallest flat end mill was selected because the cutter has the best accessibility for machining the pocket feature and can cut the pocket with a minimum amount of machining cost. The tool paths for the third operation are shown in Figure 6.9.

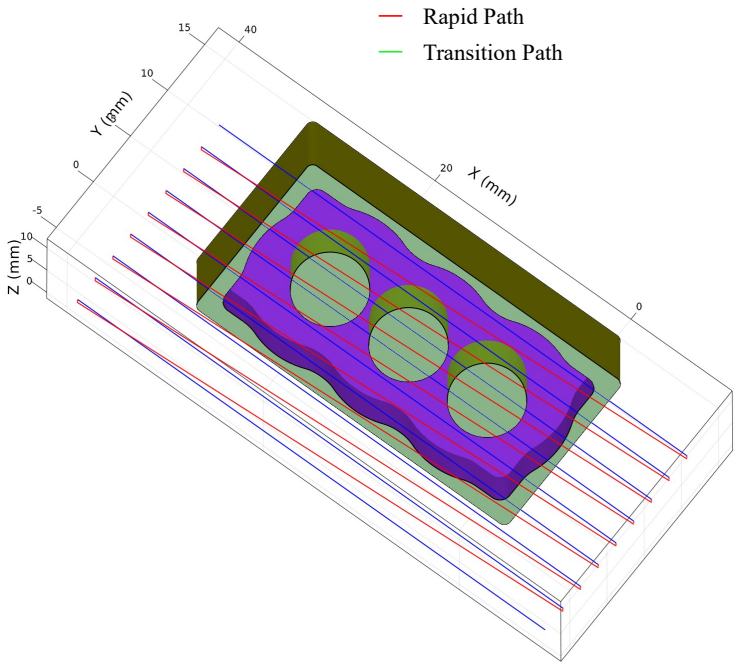


Figure 6.7: Tool paths for the first operation when machining from the -Z direction of the test part.

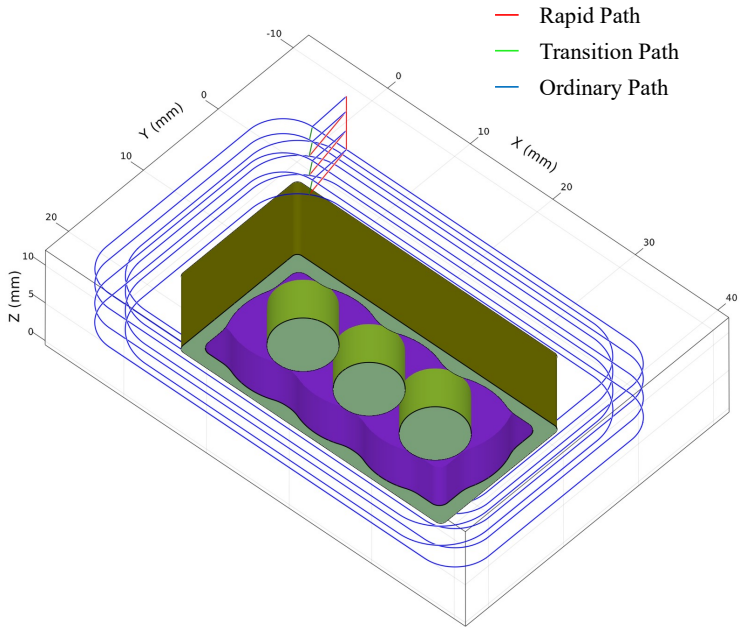


Figure 6.8: Tool paths for the second operation when machining from the -Z direction of the test part.

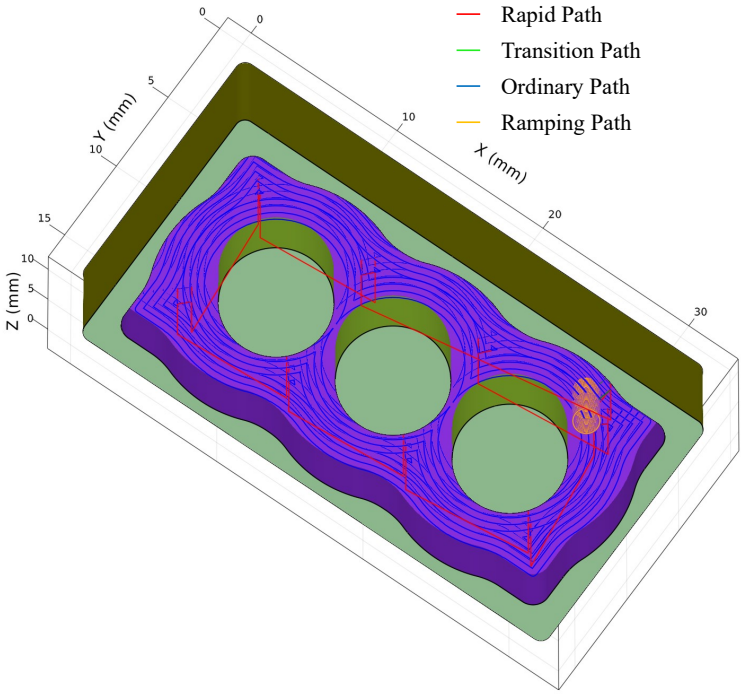


Figure 6.9: Tool paths for the third operation when machining from the -Z direction of the test part.

The tool paths for machining the Lego part from the +Z and -Z directions were

post-processed into G-code files, shown in Figure 6.10(a), to run the CNC mill so that the two parts could be machined to validate the effectiveness of the tool paths. The machining results of the two parts are shown in Figure 6.10(b), which indicates that the tool paths by using the introduced tool path planning method for milling with multiple cutters can machine the stock into the desired shapes from both the +Z and -Z directions.

(a)

```
%
O0001
(Toolpath Automatically Created  Gcode)
G90 G94 G17 G49 G40 G80
G20
G53 G0 Z0.0

G54
G30
T13 G43 H13
M6 S10000.0
M3
M8
G0 X0.9875 Y-0.51562
G0 Z1.0
G1 X0.98750 Y-0.51562 Z0.00000 F20.0
X-0.93750
G1 X-0.93750 Y-0.51562 Z0.00000 F200.0
Z0.07559
X0.98750 Y-0.42188 Z0.00000
G1 X0.98750 Y-0.42188 Z0.00000 F20.0
X-0.93750
G1 X-0.93750 Y-0.42188 Z0.00000 F200.0
Z0.07559
.....
```

(b)

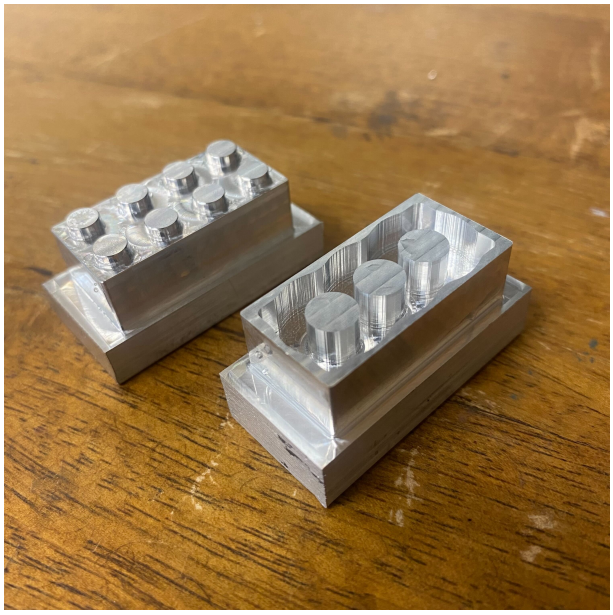

Figure 6.10: G-code and machining results of the test part: (a), G-code file; (b), picture of the machined parts from +Z and -Z directions, separately.

## 6.3 Deburring Operation

An example of a part with multiple machining features, shown in 6.11(a) and (b), was tested to validate the automated deburring tool path planning method. These machining features need to be manufactured by different machining operations and from different cutting directions on a 3-axis CNC machine. The cylindrical face and the profile need to be cut by a milling operation from the +Z direction and the four holes and the four through holes can be cut using a drilling operation also from the +Z direction. The other cylindrical face and two planar faces can only be milled from the

-Z direction. After machining the face, the profile, the blind holes, and the through holes from the +Z direction, burrs may occur on the sharp edges. These edges need to be deburred before assembling; otherwise, the assembling accuracy might be affected. For example, the through holes might be difficult for a screw to enter so that the part cannot be assembled with its mating part securely. The proposed approach was tested on this part to verify whether the sharp edges can be automatically detected, and the deburring tool paths can be successively generated to remove the burrs on the boundaries.

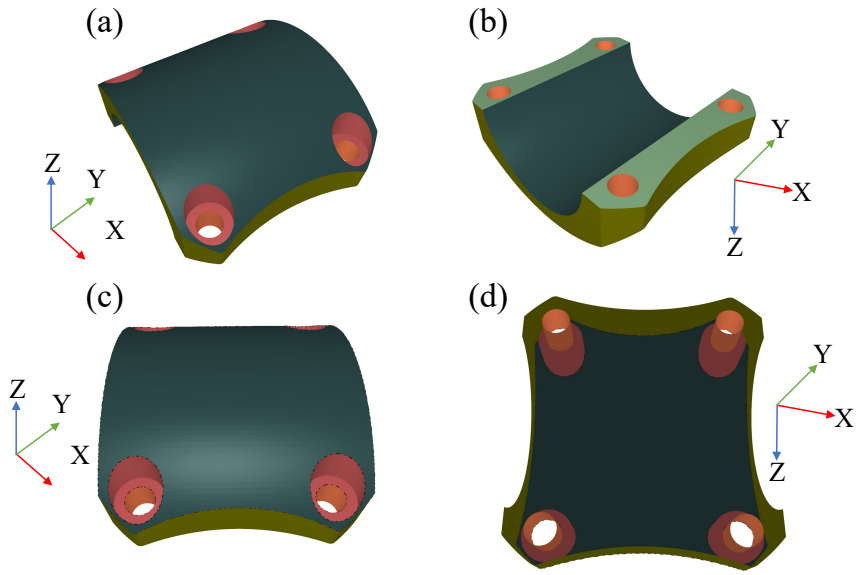


Figure 6.11: An example part for testing the deburring tool path planning methodology: (a), top view of the machining features; (b), bottom view of the machining features; (c), top view of the detected feasible boundaries and their associated features; (d), bottom view of the detected feasible boundaries and their associated features.

As shown in Figure 6.11(c) and (d), given +Z as the cutting direction, the boundaries on the top of the profile, four blind holes, and four through holes were detected as the feasible boundaries. These boundaries contain the sharp edges that need to be deburred. The related machining features were detected correctly. The gouging-free deburring tool paths were then generated automatically for each boundary using the 3.18 mm ball end cutter as shown in Figure 6.12. The generated tool paths were

post-processed into G-code and tested on the Tormach CNC machine to demonstrate the ability of the proposed approach to generate the effective deburring tool path that can create chamfer with a constant width for complex boundaries with arbitrary shapes.

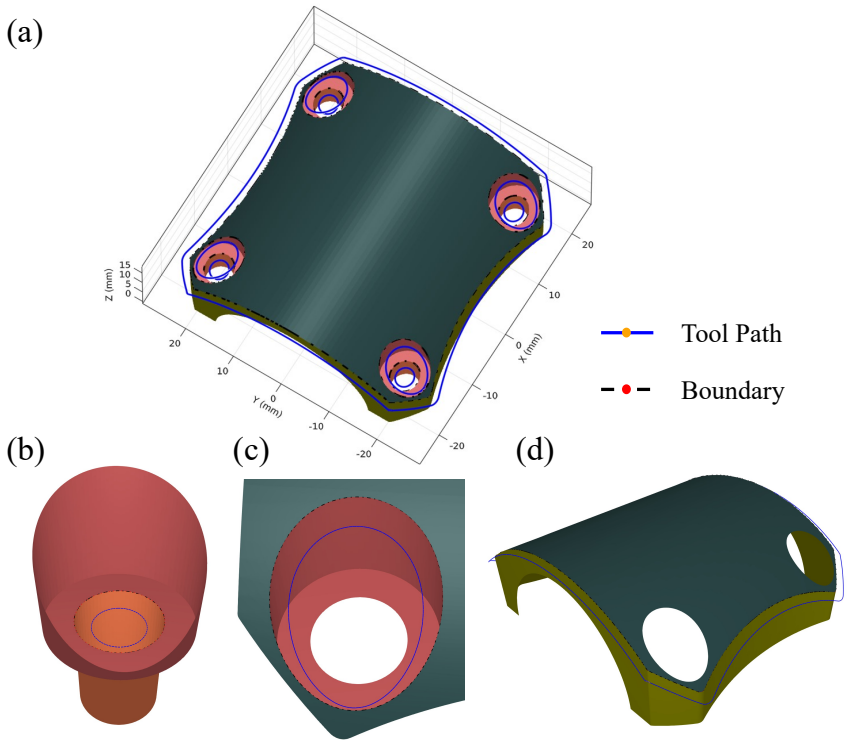


Figure 6.12: Tool path generation for the example part: (a), generated tool paths for each feasible boundary; (b), close-up view of the tool path for the boundary between the through hole and the blind hole; (c), close-up view of the tool path for the boundary between the blind hole and the face on top; (d), close-up view of the tool path for the boundary between the face on top and the profile.

The top section of the example part was first manufactured on the Tormach vertical CNC mill. The blind and through holes were created using drilling operations while the top face and profile were machined using a milling operation. As shown in Figure 6.13, burrs with irregular shapes were formed on the the edges of the holes and the profile after the drilling and milling operations.

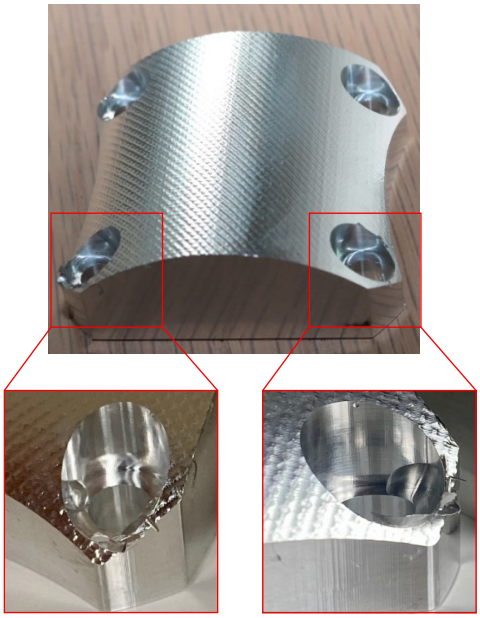

Figure 6.13: Test part before deburring the edges.

The generated deburring tool paths were post-processed into G-code to run on the CNC to deburr the part. The machining result is shown in Figure 6.14. The burrs on all the sharp edges were completely removed and relative uniform chamfers were formed along the boundaries.

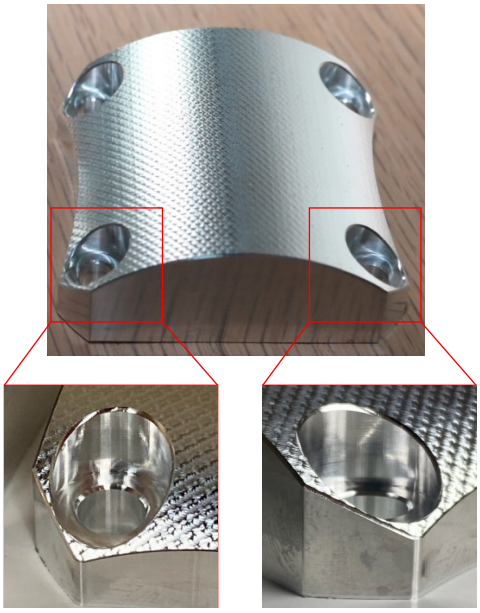

Figure 6.14: Test part after deburring the edges.

# CHAPTER 7: CONCLUSIONS AND FUTURE WORK

## 7.1 Conclusions

CNC tool path planning still currently relies heavily on the experience of the process engineer. To improve machining efficiency and reduce manufacturing costs, this dissertation has presented a feature-based automated tool path planning system. The primary contributions of this dissertation are summarized as follows:

1. A new robust segmentation algorithm with three procedures is introduced to detect primitives and freeform shapes within discrete geometry.

2. A multiple cutter selection and optimization framework is created to select the optimal set of cutters with a minimum machining time for cutting arbitrary machining features.

3. An automated deburring tool path planning algorithm is developed to generate deburring tool paths with a ball end cutter to remove the undesired burrs from the feasible boundaries.

Unlike machining feature recognition from analytic geometry, region segmentation is a fundamental and essential procedure in recognizing machining features from discrete geometry. A new robust segmentation algorithm is introduced to segment the discrete geometry represented by the triangle mesh into different surface regions. Each surface region represent a primitive shape (plane, cylinder, sphere, cone, or torus) or a freeform shape.

A new framework that can generate roughing tool paths with multiple cutters is developed to generate tool paths for cutting arbitrary $2\frac{1}{2}D$ machining features with the optimal set of cutters. The machining process based on the proposed method can produce parts with an expected surface quality with a minimum cutting time.

Automated tool path generation for deburring sharp edges is an important module for any automated machining system but is rarely available. This dissertation presents an automated deburring tool path planning technique. The feasible boundaries are detected automatically. Deburring tool paths with a ball end cutter are then automatically generated to remove the burrs and create a chamfer with a constant width along each feasible boundary.

The milling and deburring tool paths were post-processed into G-code files and run on a CNC milling machine to manufacture two examples of parts. The machining results demonstrate the validity of the milling and deburring tool paths generated by the methodology presented in this dissertation.

## 7.2 Future Work

Although the automated tool path planning methodology based on the automated recognition of machining features is promising, several important topics still need to be investigated. A fully automated tool path planning system needs to generate efficient tool paths for the whole part automatically so that no human decision is required.

### 7.2.1 Maximum Cutter Selection for Pockets

First and foremost, the method for selecting the maximum cutter to machine pocket features can be optimized. The maximum cutter selected using the MIC method can only cut a single point in many cases. Starting from this cutter to build the cost graph may increase the computational cost as the depth of the cost graph is increased. The medial axis transform approach [98], which can be used to compute the maximum circle on the topological skeleton of the object, may be applied to select the maximum cutter that can cut a certain amount of material from the pockets.

### 7.2.2 Optimal Sequence for Machining Multiple Features

In addition, an optimal sequence for machining multiple features is neccessary for a fully automated tool path planning system. Mechanical parts usually consist of multiple features. The approach to machining the multiple features in an optimal sequence considering the geometry constraints between different operations is one of the key milestones for the fully automated tool path planning system.

### 7.2.3 Cutter Selection and Optimization for Multiple Machining Features

Another important investigation should be focused on the selection and optimization for multiple machining features. When selecting the optimal set of cutters for machining a single feature, it is important to consider whether a cutter can be used for multiple features should also be considered. The graph based approach for selecting cutters can possibly be adapted for selecting cutters for multiple cutters by treating the IPM for each feature as a node of the graph.

### 7.2.4 Feeds and Speeds Optimization

Finally, the methodology for optimizing the feeds and speeds for the generated tool paths is also vital for a fully automated tool path planning system. The feeds and speeds might be optimized based on the depth of cut, width of cut, chip load, tool properties, and the property of the stock material. In addition, the acceleration and deceleration may also be considered in the optimization of feedrate because the actual feedrate may vary based on the acceleration and deceleration when the cutter moves from one G-code block to another block.

# REFERENCES


[1] A. K. Verma and S. Rajotia, "A review of machining feature recognition methodologies," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 4, pp. 353–368, 2010.

[2] J. E. Bobrow, "Nc machine tool path generation from csg part representations," *Computer-Aided Design*, vol. 17, no. 2, pp. 69–76, 1985.

[3] Y. S. Suh and K. Lee, "Nc milling tool path generation for arbitrary pockets defined by sculptured surfaces," *Computer-Aided Design*, vol. 22, no. 5, pp. 273–284, 1990.

[4] S. Ding, M. Mannan, A. N. Poo, D. Yang, and Z. Han, "Adaptive iso-planar tool path generation for machining of free-form surfaces," *Computer-Aided Design*, vol. 35, no. 2, pp. 141–153, 2003.

[5] P. Broomhead and M. Edkins, "Generating nc data at the machine tool for the manufacture of free-form surfaces," *International Journal of Production Research*, vol. 24, no. 1, pp. 1–14, 1986.

[6] G. C. Loney and T. M. Ozsoy, "Nc machining of free form surfaces," *Computer-Aided Design*, vol. 19, no. 2, pp. 85–90, 1987.

[7] G. Elber and E. Cohen, "Toolpath generation for freeform surface models," *Computer-Aided Design*, vol. 26, no. 6, pp. 490–496, 1994.

[8] Y. M. Wang and X. S. Tang, "Five-axis nc machining of sculptured surfaces," *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 1, pp. 7–14, 1999.

[9] K. Suresh and D. Yang, "Constant scallop-height machining of free-form surfaces," *Journal of Manufacturing Science and Engineering*, vol. 116, no. 2, pp. 253–259, 1994.

[10] H.-Y. Feng and H. Li, "Constant scallop-height tool path generation for three-axis sculptured surface machining," *Computer-Aided Design*, vol. 34, no. 9, pp. 647–654, 2002.

[11] C. Tournier and E. Duc, "Iso-scallop tool path generation in 5-axis milling," *The International Journal of Advanced Manufacturing Technology*, vol. 25, no. 9, pp. 867–875, 2005.

[12] B. H. Kim and B. K. Choi, "Machining efficiency comparison direction-parallel tool path with contour-parallel tool path," *Computer-Aided Design*, vol. 34, no. 2, pp. 89–95, 2002.

[13] S. Yuwen, G. Dongming, W. Haixia, *et al.*, "Iso-parametric tool path generation from triangular meshes for free-form surface machining," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 7, pp. 721–726, 2006.

[14] B. W. Niebel, "Mechanized process selection for planning new designs," *ASME paper*, vol. 737, 1965.

[15] N. Ahmad, A. Haque, and A. Hasin, "Current trend in computer aided process planning," in *Proceedings of the 7th Annual Paper Meet and 2nd Intern. Conf*, pp. 25–27, Citeseer, 2001.

[16] M. Al-Wswasi, A. Ivanov, and H. Makatsoris, "A survey on smart automated computer-aided process planning (acapp) techniques," *The International Journal of Advanced Manufacturing Technology*, vol. 97, no. 1, pp. 809–832, 2018.

[17] X. Xu, L. Wang, and S. T. Newman, "Computer-aided process planning–a critical review of recent developments and future trends," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. 1, pp. 1–31, 2011.

[18] Y. Yusof and K. Latif, "Survey on computer-aided process planning," *The International Journal of Advanced Manufacturing Technology*, vol. 75, no. 1, pp. 77–89, 2014.

[19] J. J. Shah, "Assessment of features technology," *Computer-Aided Design*, vol. 23, no. 5, pp. 331–343, 1991.

[20] S. C. Park, "Knowledge capturing methodology in process planning," *Computer-Aided Design*, vol. 35, no. 12, pp. 1109–1117, 2003.

[21] Y. Yue, L. Ding, K. Ahmet, J. Painter, and M. Walters, "Study of neural network techniques for computer integrated manufacturing," *Engineering Computations*, 2002.

[22] F. Zhang, Y. Zhang, and A. Y. C. Nee, "Using genetic algorithms in process planning for job shop machining," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 278–289, 1997.

[23] X. Xu and S. Hinduja, "Determination of finishing features in 21/2 d components," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 211, no. 2, pp. 125–142, 1997.

[24] R.-R. Wu, L. Ma, J. Mathew, and G.-H. Duan, "Optimal operation planning using fuzzy petri nets with resource constraints," *International Journal of Computer Integrated Manufacturing*, vol. 15, no. 1, pp. 28–36, 2002.

[25] D. Kiritsis, K.-P. Neuendorf, and P. Xirouchakis, "Petri net techniques for process planning cost estimation," *Advances in Engineering Software*, vol. 30, no. 6, pp. 375–387, 1999.

[26] W. Shen and D. H. Norrie, "Agent-based systems for intelligent manufacturing: a state-of-the-art survey," *Knowledge And Information Systems*, vol. 1, no. 2, pp. 129–156, 1999.

[27] F. Zhao, "A cooperative framework for process planning," *International Journal of Computer Integrated Manufacturing*, vol. 12, no. 2, pp. 168–178, 1999.

[28] F. Zhao, S. Tso, and P. S. Wu, "A cooperative agent modelling approach for process planning," *Computers in Industry*, vol. 41, no. 1, pp. 83–97, 2000.

[29] C.-F. You and C.-H. Lin, "Java-based computer-aided process planning," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 9, pp. 1063–1070, 2005.

[30] ISO 14649-10: (2003c), "Industrial automation systems and integration-physical device control-data model for computerized numerical controllers-part 10: General process data," standard, International Organization for Standardization, 2003.

[31] ISO 10303-224:2006(E), "Industrial automation systems and integration-product data representation and exchange-part 224: Application protocol: Mechanical product definition for process planning using machining features," standard, International Organization for Standardization, 2006.

[32] M. Kang, J. Han, and J. Moon, "An approach for interlinking design and process planning," *Journal of Materials Processing Technology*, vol. 139, no. 1-3, pp. 589–595, 2003.

[33] L. Wang, H.-Y. Feng, and N. Cai, "Architecture design for distributed process planning," *Journal of Manufacturing Systems*, vol. 22, no. 2, pp. 99–115, 2003.

[34] H. Wang, X. Xu, and J. D. Tedford, "An adaptable cnc system based on step-nc and function blocks," *International Journal of Production Research*, vol. 45, no. 17, pp. 3809–3829, 2007.

[35] M. Liang, S. Ahamed, and B. Van Den Berg, "A step based tool path generation system for rough machining of planar surfaces," *Computers in Industry*, vol. 32, no. 2, pp. 219–231, 1996.

[36] M. Hou and T. Faddis, "Automatic tool path generation of a feature-basedcad/capp/cam integrated system," *International Journal of Computer Integrated Manufacturing*, vol. 19, no. 4, pp. 350–358, 2006.

[37] T. Xu, Z. Chen, J. Li, and X. Yan, "Automatic tool path generation from structuralized machining process integrated with cad/capp/cam system," *The International Journal of Advanced Manufacturing Technology*, vol. 80, no. 5, pp. 1097–1111, 2015.

[38] W. C. Regli, S. K. Gupta, and D. S. Nau, "Feature recognition for manufacturability analysis," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 13805, pp. 93–104, American Society of Mechanical Engineers, 1994.

[39] A. Yildiz, N. Öztürk, N. Kaya, and F. Öztürk, "Integrated optimal topology design and shape optimization using neural networks," *Structural and Multidisciplinary Optimization*, vol. 25, no. 4, pp. 251–260, 2003.

[40] J. Han, M. Kang, and H. Choi, "Step-based feature recognition for manufacturing cost optimization," *Computer-Aided Design*, vol. 33, no. 9, pp. 671–686, 2001.

[41] A. C. Lin and S.-Y. Lin, "A volume decomposition approach to process planning for prismatic parts with depression and protrusion design features," *International Journal of Computer Integrated Manufacturing*, vol. 11, no. 6, pp. 548–563, 1998.

[42] M. Marefat and R. L. Kashyap, "Automatic construction of process plans from solid model representations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1097–1115, 1992.

[43] M.-T. Wang, M. A. Chamberlain, A. Joneja, and T.-C. Chang, "Manufacturing feature extraction and machined volume decomposition in a computer-integrated feature-based design and manufacturing planning environment," *Computers in Industry*, vol. 23, no. 1-2, pp. 75–86, 1993.

[44] N. Gindy, "A hierarchical structure for form features," *The International Journal of Production Research*, vol. 27, no. 12, pp. 2089–2103, 1989.

[45] M. Fu, S.-K. Ong, W. F. Lu, I. Lee, and A. Y. Nee, "An approach to identify design and manufacturing features from a data exchanged part model," *Computer-Aided Design*, vol. 35, no. 11, pp. 979–993, 2003.

[46] Y. Woo and H. Sakurai, "Recognition of maximal features by volume decomposition," *Computer-Aided Design*, vol. 34, no. 3, pp. 195–207, 2002.

[47] R. V. Narang, "An application-independent methodology of feature recognition with explicit representation of feature interaction," *Journal of Intelligent Manufacturing*, vol. 7, no. 6, pp. 479–486, 1996.

[48] V. Sunil and S. Pande, "Automatic recognition of features from freeform surface cad models," *Computer-Aided Design*, vol. 40, no. 4, pp. 502–517, 2008.

[49] S. Kailash, Y. Zhang, and J. Y. Fuh, "A volume decomposition approach to machining feature extraction of casting and forging components," *Computer-Aided Design*, vol. 33, no. 8, pp. 605–617, 2001.

[50] H. L. Lockett and M. D. Guenov, "Graph-based feature recognition for injection moulding based on a mid-surface approach," *Computer-Aided Design*, vol. 37, no. 2, pp. 251–262, 2005.

[51] S. Joshi and T.-C. Chang, "Graph-based heuristics for recognition of machined features from a 3d solid model," *Computer-Aided Design*, vol. 20, no. 2, pp. 58–66, 1988.

[52] S. Gao and J. J. Shah, "Automatic recognition of interacting machining features based on minimal condition subgraph," *Computer-Aided Design*, vol. 30, no. 9, pp. 727–739, 1998.

[53] A. Z. Qamhiyah, R. Venter, and B. Benhabib, "Geometric reasoning for the extraction of form features," *Computer-Aided Design*, vol. 28, no. 11, pp. 887–903, 1996.

[54] H. K. Miao, N. Sridharan, and J. J. Shah, "Cad-cam integration using machining features," *International Journal of Computer Integrated Manufacturing*, vol. 15, no. 4, pp. 296–318, 2002.

[55] H. Sakurai, "Volume decomposition and feature recognition: Part 1-polyhedral objects," *Computer-Aided Design*, vol. 27, no. 11, pp. 833–843, 1995.

[56] H. Sakurai and P. Dave, "Volume decomposition and feature recognition, part ii: curved objects," *Computer-Aided Design*, vol. 28, no. 6-7, pp. 519–537, 1996.

[57] Y. S. Kim and D. J. Wilde, "A convergent convex decomposition of polyhedral objects," *Journal of Mechanical Design*, vol. 114, no. 3, pp. 468–476, 1992.

[58] M. Marefat and R. L. Kashyap, "Geometric reasoning for recognition of three-dimensional object features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 949–965, 1990.

[59] J. H. Vandenbrande and A. A. Requicha, "Spatial reasoning for the automatic recognition of machinable features in solid models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 12, pp. 1269–1285, 1993.

[60] X. Xu and S. Hinduja, "Recognition of rough machining features in $2\frac{1}{2}d$ components," *Computer-Aided Design*, vol. 30, no. 7, pp. 503–516, 1998.

[61] S. Prabhakar and M. R. Henderson, "Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models," *Computer-Aided Design*, vol. 24, no. 7, pp. 381–393, 1992.

[62] Y. Shi, Y. Zhang, K. Xia, and R. Harik, "A critical review of feature recognition techniques," *Computer-Aided Design and Applications*, vol. 17, no. 5, pp. 861–899, 2020.

[63] Y.-J. Tseng and B. J. Sanjay, "Recognizing multiple interpretations of interacting machining features," *Computer-Aided Design*, vol. 26, no. 9, pp. 667–688, 1994.

[64] C. Zhang, K. Chan, and Y. Chen, "A hybrid method for recognizing feature interactions," *Integrated Manufacturing Systems*, vol. 9, no. 2, pp. 120–128, 1998.

[65] J. Gao, D. Zheng, and N. Gindy, "Extraction of machining features for cad/cam integration," *The International Journal of Advanced Manufacturing Technology*, vol. 24, no. 7, pp. 573–581, 2004.

[66] K. Rahmani and B. Arezoo, "Boundary analysis and geometric completion for recognition of interacting machining features," *Computer-Aided Design*, vol. 38, no. 8, pp. 845–856, 2006.

[67] K. Rahmani and B. Arezoo, "A hybrid hint-based and graph-based framework for recognition of interacting milling features," *Computers in Industry*, vol. 58, no. 4, pp. 304–312, 2007.

[68] A. K. Verma and S. Rajotia, "A hybrid machining feature recognition system," *International Journal of Manufacturing Research*, vol. 4, no. 3, pp. 343–361, 2009.

[69] S.-J. Kim and M.-Y. Yang, "A cl surface deformation approach for constant scallop height tool path generation from triangular mesh," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 3, pp. 314–320, 2006.

[70] Y. Sun, D. Guo, H. Wang, *et al.*, "Iso-parametric tool path generation from triangular meshes for free-form surface machining," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 7, pp. 721–726, 2006.

[71] S.-G. Lee, H.-C. Kim, and M.-Y. Yang, "Mesh-based tool path generation for constant scallop-height machining," *The International Journal of Advanced Manufacturing Technology*, vol. 37, no. 1, pp. 15–22, 2008.

[72] J. Xu, Y. Sun, and S. Wang, "Tool path generation by offsetting curves on polyhedral surfaces based on mesh flattening," *The International Journal of Advanced Manufacturing Technology*, vol. 64, no. 9, pp. 1201–1212, 2013.

[73] F. Liang, C. Kang, and F. Fang, "Tool path planning on triangular mesh surfaces based on the shortest boundary path graph," *International Journal of Production Research*, pp. 1–20, 2021.

[74] J. Tarbutton, T. R. Kurfess, T. Tucker, and D. Konobrytskyi, "Gouge-free voxel-based machining for parallel processors," *The International Journal of Advanced Manufacturing Technology*, vol. 69, no. 9, pp. 1941–1953, 2013.

[75] D. Konobrytskyi, M. M. Hossain, T. M. Tucker, J. A. Tarbutton, and T. R. Kurfess, "5-axis tool path planning based on highly parallel discrete volumetric geometry representation: Part i contact point generation," *Computer-Aided Design and Applications*, vol. 15, no. 1, pp. 76–89, 2018.

[76] A. Balabokhin and J. Tarbutton, "Automatic generalized cutter selection for finishing of free-form surfaces in 3-axis cnc milling by "surface tolerance and tool

performance metrics"," *The International Journal of Advanced Manufacturing Technology*, vol. 93, no. 1, pp. 423–432, 2017.

[77] F. Shen and J. Tarbutton, "A voxel based automatic tool path planning approach using scanned data as the stock," *Procedia Manufacturing*, vol. 34, pp. 26–32, 2019.

[78] T. Chaperon and F. Goulette, "Extracting cylinders in full 3d data using a random sampling method and the gaussian image.," in *Proceedings of the Vision Modeling and Visualization Conference*, vol. 2001, pp. 35–42, 01 2001.

[79] S. Xú, N. Anwer, C. Mehdi-Souzani, R. Harik, and L. Qiao, "Step-nc based reverse engineering of in-process model of nc simulation," *The International Journal of Advanced Manufacturing Technology*, vol. 86, no. 9, pp. 3267–3288, 2016.

[80] M. Fanwu, X. Chunguang, L. Haiming, H. Juan, and X. Dingguo, "Quick algorithm of maximum inscribed circle method for roundness evaluation," in *2011 International Conference on System science, Engineering design and Manufacturing informatization*, vol. 2, pp. 348–351, IEEE, 2011.

[81] M. Zhou, G. Zheng, and Z. C. Chen, "An automated cnc programming approach to machining pocket with complex islands and boundaries by using multiple cutters in hybrid tool path patterns," *The International Journal of Advanced Manufacturing Technology*, vol. 83, no. 1-4, pp. 407–420, 2015.

[82] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[83] A. S. Sofronas, *The formation and control of drilling burrs*. PhD thesis, University of Detroit Mercy, 1975.

[84] A. Peklharing, "The exit angle failure in interrupted cutting," *CIRP Annals*, vol. 27, no. 1, p. 5, 1978.

[85] L. K. Gillespie, *Deburring and edge finishing handbook*. Society of Manufacturing Engineers, 1999.

[86] Y. H. Jeong, B. HanYoo, H. U. Lee, B.-K. Min, D.-W. Cho, and S. J. Lee, "Deburring microfeatures using micro-edm," *Journal of Materials Processing Technology*, vol. 209, no. 14, pp. 5399–5406, 2009.

[87] S. A. Niknam, B. Davoodi, J. P. Davim, and V. Songmene, "Mechanical deburring and edge-finishing processes for aluminum parts-a review," *The International Journal of Advanced Manufacturing Technology*, vol. 95, no. 1, pp. 1101–1125, 2018.

[88] G. Bone, M. Elbestawi, R. Lingarkar, and L. Liu, "Force control for robotic deburring," *Journal of Dynamic Systems, Measurement, and Control*, vol. 113, no. 3, pp. 395–400, 1991.

[89] M. Her and H. Kazerooni, "Automated robotic deburring of parts using compliance control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 113, no. 1, pp. 60–66, 1991.

[90] H.-C. Song and J.-B. Song, "Precision robotic deburring based on force control for arbitrarily shaped workpiece using cad model matching," *International Journal of Precision Engineering and Manufacturing*, vol. 14, no. 1, pp. 85–91, 2013.

[91] F. L. Princely and T. Selvaraj, "Vision assisted robotic deburring of edge burrs in cast parts," *Procedia Engineering*, vol. 97, pp. 1906–1914, 2014.

[92] J. Hu, A. M. Kabir, S. M. Hartford, S. K. Gupta, and P. R. Pagilla, "Robotic deburring and chamfering of complex geometries in high-mix/low-volume production applications," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1155–1160, IEEE, 2020.

[93] T. Sato, Y. Sato, and T. Maekawa, "Tool path generation for chamfering drill holes of a pipe with constant width," *Computer-Aided Design*, vol. 78, pp. 26–35, 2016.

[94] E. Abele, K. Schützer, S. Güth, and A. Meinhard, "Deburring of cross-drilled holes with ball-end cutters-modeling the tool path," *Production Engineering*, vol. 12, no. 1, pp. 25–33, 2018.

[95] K. Schützer, E. Abele, A. Meinhard, and J. Donnelly, "Deburring of complex-shaped drilling intersections: a numerical method for modelling the tool path," *The International Journal of Advanced Manufacturing Technology*, vol. 102, no. 1, pp. 67–79, 2019.

[96] L. K. Gillespie and P. T. Blotter, "The formation and properties of machining burrs," *Journal of Manufacturing Science and Engineering*, vol. 98, no. 1, pp. 66–74, 1976.

[97] G.-L. Chern, "Experimental observation and analysis of burr formation mechanisms in face milling of aluminum alloys," *International Journal of Machine Tools and Manufacture*, vol. 46, no. 12-13, pp. 1517–1525, 2006.

[98] H. Blum, "A transformation for extracting new descriptors of shape," *Models for Perception of Speech and Visual Forms, 1967*, pp. 362–380, 1967.