FEATURE-BASED METHODOLOGIES FOR MILLING PROCESS MODELING IN VIRTUAL MACHINING

by

Jue Wang

B.Eng., Wuhan University of Technology, 1989
M.Eng., Huazhong University of Science and Technology, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate Studies

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

September 2006

Abstract

Virtual Machining is used to simulate the machining process prior to actual machining, thereby avoiding costly test trials on the shop floor. To realize Virtual Machining, two major methodologies, Geometric Modeling and Process Modeling, are required. In geometric modeling, Cutter/Workpiece Engagements (*CWE*s) are extracted to support force prediction in process modeling. In process modeling, the physics of the machining process, such as cutting forces, torque and power, are predicted by integrating the laws of the metal cutting process with *CWE*s generated in geometric modeling. Based on these predictions, process parameters can be optimized for productivity.

Methodologies in geometric modeling for *CWE* extraction require a large number of calculations, however, the robustness and computational stability of these approaches is a significant challenge. In this thesis, methodologies are developed to address these problems in *CWE* extraction for the milling process. These methodologies achieve computational efficiency and stability by reducing the number of intersections that need to be performed and by parallelizing computational activities in the process of *CWE* extraction. A feature-based approach is presented for *CWE* extraction in 2½D end milling by introducing in-process machining features into process modeling. In hole milling, an analytical approach is presented for *CWE* extraction, and a NURBS based approach is developed for generating the swept volume for in-process workpiece modeling. A Multi-Agent System based framework is developed to improve efficiency of the *CWE* calculation, by parallelizing computational activities and utilizing free resources over a network.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgements

I would like to express my sincere appreciation to my advisor Professor Derek Yip-Hoi for his guidance in my studies at the University of British Columbia, and for his many insightful suggestions regarding the development of the ideas in this thesis. I am especially grateful for the many early morning hours he spent discussing my thesis, which helped me complete it.

I would also like to thank Professor Yusuf Altintas for extending me the opportunity to work on the development of the Virtual Machining project.

I would like to express many thanks to my colleagues in the CAD group and the Manufacturing Automation Laboratory (MAL) for their friendship and help.

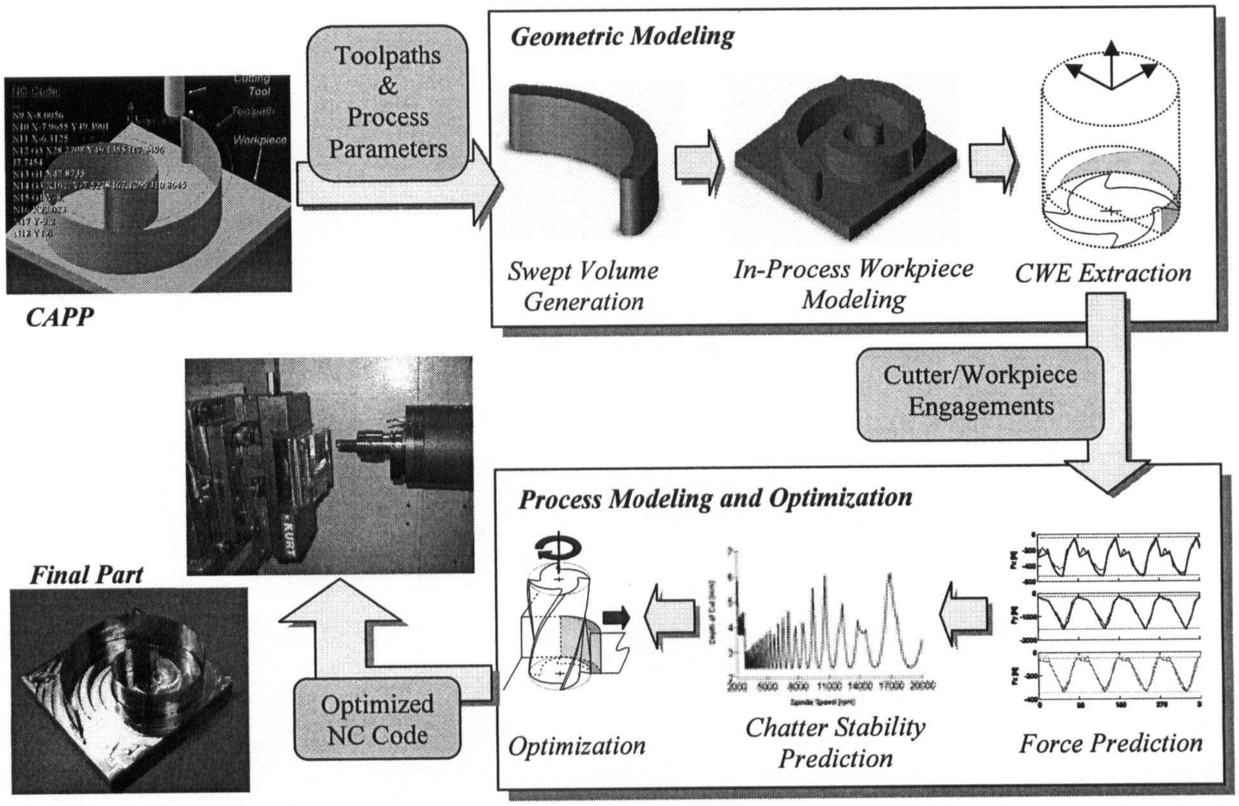
I am grateful to my parents and brother for their encouragement and support. Finally, I would like to thank my wife Qing Liu and her parents for their understanding and for taking care of my son Minghan and daughter Helen, who were born during my studies at UBC. Their support allowed me to focus on my education and research.

Chapter 1
Introduction

1.1 Virtual Machining

To produce mechanical parts correctly and optimally in the manufacturing industry, *Virtual Machining* (*VM*) is necessary to simulate the machining process prior to actual machining. There are two methodologies in *VM*: geometric modeling and process modeling. In geometric modeling, NC toolpaths generated by a CAM system are verified to prevent collisions between the cutter and the workpiece or fixture, to eliminate gouges and uncut material on the surfaces of the final part. Verification is done by examining the in-process workpiece, starting with a model of the initial workpiece. As machining progresses, the geometry of the workpiece gets updated by subtracting the volume swept by the cutter during each motion. Cutter/Workpiece Engagements (*CWE*s) are also extracted in geometric modeling to support force prediction in process modeling. In process modeling, the physics of the machining process, such as cutting forces, torque and power, are predicted by integrating the laws of the metal cutting process with *CWE*s generated in geometric modeling. Furthermore, process parameters can be optimized for productivity while ensuring that part quality is achieved.

Figure 1-1 illustrates the relationship between geometric and process modeling in *VM*. A CAD model is input into a CAPP system for process planning. A series of machining operations, which include toolpaths and process parameters such as feed rate and spindle speed, are generated. These toolpaths and process parameters are input into geometric modeling for generating the in-process workpiece for NC verification and *CWE* extraction. In process modeling, cutting forces and chatter stability are predicted using the *CWE*s provided from geometric modeling. From the results in process modeling, the process parameters are optimized for productivity while respecting process constraints such as the torque and power limits of the machine, strength of the tool, and dimensional tolerances of the part, as well as chatter vibration limits of the machine tool, fixture and workpiece structures. The optimized process parameters and toolpaths are sent to the CNC machine to efficiently produce the part.



Toolpaths & Process Parameters
Geometric Modeling
Swept Volume Generation
In-Process Workpiece Modeling
CWE Extraction
CAPP
Cutter/Workpiece Engagements
Process Modeling and Optimization
Final Part
Optimized NC Code
Optimization
Chatter Stability Prediction
Force Prediction

Figure 1-1: Virtual Machining

1.2 Process Modeling and Optimization

In process modeling, the cutting forces are predicted based on the *CWE*s extracted using geometric modeling techniques. In order to properly define the input format from *CWE* for force prediction, a force model must be identified. In this thesis, the force prediction model used is that proposed by Altintas and Spence [4]. This model solves the Cartesian force components by analytically integrating the differential cutting forces along the in-cut portion of each flute. A single engagement zone at a fixed axial depth of cut is defined by flattening the engagement region on the tool envelop cylinder to determine the integration. As shown in Figure 1-2 (a), the *CWE* from geometric modeling required by this force model in process modeling should be represented as an engagement zone bounded by the entry (ϕ_{st}) and exit angles (ϕ_{ex}) of the cutter, as well as by the axial location (d_{min}, d_{max}) at a given feed step. For general engagement conditions, the *CWE* zone can be discretized into a set of single engagement zones to satisfy the requirements of the force prediction model. As shown in Figure 1-2 (b), a general *CWE* is parametrized by applying a decomposition procedure.

Therefore, the general *CWE* zone is decomposed into a set of standard formats required by the force prediction model.

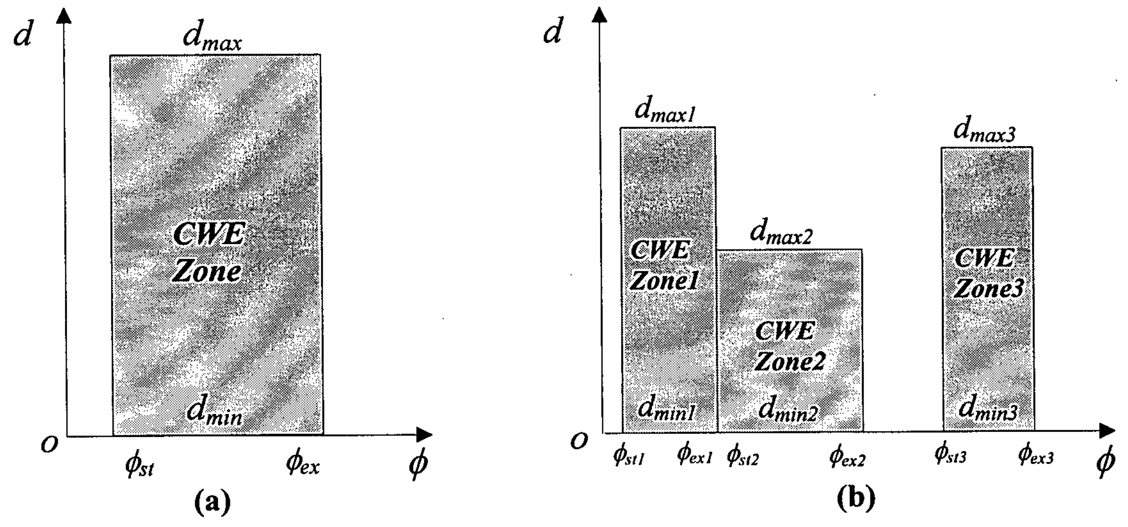


Figure 1-2: *CWE* Formats for Force Prediction Model

Based on the cutting forces, the chatter stability in milling operations can be predicted by an analytical chatter prediction model. The process parameters, such as depth of cut and spindle speed, can be optimized to achieve maximum productivity according to the predicted chatter stability constraints of the milling operation.

1.3 Geometric Modeling for Virtual Machining

*CWE*s are extracted in geometric modeling based on the input format required by process modeling. Figure 1-3 illustrates the geometric modeling process for *VM*. The inputs include the initial workpiece, CLData generated by a CAM system for machining the final part from the workpiece, and the cutting tool descriptions. Based on the latter two pieces of information, swept volumes are generated for the tool moving along each toolpath. By subtracting each swept volume sequentially, a series of in-process workpieces are generated. These in-process workpieces play two important roles in geometric modeling: one is to verify NC toolpaths, and the other is in *CWE* extraction. *CWE*s can be extracted by identifying the engagement conditions at feed steps along each toolpath. This procedure involves intersecting a representation of the cutting tool with the in-process workpiece then manipulating the intersection graph to extract entry and exit angles as a function of depth of cut.

Three steps are therefore required to extract *CWE*s in geometric modeling: (1) swept volume generation; (2) in-process workpiece modeling; and (3) *CWE* extraction. Each step is introduced in the next sections, following an introduction to the different 3D solid modeling representations that can be used in generating *CWE*s.

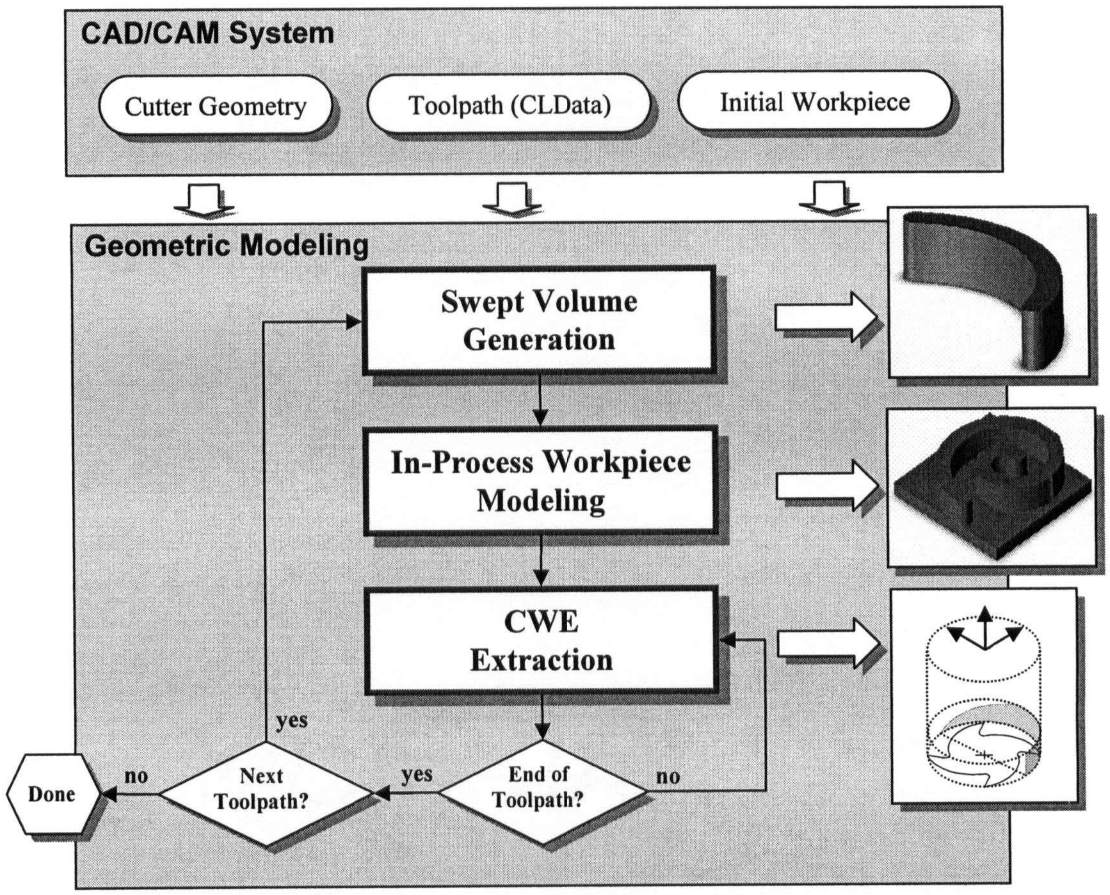


Figure 1-3: Steps for *CWE* Extraction in Geometric Modeling

<u>Approaches to *CWE* Extraction</u>

Based on 3D solid modeling representations of the workpiece, reported research on geometric modeling identifies four major approaches: B-rep, Z-buffer, normal vector and polyhedral based approaches. The B-rep based approach represents the workpiece using a Boundary Representation (B-rep) model that describes the topological relationship among the different level geometric entities of the boundary of the solid model, such as faces, coedges,

edges and vertices. A B-rep structure is the 3D solid representation model used in solid modelers. The *CWE* can be extracted by using numerical Surface/Surface Intersections (SSI) between the workpiece and the cutting tool geometry in solid modelers. The Z-buffer based approach characterizes the workpiece using a z-directional vector model, which consists of a set of z-directional vectors (ZDV) emanating from a grid on a workpiece surface. The *CWE* can be determined by finding intersections between the tool swept volume and these vectors. This procedure involves calculating intersections between surfaces and lines. The normal vector based approach is derived originally from the technique for accurate color shading of sculptured surfaces. In this approach, the workpiece is discretized into arrays containing surface coordinates and the corresponding normal vectors. Similarly, the *CWE* is extracted by calculating intersections between implicit solid models representing the motions of the cutting tool and surface normal vectors. The polyhedral based approach represents the workpiece with a polyhedral model. The boundary curves of the *CWE* can be extracted by performing intersections between the cutting tool geometry and the triangular planes that define the boundary of the polyhedral model. Table 1-1 illustrates a comparison of these four *CWE* extraction approaches.

Although Z-buffer, normal vector and polyhedral based approaches require a shorter computational time than does the B-rep based approach, the accuracy of these approaches depends greatly on the resolution of the workpiece. There is always a tradeoff between accuracy and computational efficiency in these approaches. The B-rep based approach can provide an accurate geometric representation for the workpiece. However, the SSI approaches in solid modelers use numerical techniques that are limited primarily by efficiency and robustness. These limitations have great influence on the stability and efficiency of the B-rep based approach.

This thesis focuses on a B-rep based approach to providing an accurate geometric representation for the *CWE* to support process modeling. Methodologies for improving the stability and efficiency of this approach are also studied. The following sections describe the three main steps in this procedure: (1) swept volume generation, (2) in-process workpiece modeling and (3) *CWE* extraction in a B-rep based approach.

	Solid modeling representations	Intersection calculations	Computational complexity	Solutions	Robustness
B-rep	boundary representation	Surface/Surface	high	exact	low
Z-buffer	z-directional vectors	Surface/Line	low	approximate	high
Normal Vector	normal vectors	Surface/Line	low	approximate	high
Polyhedral	triangular mesh	Surface/Plane	medium	approximate	high

Table 1-1: Comparison of *CWE* Extraction Approaches

Swept Volume Generation

In 2½D end milling, the swept volumes can be generated by a profile curve sweeping approach using a solid modeler. Figure 1-4 illustrates the swept volumes generated by a linear and a circular tool motion. The great challenge in swept volume generation is in 3- and 5-axis machining where the cutting tools with various geometries move along 3D spatial curves and the orientation of the cutter axis changes. Although significant research has been reported on this topic, general swept volume generation is not a component included in today's CAD systems. Some researchers have reported skinning techniques to generate swept volumes. However, only approximate solutions can be provided by these approaches. The stability and efficiency of these methodologies still need to be improved.

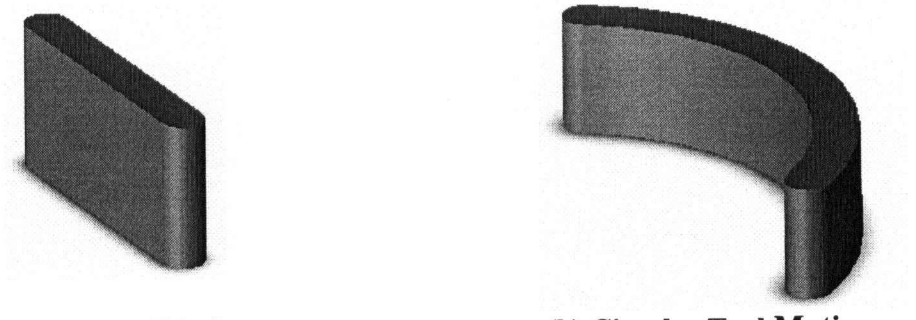

(a) Linear Tool Motion (b) Circular Tool Motion

Figure 1-4: Swept Volume from 2½D Flat End Milling

In-Process Workpiece Modeling

An accurate in-process workpiece representation is required for both NC verification and *CWE* extraction. In the B-rep based approach, the in-process workpieces can be generated by subtracting a swept volume from the workpiece. Figure 1-5 shows the generation of the in-process workpiece for the *i-th* tool motion. SV_i represents the swept volume in the *i-th* tool motion, and W_{i-1} is the in-process workpiece before the *i-th* tool motion. The in-process workpiece W_i is updated by subtracting SV_i from W_i (W_{i-1}-* SV_i). For all tool motions, a series of in-process workpiece states are captured when swept volumes are sequentially subtracted from the initial workpiece. Figure 1-6 illustrates a sequence of in-process workpieces generated by a machining operation with n tool motions. The initial workpiece (W_0) evolves into the final part (W_n) by sequentially subtracting n swept volumes from W_0. The in-process workpieces W_i and W_j represent the in-process workpiece states after the i-th and j-th tool motions, respectively.

The great challenge in in-process workpiece modeling is that the geometry of the workpiece becomes more and more complicated as machining progresses. A B-rep solid representation of the complicated in-process workpiece has a large data structure that leads to slow and unstable Boolean operations in the solid modeler. The in-process workpiece may be replaced by the removal volume, which has a less complicated geometric structure for *CWE* extraction in the B-rep based approach. This variation will be used in this research.

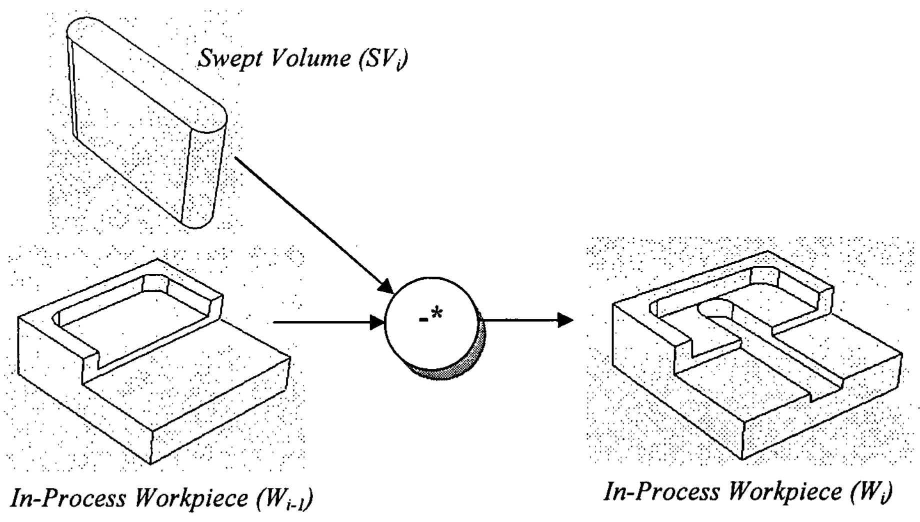

Swept Volume (SV_i)
-*
In-Process Workpiece (W_{i-1})
In-Process Workpiece (W_i)

Figure 1-5: In-Process Workpiece Generation

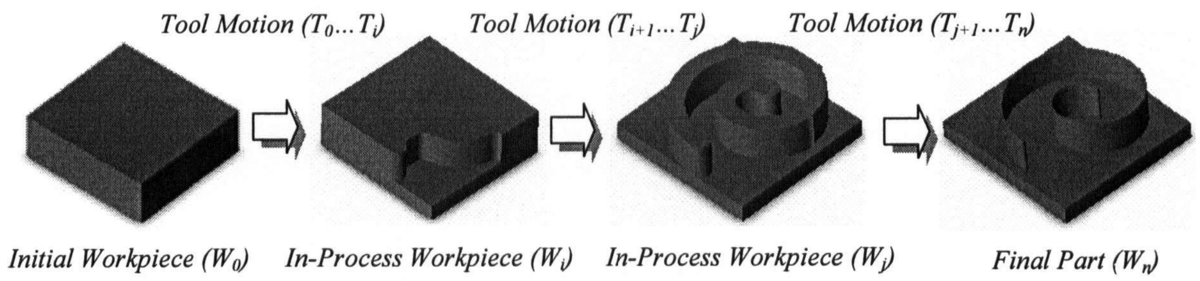

Tool Motion ($T_0 \ldots T_i$)
Tool Motion ($T_{i+1} \ldots T_j$)
Tool Motion ($T_{j+1} \ldots T_n$)
Initial Workpiece (W_0)
In-Process Workpiece (W_i)
In-Process Workpiece (W_j)
Final Part (W_n)

Figure 1-6: In-Process Workpiece States during a Machining Operation

Cutter/Workpiece Engagement Extraction

Cutter/workpiece engagements can be extracted by performing Surface/Surface Intersections (SSI) between the advancing semi-cylindrical surface of a cutter and the in-process workpiece (or removal volume) at a series of consecutive cutter locations along each tool motion. As shown in Figure 1-7, the in-process workpiece (W_{i-1}) is a workpiece state immediately before the *i*-th tool motion (T_i). SSIs between the semi-cylinder and W_{i-1} are performed at each feed step along the tool motion (T_i) to extract *CWE*s. It can be seen that a large number of SSIs are needed for machining a complicated part. The robustness and computational efficiency of this approach are therefore primary concerns.

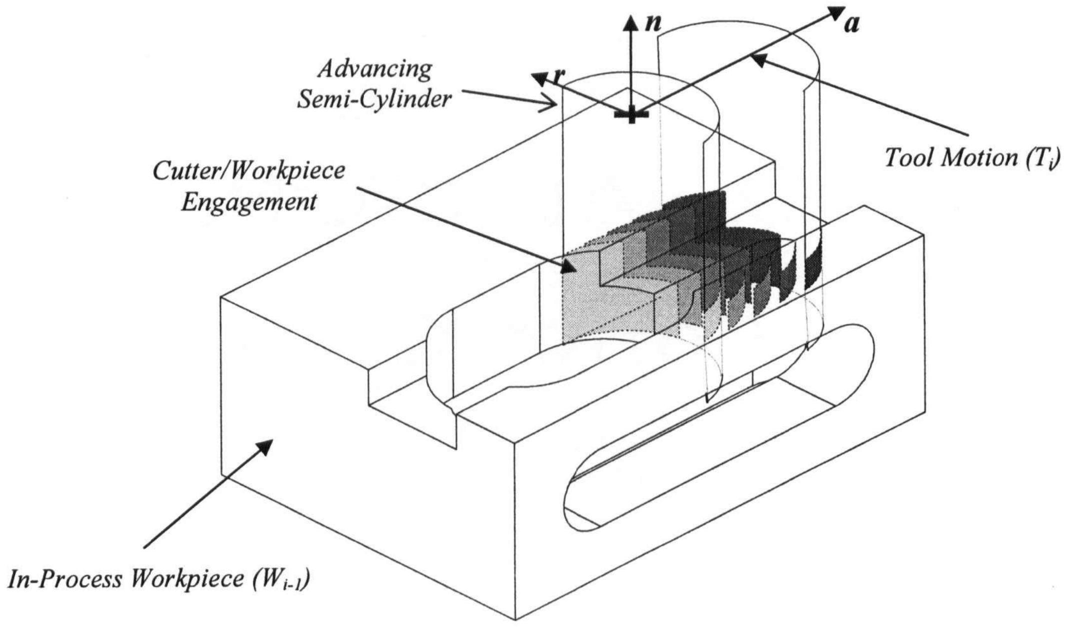


Figure 1-7: Cutter/Workpiece Engagement Extraction

From the discussion above, it can be seen that methodologies in geometric modeling for *CWE* extraction require a large number of calculations. Therefore, the robustness and computational stability of these approaches is a significant challenge. To address these problems, this thesis studies the potential of *features* to characterize regions of the removal volume for reducing the number of intersections that need to be performed. On a parallel track, this thesis also develops a new computational framework for improving the efficiency of the *CWE* calculation by parallelizing computational activities and utilizing free resources over a network.

1.4 Feature-Based Modeling and Planning

Computer Aided Process Planning (CAPP) is a system for automating process planning, and is vital for integration of CAD and CAM. CAPP utilizes features as the link between CAD and CAM systems. A *feature* is an abstract concept that refers to a region of interest on a component within a specific application domain. Features can be defined with respect to domains such as design, process planning and manufacturing. Machining features capture the manufacturing perspective of a product. A machining feature is defined as a region of interest on a workpiece generated by the removal of material with cutting tools during a sequence of machining operations. In CAPP, one strategy utilizes feature recognition to extract machining features from a CAD model. As shown in Figure 1-8, machining features such as holes, slots and pockets are recognized from the final part. After feature recognition, machining features are mapped to a set of machining operations for creating each feature using knowledge based techniques, e.g., an Expert System. As a result, each machining feature becomes associated with a set of machining operations. For instance, a hole feature might be linked to a center-drill, drill, counter-sink and ream operation sequence.

To facilitate *CWE* extraction, this thesis proposes that in-process machining features be introduced into process modeling to represent regions of interest within the volumes removed during machining operations. To illustrate, in Figure 1-9, the removal volume (RV_i) can be generated by performing a Boolean intersection between the swept volume (SV_i) and the in-process workpiece (W_{i-1}). This approach uses the removal volume to calculate *CWE* instead of using the in-process workpiece, as it typically has a less complicated geometric structure than the in-process workpiece. As shown in Figure 1-10, a removal volume can be decomposed into two classes of in-process machining features. Geometric Invariant Machining Features (*giF*) and Form Invariant Machining Features (*fiF*) are defined to characterize regions of the removal volume where the *CWE* is constant or changing in a predicable way. The advantage of defining *giF*s and *fiF*s is that *CWE*s can be analytically extracted from *giF*s and *fiF*s without applying repetitive SSI operations. Computational stability and efficiency can thereby be improved significantly.

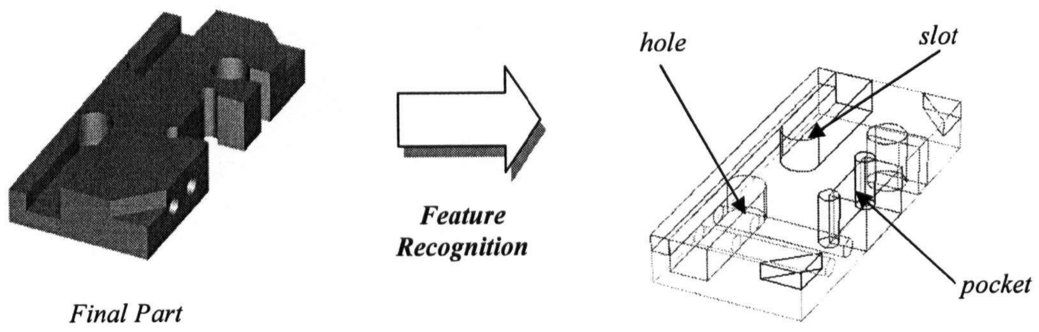


Figure 1-8: Machining Features

It can be seen from the above that the introduction of an in-process machining feature into the process modeling domain leads to a comprehensive and unambiguous description of the characterized regions during a machining operation. This description helps develop algorithms to extract these features for facilitating process modeling more efficiently.

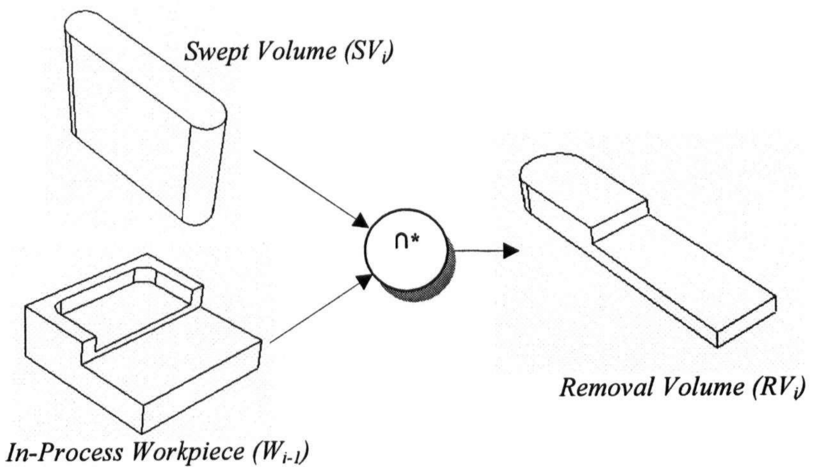


Figure 1-9: Removal Volume Generation

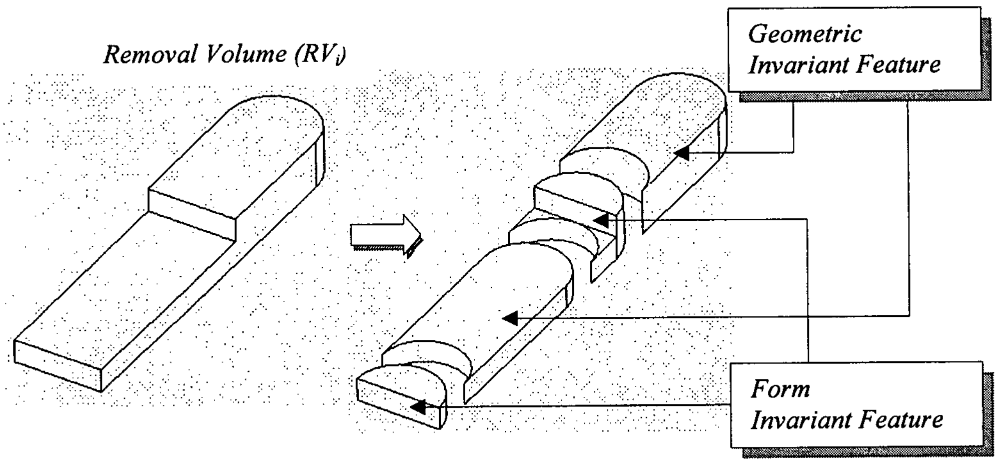


Figure 1-10: Removal Volume Decomposition

1.5 Multi-Agent Systems

Multi-Agent Systems (*MAS*) have been developed to provide methodologies for the construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' actions. An independent agent can be considered as an entity with goals, actions, and knowledge, situated in an environment. *MAS* can speed up the operation of a system by providing a framework for parallel computing. In particular, the parallelism within a *MAS* can help mitigate limitations imposed by time-critial requirements. To achieve parallel computing, a *MAS* breaks tasks into several independent subtasks that can be handled by separate agents.

To achieve computational efficiency in *CWE* extraction, a *MAS* based framework is proposed to facilitate parallel processing. As discussed previously, the B-rep based *CWE* extraction approach decomposes the *CWE* extraction procedure into the following three computational activities:

- **Swept Volume Generation**

  This activity can achieve full parallelism, since swept volume generation depends only on the toolpath and the associated cutting tool.

- **Removal Volume Generation**

  This activity can only achieve partial parallelism, because the creation of removal volumes requires an accurate in-process model of the workpiece after the previous tool motion. It may be possible to group toolpaths (e.g., on different regions of the part) and to partition the in-process model of the workpiece for parallelism.

- ***CWE* Extraction**

  This activity can achieve full parallelism since *CWE* extraction depends only on the removal volume (generated from the previous step), toolpath and cutting tool geometry.

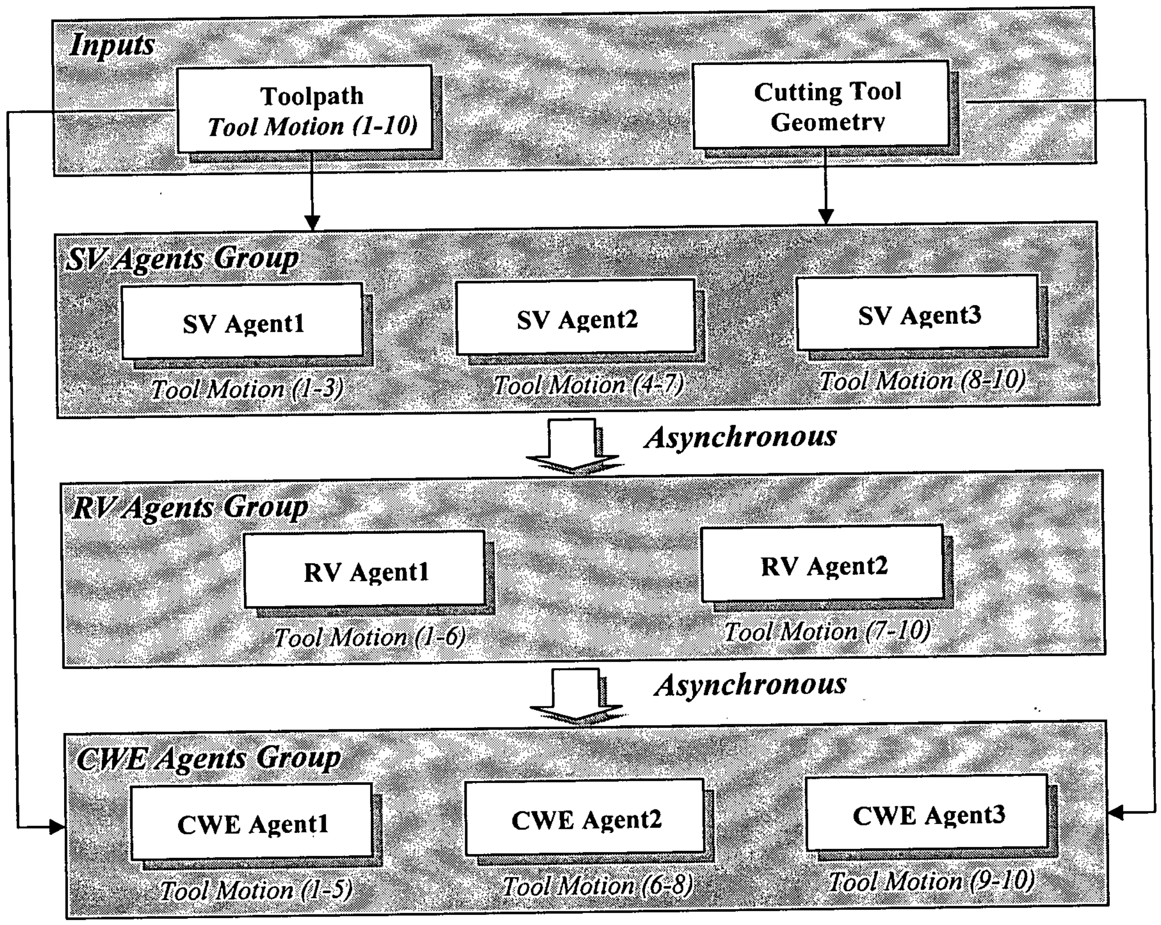


Figure 1-11: Proposed *MAS* Based Framework for *CWE* Extraction

In the proposed *MAS* based framework, three computational agents, the Cutter/Workpiece Engagement Agent (CWE Agent), Removal Volume Agent (RV Agent) and Swept Volume Agent (SV Agent), are designed with respect to the corresponding computational activities.

Figure 1-11 illustrates how these agents achieve parallelism within the framework. It can be seen that computational tasks are shared among agents within each group to achieve parallelism by using the strategy of tasks allocation. In addition, the different groups can also achieve parallelism by using an asynchronous results passing strategy.

1.6 Objectives

The objective of this thesis is to develop feature based methodologies that facilitate the extraction of *CWE* geometry in support of milling process modeling for Virtual Machining. This objective is achieved through methods that reduce the number of intersections that need to be performed and parallelize computational activities in the process of *CWE* extraction. Toward this objective, the following contributions are presented in this thesis:

- A novel in-process machining feature to address the computational complexity and robustness issues in *CWE* calculations. In-process machining features provide an unambiguous description of removal volume regions during a machining operation. They help in the development of robust and efficient algorithms that extract these features.
- A feature-based approach to *CWE* calculation in 2½D end milling. Geometric invariant and form invariant machining features are modeled to help analytically represent engagement conditions. Volume decomposition and composition algorithms are described that extract these two types of machining features from the removal volumes generated from each toolpath. *CWE*s can be analytically extracted from geometric invariant and form invariant machining features without applying repetitive SSI operations. The robustness and computational efficiency of the *CWE* extraction algorithm are significantly improved.
- An analytical approach to *CWE* extraction for hole milling. A Milled Hole Machining Feature (*mhF*) and a milling operation that uses a helical toolpath and a flat end-mill are introduced into the domain of process modeling. An analytical approach is presented to parametrize the *CWE*s based on parameters of the *mhF*. The parametric representation of *CWE*s provides a closed-form formula to the complicated *CWE*

extraction problem in hole milling, and brings significant advantages with respect to accuracy and computational efficiency.

- An advanced NURBS based approach for swept volume generation for hole features generated by helical milling using flat, conical and ball nose end-mills. This approach generates an exact and efficient NURBS representation for the envelop surfaces of the swept volume given the helical toolpath and tool geometry. Boundary surfaces are trimmed at intersections and internal patches are discarded. The boundary patches are then stitched together to form a closed volume. Reported NURBS based approaches can provide approximate solutions only, while this approach can produce an exact solid model of the swept volume. The generated swept volumes are then used for in-process workpiece modeling for geometric verification purposes.
- A multi-agent system for B-rep based *CWE* extraction that allows distributed processing of the modeling steps over the Internet. The *CWE* calculation utilizes distributed agents for performing swept volume and removal volume construction in addition to the extraction of the *CWE* geometry itself. These distributed agents provide the capability to perform many of the calculations in parallel. The proposed methodology thus makes the best use of available, distributed computing resources, leading to greatly improved efficiency in the *CWE* calculations. If agents are available to perform these calculations using other non-B-rep approaches the proposed framework facilitates their integration.

1.7 Organization of Thesis

A review of relevant literature is presented in Chapter 2. Chapter 3 introduces a classification of machining features for process modeling. In Chapter 4, a feature-based approach to *CWE* extraction in 2½D end milling is presented. An analytical approach to *CWE* extraction in hole milling is developed in Chapter 5. In Chapter 6, a NURBS based approach to swept volume generation and in-process workpiece modeling in hole milling is introduced. In Chapter 7, a multi-agent system for distributed, Internet-enabled *CWE* extraction is presented. Finally, conclusions and future research directions are given in Chapter 8.

Chapter 2
Literature Review

2.1 Introduction

As mentioned in Chapter 1, process modeling in Virtual Machining requires the calculation of *CWE* from geometric modeling to predict the cutting forces. This is a challenging procedure when the geometry of the workpiece becomes more and more complicated as the machining simulation progresses. An extensive amount of research has focused on geometric and physical simulations of machining processes. These research achievements have contributed to the important techniques that are integrated in a virtual machining environment. Some of important research contributions in this field will be reviewed in this chapter.

Specifically, research into force prediction models, swept volume generation and *CWE* extraction techniques is reviewed. Since machining features and multi-agent systems are introduced for facilitating *CWE* extraction in this thesis, an overview of previous research work on feature recognition and multi-agent systems is provided.

2.2 Force Prediction Models in Virtual Machining

The mechanistic model of cutting forces, first proposed by Tlusty and MacNeil [59], has received much attention over the past two decades. Since this model can be used to estimate instantaneous cutting forces, it is widely used in the physical simulation of machining processes to predict cutting tool deflections and optimize machining parameters.

Significant research has addressed the mechanistic model of cutting forces. DeVor *et al.* [16] and Kline *et al.* [35] refined and extended Tlusty's model by integrating a discrete model of the cutting tool. Fussell [19] extended this discrete model to handle workpiece geometry changes. One of the most well-known approaches to using the mechanistic model for milling processes is that proposed by Altintas and Spence [4].

In Altintas' model, the elemental forces of each flute: feed (F_x), normal (F_y), and axial (F_z) forces, can be represented by integrating analytically the differential cutting forces along the in-cut portion of each flute as:

$$F_q(\phi_j(z)) = \int_{z_{j,1}}^{z_{j,2}} dF_q(\phi_j(z))dz \qquad q = x, y, z$$

where $dF_q(\phi_j(z))$ are the differential cutting forces of the flute *j*. $z_{j,1}(\phi_j(z))$ and $z_{j,2}(\phi_j(z))$ are the lower and upper axial engagement limits of the in-cut portion of the flute *j*. To determine the axial integration limits $z_{j,1}$ and $z_{j,2}$ for each flute, a single engagement zone at a fixed axial depth of cut is defined by flattening the engagement region on the tool envelop cylinder. As shown in Figure 2-1, the axial integration limits for each flute are identified based on how the helix representing the cutter flute intersects the boundary of the engagement zone. There are five cases with regard to intersection conditions. Each case provides a set of integration limits for the flute. From Figure 2-1, it can be seen that the *CWE* from geometric modeling required by this force model in process modeling should be represented as an engagement zone bounded by the entry (ϕ_{st}) and exit angles (ϕ_{ex}) of the cutter, and the axial location (d_{min}, d_{max}) at a given feed step. Mathematically, the *CWE* at the cutter location $\boldsymbol{P}_{CL}$ can be represented as $CWE = f(\phi_{st}, \phi_{ex}, d_{\min}, d_{\max})$. For general engagement conditions, the *CWE* zone can be discretized into a set of standard formats. Yip-Hoi and Huang [70] parametrized general *CWE*s by applying a decomposition procedure to meet the requirements of this force prediction model.

In this thesis, Altintas' model is used for force prediction in process modeling. Therefore, the *CWE*s extracted from geometric modeling should follow the standard input format (*CWE* zone illustrated in Figure 2-1) in order to satisfy the requirements of the force prediction model. As discussed in Chapter 1, a general shape of C*WE*s has to be decomposed into a set of standard *CWE* zones to be accepted by the force model.

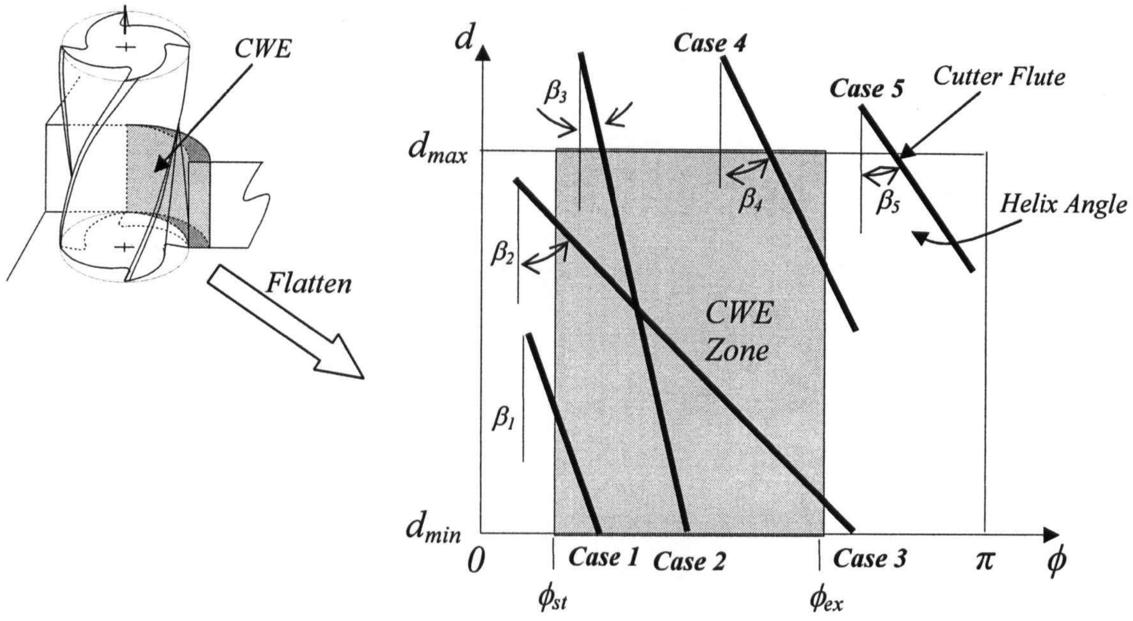


Figure 2-1: *CWE* Zone in the Force Prediction Model

2.3 Cutter/Workpiece Engagement Extraction

The purpose of *CWE* extraction is to identify engagement conditions between the cutter and the in-process workpiece during machining operations for predicting the cutting forces. Research work on *CWE* extraction can be classified into the following areas:

- B-rep based approaches
- Discrete vector approaches
- Polyhedral based approaches

These approaches to *CWE* extraction are discussed in the following subsections.

2.3.1 B-rep Based Approaches

Research into B-rep based methodologies to support modeling machining processes has been reported [4, 51-54, 57, 58, 64]. In these approaches, an in-process workpiece is represented by using a B-rep model. In-process workpiece updating and *CWE* extraction are performed using geometric and topologic algorithms within the solid modeler system.

Spence *et al.* [52] presented an approach to calculating *CWE* extraction for 2½D end milling. Figure 2-2 (a) shows a 2½D end milling process where a flat end mill engages the in-process workpiece (W_i) at a constant depth of cut *d*. As shown in Figure 2-2 (b), the *CWE*s can be determined by performing the intersection between an advancing semi-circle C_i moving in the tangential direction of the toolpath and boundary edges of the in-process workpiece on an engagement plane PL_{Z_Top}. As shown in Figure 2-2 (c), the *CWE* is identified as entry (ϕ_{st}) and exit (ϕ_{ex}) angles in the local coordinates of the cutter by the intersection of 2D geometry on the plane PL_{Z_Top}. Due to an assumed constant depth of cut, this approach is limited to 2½D end milling where the axial depth of the cutter engagement remains constant during machining.

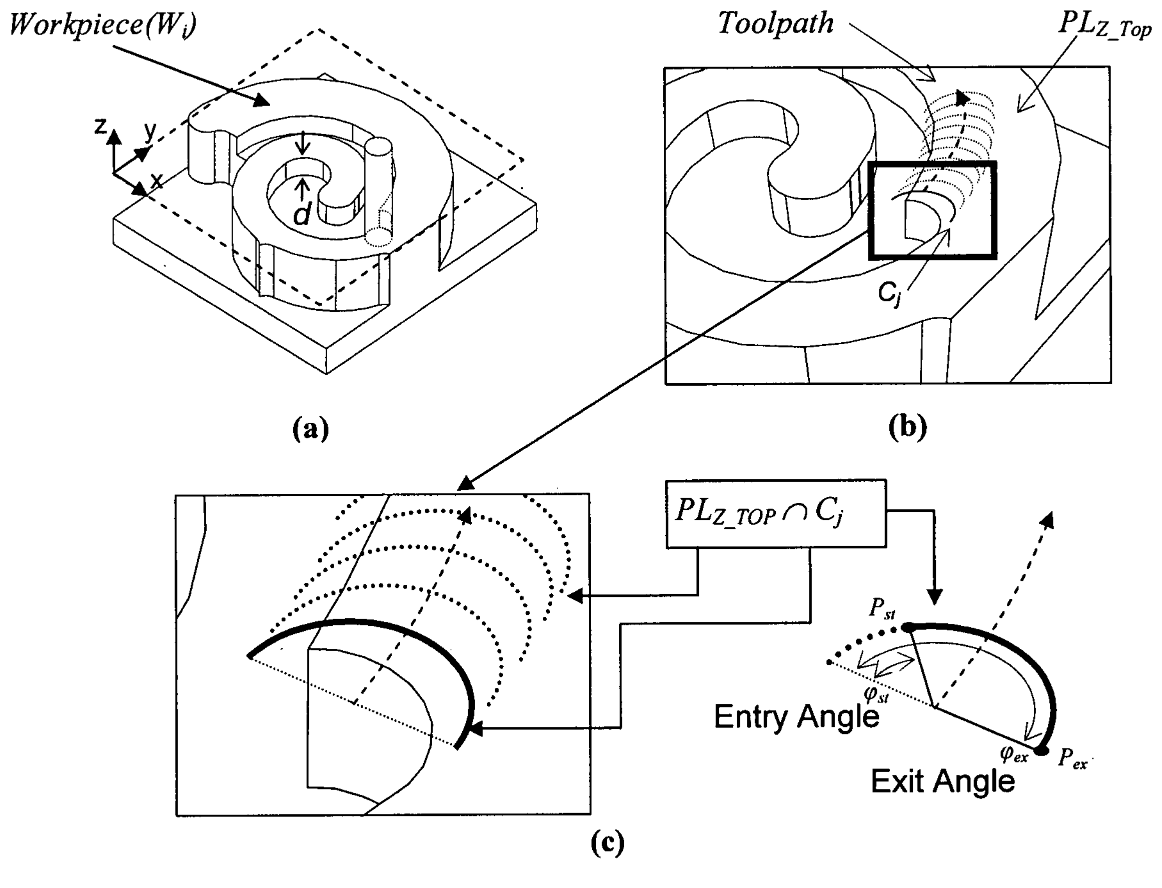


Figure 2-2: *CWE* Calculations for 2½D End Milling: the Spence Approach (derived from [70])

Yip-Hoi *et al.* [70] extended Spence's approach by applying the intersection operation between a semi-cylindrical surface and surfaces of the in-process workpiece. As shown in Figure 2-3, this approach replaces the advancing semi-circle used in Spence's approach with the advancing semi-cylinder. Surface/surface intersections are performed by the solid modeler to imprint the engagement regions on the surface of the cutter. These engagement regions are then decomposed into a set of simpler regions ($CWE = \cup R_i,\ i = 1, 2, \ldots n_R$) to satisfy the *CWE* format required by the force prediction model. This approach extends the *CWE* calculation to encompass a wider range of geometries, including conditions when the initial workpiece is non-prismatic and when the tooling and setup changes.

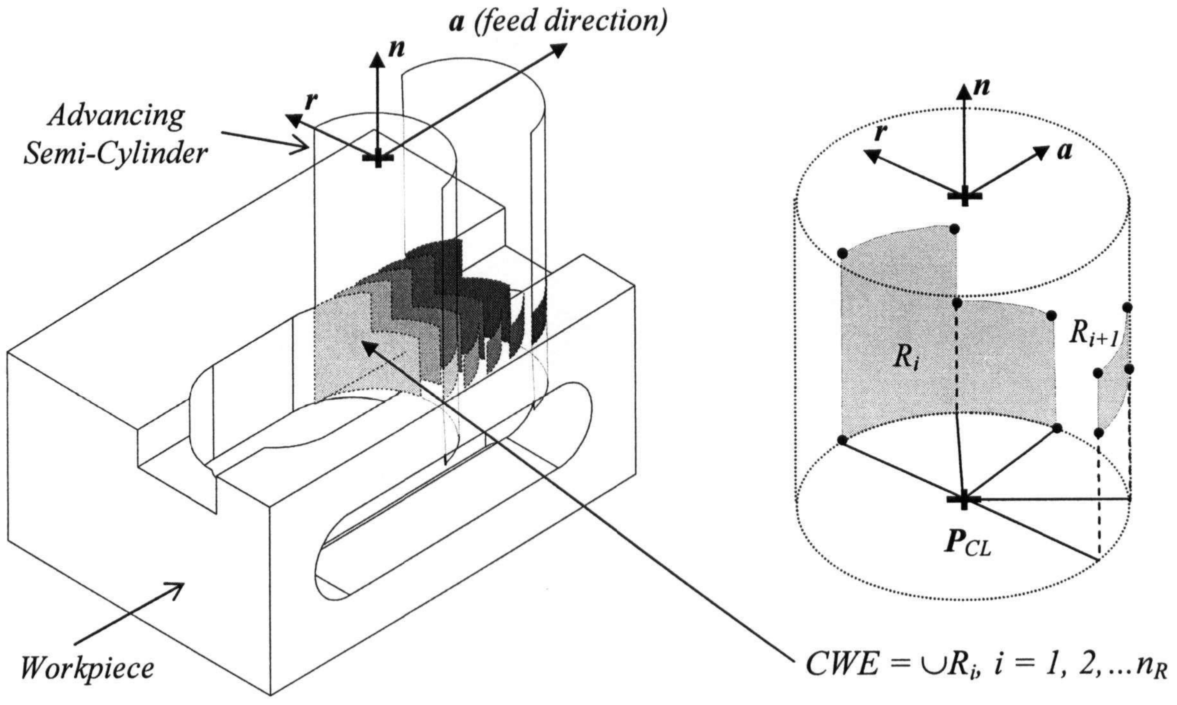


Figure 2-3: *CWE* Calculations for 2½D End Milling: Yip-Hoi Approach

2.3.2 Discrete Vector Approaches

Research into NC verification has resulted in methodologies for determining the correctness of NC toolpaths generated by CAM applications. The primary concern here is to identify gouges or uncut material on a part's surfaces prior to actual machining, to avoid costly rework. However, these approaches can also be adopted for *CWE* calculations, since calculations of the intersection of the cutting tool and the workpiece are involved in these approaches. A number of approaches are reported in this area, and can be classified into Solid Modeling [52, 53, 70], Z-Map Model [61, 65] and Discrete Vector Approaches [29-32, 41, 42]. In this section, the dominant approaches in NC verification, Discrete Vector Approaches which include the Normal Vector and Z-buffer approaches will be discussed.

The Normal Vector approach has been developed by Oliver [41]. As shown in Figure 2-4, the approach discretizes the workpiece into arrays containing surface coordinates and the corresponding normal vectors. The discretization of the workpiece is based on a computer graphics image of the workpiece surface. Each pixel on the image is projected onto the workpiece surface, and this set of points becomes the approximation to the surface. The normal vector is then determined at each point on the surface. The *CWE* can be determined by calculating intersections between implicit solid models that represent the motions of the cutting tool and these normal vectors. The problem in this approach is that the pixel-based discretization generates a large number of vectors on the workpiece surface. The computational complexity is high, since every vector has to be separately intersected with every tool movement envelope. Localization methods have been developed by Chang [8] and Huang [26] to eliminate many of the vectors from consideration.

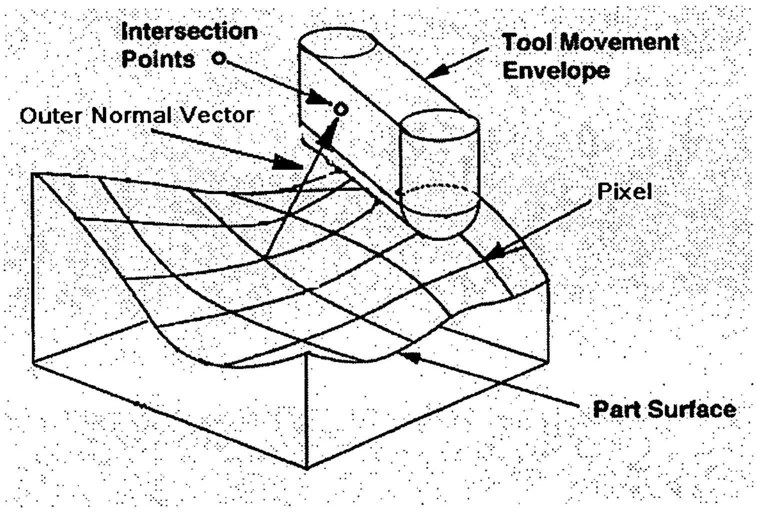


Figure 2-4: Normal Vector Approach

The Z-buffer based approach was developed by Jerard [29, 30, 31, 32]. This approach provides a more flexible strategy to discretize the workpiece than the pixel-based discretization in the Normal Vector based approach. In this approach, the spacing between points can be determined based on the desired accuracy, tool size and shape and local surface curvature. Fussell and Jerard in their recent research [21] presented an extended Z-buffer approach for *CWE* calculation for 5-axis sculptured surface machining. This approach represents the workpiece with an extended Z-buffer model and the cutting tool with a discrete axial slice model. The extended Z-buffer model represents the workpiece with a set of evenly distributed discrete vectors, called z-directional vectors (ZDVs), with the length of each vector representing the depth of the workpiece. As shown in Figure 2-5, the *CWE*s can be determined by finding intersections between the tool swept volume and these vectors. Since the tool swept volume can be modeled as a set of geometric primitives such as planes and quadric surfaces, calculation of the intersections between the cutting tool and the workpiece can be simplified to a set of line/surface intersection calculations instead of expensive surface/surface intersection calculations performed in the solid modeler based approaches.

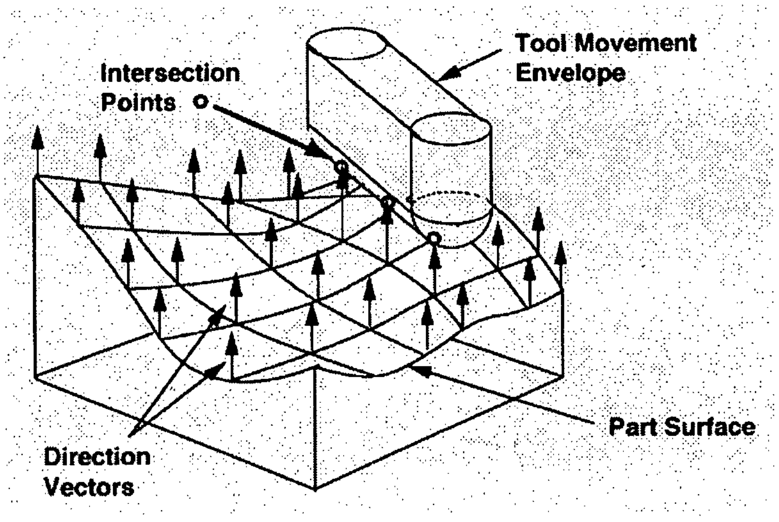


Figure 2-5: Extended Z-buffer Approach

2.3.3 Polyhedral Based Approaches

Some research has been reported on polyhedral based methodologies. Peng e*t al.* [43] presented a polyhedral model based *CWE* calculation method. To improve computational efficiency, in this approach, *CWE* extraction is based on the calculation of the intersections between the cutter and the polyhedral model of removal volumes instead of on in-process workpieces. Efficiency is improved, since the removal volumes have simpler geometry. To further reduce the number of intersection calculations, an R*-tree based localization technique is adopted to localize those facets that have potential intersections with the cutter (see Figure 2-6). The intersection points between the facet edges and the cutter are calculated by performing line/surface intersections. A validity checking method based on the kinematics model of the cutter is utilized to remove invalid intersection points. After removing these points, a Mark Circle method is used to generate intersection segments between facets and the cutter. Finally, an undirected graph based method is used to connect all the intersection segments and the boundary curves of the cutter surfaces to generate the resultant intersection lines. This approach aims to improve computational efficiency and robustness for *CWE* extraction by using a polyhedral model for the removal volumes and through the localization technique. It guarantees that the calculations of *CWE* boundaries can be performed analytically, thus avoiding numerical calculations and the accompanying problems of stability and robustness.

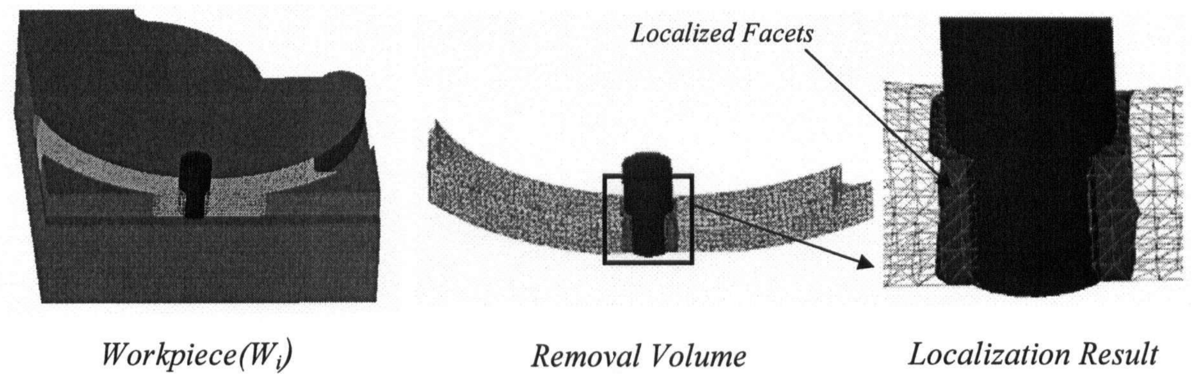


Figure 2-6: *CWE* Calculations by Polyhedral Based Approach

2.3.4 Discussion

From the above discussion, it can be seen that B-rep based approaches can provide accurate mathematical representations for intersections. However, the greatest challenge of these approaches is computational complexity and robustness, since numerical surface/surface intersections are involved. Discrete vector approaches and polyhedral based approaches simplify these calculations into line/surface intersection calculations by using discrete representations for the workpiece or removal volume. Although the simplicity of the model results in a shorter computation time than in the B-rep based approach which performs surface/surface intersections, the accuracy of these approaches greatly depends on the resolution of the workpiece or removal volume model. As such, there is always a tradeoff between accuracy and computational efficiency in these approaches. These problems with the *CWE* extraction approaches motivate the present research to utilize feature-based techniques for improving the computational efficiency and robustness of B-rep based approaches.

2.4 Feature Recognition

Computer-aided process planning (CAPP) plays an important role in CAD/CAM integration. A CAD model is interpreted as a set of machining features in a CAPP system. These machining features are extracted from the CAD model by utilizing feature recognition techniques. Based on these machining features, a sequenced set of machining operations can be generated to produce the final part. A wide variety of techniques have been developed for feature recognition. A comprehensive survey and review of feature recognition methodologies is given by Han *et al.* [23]. These techniques can be classified as follows:

- Graph Based Recognition
- Volume Decomposition
- Convex Hull Decomposition
- Hint Based Recognition

The approaches of these three areas are briefly described in the following subsections.

2.4.1 Graph Based Recognition

The Graph Based approach was first formalized by Joshi [28]. As illustrated in Figure 2-7, it consists of the following steps:

Step 1. Translation

In this step, the boundary representation of the part is translated into a part graph where nodes represent faces and arcs represent edges. The part graph has been represented as a *B-rep Graph* by Sakurai and Gossard [46], *Attributed Adjacency Graphs (AAG)* by Joshi [28], *Face Edge Graphs (FEG)* by De Floriani [15], *Face Adjacency Hypergraphs (FAH)* by Falcidieno and Giannini [18], *Vertex Edge Graphs (V-E)* by Chuang and Henderson [9] and *Aspect Face Edge Graphs (AFEG)* by Corney [12]. Additional information such as edge-convexity is incorporated into the graph.

Step 2. Decomposition

The graph is decomposed into sub-graphs by removing all faces that cannot form part of a feature. For example, for a depression a face whose incident edges are all convex will be removed from the part graph since this face cannot form part of this type of feature.

Step 3. Matching

The resulting sub-graphs are analyzed to determine their feature types by matching some predefined feature patterns such as holes and slots.

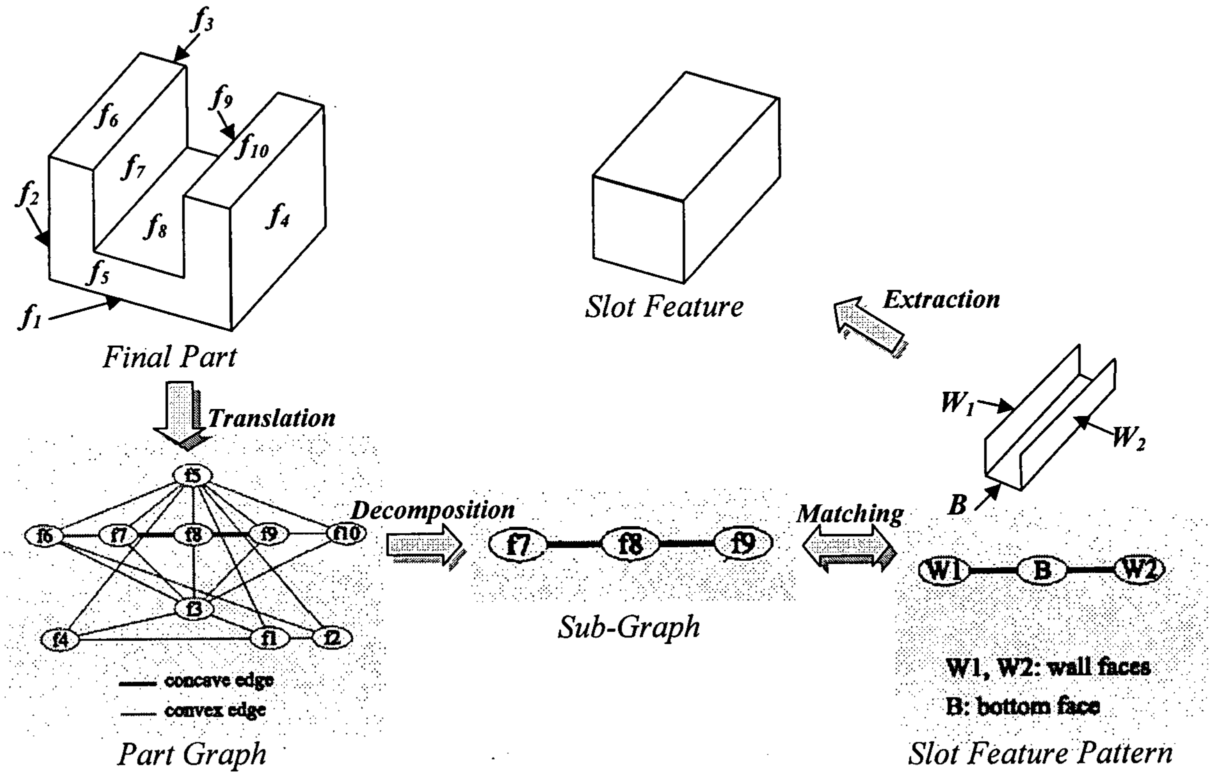


Figure 2-7: Graph Based Approach to Feature Recognition

The Graph Based approach has been quite successful in recognizing features without intersections, but encounters many difficulties in recognizing intersecting features. Marefat and Kashyap [40] observed that the arcs between face nodes in the part graph may be missing when features intersect. They proposed a novel solution to restore highly ranked missing arcs into the part graph. Trika and Kashyap [60] devised algorithms that can compute the exact set of missing arcs. However, their algorithms place strong restrictions on input parts, i.e., they must be polyhedral without inclined faces.

2.4.2 Volume Decomposition

Volume Decomposition approaches decompose the CAD model into a set of intermediate volumes which are then manipulated to produce features. One of the most well-known approaches using volume decomposition is that adopted by Sakurai and Chin [45]. This approach is described in the following steps and illustrated in Figure 2-8.

Step 1. Delta Volume Generation

The delta volume is the volume that must be removed from a stock piece (initial workpiece) to generate the final part. In this step, the delta volume is generated by subtracting the final part from the stock using a solid modeler.

Step 2. Decomposition

In this step, the delta volume of the part is decomposed into minimal cells by extending and intersecting all the surfaces or halfspaces of the delta volume.

Step 3. Combination

The set of minimal cells are combined to generate a volume that can be removed by a machining operation. Therefore, the volume is classified as a machining feature.

Cell decomposition may generate a large number of cells. The difficulty is how to combine such cells and produce suitable features. Sakurai and Chin [45] proposed generating all possible features. Coles *et al.* [11] proposed composing the cells into convex volumes only. Even though some heuristics are used to prune unpromising compositions in their approaches, these composition algorithms still have difficulty avoiding combinatorial explosion.

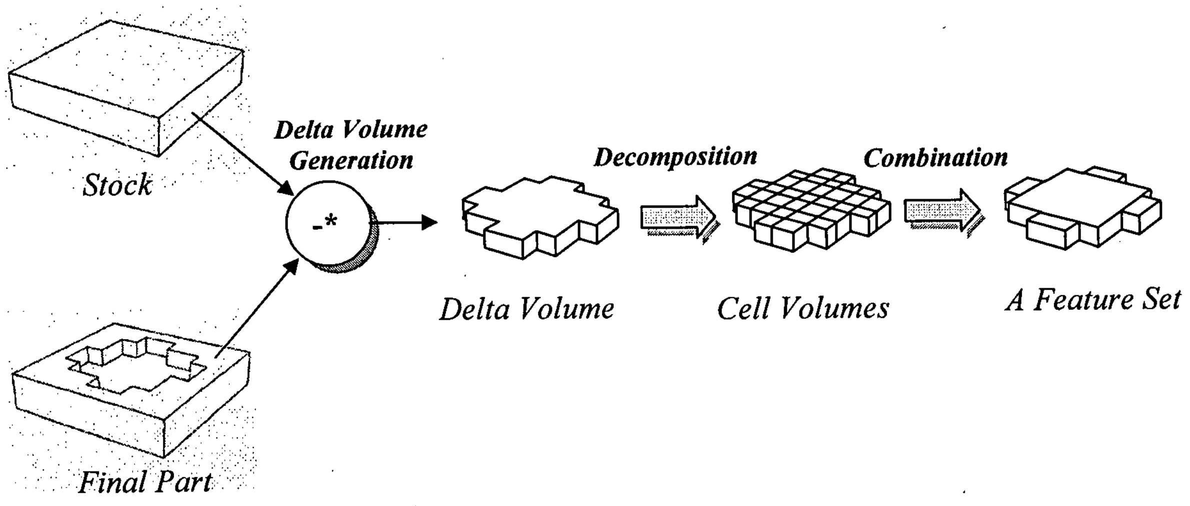


Figure 2-8: Volume Decomposition Approach to Feature Recognition

2.4.3 Convex Hull Decomposition

The Convex Hull Decomposition approach was initiated by Kyprianou in 1980 [37], and formalized by Woo [69]. In this approach, a final part is decomposed by using an Alternating Sum of Volumes (ASV) decomposition. The ASV decomposition recursively performs Boolean difference operations (CHD*(P)) to subtract the part (P) from its convex hull (CH(P)) until the null set is reached. Therefore, a sequenced set of convex volumes $\{P^i\}$ are generated. The final part can be represented by summing these convex volumes with alternating signs. As illustrated in Figure 2-9, a final part P is decomposed into a set of convex volumes $\{P^1, P^2, P^3\}$. The final part P is represented by an alternating sum of convex volumes as $P = P^1$ -* P^2 +* P^3, where -* denotes a difference operation, and +* denotes a union operation.

The main problem with this approach is that it does not guarantee success and may result in an undesirable machining feature model. Another problem with this approach is that it is inherently based on a polyhedral representation of the part. Parts with curved surfaces must first be mapped to the polyhedral models. Kim [33] has extended the ASV approach by introducing a modified ASV decomposition, Alternating Sum of Volumes with Partitioning (ASVP) decomposition, to provide a remedy for non-convergence in the ASV approach.

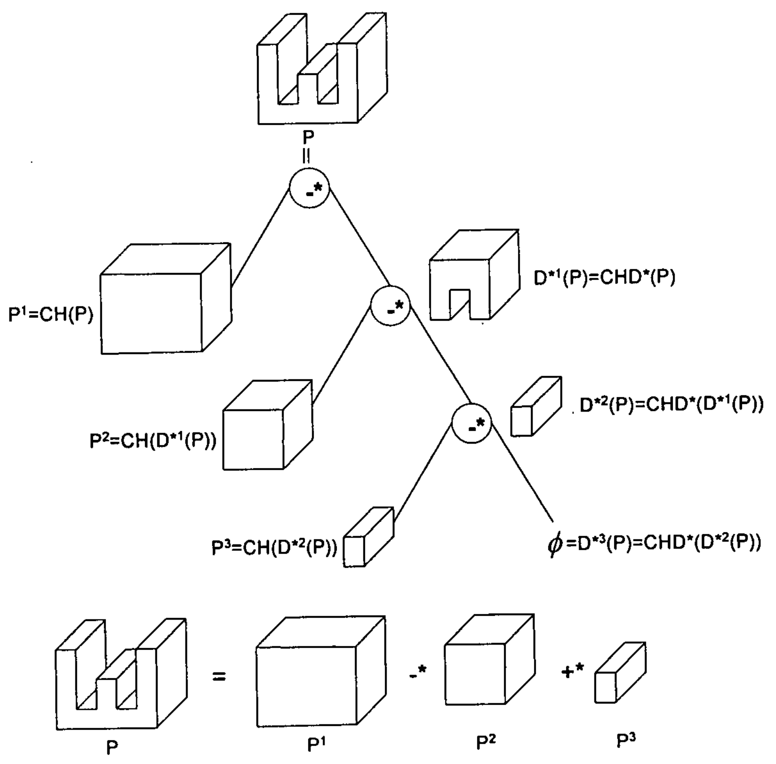


Figure 2-9: Alternating Sum of Volumes Decomposition

2.4.4 Hint Based Recognition

Several Hint Based approaches [22, 62, 63] have been developed to address the difficulty of intersecting feature recognition. A Hint Based approach is a strategy that is based on feature hints rather than on rigid feature definitions such as the part graph used in graph based approaches. The IF^2 (Integrated Incremental Feature Finder) methodology developed by Vandenbrande and Requicha [62] follows these steps:

Step 1. Generate

In the *generate* step, a proposed removal volume that is a portion of the delta volume is generated by finding a feature trace such as a floor for a slot feature.

Step 2. Test

The *test* step checks the boundary of the proposed volume to identify stock faces that originate from the stock and part faces that originate from the part. If the boundary of the proposed removal volume contains any part faces that have the potential of being removed by the cutter when machining this removal volume, the proposed removal volume is identified as non-machinable as a whole.

Step 3. Repair

The *repair* step will be applied to remove a subset of the proposed removal volume such that the machining operation does not intrude into the part faces. After the *repair* step, the removal volume left is taken as a machining feature.

The IF^2 approach can recognize intersecting holes, slots and pockets. However, the problem with the IF^2 approach is that a significant number of traces may not lead to valid features; on the other hand, several traces may lead to an identical volumetric feature. These situations lead to computational inefficiency because of the need to perform expensive geometric reasoning on every trace.

2.4.5 Discussion

From the above discussion, it can be seen that a significant amount of research on feature recognition has focused on the domain of process planning. Still, a comprehensive solution has not yet been found in this area. Some problems, such as intersecting feature recognition, computational complexity, and integration of manufacturing information with the recognized features, are still under investigation.

On the other hand, process planning is just one feature based application domain. There are other areas in the machining domain that have the potential to benefit from the use of feature based techniques. In this research, feature based approaches are applied to a new application area, process modeling, to facilitate *CWE* extraction. Since this is a new application area being addressed in terms of features, feature recognition methodologies will be investigated in this thesis to extract in-process machining features.

2.5 Swept Volume Generation

A swept volume can be defined as the volume generated by the motion of an arbitrary object along an arbitrary path, possibly with arbitrary rotations. As mentioned in Chapter 1, swept volume generation plays an important role in the field of geometric modeling for Virtual Machining, since in-process workpiece modeling and removal volume generation require swept volumes. Although many studies on this topic have been reported during the last two decades, the problem is still not considered to be sufficiently well solved. Abdel-Malek *et al.* [2] presented a comprehensive survey and review on swept volume generation techniques. In the following sections, some of the dominant techniques will be discussed.

2.5.1 Jacobian Rank Deficiency Approach

The Jacobian Rank Deficiency (JRD) approach was first proposed by Abdel-Malek *et al.* [1]. This approach is based on singularity theory in differential geometry. In singularity theory, a manifold with singularities represents a manifold with boundary parts of lower dimensions. In the JRD approach, a rank-deficiency condition is imposed on the constraint Jacobian of the sweep to determine singular sets. The boundary to the resulting swept volume is generated by substituting the resulting singularities into the constraint equation.

One advantage of the JRD approach is that the formulation is applicable to entities of any dimension because of the generality of the rank-deficiency condition. Another advantage is that this approach can generate the exact boundary envelope of a swept volume in closed form. The problem with the JRD approach is that its applications are limited to parametric and implicit sweeping. The resulting swept volumes cannot be directly used in solid modeler based simulations since its analytical representation of the boundary does not directly map to any of the native representation forms.

2.5.2 Sweep Differential Equation Approach

The Sweep Differential Equation (SDE) approach was first presented by Blackmore *et al.* [5]. This approach proposes an algorithm for generating swept volumes using the trajectories of sweep differential equations, which is called the Sweep Vector Field (SVF). As illustrated in Figure 2-10, the SVF decomposes the boundary of the moving object into three sets of points: Ingress Points, Egress Points and Grazing Points. The set of ingress (egress) points are all points on the boundary of the object at which the SVF points into (out of) the interior of the object. The set of grazing points are those points that are neither ingress nor egress points. The full curves of these grazing points at each time step are calculated for constructing the boundary envelope of the swept volume.

The main problem with the SDE approach is its computational complexity, since the grazing point curves must be calculated at every time step along the sweep. Blackmore *et al.* [6] extended the SDE approach by introducing the Sweep-Envelope Differential Equation (SEDE). The SEDE approach only needs to calculate the grazing point set at the initial position of the object; the remaining grazing points can be generated by the flow of the sweep-envelop differential equation. Computational efficiency is thereby dramatically improved.

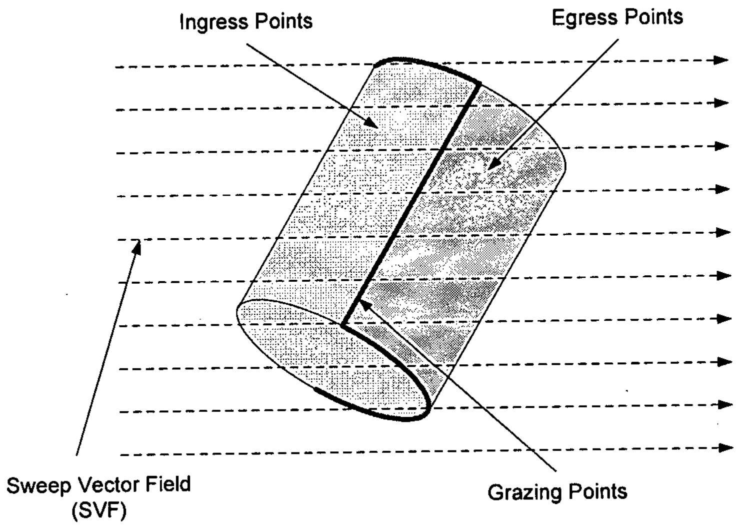


Figure 2-10: Ingress, Egress and Grazing Points on the Boundary of the Object

2.5.3 Solid Model Based Approaches

Mathematical approaches such as JRD and SDE represent the boundary of the resulting swept volume with parametric or implicit equations. It is difficult to integrate these swept volumes into solid modeler based approaches for machining simulations. Attempts have been made to generate swept volumes using a B-rep model for ease of integration. Many researchers have proposed curve-skinning techniques to generate these swept volumes. As illustrated in Figure 2-11, the basic philosophy of these approaches is to interpolate a series of profile curves along the path with parametric surfaces (e.g., B-Spline, NURBS surfaces). These profile curves can be silhouette curves of the moving cutter geometry at a sequence of cutter locations along the path or a set of curves that can be considered on the boundary of the swept volume. These envelope surfaces are then stitched together by solid modeler operators to form a B-rep solid model of the swept volume. In these approaches, the key is how to generate these profile curves. The following approaches use different techniques to construct the profile curves.

Sheltami *et al.* [49] proposed a technique that is based on identifying generating curves along the path and connecting them into a solid model of the swept volume. In this approach, a generating curve is approximated by a circle. It is valid only when the cutter moves along a circular path. For 5-axis motions, this approach has difficulty in achieving accuracy.

Roth *et al.* [44] presented a method to generate swept volumes for a 5-axis toolpath. This method discretizes the cutter into pseudo-inserts and identifies imprint points using a modified principle of silhouettes. An imprint curve at each cutter location is formed by connecting imprint points that exist for each pseudo-insert. A collection of imprint curves are interpolated by curve-skinning techniques to approximate the swept surface.

Chung *et al.* [10] presents an analytical approach for calculating the silhouette curves of a generalized tool. This approach can provide a closed-form solution, but is limited to 3-axis machining simulations.

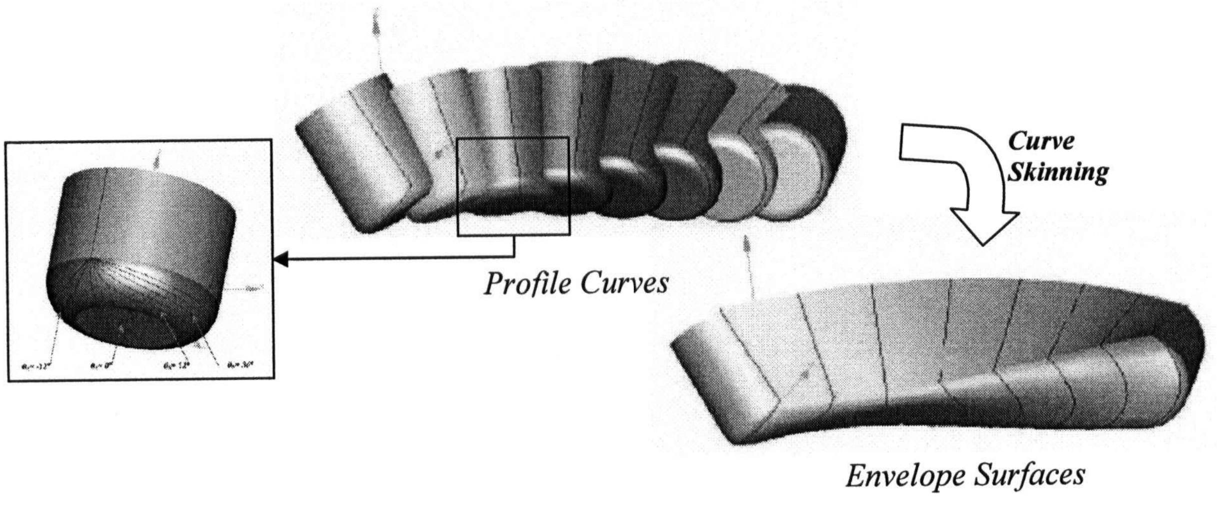


Figure 2-11: Solid Model Based Approaches

Weinert *et al.* [68] proposed an approach for swept volume generation for 5-axis machining. This approach approximates the silhouette curve of a fillet-end cutter by interpolating a set of points on the surface of the cutter. A set of silhouette curves are calculated when the cutter is located at a sequence of cutter locations along the toolpath. The advantage of this approach is that the silhouette curve of a fillet-end cutter can always be represented in the form of parametric curves for 5-axis motions. The disadvantage of this approach is that the parametric curve is only an approximation to the silhouette curve. Therefore, only approximate solutions to swept volume generation can be made in this approach.

2.5.4 Discussion

From the above discussion, it can be seen that the JRD and SDE approaches generate the exact boundary envelope of a swept volume in the form of parametric or implicit equations, making it difficult to integrate with a solid modeler based simulation. On the other hand, the nature of the curve-skinning technique in solid model based approaches only promises an approximation to the swept volume. For some special cases, for example, the helical toolpath in hole milling, there is a potential for the swept volume to be represented as an exact solid model. Unfortunately these approaches can only provide approximate solutions to these special cases. This limitation motivates research in this thesis into methodologies that provide an accurate solution to the swept volumes generated by special motions such as a helical motion.

2.6 Distributed, Collaborative and Multi-Agent Systems for Engineering

As discussed in Chapter 1, Multi-Agent System (*MAS*) can provide a framework for parallel computing. Reported research in this area focuses on the development of distributed, collaborative systems for engineering analysis, design and process planning. Applications have not been found in the field of process modeling, which inspires research efforts to develop a *MAS* based framework for facilitating *CWE* extraction in process modeling. In this section, previous work on distributed, collaborative and multi-agent systems for engineering will be reviewed.

2.6.1 Distributed and Collaborative Systems for Engineering

Web technologies stimulate research work on developing distributed and collaborative systems for engineering design and manufacturing. These systems normally are implemented by integrating existing engineering techniques with emerging web technologies. This integration provides the possibility of collaborating among geographically distributed teams to achieve their goals. Cutkosky *et al.* [13] proposed a real-time collaboration environment based on the Internet and knowledge-sharing agreements to facilitate communication among specialists and their tools. Stori and Wright [55] built an internet-based design and manufacturing environment to provide a networked machining service. Li and Lu [39] developed a web-based process planning system to support distributed design and manufacturing analysis. Wang et al. [66] proposed a scheduled role-based distributed data access control model to support data security management in a distributed environment. CAD-based Application Service Providers (ASP) have also evolved in recent years to offer complete CAD modeling via the Internet on a pay-per-month or pay-per-use basis (e.g., Alibre© [38]).

These web-based collaborative systems publish their services on the Internet through web service architectures. The web service architecture decomposes the engineering kernel into a set of software components and distributes them on different machines on the server side. These components are not intelligent; no collaborative behaviors can be conducted among them. Therefore, a control unit is always needed to coordinate their work. This architecture imposes a limitation that software components of the engineering kernel can only be distributed among a limited number of machines on the sever side. The user's machine acts as a thin-client whose job is only to collect user inputs and visualize the results. *MAS* technologies provide an technique called agent mobility to allow a much wider range of distribution, even including the whole of the Internet if web technologies are integrated. Recent research has demonstrated the trend of integration of web and *MAS* technologies.

2.6.2 Integration of Multi-Agent Systems and Web Technologies for Engineering

Reported research shows that *MAS* technologies are playing an increasingly important role in developing web-based, intelligent, distributed and collaborative applications. [50] and [67] give a comprehensive survey and review of *MAS* applications in engineering design and manufacturing. There is significant research on the application of agents in the design field, such as PACT, SHARE, SiFA, DIDE, ICM, Co-Designer, and A-Designer. These earlier applications by different groups reveal the important issue of the interoperability between heterogeneous agents and agent systems. FIPA (Foundation of Intelligent Physical Agents)

[71] has been built to improve the interoperability between heterogeneous agents and agent systems. Following FIPA specifications, agent systems can be designed to interoperate with each other for collaborations. FIPA also stimulates the development of FIPA compliant agent platforms such as JADE, ZEUS and FIPA-OS.

Hao *et al.* [24] reported a FIPA compliant multi-agent framework called AADE (Autonomous Agent Development Environment). This framework can facilitate the development of collaborative engineering applications by using its development toolkits. In [24], a web-based design and optimization project is implemented based on the AADE framework. The authors focused only on the development of the FIPA compliant multi-agent framework. The implemented project demonstrates that the job distributions are created only on the server side, and no collaboration actions taken by agents are observed. Therefore, this framework does not fully take advantage of the capabilities of *MAS*.

Huang *et al.* [27] presented a web-based framework for collaborative product development. Their framework integrates the concept of agents into workflow management. In their research, the workflow of a project is modeled as a network. The concept of agents is introduced to define nodes and the concept of messages to define edges. Through flow messages, agents can collaborate with each other. The advantage of the agent-based framework is that the same agents may be reused in the workflow of different projects without any changes.

2.6.3 Discussion

It can be seen that recent research has focused on the integration of web technologies and multi-agent systems. The attractiveness of the Web for propagating information, and multi-agent systems for their distributed, collaborative and intelligent capabilities, can be integrated to make it more efficient in accessing and manipulating information. However, the challenge is how to build a web-based framework that facilitates integration of related emerging technologies. In this thesis, a distributed, web-enabled, *MAS* based framework is proposed for *CWE* extraction. The aim of this research is to develop a framework to facilitate the integration of *CWE* extraction techniques implemented by using different methodologies such as B-rep, Z-buffer and polyhedral based approaches.

2.7 Summary

In this chapter, a review of the literature in force prediction models, *CWE* extraction, feature recognition, swept volume generation, and multi-agent systems are presented. It has been shown that force prediction models have been well developed for predicting cutting forces. The reported *CWE* extraction approaches have demonstrated difficulty in computational efficiency and robustness. Feature recognition methodologies for process planning have been well addressed, and feature-based methodologies for process modeling need to be investigated. General swept volume generation approaches have been shown to be deficient in providing exact solutions to swept volume generation with the helical path. Distributed, collaborative and multi-agent systems and their application in the field of engineering have been addressed in this chapter.

Chapter 3
Feature Taxonomy for Process Modeling

3.1 Introduction

As mentioned in the previous chapters, machining process modeling requires *CWE* geometry in order to predict cutting forces. The calculation of these engagements is challenging due to the complicated and changing intersection geometry that occurs between the cutter and the in-process workpiece. B-rep solid modelers can be used to perform these calculations by executing intersection operations between the cutter and the workpiece or removal volume at successive cutter locations. These operations utilize parametric surface/surface intersection algorithms. For the large number of engagements that can occur in machining a complicated workpiece, this can be a time-consuming and sometimes unreliable process. To improve the computational efficiency and stability of B-rep based approaches, in this research, in-process machining features are introduced for reducing the number of intersections that need to be performed in process modeling. One of the primary in-process machining features, the cutter engagement machining feature, was first proposed by Yip-Hoi *et al.* [70] for characterizing the *CWE*. In this chapter, several in-process machining features are defined for facilitating *CWE* extraction in process modeling.

First, a definition of machining features is introduced. And then, a classification of machining features for process modeling is presented, followed by formal definitions of related in-process machining features in process modeling: Cutter Engagement Machining Features, Removal Volume Machining Features, Geometric Invariant Machining Features and Form Invariant Machining Features. In this chapter, Removal Volume, Geometric Invariant and Form Invariant Machining Features are assumed to be features that can be created using 2½D milling operations.

3.2 Machining Features

There are many definitions of the concept of a *feature* in the literature. Commonly, a feature is considered to be an abstract concept that refers to regions of interest on a component defined within a specific application domain. Features can be defined from different viewpoints, such as design, analysis and manufacturing. Machining features are defined from the manufacturing point of view. A *Machining Feature (MF)* is defined by Shah as follows:

> *A Machining Feature is a collection of related geometric elements which correspond to a particular manufacturing method or process, or which can be used to reason about the suitable manufacturing methods or processes for creating that geometry.*
>
> (Shah *et al.* [48])

Traditionally, machining features are used in the context of process planning. In this domain, machining features are extracted from a CAD model of the final part to aid in the creation of process plans and toolpaths. However, machining feature-based approaches can be useful in a much broader range of application domains. In the process modeling domain, machining features can be classified into a new category: in-process machining features, to be used for capturing regions of interest within volumes removed during machining operations to facilitate *CWE* extraction.

3.3 Feature Taxonomy

There have been many efforts to classify machining features and create feature taxonomies. No agreement on a canonical set of features for any application has yet been reached by the features technology community. Since features are application dependent, it is more practical to classify features based on the specific application domain.

As illustrated in Figure 3-1, a classification of machining features for process modeling is proposed. In this feature hierarchy, final part machining features (*fpF*) are used in process planning. A large number of feature recognition techniques reviewed in the last chapter address the problem of identifying these machining features from a CAD model of the final part. The result is a set of final part machining features $\{fpF_i\}$. A sequence of machining operations is then generated by process planning. Each operation in turn when executed leads to the creation of in-process machining features (*ipF*).

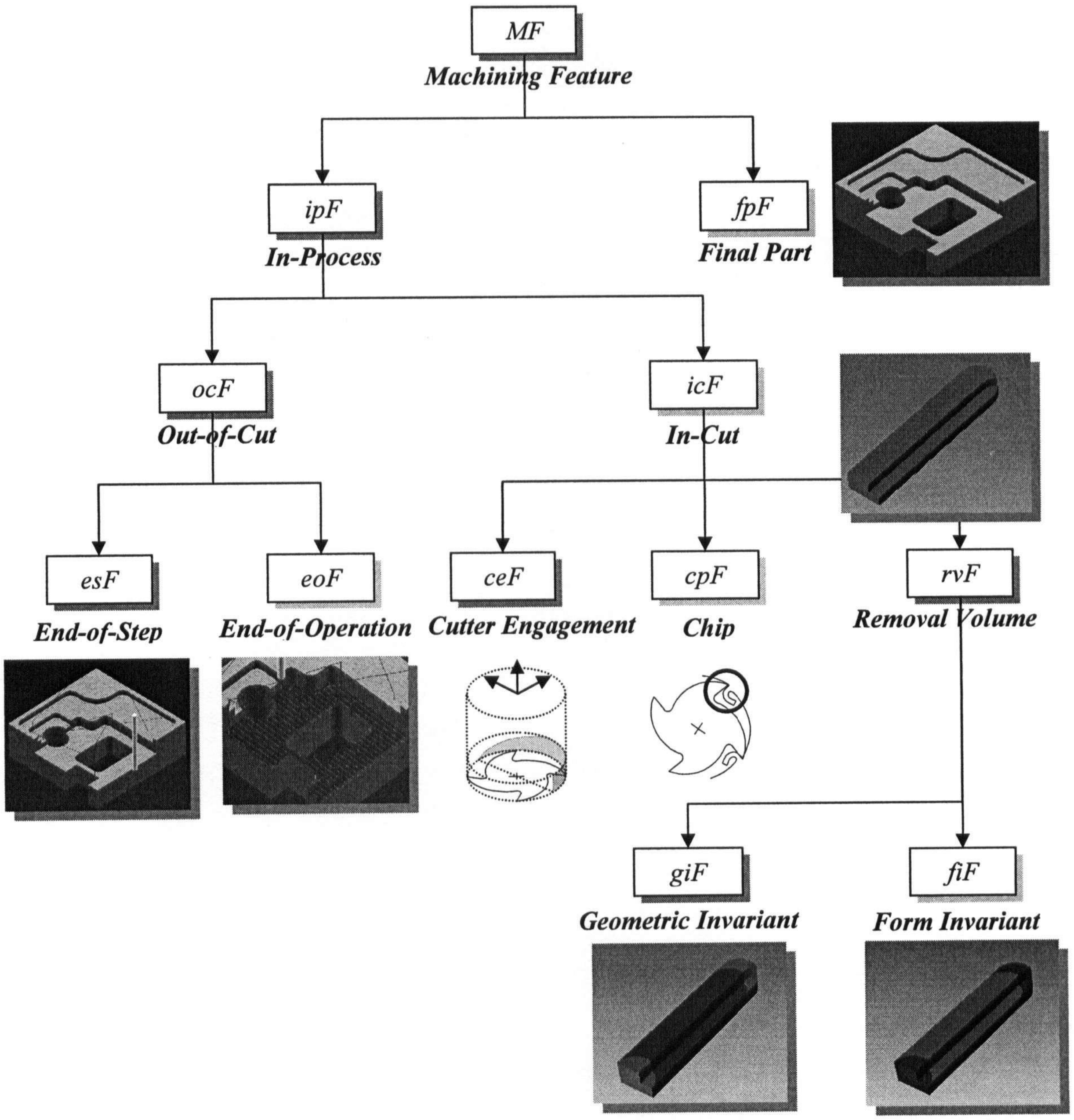


Figure 3-1: Machining Feature Classification for Process Modeling

An in-process machining feature is defined as a set of surfaces generated on the workpiece by a sequence of machining operations that lead to a final part machining feature. Distinct from *fpF*s, *ipF*s describe intermediate states of the workpiece before a *fpF* is completely machined. For the purposes of integrating process modeling into process planning, it is necessary to add an *ipF* node into the feature hierarchy. As shown in Figure 3-1, an in-process machining feature is classified into In-Cut (*icF*) and Out-of-Cut (*ocF*) machining features. An *ocF* is defined as the intermediate state of the workpiece when the operation or a step is completed and a new operation or another step of the operation is about to be started. It is obvious that the cutter is not engaged with the workpiece at the moment when an *ocF* is generated. This class of feature can be used in the field of tooling, fixture and gauging design, and inspection planning. On the other hand, *icF*s capture characteristics of the workpiece during a machining operation when the cutter is engaged with the workpiece. These features help in describing workpiece states during machining operations in process modeling.

In-cut machining features can be classified into three categories. Chip machining features (*cF*) capture the characteristics of chip geometries generated within a single revolution of the cutter. Cutter engagement machining features (*ceF*) are used to describe the cutter/workpiece engagement over a single revolution of the cutter. *ceF*s are the primary concern in this thesis, since parametrizations of *ceF*s are required to provide geometric inputs to a process model. Removal volume machining features (*rvF*) represent volumes removed by the cutter as it moves along a predefined toolpath. This feature is significant because each *rvF* contains engagement information and has much less geometric complexity than the in-process workpiece being generated concurrently. Therefore, in-process workpieces can be replaced by *rvF*s for cutter/workpiece calculations in order to reduce computational complexity. To reduce the number of intersection calculations in B-rep based solutions, an *rvF* is decomposed into two classes: Geometric Invariant Machining Features (*giF*) and Form Invariant Machining Features (*fiF*). *giF*s and *fiF*s are defined to characterize regions of the *rvF* where the engagement is constant or changing in a predictable way. The definition of *giF*s and *fiF*s helps in improving computational efficiency and robustness.

In the following sections, formal definitions and mathematical descriptions of the features described above are provided.

3.4 Cutter Engagement Machining Features

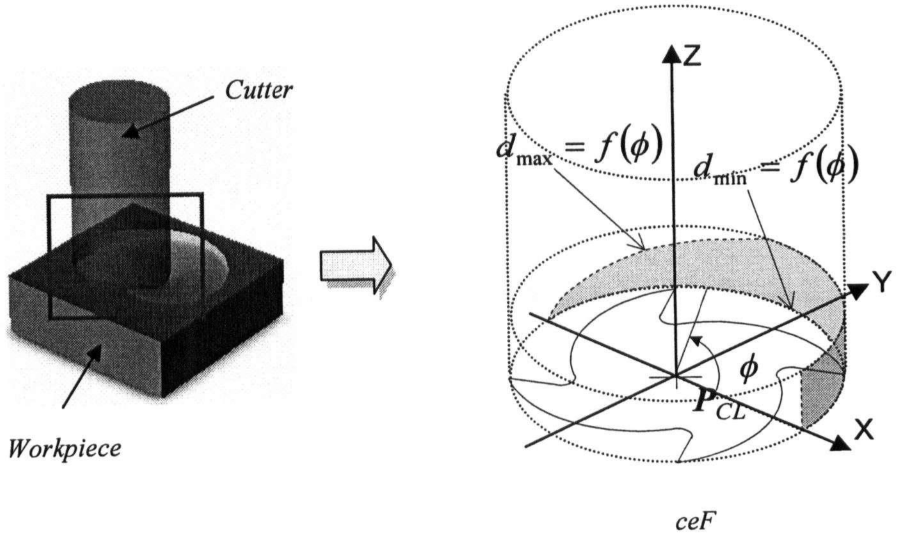


Figure 3-2: Cutter Engagement Machining Features

In process modeling, the definition of *Cutter Engagement Features* (*ceF*) is driven by the requirements of a process model, which dictates the type of geometric input required. *ceF*s are used to describe the engagement conditions between the cutter and the workpiece in the standard format required by the force model. A *ceF* is defined as follows:

> *A Cutter Engagement Feature is a collection of one or more connected regions on the envelop surface of a cutting tool that approximates the surface generated on the workpiece during one revolution of the cutter.*
>
> (Yip-Hoi *et al.* [70])

In their research, a classification of *ceF*s is presented to represent *CWE*s generated in 2½D end milling. In this thesis, *ceF*s need to be extended to represent *CWE*s in 3-axis end milling process such as hole milling. In this case, the *CWE* may be bounded by several free-form curves.

Generally, a *ceF* is expressed as follows:

$$ceF = \bigcup_{i=1}^{m} Zone_i$$

where:

$Zone_i$ is called an engagement zone in the axial depth of cut-engagement angle (d-ϕ) domain. It can be represented as $Zone_i[d_{mini}, d_{maxi}, \phi_{sti}, \phi_{exi}]$.

Figure 3-2 illustrates a *ceF* for a hole milling process based on a helical toolpath. This *ceF* consists of one connected region on the surface of the cutter which is bounded by the entry (ϕ_{st}) and exit engagement angles (ϕ_{ex}) of the cutter and limits on the axial depth of cut [d_{min}, d_{max}]. The lower (d_{min}) and upper (d_{max}) engagement limits can be represented as a function of the engagement angle ϕ: $d_{min} = f(\phi)$ and $d_{max} = f(\phi)$ respectively, where $\phi \in [\phi_{st}, \phi_{ex}]$. When the region on the surface of the cutter is mapped into the d-ϕ domain, the *ceF* is then represented as:

$$ceF = Zone_1[d_{min1}, d_{max1}, \phi_{st1}, \phi_{ex1}]$$

Since *CWE*s extracted from geometric modeling are required to be represented as *ceF*s, *CWE* extraction is alternately called *ceF* extraction in the following chapters.

3.5 Removal Volume Machining Features

The goal of defining *Removal Volume Machining Features* (*rvF*) is to improve the computational efficiency of B-rep based approaches for *CWE* extraction by replacing the in-process workpiece with the removal volume. A *rvF* is defined as follows:

A Removal Volume Machining Feature is a set of one or more material volumes removed by the action of a cutter as it advances along a linear or circular tool motion.

An *rvF* is generated by a regularized Boolean intersection ($\cap^*$) between the in-process workpiece and the tool swept volume. The latter is formed by sweeping the cutter geometry along a linear or circular tool motion. An *rvF* contains all the geometric information about engagement conditions between the cutter and workpiece while machining this portion of the workpiece. Its formal definition is expressed as follows:

$$rvF = \bigcup_{i=1}^{k} R_i$$

where:

$R_i \subset E^3$: A connected regular closed set in 3D Euclidean space.

There are two types of surface geometries in the boundary of each *rvF* element R_i. One type is from the in-process workpiece, the other is from the tool swept volume:

$$sR_i = \bigcup_{j=1}^{m} sW_{ij} \cup \bigcup_{j=1}^{n} sS_{ij} :$$

where:

sW_{ij}: Surface geometry which originates from the in-process workpiece, called the W-Boundary.

sS_{ij}: Surface geometry which originates from the tool swept volume, called the S-Boundary.

Figure 3-3 (a) illustrates an example of a linear *rvF*. In this example, this *rvF* is composed of two disjoint volumes:

$$rvF = R_1 \cup R_2$$

The surfaces of R_1 includes five W-Boundary and two S-Boundary surfaces:

$$sR_1 = \{sW_{11}, sW_{12}, sW_{13}, sW_{14}, sW_{15}\} \cup \{sS_{11}, sS_{12}\}$$

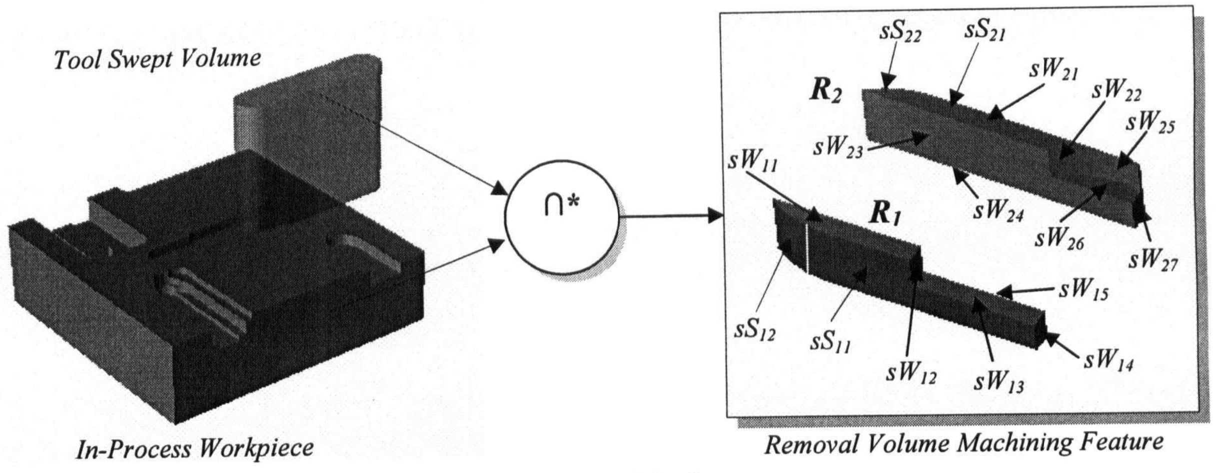
The surfaces of R_2 includes seven W-Boundary and two S-Boundary surfaces:

$$sR_2 = \{sW_{21}, sW_{22}, sW_{23}, sW_{24}, sW_{25}, sW_{26}, sW_{27}\} \cup \{sS_{21}, sS_{22}\}$$

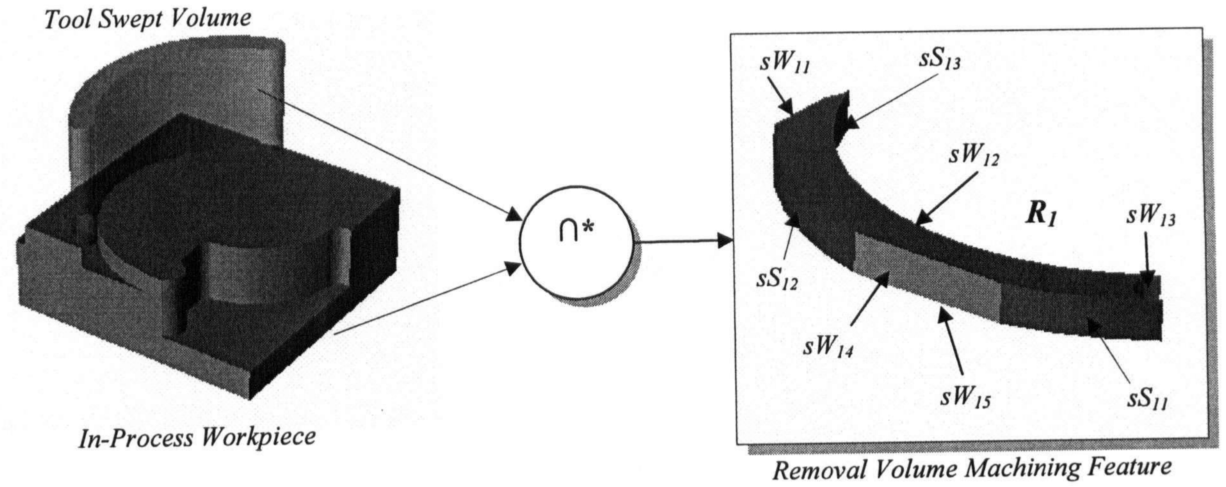
A circular *rvF* example is shown in Figure 3-3 (b). A single removal volume R_1 in this case has five W-Boundary and three S-Boundary surfaces as follows:

$$rvF = R_1$$

$$sR_1 = \{sW_{11}, sW_{12}, sW_{13}, sW_{14}, sW_{15}\} \cup \{sS_{11}, sS_{12}, sS_{13}\}$$



(a) Linear Tool Motion



(b) Circular Tool Motion

Figure 3-3: Removal Volume Machining Features

3.6 Geometric Invariant and Form Invariant Machining Features

The motivation to define geometric invariant and form invariant machining features is to improve the computational stability and efficiency of B-rep based *CWE* extraction approaches. A geometric invariant feature characterizes a portion of a removal volume where *CWE*s are constant while a form invariant feature characterizes a portion where *CWE*s have a certain topological shape. The characteristics of these features not only reduce the number of intersection calculations but also eliminate numerical surface/surface intersections. *CWE*s can be extracted from these features analytically. Computational stability and efficiency can be improved significantly as a result.

In this section, formal definitions of geometric invariant and form invariant machining features will be given.

3.6.1 Geometric Invariant Machining Features

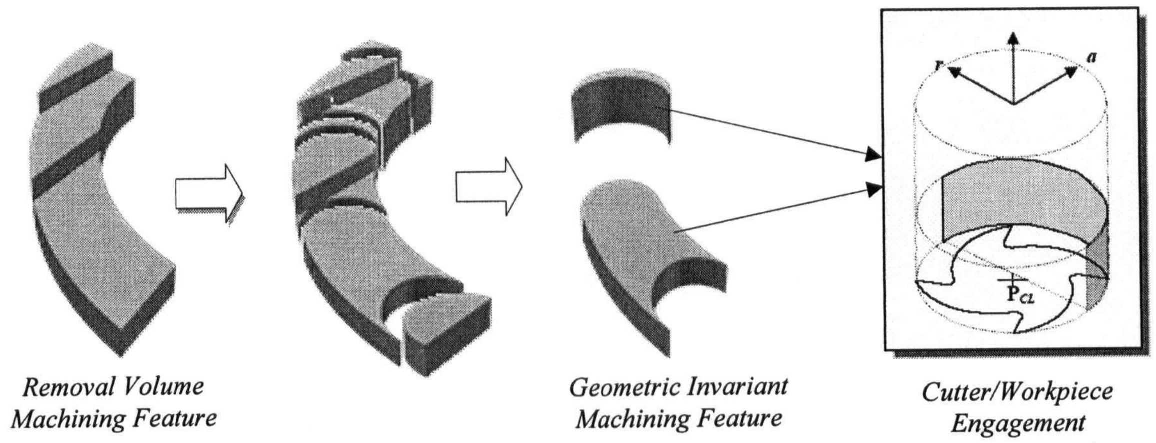

r
a
P_{CL}
Removal Volume Machining Feature
Geometric Invariant Machining Feature
Cutter/Workpiece Engagement

Figure 3-4: Geometric Invariant Machining Features

Figure 3-4 illustrates an example of a *Geometric Invariant Machining Feature* (*giF*). The removal volume machining feature is decomposed into several sub-volumes. Two of them have constant cutter/workpiece engagements while the cutter moves along the toolpath to machine them from the workpiece. This feature type is defined as follows:

A Geometric Invariant Machining Feature is a portion of a Removal Volume Machining Feature where the cutter engagement features do not change during machining.

The mathematical definition is given as follows:

$$giF = \bigcup_{i=1}^{k} R_i$$

where:

$R_i \subset E^3$: A connected regular closed set in 3D Euclidean space.

For any two *ceF*s in a *giF*,

$$ceF_1 = \bigcup_{i=1}^{m} Zone_{1,i}\left[d_{\min 1,i}, d_{\max 1,i}, \phi_{st1,i}, \phi_{ex1,i}\right] \text{ and } ceF_2 = \bigcup_{i=1}^{n} Zone_{2,i}\left[d_{\min 2,i}, d_{\max 2,i}, \phi_{st2,i}, \phi_{ex2,i}\right],$$

the following constraints should be satisfied:

- The number of engagement zones of each *ceF* is constant:

  $m = n$

- The corresponding engagement zones are identical:

  $d_{min1,i} = d_{min2,i}$ and $d_{max1,i} = d_{max2,i}$

  $\phi_{st1,i} = \phi_{st2,i}$ and $\phi_{ex1,i} = \phi_{ex2,i}$

  where: i=1,2,...*m*.

3.6.2 Form Invariant Machining Features

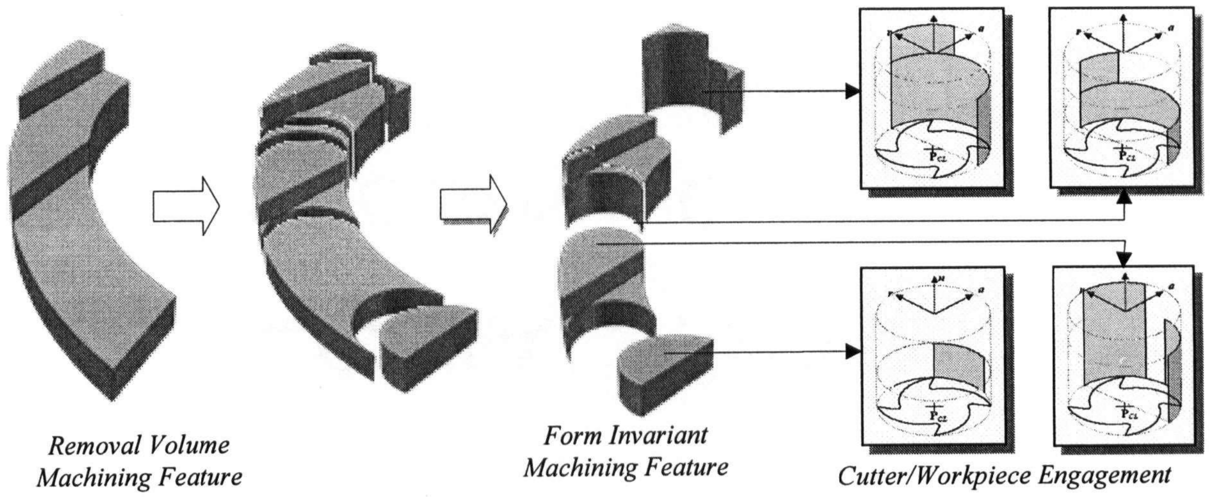

Removal Volume Machining Feature
Form Invariant Machining Feature
Cutter/Workpiece Engagement

Figure 3-5: Form Invariant Machining Features

Figure 3-5 illustrates an example of a *Form Invariant Machining Feature* (*fiF*). The removal volume machining feature is decomposed into several sub-volumes. Four of them have similar forms while the cutter moves along the toolpath to machine them from the workpiece. This feature type is defined as follows:

A Form Invariant Machining Feature is a portion of a Removal Volume Machining Feature where the cutter engagements features have similar shapes during machining.

The mathematical definition is given as follows:

$$fiF = \bigcup_{i=1}^{k} R_i$$

where:

$R_i \subset E^3$: A connected regular closed set in 3D Euclidean space.

For any two *ceF*s in a *fiF*,

$$ceF_1 = \bigcup_{i=1}^{m} Zone_{1,i}\left[d_{\min 1,i}, d_{\max 1,i}, \phi_{st1,i}, \phi_{ex1,i}\right] \text{ and } ceF_2 = \bigcup_{i=1}^{n} Zone_{2,i}\left[d_{\min 2,i}, d_{\max 2,i}, \phi_{st2,i}, \phi_{ex2,i}\right],$$

the following constraints should be satisfied:

- The number of engagement zones of each *ceF* is constant:

  $m = n$

- The corresponding engagement zones are similar:

  $d_{min1,i} = d_{min2,i}$ and $d_{max1,i} = d_{max2,i}$

  $\phi_{st1,i} \neq \phi_{st2,i}$ or $\phi_{ex1,i} \neq \phi_{ex2,i}$

  where: i=1,2,…m.

3.7 Summary

In this chapter, in-process machining features are added into the machining feature hierarchy to capture the characteristics of the workpiece geometry during machining operations. Some related in-process machining features, such as Cutter Engagement, Removal Volume, Geometric Invariant and Form Invariant, are formally defined in this chapter. The definition of these in-process machining features in process modeling helps to improve the computational efficiency and robustness of B-rep based *CWE* extraction approaches by reducing geometric complexity and the number of numerical surface/surface intersection calculations. These definitions are one of contributions of this research.

Chapter 4

A Feature-Based Approach to Cutter/Workpiece Engagement Extraction in 2½D End Milling

4.1 Introduction

As described in the previous chapters, cutting forces are a key input in simulating the vibration of machine tools prior to implementing the real machining process. This simulation can be used to optimize instantaneous process parameters to avoid chatter and improve machining quality. Instantaneous cutting forces are determined by the feed rate, spindle speed, and *CWE* (which captures the depth of cut). Performing *CWE* calculations is difficult due to the complex geometry of the in-process workpiece.

In 2½D end milling, a B-rep solid modeler based *CWE* calculation approach determines engagements by performing Surface/Surface Intersections (SSI) between the advancing semi-cylindrical surface of a cutter and either the in-process workpiece or the removal volume (see Figure 4-1). Intersecting the removal volume is preferred since the removal volume typically has less complicated geometry than the in-process workpiece. The robustness and computational efficiency of this advancing semi-cylinder approach significantly depends on the solid modeler, since it utilizes the SSI algorithm within the modeler to retrieve *CWE* boundary curves. The following are several problems in realizing this advancing semi-cylinder approach:

- The SSI approaches in most solid modelers use numerical techniques that are limited by accuracy, efficiency and robustness. These have great influence on the stability and efficiency of the advancing semi-cylinder approach.
- Numerical techniques in SSI provide only an approximation to *CWE* even though the intersecting surfaces are quadric surfaces, which are common in 2½D end milling process. Quadric Surface/Surface Intersection can be solved by algebraic or geometric approaches to obtain exact solutions for *CWE*s.

- *CWE*s at a series of consecutive cutter locations may have the same geometry, which leads to duplication of identical SSI computations.

As illustrated in Figure 4-1, an advancing semi-cylinder intersects the removal volume at several cutter locations with constant interval δ. There are two groups of consecutive Cutter Engagement Features (*ceF*s), {ceF_2, ceF_3, ceF_4} and {ceF_6, ceF_7, ceF_8}, where the *ceF*s are identical. If a subset of volumes with invariant *ceF*s can be identified from the removal volume, only a single SSI is needed for each sub-volume. Computational efficiency will be significantly improved.

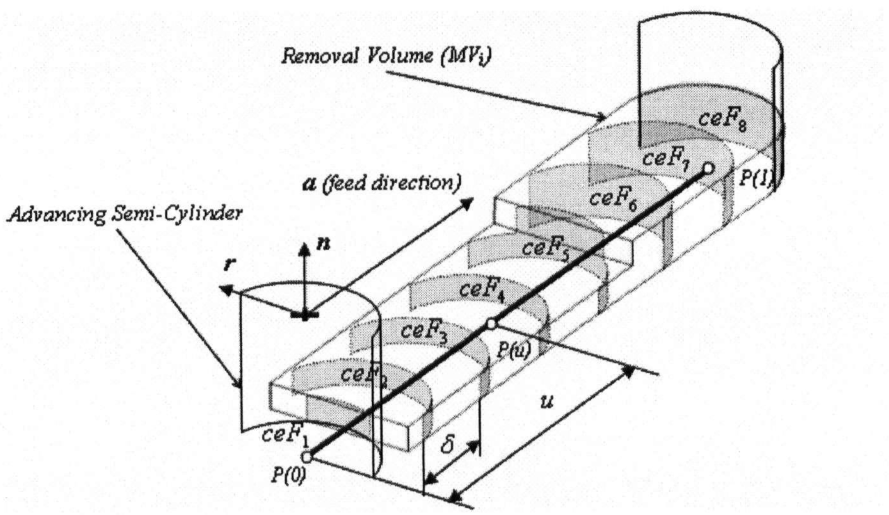


Figure 4-1: *ceF*s Extraction using Advancing Semi-Cylinder

In this chapter, a feature-based approach is presented that addresses the above problems in the B-rep solid modeler based approach. This feature-based approach decomposes a removal volume into two classes of features by using a feature recognition technique. As shown in Figure 4-2, volumes with invariant *ceF* are defined as Geometric Invariant Features (*giF*), and volumes with invariant form are classified as Form Invariant Features (*fiF*). The formal definitions of *giF* and *fiF* were given in the last chapter. The extraction of *giF*s and *fiF*s reduces duplication of the identical engagement calculation. Accurate analytical representations of *ceF*s based on these features parameters are also presented in this chapter. Formally, *ceF* can be represented as a closed-form single-variable function *ceF(u)*, where variable *u* is a parameter representing position along a linear or circular segment of tool motion. As shown in Figure 4-1, *P(u)* is a cutter location within a segment of a tool motion,

$u \in [0,1]$. For example, ceF_4 is the cutter engagement feature with the cutter located at cutter location $P(u_4)$.

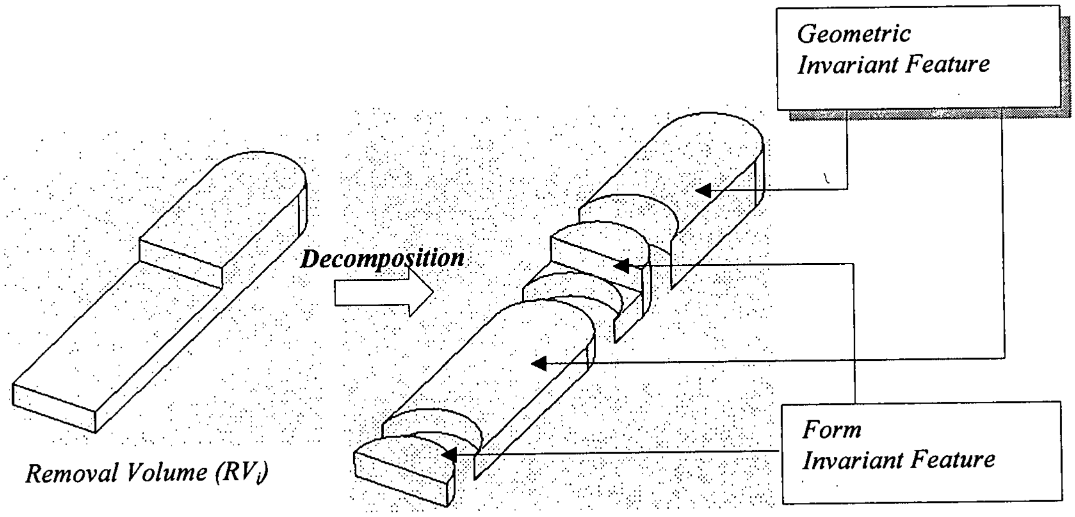


Figure 4-2: Removal Volume Decomposition

Following an overview of this feature-based approach, swept volume and removal volume generation methods are presented. A set of operators used to decompose the removal volume into these feature types are then introduced. A feature recognition algorithm based on these operators is presented to recognize these machining features. After feature recognition, the *CWE* is retrieved from the recognized machining features. Finally, implementation and validation are presented.

4.2 Overview

A feature-based methodology for *CWE* extraction in 2½D end milling is illustrated in Figure 4-3. This approach has four major steps: swept volume generation, removal volume generation, in-process feature recognition and *CWE* extraction. The first step is to generate a swept volume for each tool motion. In the second step, the swept volume is used to create a removal volume by applying a Boolean intersection operation between the swept volume and the in-process workpiece, and to create an in-process workpiece by subtracting the swept volume from the previous in-process workpiece as well. The removal volume is decomposed into *giF*s and *fiF*s in the In-Process Feature Recognition step. After this step, a set of *giF*s and/or *fiF*s are generated. The final step is to analytically extract *CWE*s from both *giF*s and *fiF*s. This is significant because analytical *CWE* extraction reduces the problem of computational efficiency and robustness found in numerical extraction methods. These four steps will be described in the following sections.

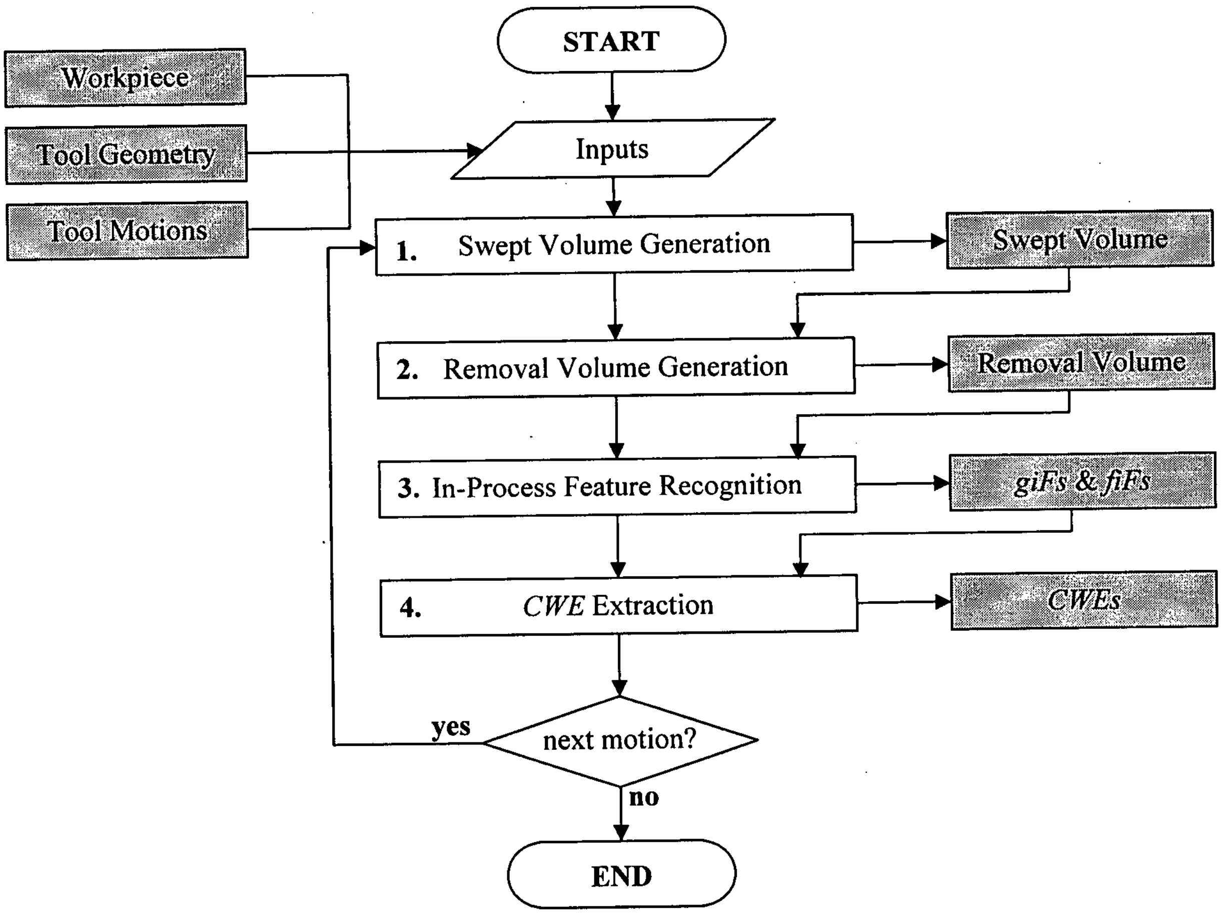


Figure 4-3: Steps in *CWE* Extraction for 2½D End Milling

4.3 Swept Volume Generation

In 2½D end milling, swept volumes are easily generated, since the cutting tool moves on the XY plane. Figure 4-4 illustrates how swept volumes are generated when the cutting tool moves in linear and circular motions, which happens in 2½D end milling. A 2D closed profile curve on the XY plane where the tip of the cutting tool is located is created based on the cutting tool geometry and tool motion. The profile curve is then swept along the tool axis direction (Z direction in 2½D milling) to form a closed volume. Normally, a swept volume is represented as a B-Rep solid to facilitate the *CWE* calculation using a solid modeler.

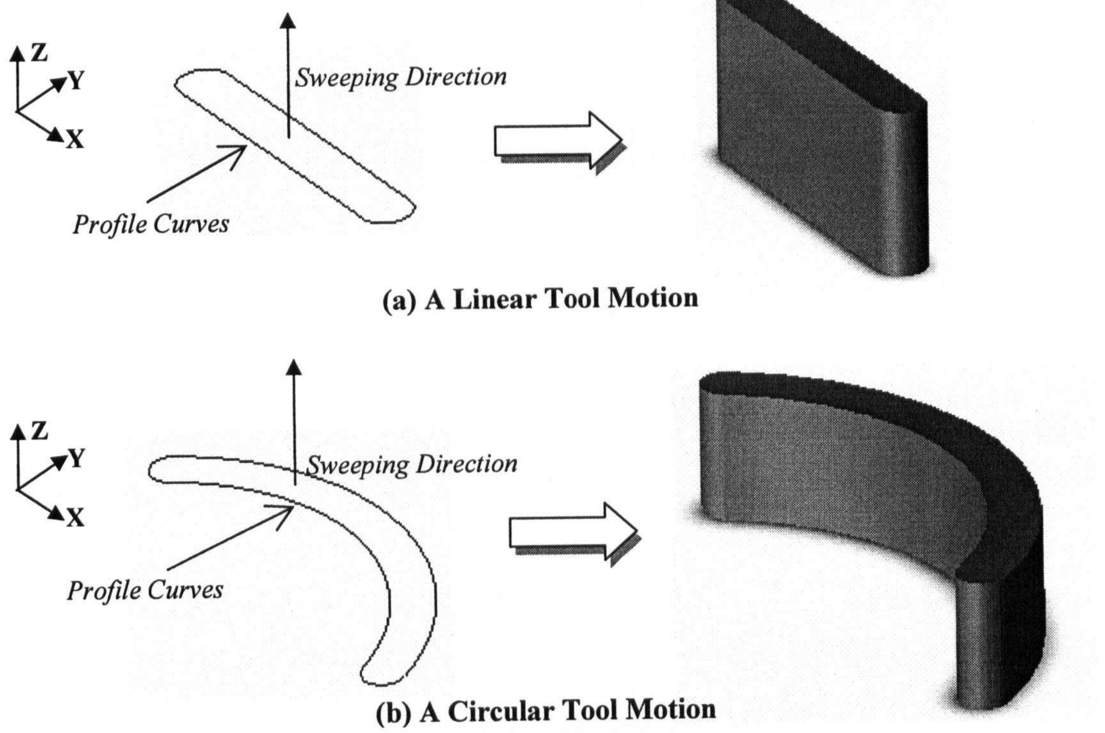


Figure 4-4: Swept Volume Generation for 2½D End Milling

4.4 Removal Volume Generation

The in-process workpiece is defined as a state of the workpiece during a machining process. These intermediate models are updated as the workpiece is machined by the cutter moving along a series of predefined tool motions. The in-process workpiece can be generated by subtracting a swept volume, which is created by sweeping the geometry of the cutter along the current tool motion from the current workpiece state. A regularized Boolean subtraction is used for this operation in a solid modeler based methodology. The removal volume is the geometry of material removed from the in-process workpiece by a single tool motion. The *CWE* calculations require accurate removal volume models for each tool motion. Removal volumes can be generated by performing a regularized Boolean intersection operation between the in-process workpiece and the swept volume. Figure 4-5 shows the generation of the in-process workpiece and the removal volume for the *i-th* tool motion. SV_i represents the tool swept volume in the *i-th* tool motion, W_{i-1} is the in-process workpiece before the *i-th* tool motion. The in-process workpiece W_i is updated by subtracting SV_i from W_{i-1} ($W_{i-1} -^* SV_i$). The removal volume MV_i is generated by a Boolean intersection operation between SV_i and W_{i-1} ($SV_i \cap^* W_{i-1}$).

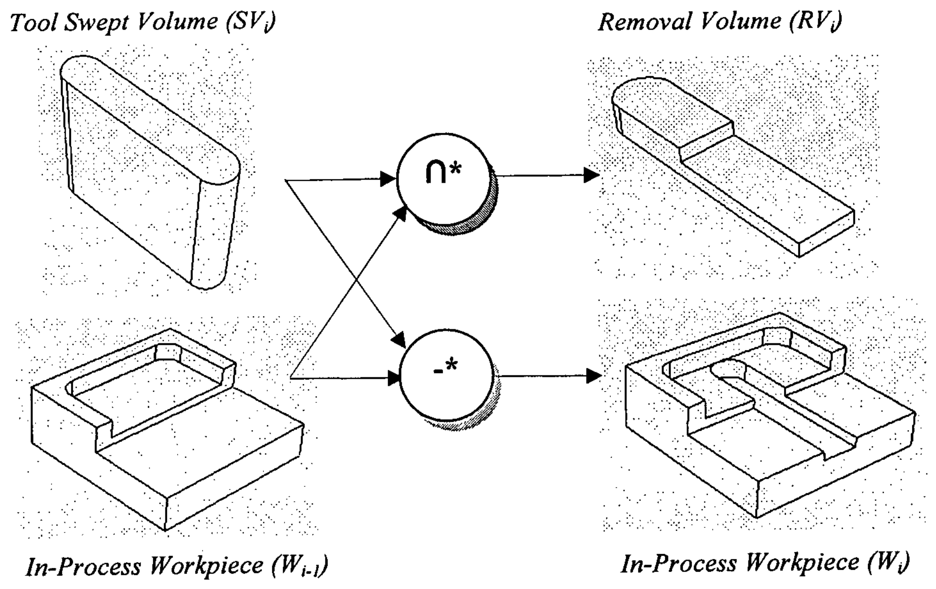

Tool Swept Volume (SV_i)
Removal Volume (RV_i)
∩*
-*
In-Process Workpiece (W_{i-1})
In-Process Workpiece (W_i)

Figure 4-5: In-Process Workpiece and Removal Volume in the *i*-th Tool Motion

4.5 In-Process Feature Recognition

The in-process feature recognition step decomposes a removal volume RV into a set of geometric invariant volumes $\{giF_i\}$ and form invariant volumes $\{fiF_i\}$. Figure 4-6 illustrates this process and its three operators: ***Decomposition***, ***Segmentation*** and ***Extraction***. The **decomposition** operator decomposes the input removal volume into a set of minimal volumes $\{MV_i\}$. These minimal volumes along with the tool geometry and the associated tool motion are input into the **segmentation** operator for generating a set of geometric invariant segments $\{giS_i\}$ and form invariant segments $\{fiS_i\}$. $\{giF_i\}$ and $\{fiF_i\}$ are then extracted from RV based on $\{giS_i\}$ and $\{fiS_i\}$ in the **extraction** operator. These three operators will be described in this section.

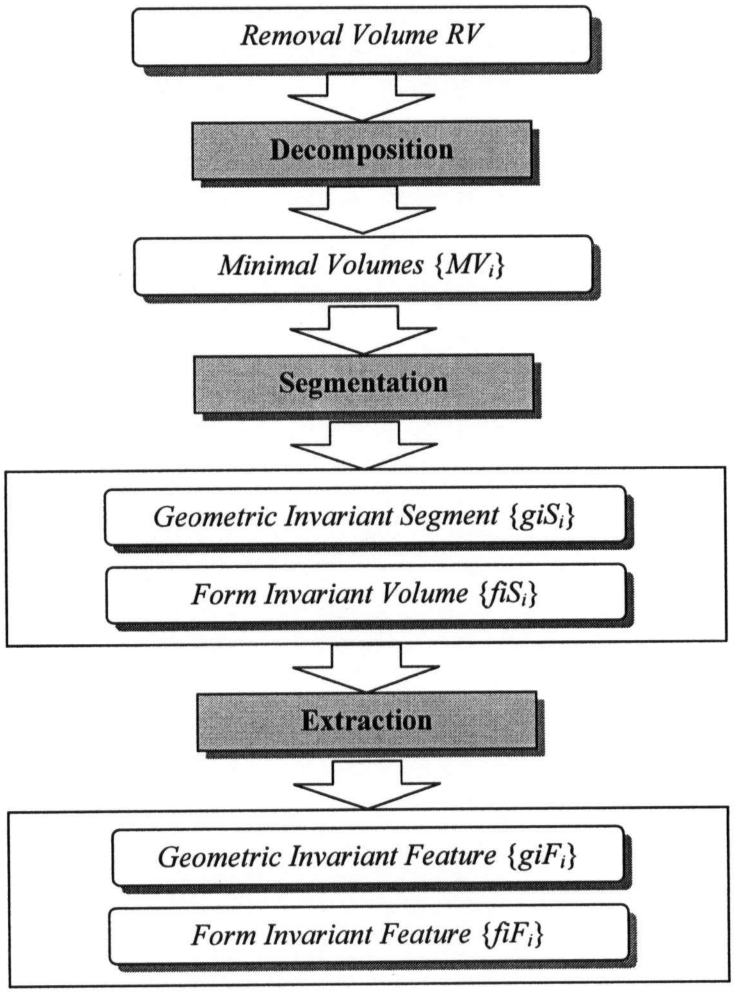

Removal Volume RV
Decomposition
Minimal Volumes {MV_i}
Segmentation
Geometric Invariant Segment {giS_i}
Form Invariant Volume {fiS_i}
Extraction
Geometric Invariant Feature {giF_i}
Form Invariant Feature {fiF_i}

Figure 4-6: In-Process Feature Recognition

4.5.1 Decomposition Operator

A decomposition operator has been defined to decompose a volume into a set of minimal volumes. This operator can be represented as $\{MV_i\}=Decompose(RV)$, where the input is a removal volume RV and output is a set of minimal volumes $\{MV_i\}$.

In 2½D end milling, a minimal volume is considered to be a solid model in E^3 space. This solid model for a minimal volume should satisfy the following constraints:

- For each horizontal face fh_i in the solid model RV, there exists a halfspace h_i. The normal of the face points to the outside of the corresponding half space.

- For each halfspace h_i, all of the geometric entities (vertices v_i, edges e_i and faces f_i) in the solid model RV should be inside or on the halfspace (i.e. $v_i \in h_i$, $e_i \in h_i$ and $f_i \in h_i$).

Figure 4-7 illustrates an example of minimal volumes and non-minimal volumes. It can be seen that halfspace h_2 on the horizontal face fh_2 in (b) does not satisfy the above constraints. Therefore, this volume is not considered as a minimal volume.

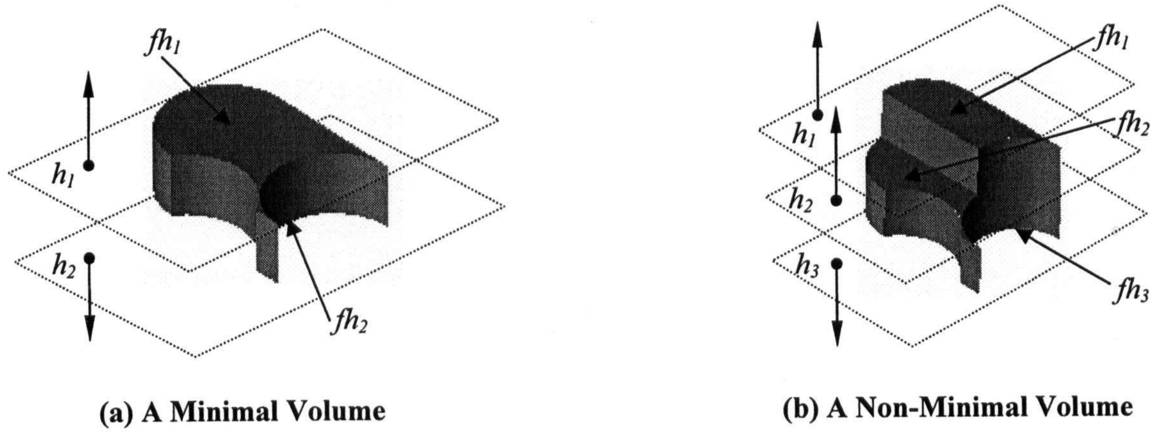


Figure 4-7: Definition of Minimal Volumes

The decomposition operator traverses all faces of the input volume RV to find horizontal faces that do not satisfy the above constraints. Such faces are referred to as splitting-faces. A halfspace h_i is constructed for each splitting-face. The input volume RV can be divided by performing Boolean operations between the halfspace h_i and the removal volume RV. As illustrated in Figure 4-8, a Boolean difference generates an upper minimal volume MV_1, and a Boolean intersection creates a lower minimal volume MV_2. In some solid modelers, these two Boolean operations can be combined into one for splitting the two volumes at the same time, for example, the routine *api_boolean_chop_body* in the solid modeler ACIS.

Algorithm 4-1 describes how a decomposition operator is implemented to decompose a removal volume into a set of minimal volumes. The first step is to search for all of the horizontal faces by traversing the B-rep of the removal volume. These horizontal faces are inserted into a face list. This list is sorted in an ascending order of z coordinates of each face within the list. The head and tail faces in the face list are then removed. When the face list becomes empty after removing the head and tail faces, the removal volume RV is considered as a minimal volume, and is inserted into the output minimal volume list *lstMV*. Otherwise, the removal volume *RV* is split by the halfspaces created based on each face within the face list. A set of minimal volumes are generated and added into the minimal volume list *lstMV*.

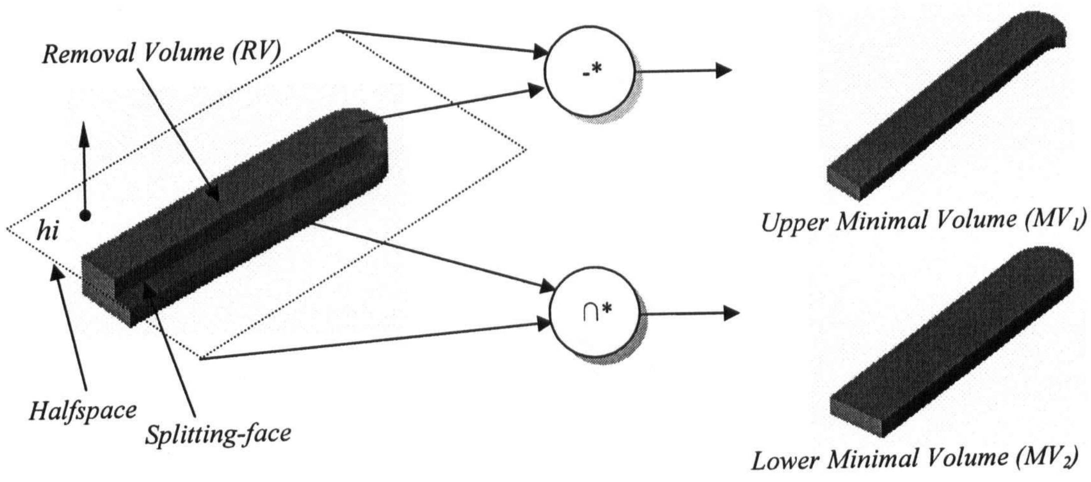


Figure 4-8: Removal Volume Decomposition

```
Routine: to decompose a removal volume into a set of minimal volumes
Input: a removal volume RV
Output: a list of minimal volumes lstMV
lstMV = Decompose (RV){
        ALLOCATE a horizontal face list: lstFace
        for ( each face $f_i$ in RV ) {
                if ( $f_i$ is a horizontal plane )
                        lstFace ← $f_i$
        }
        SORT lstFace in an ascending order of z coordinate of $f_i \in lstFace$
        REMOVE head & tail faces in lstFace
        if ( $lstFace \in \phi$ )
                lstMV ← RV
        else {
                for (each face $f_i$ in lstFace ) {
                        CONSTRUCT a halfspace $H_i$ at $f_i$
                        SPLIT R into a MV with $H_i$
                        lstMV ← MV
                }
        }
}
```

Algorithm 4-1: Removal Volume Decomposition

4.5.2 Segmentation Operator

The segmentation operator divides a linear or circular tool motion into a set of labeled segments according to the engagement conditions between the tool geometry T and a set of minimal volumes $\{MV_i\}$ decomposed from a removal volume. This operator can be represented as $\{S_i\}=Segment(S, T, \{MV_i\})$, where the inputs are a line or circular tool motion S, a tool geometry T and a set of minimal volumes $\{MV_i\}$. The outputs are a set of labeled segments $\{S_i\}$.

As shown in Figure 4-9, the outputs include two types of segments: Geometric Invariant Segments (*giS*) and Form Invariant Segments (*fiS*), which correspond to Geometric Invariant and Form Invariant Volumes, respectively. The type is determined by the engagement conditions between the tool geometry T and the set of minimal volumes $\{MV_i\}$. The engagement condition is invariant when the tool is moving along a *giS*. Otherwise, the engagement condition is changing while the tool is moving along a *fiS*.

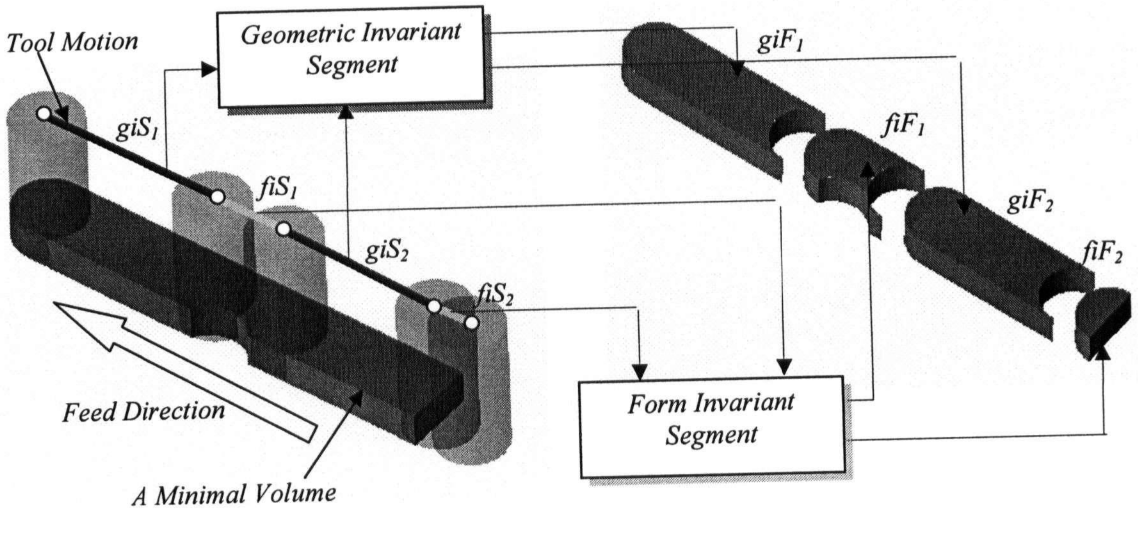


Figure 4-9: Geometric Invariant and Form Invariant Segments

Figure 4-10 illustrates a flowchart of the segmentation operator. This operator includes four steps: **(1) projection, (2) boundary tracing, (3) segment combination for one *MV*** and **(4) segment combination for overlapping *MV*s**. Each minimal volume and the tool geometry are projected onto the XY plane in order to reduce the segmentation calculation in the following steps to a two-dimensional problem. In the boundary tracing step, a set of *giS*s and *fiS*s are generated based on the engagement condition between the tool projection and each edge of the minimal volume projection. Following this step, there are two level segment combination steps. The first level is to combine the overlapping segments generated from a single minimal volume in the boundary tracing step. The second level is to combine the overlapping segments generated from different minimal volumes into a final set of *giS*s and *fiS*s. These four steps are described in the following sections.

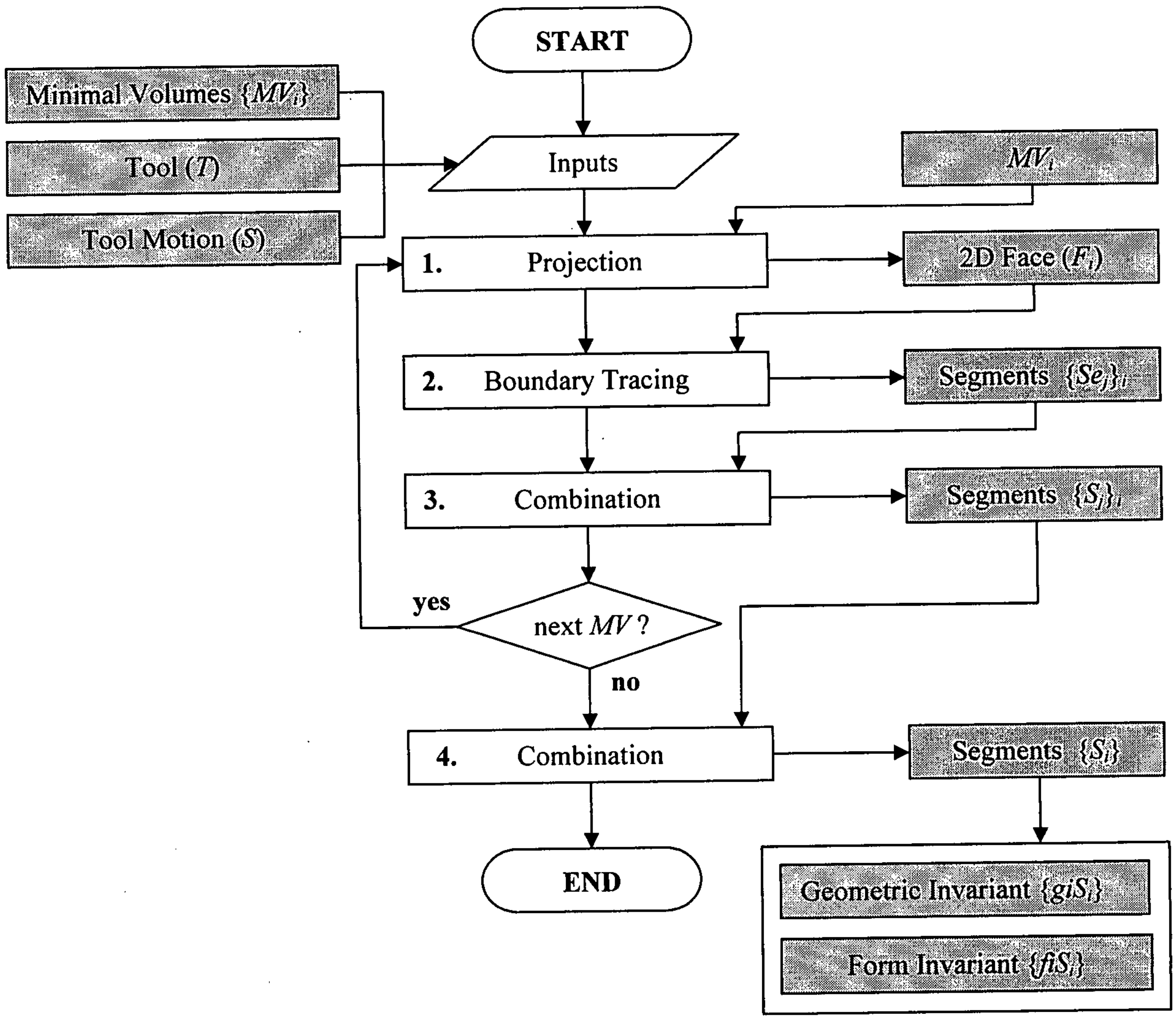


Figure 4-10: A Flowchart of the Segmentation Operator

4.5.2.1 Projection

Both the tool geometry *T* and each minimal volume MV_i are projected onto the XY plane in order to reduce the segmentation calculation to a two-dimensional problem. As shown in Figure 4-11, tool geometry *T* is projected as a circle *C* and MV_i as a planar region F_i bounded by a set of linear and circular edges E_1-E_6 in the XY plane. Therefore, a minimal volume MV_i can be alternatively represented by its axial limits (d_{min}, d_{max}) and projection F_i in the coordinate system: MV_i (d_{min}, d_{max}, F_i,).

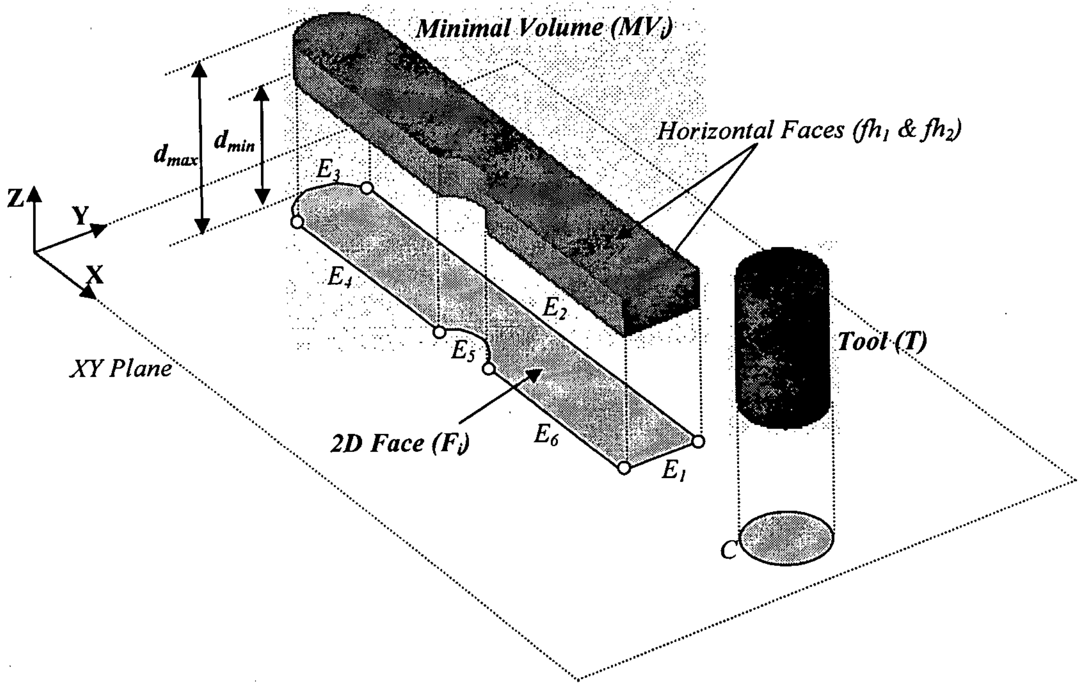


Figure 4-11: Minimal Volume Projection

Details of the algorithm to generate the projection are described in Algorithm 4-2. In summary, the algorithm searches for a horizontal face by traversing the B-rep of each minimal volume. All of the edges on the horizontal face are inserted into an edge list for output. The Z coordinates of vertices on each edge within the edge list are set to zero. The axial limits (d_{min}, d_{max}) can be determined by traversing all of the vertices on the minimal volume to find a minimum and a maximum Z coordinate. As shown in Algorithm 4-2, d_{min} and d_{max} are found by the routines *FindMinimumZCoordinate* and *FindMaximumZCoordinate*, respectively.

```
Routine: to project a minimal volume into a set of edges on the XY plane
Input: a minimal volume $MV$
Output: a list of edges $lstEdge$, the axial limits $(d_{min}, d_{max})$
$\{lstEdge, (d_{min}, d_{max})\}$ = Project ($MV$){
        for ( each face $f_i$ in $MV$ ) {
                if ( $f_i$ is a horizontal plane ) {
                        for ( each edge $E_j$ in $f_i$ ) {
                                $z$ coordinate of the $E_j \leftarrow 0$
                                $lstEdge \leftarrow E_j$
                        }
                        $d_{min}$ = FindMinimumZCoordinate($MV$)
                        $d_{max}$ = FindMaximumZCoordinate($MV$)
                        return
                }
        }
}
```

Algorithm 4-2: Minimal Volume Projection

4.5.2.2 Boundary Tracing

The inputs to the boundary tracing step are a minimal volume projection F_i, a tool projection C and a tool motion S. As mentioned in the projection step, the minimal volume projection F_i consists of a set of edges $\{E_i\}$ in the XY plane. The tool projection C is a circle in the plane. The tool motion S is the toolpath from which the removal volume is generated. The outputs to this step are a set of geometric invariant and form invariant segments. The boundary tracing step can be divided into the following steps:

Step 1. Segment Generation:

A segment is generated by identifying the entry and exit positions where the tool projection C starts and ends the intersection with an edge of the minimal volume projection F_i along the tool motion S.

Step 2. Segment Classification:

The segment generated from the last step is classified as either a geometric invariant segment *giS* or a form invariant segment *fiS*, based on the intersection condition between the tool projection C and the edge of the minimal volume projection F_i.

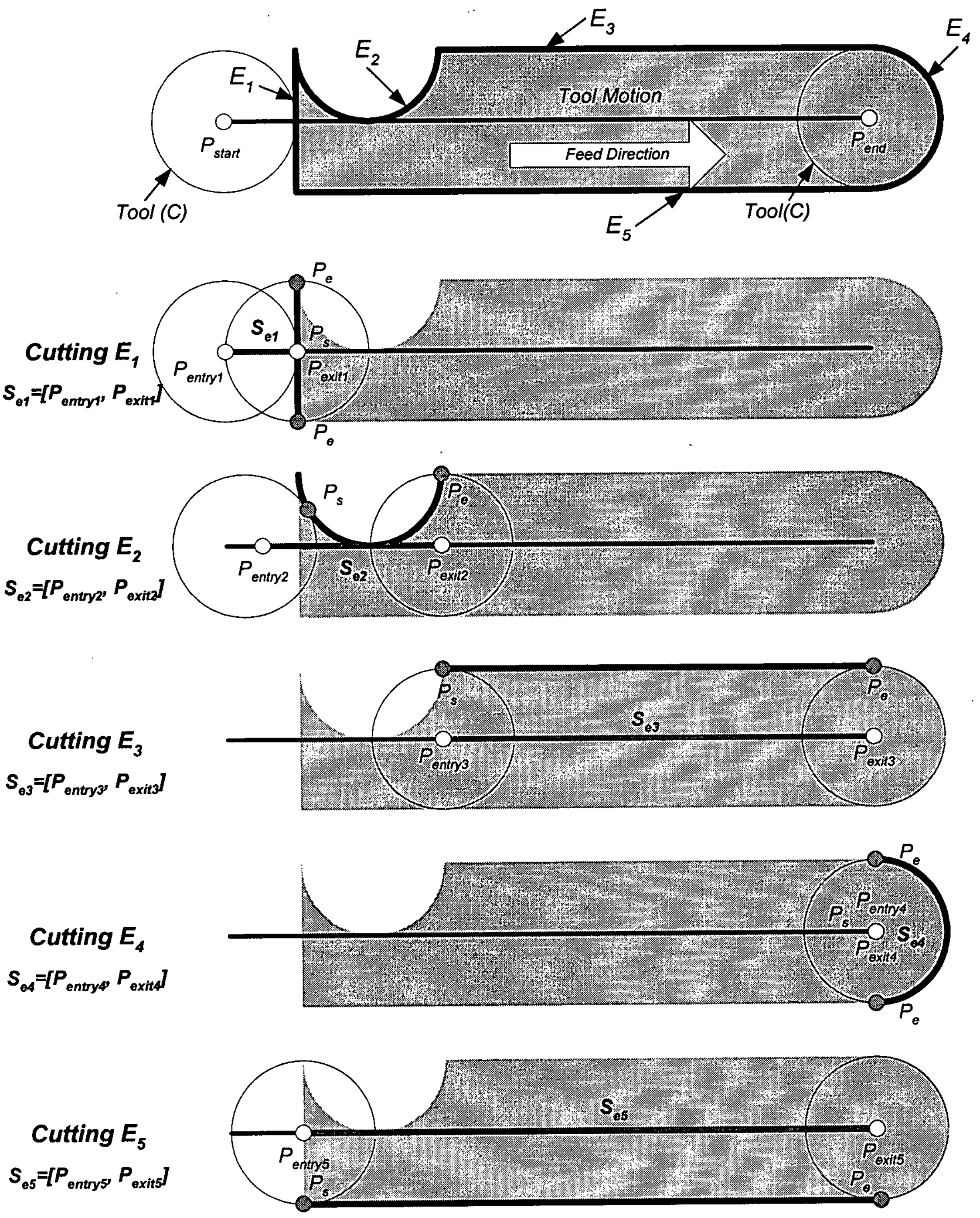


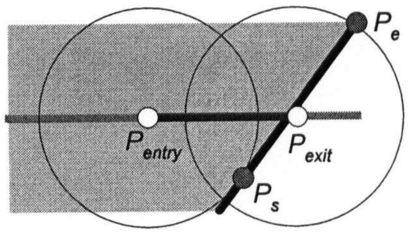
Figure 4-12: Segment Generation

These two steps are described as follows:

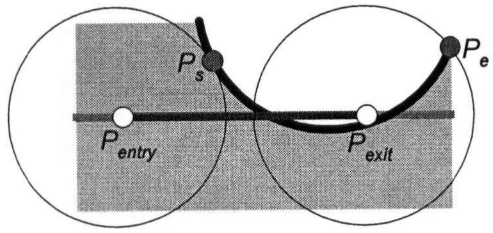
Step 1: Segment Generation

A segment is bounded by the entry and exit positions where the tool projection C starts and ends the intersection with an edge of the minimal volume projection. Figure 4-12 illustrates five segments (S_{e1}, S_{e2}, S_{e3}, S_{e4}, S_{e5}) corresponding to five edges of a minimal volume projection (E_1, E_2, E_3, E_4, E_5). Segment S_{e1} = [P_{entry1}, P_{exit1}] is for the edge E_1, segment S_{e2} = [P_{entry2}, P_{exit2}] is for the edge E_2 and so on. In Figure 4-12, P_s is an intersection point where C starts the intersection with the edge, and P_e is an intersection point where C ends the intersection with the edge.

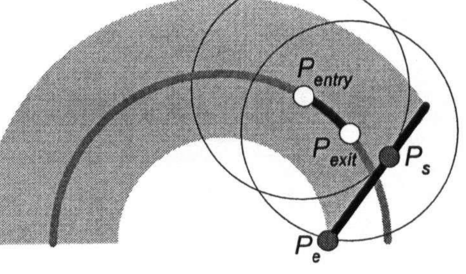
Entry (P_{entry}) and exit (P_{exit}) positions of each segment can be calculated by performing line/arc or arc/arc intersections between the tool projection C and each edge E_i depending on the type of the tool motion and the edge. Figure 4-13 illustrates four cases for calculating entry and exit positions of the segment in 2½D end milling. The tool projection may intersect a linear or a circular edge when it moves along a linear or circular motion. The calculation for each case is described in Appendix A.



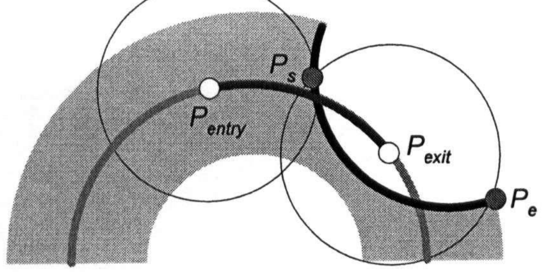
CASE1: Linear Tool Motion/Linear Edge



CASE2: Linear Tool Motion/Circular Edge



CASE3: Circular Tool Motion/Linear Edge



CASE4: Circular Tool Motion/Circular Edge

Figure 4-13: Entry and Exit Positions of the Segment

An algorithm to calculate entry and exit positions of each edge is presented in Algorithm 4-3. It can be seen that entry and exit positions of each edge can be calculated based on the tool motion type and the edge type.

```
Routine: to generate a set of segments by tracing the edges of a minimal volume
projection
Input: the edges of a minimal volume projection (lstEdge), tool projection (C), tool
motion (T)
Output: a list of segments lstS
lstS = GenerateSegments (lstEdge, C, T) {
        for ( each edge $E_i$ in lstEdge ) {
                case ( T is Linear ) {
                        case ( $E_i$ is a Line )
                                S = CalculateLineLine($E_i$, C, T)
                        case ( $E_i$ is a Arc )
                                S = CalculateLineArc($E_i$, C, T)
                }
                case ( T is Circular ) {
                        case ( $E_i$ is a Line )
                                S = CalculateArcLine($E_i$, C, T)
                        case ( $E_i$ is a Arc )
                                S = CalculateArcArc($E_i$, C, T)
                }
                lstS ← S
        }
}
```

Algorithm 4-3: Segment Generation

Step 2: Segment Classification

A set of segments is generated in the last step. Each segment needs to be classified as either a geometric invariant segment or a form invariant segment. A *giS* is considered as a segment where the intersection condition between the tool projection C and the edge E_i is constant. Figure 4-14 illustrates the geometric invariant segments for a linear tool motion and a circular tool motion. It can be seen that the immersion angles on *C* at any two positions (P_1, P_2) along the segment are constant ($\theta_1 = \theta_2$). *giS*s can be determined by identifying that the linear edge E_i is parallel to the linear segment in linear tool motion, and the circular edge E_i has the same centre as the circular segment in circular tool motion. On the other hand, a variant intersection condition between the C and E_i along an initial segment leads to a *fiS*. For the example described in Figure 4-12, the segments can be labeled as in Table 4-1.

	Segments				
	S_{e1}	S_{e2}	S_{e3}	S_{e4}	S_{e5}
Entry & Exit Positions	$[P_{entry1}, P_{exit1}]$	$[P_{entry2}, P_{exit2}]$	$[P_{entry3}, P_{exit3}]$	$[P_{entry4}, P_{exit4}]$	$[P_{entry5}, P_{exit5}]$
Cutting Edge	E_1	E_2	E_3	E_4	E_5
Labels	*fiS*	*fiS*	*giS*	Ø*	*giS*

* S_{e4} is not labeled because its entry and exit positions are the same

Table 4-1: Labeled Segments

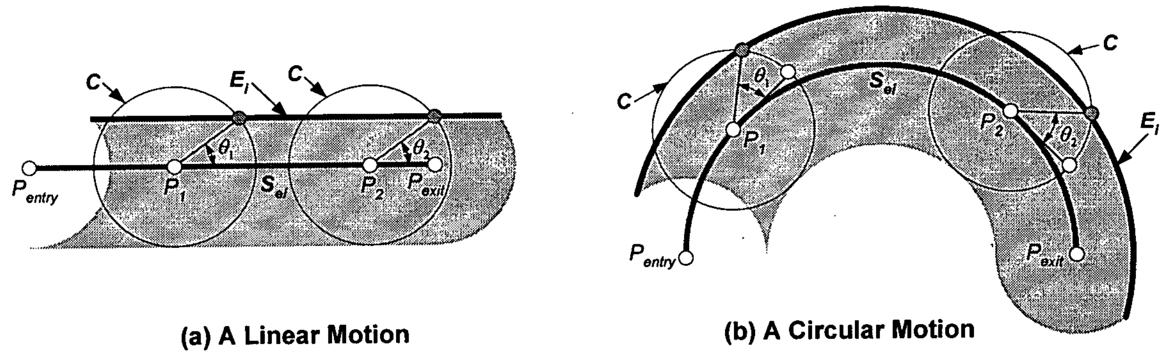


Figure 4-14: Geometric Invariant Segments

An algorithm to classify the segments is presented in Algorithm 4-4. For each segment S_{ei} in the segment list *lstS*$_{ei}$, its associated cutting edge E_i has been checked. In a linear tool motion, if E_i is parallel to S_{ei}, S_{ei} is labeled as a *giS*, otherwise S_{ei} is labeled as a *fiS*. In a circular tool motion, both S_{ei} and E_i are arcs. If S_{ei} and E_i have the same center point, S_{ei} is labeled as a *giS*, otherwise S_{ei} is labeled as a *fiS*.

```
Routine: to classify a set of segments as $fiS$s or $giS$s
Input: a list of segments ($lstS_e$), the associated edges ($lstEdge$)
Output: a list of labeled segments $lstS_e$
$lstS_e$ = ClassifySegments ($lstS_e$, $lstEdge$) {
        for ( each segment $S_{ei}$ in $lstS_e$ ) {
                case ( $S_{ei}$ is Linear ) {
                        if ( $E_i$ in $lstEdge$ is parallel to $S_{ei}$ )
                                $S_{ei} \leftarrow giS$
                        else
                                $S_{ei} \leftarrow fiS$
                }
                case ($S_{ei}$ is Circular ) {
                        if ($E_i$ in $lstEdge$ has the same center point as $S_{ei}$)
                                $S_{ei} \leftarrow giS$
                        else
                                $S_{ei} \leftarrow fiS$
                }
        }
}
```

Algorithm 4-4: Segment Classification

4.5.2.3 Segment Combination for One *MV*

A set of labeled segments $\{S_{ei}\}$ is created in the previous steps. Among these segments, there may be some overlap. By combining these overlapping segments, a set of new segments $\{S_i\}$ is generated. These new segments will be classified based on the label of each overlapping segment S_{ei} and the following combination rules:

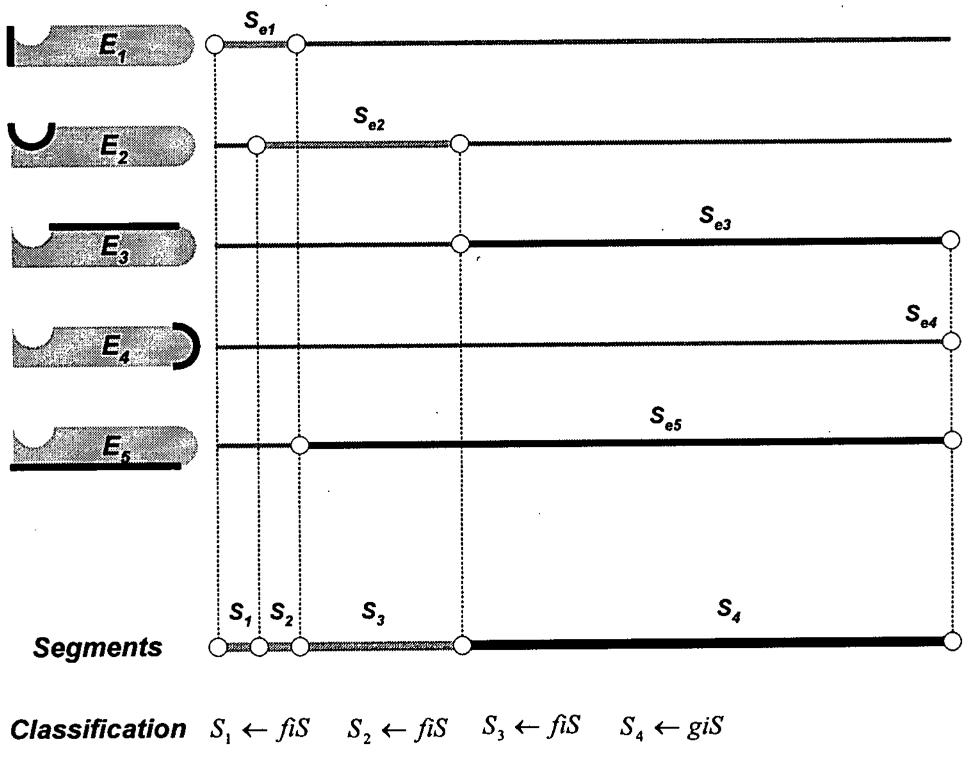
Rule 1. Overlapping geometric invariant segments produce a new geometric invariant segment leading to a geometric invariant feature.

Rule 2. Overlapping form invariant segments produce a new form invariant segment leading to a form invariant feature.

Rule 3. Overlapping geometric invariant and form invariant segments produce a new form invariant segment leading to a form invariant feature.

Figure 4-15 illustrates the combination based on the example described in Figure 4-12. From Figure 4-15, it can be seen that segment S_2 is extracted from overlapping segments (S_{e1},

S_{e2}), segment S_3 from overlapping segments (S_{e2}, S_{e5}), and segment S_4 from overlapping segments (S_{e3}, S_{e5}). By applying the rules above, S_2 is labeled as a *fiS* since both S_{e1} and S_{e2} are *fiS*s (see the second rule), S_3 is labeled as a *fiS* since S_{e2} is a *fiS* and S_{e5} is a *giS* (see the third rule), and S_4 is labeled as a *giS* since both S_{e3} and S_{e5} are *giS*s (see the first rule). In this example, the output contains four segments (S_1, S_2, S_3, S_4}, where S_4 is a *giS*, and S_1, S_2, S_3 are *fiS*s.



S_{e1}
E_1
S_{e2}
E_2
S_{e3}
E_3
S_{e4}
E_4
S_{e5}
E_5
S_1
S_2
S_3
S_4
Segments
Classification $S_1 \leftarrow fiS$ $S_2 \leftarrow fiS$ $S_3 \leftarrow fiS$ $S_4 \leftarrow giS$

Figure 4-15: Segment Combination for One *MV*

An algorithm to combine these overlapping segments in the segment list $lstS_e$ is presented in Algorithm 4-5. This algorithm extracts entry and exit points from each segment S_{ei} in $lstS_e$ and inserts them into a point list *lstPoint*. The duplicated points in *lstPoint* are removed. The point list *lstPoint* is then sorted in ascending order by the distance from each point to the start point (see P_{start} in Figure 4-12) of the tool motion. A new segment set *lstS* can be generated by extracting the adjacent point pairs such as [P_1, P_2], [P_2, P_3] and so on from the point list *lstPoint*. To classify these new segments, each new segment S_i in the segment list *lstS* is examined to see whether it intersects with each segment S_{ei} in the list $lstS_e$.

If multiple intersected segments *lstInterS*$_e$ are found, the rules above are applied to determine the classification of the new segment S_i.

```
Routine: to combine a set of overlapping segments
Input: a list of overlapping segments ($lstS_e$)
Output: a list of segments $lstS$
$lstS$ = CombineSegments ($lstS_e$) {
        ALLOCATE a point list: $lstPoint$
        for ( each segment $S_{ei}$ in $lstS_e$ ) {
                $lstPoint \leftarrow$ EntryPointOf($S_{ei}$)
                $lstPoint \leftarrow$ ExitPointOf($S_{ei}$)
        }
        REMOVE duplicated points in $lstPoint$
        SORT $lstPoint$ in an ascending order by the distance $|P_iP_{start}|$
        for ( each pair $[P_i, P_{i+1}]$ in $lstPoint$ ) {
                $S_i = [P_i, P_{i+1}]$
                $lstS \leftarrow S_i$
        }
        ALLOCATE an intersected segment list: $lstInterS_e$
        for ( each segment $S_i$ in $lstS$ ) {
                $lstInterS_e \leftarrow$ FindIntersectSegments($S_i$, $lstS_e$)
                if ( each intersected segment $S_{ei}$ in $lstInterS_e$ is $giS$ )
                        $S_i \leftarrow giS$
                else
                        $S_i \leftarrow fiS$
        }
}
```

Algorithm 4-5: Segment Combination

4.5.2.4 Segment Combination for Overlapping *MV*s

Since the decomposition operator described in the last section can generate overlapping minimal volumes, the final classification of regions of the removal volume into geometric or form invariant types depends on the length and types of the segments for the overlapping minimal volumes. The combination rules and the algorithm used for overlapping *MV*s are the same as those used for one *MV*.

As shown in Figure 4-16, a removal volume is decomposed into two overlapping minimal volumes, with which labeled segments ($fiS_{1,1}$, $giS_{1,1}$, $fiS_{1,2}$, $giS_{1,2}$) and ($fiS_{2,1}$, $giS_{2,1}$)

are associated. By applying the combination rules above it can be seen that overlap of $fiS_{1,1}$ and $fiS_{2,1}$ leads to the form invariant segment fiS_1, overlap of $giS_{1,1}$ and $giS_{2,1}$ leads to the geometric invariant segment giS_2, overlap of $fiS_{1,2}$ and $giS_{2,1}$ leads to the form invariant segment fiS_2 and overlap of $giS_{1,2}$ and $giS_{2,1}$ leads to the geometric invariant segment giS_2.

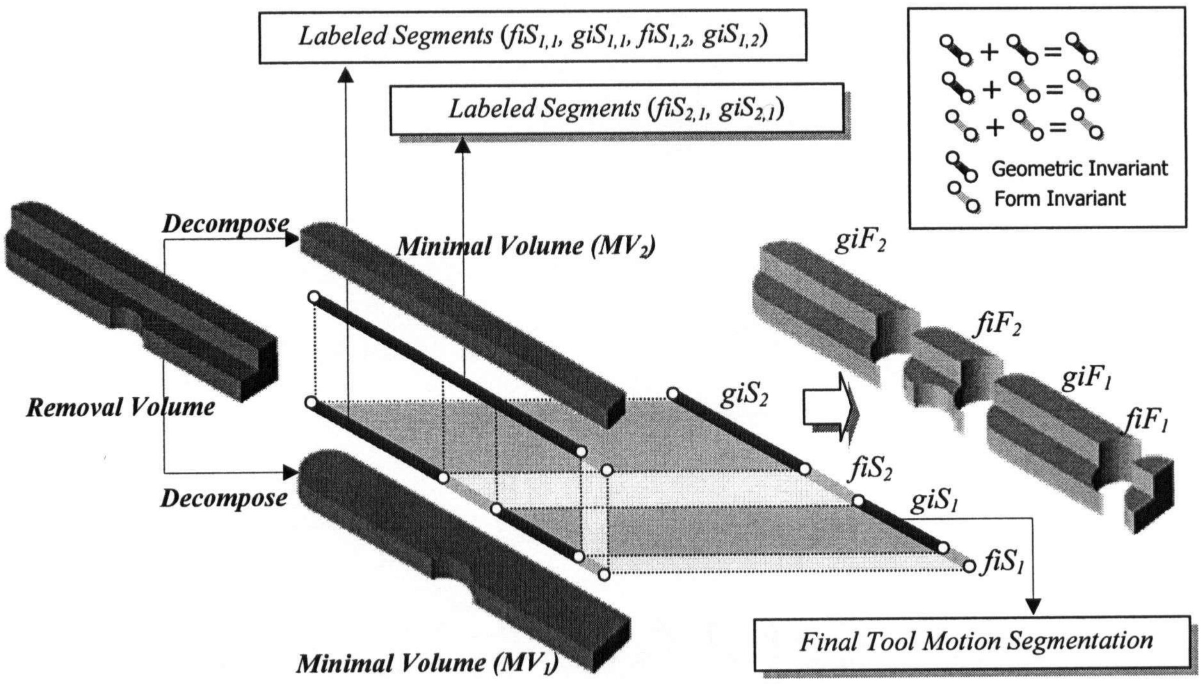


Figure 4-16: Segment Combination for Overlapping *MV*s

In the segmentation operator, the tool motion is divided into a set of Geometric Invariant and Form Invariant Segments based on the intersection condition between the tool and the removal volume. In the following section, an extraction operator will be introduced to extract Geometric Invariant and Form Invariant Features by decomposing the removal volume along the *giS*s and *fiS*s.

4.5.3 Extraction Operator

The extraction operator is to extract a set of $\{giF_i, fiF_i\}$ by decomposing a removal volume RV according to the labeled segment set $\{giS_i, fiS_i\}$ generated in the previous steps. This operator can be represented as $\{giF_i, fiF_i\}=Extract(RV, \{giS_i, fiS_i\})$, where the inputs are a removal volume RV and a set of Geometric Invariant and Form Invariant Segments $\{giS_i,$

fiS$_i$}. The outputs are a set of Geometric Invariant and Form Invariant Machining Features {*giF$_i$*, *fiF$_i$*}.

Figure 4-17 illustrates an algorithm for extracting *giF*s and *fiF*s from a removal volume. It can be seen that a semi-cylinder surface halfspace h_i is generated at each internal boundary point P_i of the labeled segment set, starting from the first point P_{start}. The halfspace h_i is used to split the removal volume into two parts. As shown in Figure 4-17, the first part is the feature *fiF$_1$*, which is generated by performing a Boolean intersection between the halfspace h_1 and the removal volume RV ($RV \cap^* h_1$). The second part is created by subtracting the halfspace h_1 from the removal volume RV ($RV -^* h_1$). The second part will be split by the next halfspace h_2 to extract another feature, if possible. Similar to the splitting operation in the decomposition operator, the ACIS routine *api_boolean_chop_body* can be used to perform these two Boolean operations at the same time in this operator.

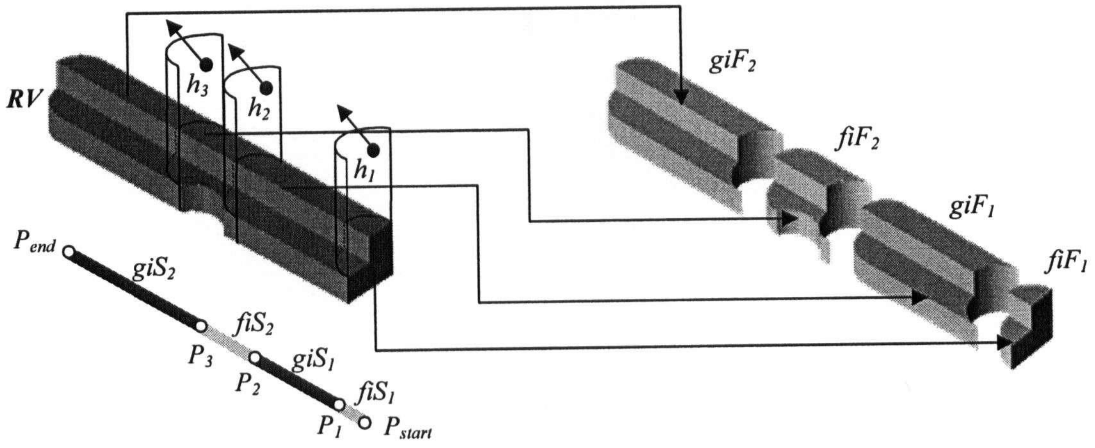


Figure 4-17: Feature Extraction from a Removal Volume

The feature extraction algorithm is described in Algorithm 4-6. A semi-cylinder halfspace h_i is generated at the internal boundary point P_i of the labeled segment set *lstS*. The removal volume RV is split by the halfspaces h_i. The decomposed volume F is inserted into *lstGIF* if the corresponding segment S_i is a *giS*. Otherwise, the volume F is a *fiF* and inserted into *lstFIF*.

```
Routine: to decompose a removal volume into a set of giFs and fiFs
Input: a removal volume RV, a list of segments lstS
Output: a list of giFs lstGIF, a list of fiFs lstFIF
{lstGIF, lstFIF} = Extract (RV, lstS) {
        for (each segment $S_i$ in lstS) {
                $P_i$ = ExitPointOf($S_i$)
                CONSTRUCT a halfspace $H_i$ at $P_i$
                SPLIT RV into a volume F with $H_i$
                if ( $S_i$ is a giF ) {
                        lstGIF ← F
                } else if ( $S_i$ is a fiF ) {
                        lstFIF ← F
                }
        }
}
```

Algorithm 4-6: Feature Extraction

After recognition of Geometric Invariant and Form Invariant Features, Cutter/Workpiece Engagements can be analytically extracted from *giF*s and *fiF*s. In the following section, a *CWE* extraction algorithm will be presented to do this.

4.6 Extraction of Cutter/Workpiece Engagement

A *giF* or *fiF* extracted from the feature recognition step may contain several sub-volumes originated from minimal volumes. Each sub-volume is represented by its position in the Z direction and projection in the XY plane. The representation of the *giF* or *fiF* is a union of each sub-volume representation. As shown in Figure 4-18, a *fiF* is composed of sub-volume V_1 and V_2. The upper volume V_1 can be represented by its position in the Z direction (d_{min1}, d_{max1}) and projection (F_1) in the coordinate system: V_1 (d_{min1}, d_{max1}, F_1). The lower volume can be similarly represented: V_2 (d_{min2}, d_{max2}, F_2). Therefore, the *fiF* is represented as (d_{min1}, d_{max1}, F_1) ∪ (d_{min2}, d_{max2}, F_2).

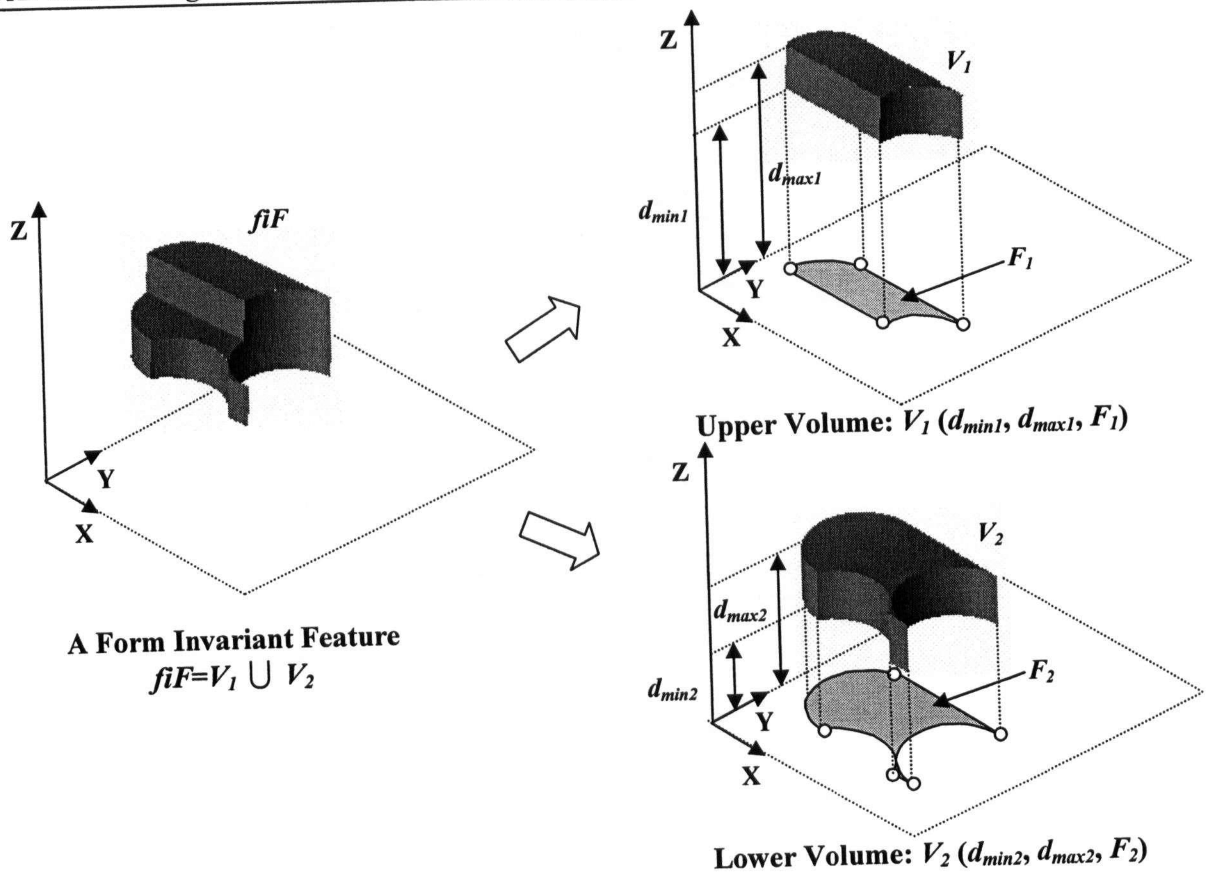

Z
Y
X
fiF
A Form Invariant Feature
fiF=V1 ∪ V2
V1
dmax1
dmin1
F1
Upper Volume: V1 (dmin1, dmax1, F1)
V2
dmax2
dmin2
F2
Lower Volume: V2 (dmin2, dmax2, F2)

Figure 4-18: Representation of a *fiF*

In 2½D end milling, a *CWE* in a *fiF* or *giF* is composed of engagements between the tool and each sub-volume. These engagements can be constructed by entry and exit angles and the depth of cut in the Z direction. Entry and exit angles are determined by calculating the intersection between the tool projection and the volume projection. As shown in Figure 4-19, the upper volume projection F_1 intersects with the tool projection C to determine the entry and exit angles. Entry and exit angles are calculated by finding valid intersection points between C and edges (E_1, E_2) of F_1. The entry and exit angle $[\phi_{st1}, \phi_{ex1}]$, along with the depth of cut $[d_{min1}, d_{max1}]$, are used to construct the engagement zone: $Zone_1[d_{min1}, d_{max1}, \phi_{st1}, \phi_{ex1}]$. $Zone_2[d_{min2}, d_{max2}, \phi_{st2}, \phi_{ex2}]$ is constructed from the lower volume V_2. Both $Zone_1$ and $Zone_2$ are combined into a *ceF* for the *fiF* at the cutter location P_u.

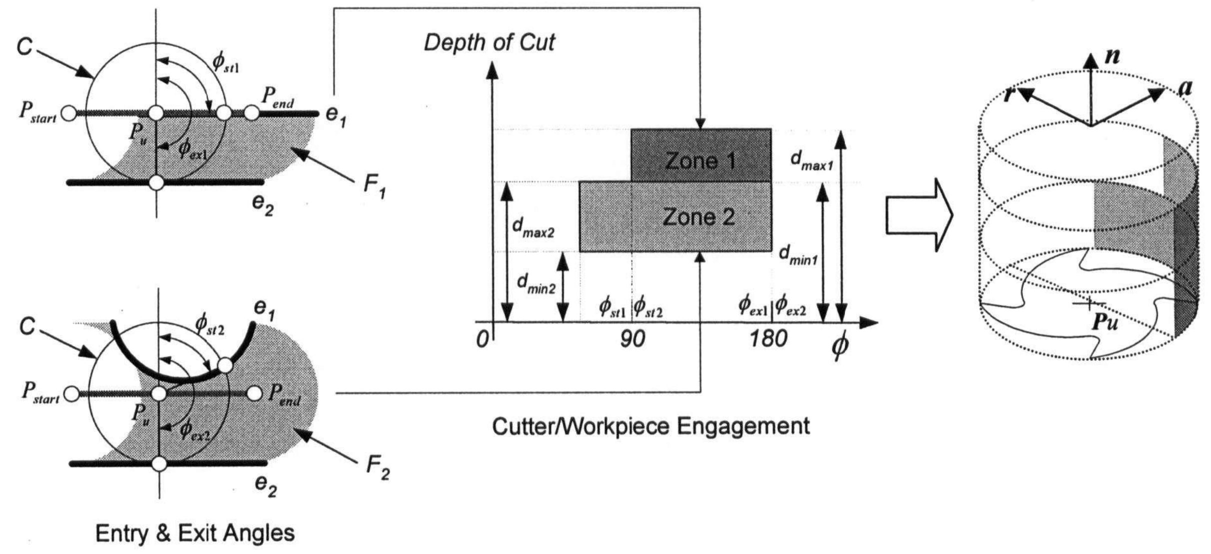


Figure 4-19: *CWE* Extraction from a *fiF*

Generally, the *CWE* extraction algorithm is described in Figure 4-20. This algorithm includes the following steps:

Step 1. Calculate entry and exit angles $[\phi_{sti}, \phi_{exi}]$ for a sub-volume V_i by intersecting the tool project *C* with edges of the sub-volume projection F_i.

Step 2. Construct an engagement zone $Zone_i$ for the sub-volume V_i based on entry & exit angles $[\phi_{sti}, \phi_{exi}]$ and the depth of cut $[d_{mini}, d_{maxi}]$. The engagement zone is represented as $Zone_i[d_{mini}, d_{maxi}, \phi_{sti}, \phi_{exi}]$.

Step 3. Combine the engagement zones $\{Zone_i\}$ into a *ceF* for *giF* or *fiF*.

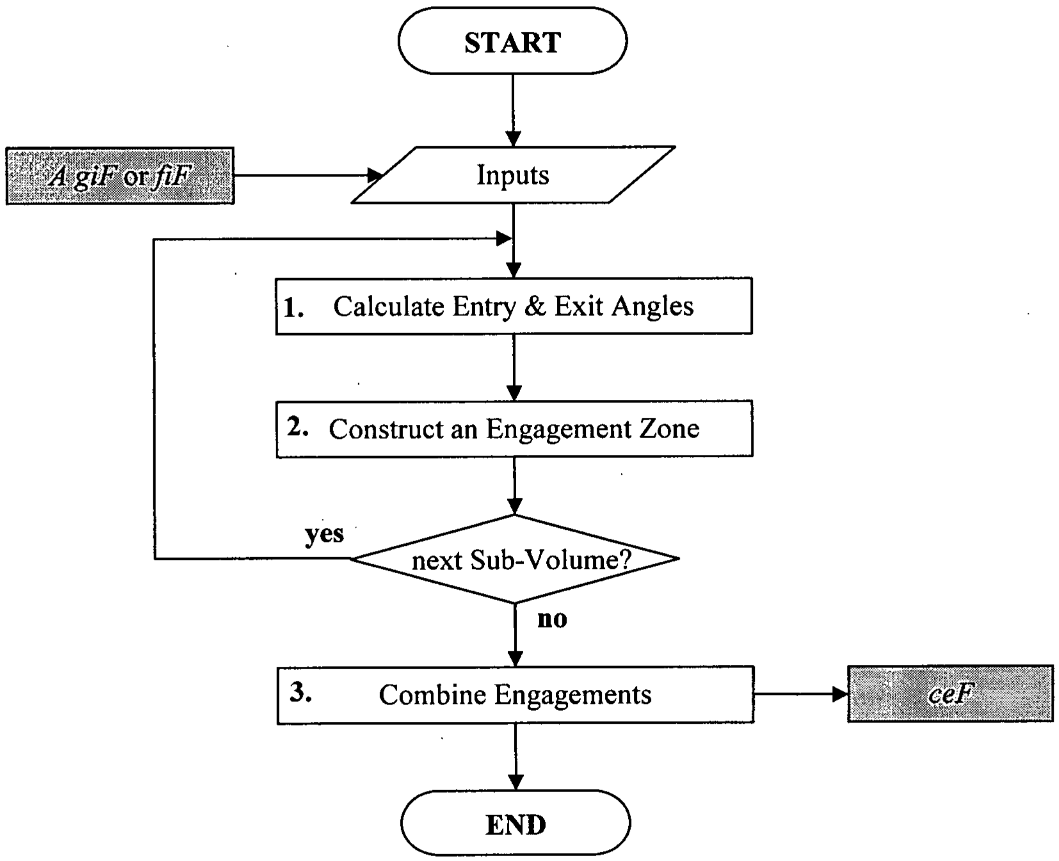


Figure 4-20: Steps in *CWE* Extraction

4.7 Implementation and Validation

4.7.1 Implementation

The feature recognition algorithm and *CWE* extraction from *giF*s and *fiF*s algorithm described in this chapter were implemented within the Microsoft Windows XP professional edition operating system. Matlab and Visual C++ were used as the development tool. This implementation of feature recognition uses the ACIS 3D modeling kernel as the solid modeling engine, and the HOOP 3dGS as the graphics system. The implementation of *CWE* extraction uses Matlab to calculate and display engagement angles and engagement zones. This program was tested using a computer with a Pentium 4 processor, with 3GHz/0.99GB.

Figure 4-21 illustrates major classes and their relationships in the *CWE* extraction system. Class *GeometricInvariantFeature* and *FormInvariantFeature* represent *giF* and *fiF* respectively. Class *Segment* within the above classes is used to stand for the corresponding *giS* and *fiS*. In the implementation of the decomposition and extraction operator of the in-process feature recognition step, limited single-sided open shell Booleans in ACIS are used to perform Boolean operations between the halfspace and the solid model of the removal volume. The ACIS routine *api_boolean_chop_body* is applied for the decomposition.

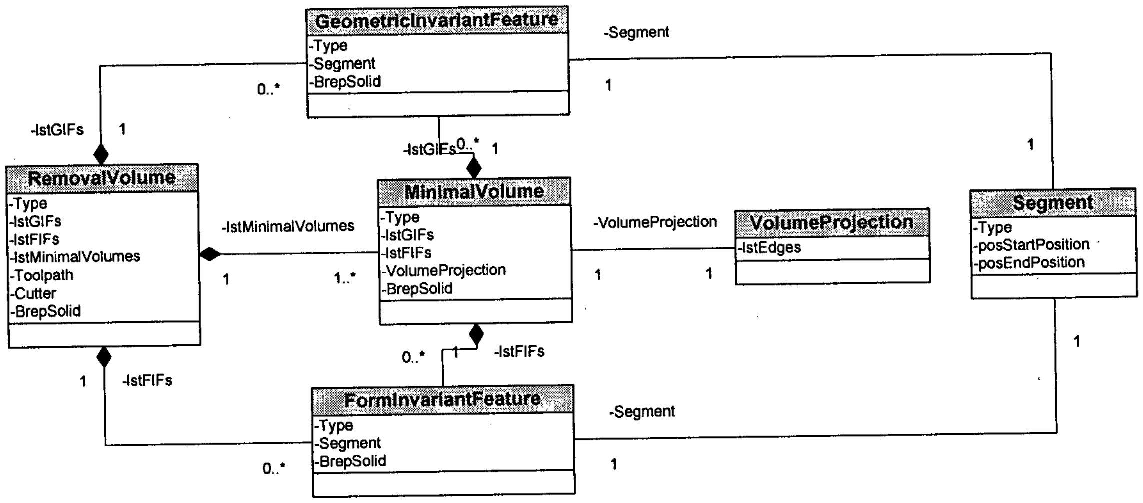
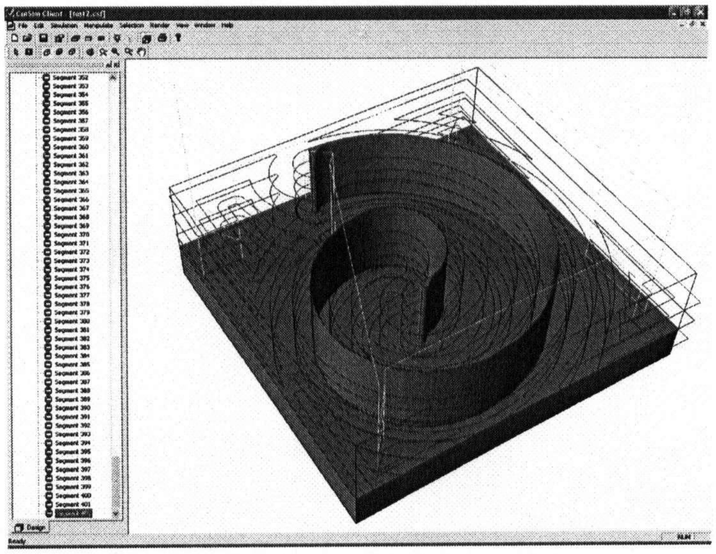


Figure 4-21: Class Diagram in the *CWE* Extraction System

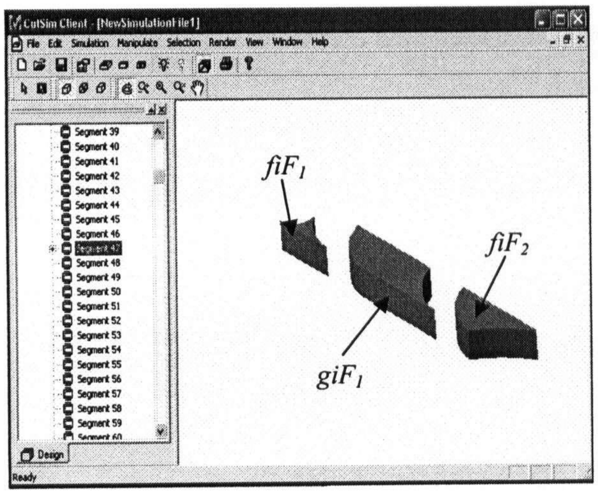
4.7.2 Validation of Feature Recognition

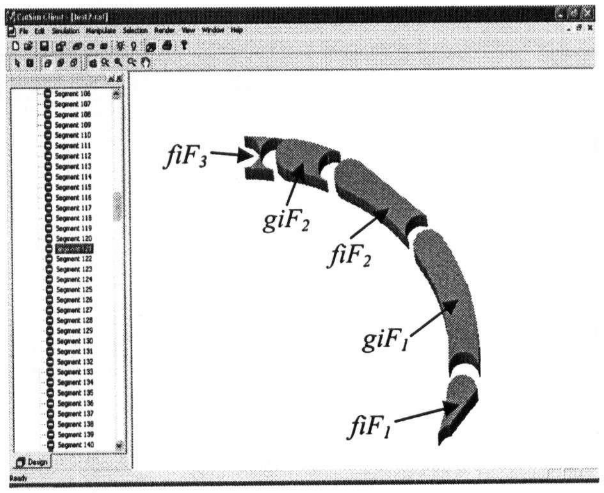
Figure 4-22 illustrates a test part for validating the feature recognition algorithm presented in this chapter. The geometry of the test part and toolpath are presented in Figure 4-22 (a). There are a total of 410 tool motions in the toolpath. 301 out of 410 tool motions are valid tool motions that cut the in-process workpiece. Table 4-2 summarizes the computation times taken to complete the feature recognition for all of the removal volumes. The number of *giF*s and *fiF*s involved is also included.

Figure 4-22 (b) and (c) show the results of feature recognition for the removal volumes generated by the 47th (linear) and the 121st (circular) tool motions. It can be seen that the linear removal volume is decomposed into 1 *giF* and 2 *fiF*s, and the circular removal volume into 2 *giF*s and 3 *fiF*s.



(a) A Test Part





(b) A Linear Tool Motion **(c) A Circular Tool Motion**

Figure 4-22: Examples for Feature Recognition

Inputs		
Tool Motions	**Removal Volumes**	
410	301	
Machining Features		
***giF*s**	***fiF*s**	
326	457	
Times (seconds)		
Removal volume	**Feature Recognition**	**Total**
40.248	4.183	44.431

Table 4-2: Simulation Times for Feature Recognition

4.7.3 Validation of *CWE* Extraction

Figure 4-23 illustrates an example of *CWE* extraction from a *fiF*. The test part is machined by 117 linear tool motions. The removal volume generated by the 95th tool motion is decomposed into several *giF*s and *fiF*s. A Form Invariant Feature fiF_1 is selected for the *CWE* extraction algorithm implemented by the Matlab software. fiF_1 is composed of three minimal volumes. Figure 4-23 (c) shows the engagement angle calculation by Matlab for each minimal volume of fiF_1. Figure 4-23 (d) illustrates the final *ceF* at the cutter location u=0.2.

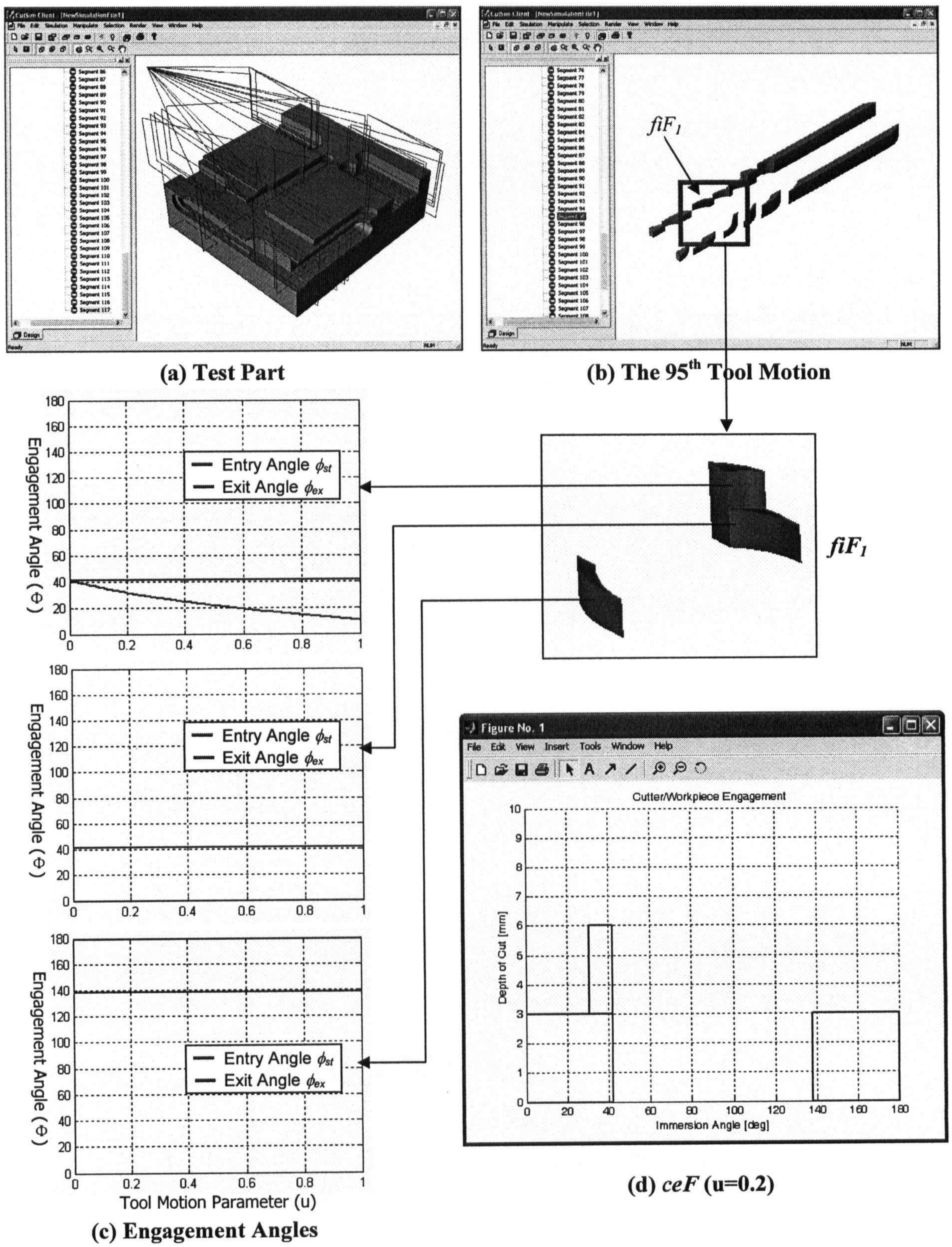


Figure 4-23: An Example of *CWE* Extraction

4.8 Discussion

The feature recognition algorithm is verified based on the removal volumes generated by both the linear and circular tool motions. As shown Figure 4-22, these two types of removal volume can be correctly decomposed into *giF*s and *fiF*s. From the results presented in Table 4-2, it can be seen that feature recognition has been done without any computational problem for this intermediary complicated part. Robustness and computational efficiency of this algorithm has been verified. In addition, *CWE* extraction from a *giF* or *fiF* has been tested as shown in Figure 4-23. From Figure 4-23 (c), it can be seen that the entry and exit engagement angles can be calculated analytically. Therefore, it is clear that an exact solution to *CWE* is provided by this approach.

However, this methodology is based on the assumption that a rectangular prismatic initial workpiece is machined by a flat end-mill in 2½D end milling. It is possible to extend this methodology to various end-mills, such as ball, cone and bull end-mills if analytical quadric surface/surface intersection calculations are used for the segmentation operator, rather than the calculation among the two-dimensional geometries presented in this algorithm. This methodology can be also extended to different machining operations, such as turning and boring.

Chapter 5
Cutter/Workpiece Engagement Extraction for Hole Milling

5.1 Introduction

In process planning, a hole is considered as a final part machining feature. To machine a hole feature, two types of machining operation, drilling and milling, can be planned. Since hole milling has a larger material removal rate than hole drilling, a milling operation is normally selected for large-size hole machining during process planning. In a milling operation, a hole is generated by moving an end mill along a helical toolpath. Since the radius of the end mill is greater than the radius of the helical toolpath, the swept volume of the end mill has self-intersections, which leads to the following two difficulties in the areas of geometric verification and process modeling:

- In geometric verification, swept volumes are needed for updating the in-process workpiece. The self-intersection in hole milling operations makes swept volume generation difficult.
- In process modeling, *CWE*s are needed for predicting cutting forces. Traditional *CWE* extraction approaches apply Surface/Surface Intersection between the cutting tool and the removal volume. Since a self-intersected swept volume loses the boundary geometry of common regions, the Surface/Surface Intersection approach cannot extract the correct *CWE*s when the cutting tool engages these common regions.

In this chapter, an analytical approach to *CWE* extraction for hole milling with a flat end-mill is discussed. This approach provides a closed-form solution without numeric Surface/Surface Intersection calculations. In the next chapter, a self-intersected swept volume generation approach is investigated for in-process workpiece modeling. This approach can generate an exact representation for the self-intersected swept volume for a variety of end-mills with helical toolpaths.

A definition of a milled hole machining feature will be given in the next section, followed by an overview of its parametrization. This parametrization provides a closed-form representation for the *ceF*. Finally, examples will be presented to compare the *ceF*s extracted using the analytical approach developed with those generated by the commercial software NC application VERICUT.

5.2 Milled Hole Machining Feature

A *Milled Hole Machining Feature* can be defined as follows:

> *A Milled Hole Machining Feature (mhF) is a cylindrical surface with an optional planar surface on the final part generated using an end mill moving along a helical toolpath.*

As shown in Figure 5-1, a *mhF* is machined by a flat end mill moving along a helical toolpath. The cylindrical hole geometry is defined by the following parameters:

R: Radius of the hole

H: Height of the hole

The flat end-mill is defined by the following parameters:

r: Radius of the cutting tool

h: Height of the cutting tool

The helical toolpath is defined by the following parameters:

R-r: Radius of the helical toolpath

p: Pitch of the helical toolpath

In addition, the following two variables are used in defining the *CWE*:

ϕ: Engagement angle of the cutting tool $\phi \in [0, 2\pi]$.

θ: Rotation angle of the cutting tool $\theta \in [\theta_s, \theta_e]$, where θ_s and θ_e are the start and end rotation angles of the cutting tool to finish machining a hole. $\theta_s = 0$ and θ_e can be calculated by Eq. (5.1):

$$\theta_e = 2\pi\left(\frac{H}{p}+1\right) \tag{5.1}$$

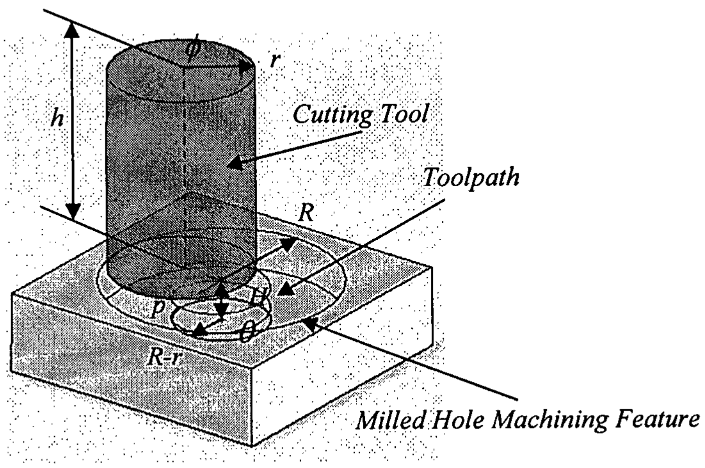


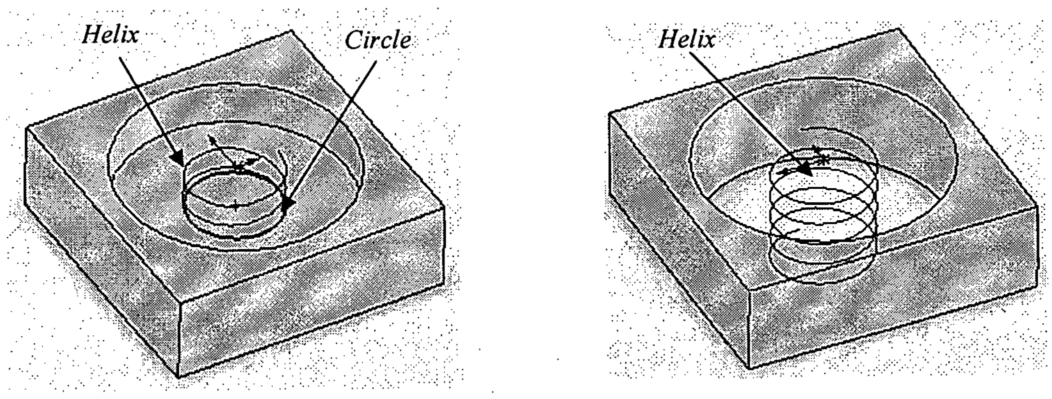
Figure 5-1: Milled Hole Machining Feature

As illustrated in Figure 5-2, there are two types of *mhF*: Blind Hole and Through Hole. The toolpaths to machine these two types of hole are described as follows:

- Blind Hole: The toolpath for a blind hole is composed of two curves: a helix and a circle. The rotation angle range $[\theta_s, \theta_e]$ is divided by a rotation angle θ_b where the cutting tool reaches the bottom surface of the hole. The toolpath is a helix where $\theta \in [\theta_s, \theta_b)$, and a circle where $\theta \in [\theta_b, \theta_e]$.
- Through Hole: The toolpath of a through hole only contains a helix. θ_b is the rotation angle where the cutting tool reaches the bottom surface of the workpiece.

For the two types of hole described above, the rotation angle θ_b can be calculated as follows:

$$\theta_b = 2\pi \frac{H}{p} \tag{5.2}$$



(a) Blind Hole **(b) Through Hole**

Figure 5-2: Toolpaths for Blind Hole and Through Hole

5.3 Parametrization of Cutter Engagement Feature for a Hole

As described in Chapter 3, a Cutter Engagement Feature (*ceF*) is defined to describe the standard engagement format required by the force model. Mathematically, a *ceF* can be represented by its limits on the axial depth of cut [d_{min}, d_{max}] on the surface of the cutting tool.

For blind hole machining, the lower engagement limit d_{min} is always zero, and the upper engagement limit d_{max} is represented as a function of the parameters defined in the last section as follows:

$$\begin{cases} d_{\min} = 0 \\ d_{\max} = f(R, r, p, \theta, \phi) \end{cases} \tag{5.3}$$

For through hole machining, d_{min} is divided into two portions. d_{min} is zero before the cutting tool reaches the bottom surface of the workpiece $\theta \in [\theta_s, \theta_b)$. For the segment of the helical tool path after reaching the bottom where $\theta \in [\theta_b, \theta_e]$, d_{min} is a function of the rotation angle θ. d_{max} is represented as a function of the parameters (R, r, p, θ, ϕ). Both d_{min} and d_{max} are defined as:

$$\begin{cases} d_{\min} = 0 & \theta_s \le \theta < \theta_b \\ d_{\min} = f(\theta) & \theta_b \le \theta \le \theta_e \end{cases}, \quad d_{\max} = f(R, r, p, \theta, \phi) \tag{5.4}$$

From Eq. (5.3) and (5.4), it can be seen that the parametric representation of the *ceF*s depends on the parameters of the hole geometry, the tool geometry and the toolpath. In other words, the *ceF* can be determined by the hole, the tool and the toolpath parameters. This is significant, since the *ceF*s can be obtained from these equations for any size hole, any size tool and different toolpath parameters.

5.3.1 Intersection between Cylinder and Helicoid

A *CWE* is a region on the surface of the cutting tool that engages the in-process workpiece. A *CWE* is bounded by a set of curves originating from the intersection curves between the surfaces of the cutting tool and the surfaces on the workpiece. In the case of hole milling, a helicoid is generated on the in-process workpiece by the flat bottom of the cutting tool as it moves along the helix. To find the *CWE*, the intersection curve between this helicoid and the cylindrical surface of the flat end mill needs to be determined. Figure 5-3 illustrates the intersection between a cylinder representing the end mill and a helicoid.

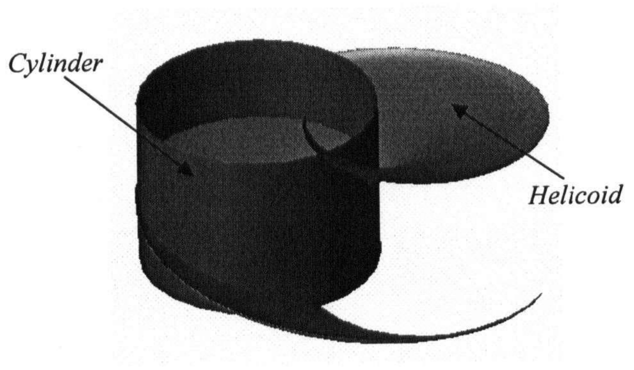


Figure 5-3: Intersection between a Cylinder and a Helicoid

A cylinder with height *p* and radius *r* at any position along a helix can be parametrically described in Eq. (5.5).

$$\begin{cases} x = (R-r)\cos\theta + r\cos\phi \\ y = (R-r)\sin\theta + r\sin\phi \\ z = z \end{cases} \tag{5.5}$$

where $z \in [-p, 0]$, $\theta \in [0,2\pi]$ and $\phi \in [0,2\pi]$.

A helicoid with a semi-circle profile and pitch p is given by Eq. (5.6) in explicit form. The derivation of Eq. (5.6) is presented in Appendix B.

$$z = -\frac{p}{2\pi}\left(\tan^{-1}\left(\frac{y}{x}\right) + \cos^{-1}\left(\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}}\right)\right) \tag{5.6}$$

By manipulating Eq. (5.6) and Eq. (5.5), a parametric form of the cylinder/helicoid intersection is obtained using only the parameters identified in Section 5.2. As a result, Eq. (5.7) is obtained as the equation of the intersection curve.

$$\begin{cases} x = (R-r)\cos\theta + r\cos\phi \\ y = (R-r)\sin\theta + r\sin\phi \\ z = -\frac{p}{2\pi}(\alpha+\beta) \end{cases} \tag{5.7}$$

where:

$$\alpha = \tan^{-1}\left[\frac{(R-r)\sin\theta + r\sin\phi}{(R-r)\cos\theta + r\cos\phi}\right]$$

$$\beta = \cos^{-1}\left[\frac{(R-r) + r\cos(\phi-\theta)}{\sqrt{(R-r)^2 + 2r(R-r)\cos(\phi-\theta) + r^2}}\right]$$

$$\alpha + \beta \in [0, 2\pi)$$

The z coordinate expression in Eq. (5.7) is used to represent the upper limit of the *ceF*, i.e.

$$d_{\max} = z = -\frac{p}{2\pi}(\alpha+\beta) \tag{5.8}$$

Figure 5-4 illustrates an example of Eq. (5.8), where the cylinder representing the end mill is located at several positions along the helix (θ = 0°, 60°, 120°, 180°, 240°, 300°). It can be seen that Eq. (5.8) describes the intersection curve between the cylinder and the helicoid. It should be noted that this equation represents the full intersection curve. In hole milling, the material behind the cutter has been removed. Hence, to find the true *CWE* between the flat end-mill and the in-process workpiece during a hole milling operation, some constraints have to be applied to the Eq. (5.8). The following are three constraints that must be satisfied:

Constraint 1. For a flat end-mill, the valid engagement range is always located on the front semi-cylinder of the cutting tool. This engagement range is changing when the cutting tool is rotating along the helical toolpath. Therefore, the engagement range at any cutter location has to be identified for extracting the valid portion from the intersection curves shown in Figure 5-4.

Constraint 2. When moving along the first or last turn of the helical toolpath, the cutting tool intersects with both the helicoid and the top or bottom plane of the workpiece. The upper limit of the *CWE* is in these cases a composite curve with two segments: one is the intersection curve between the cutting tool and the helicoid, another is the intersection curve between the cutting tool and the top or bottom plane. The intersection point of these two segments needs to be identified in order to construct the complete *CWE* when the tool is moving along the first or last turns of the helix.

Constraint 3. Eq. (5.8) represents the intersection curve between a helicoid and a cylinder with a height p (the pitch of the helix). The curve within valid engagement range is disconnected at the intersection point where the bottom edge of the cylinder intersects with the helicoid. To construct a correct *CWE* from the intersection curve, the curve within valid engagement range must be connected by shifting the segment a pitch distance p upward, since the intersection curve is periodic.

The above three constraints will be addressed in the following sections. Eq. (5.8) will be modified to represent the *CWE* based on these two constraints.

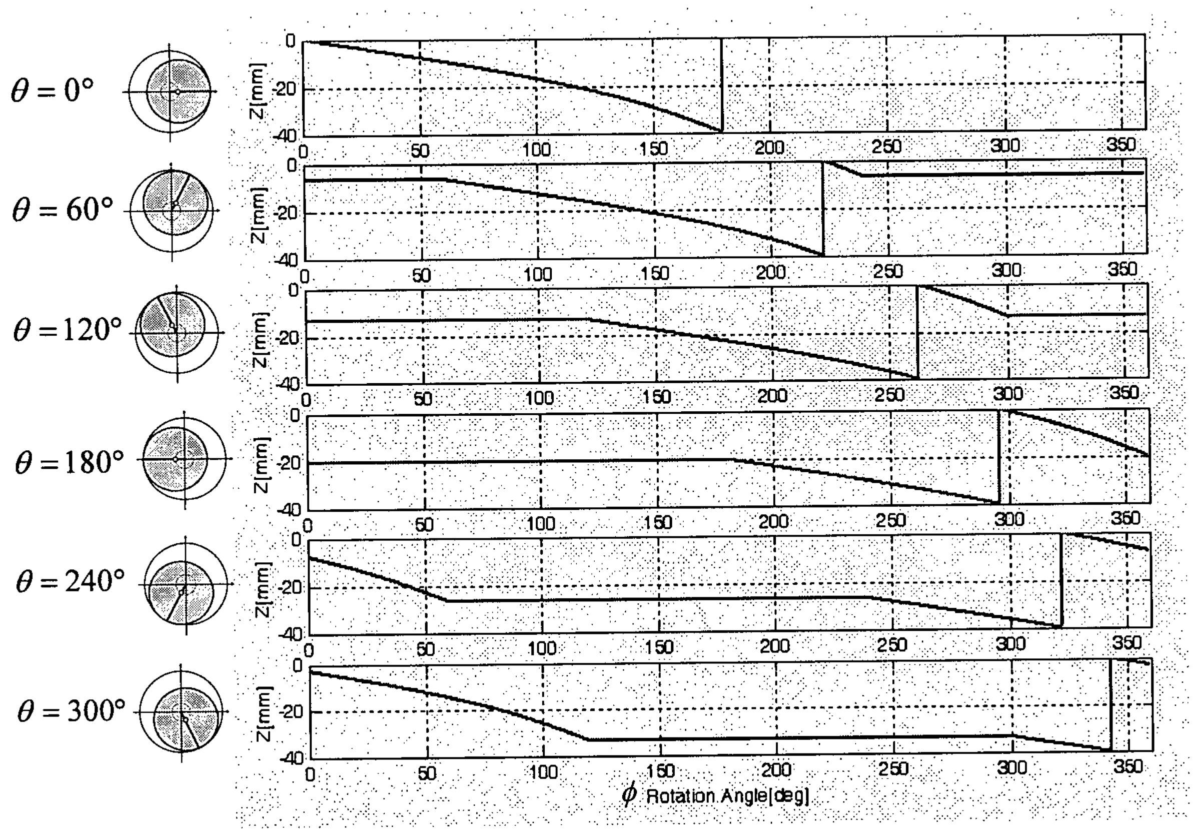


Figure 5-4: Z coordinate of the intersection curve: (radius of hole: *R*=10mm; radius of cylinder: *r*=7mm; pitch of helix: *p*=40mm)

5.3.2 Constraint 1: Valid Engagement Range

Figure 5-5 illustrates a valid engagement range [ϕ_s, ϕ_e] on the surface of an end-mill where the end-mill has rotated an angle θ along a helical toolpath. ϕ_s and ϕ_e can be obtained from Eq. (5.9) as follows:

$$\begin{cases} \phi_s = \theta \\ \phi_e = \theta + \pi \end{cases} \tag{5.9}$$

For example, if the end-mill rotates 60° along the helical toolpath, θ=60°. The valid engagement range should then be $\phi \in$ [60°, 240°].

Once the valid engagement range [ϕ_s, ϕ_e] is identified, the intersection curve portion within [ϕ_s, ϕ_e] is considered as a valid intersection curve.

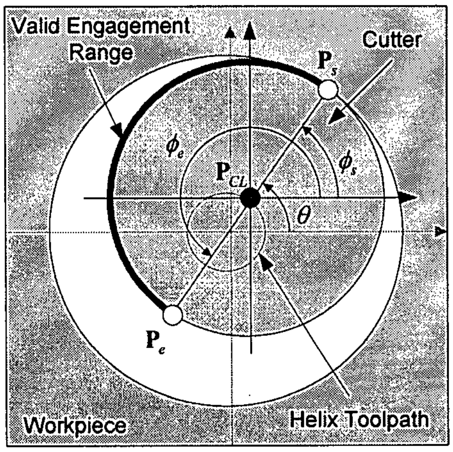


Figure 5-5: Valid Engagement Range

5.3.3 Constraint 2: Intersection Point

Figure 5-6 illustrates an end-mill at its first turn in machining a hole. It can be seen that the end-mill has intersections with both the machined area and the non-machined area on the in-process workpiece within its valid engagement range. The machined area is a helicoid generated by the bottom surface of the end mill, and the non-machined area is the top plane of the initial workpiece. An intersection point (P_b) that lies inside the valid engagement range as identified above joins the two curves that are generated by the cylinder/helicoid and cylinder/plane intersection, respectively. After identifying this intersection point and its angular location ϕ_b, it can be seen that the curve at $[\phi_s, \phi_b)$ is from a cylinder/plane intersection, and the curve at $[\phi_b, \phi_e]$ is from a cylinder/helicoid intersection.

The intersection point can be calculated by intersecting a circle representing the end-mill at the rotation angle θ along the helix with a circle representing the end-mill at the initial location $\theta = 0°$. The intersection point angle is described in Eq. (5.10). The derivation of this equation is given in Appendix C:

$$\phi_b = \frac{\theta}{2} + \pi - \sin^{-1}\left(\frac{R-r}{r}\sin\frac{\theta}{2}\right) \qquad (5.10)$$

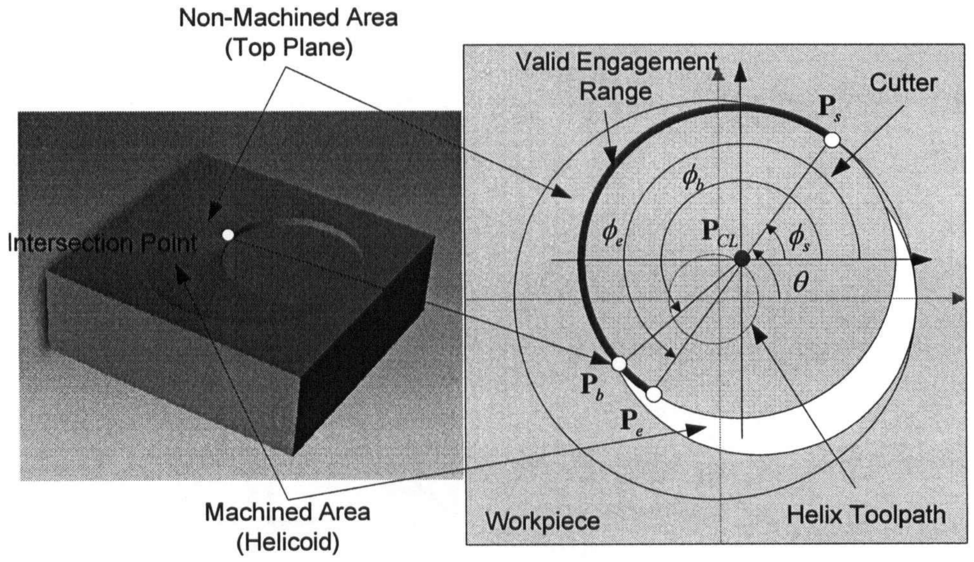


Figure 5-6: Intersection Point

5.3.4 Constraint 3: Connection Point

Figure 5-7 illustrates a connection point $\mathbf{P}_b$ that lies inside the valid engagement range. It can be seen that $\mathbf{P}_b$ can be identified as the intersection point between the top or bottom edge of the helicoid and the top or bottom edge of the semi-cylinder representing the front surface of a flat end-mill. The upper segment can be connected by shifting the lower segment a distance p upward. Since the connection point is identical to the intersection point in constraint 2, Eq. (5.10) can be used to calculate the connection point.

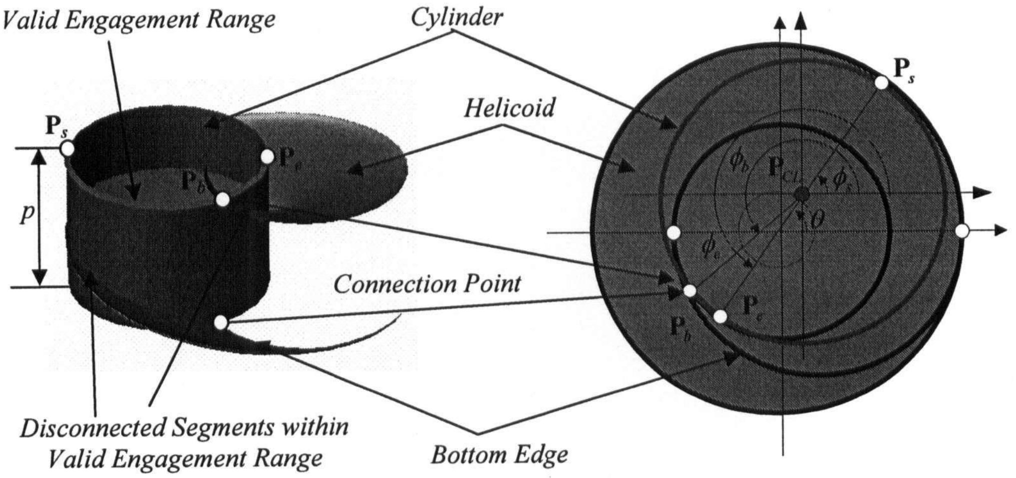


Figure 5-7: Connection Point

5.3.5 Parametric Representation of Cutter Engagement Feature

Based on the constraints above, Eq. (5.8) is adapted to represent the *ceF*s in the following three cases. An example with the same input parameters as those described in Figure 5-4 is presented for each case.

First Turn

For the first turn, the lower limit of the depth of cut d_{min} is always zero. The upper limit of the depth of cut d_{max} is composed of two curves joined at the intersection point P_b. The engagement angle is divided into two portions: $[\phi_s, \phi_b)$ and $[\phi_b, \phi_e]$. As represented in Eq. (5.11), d_{max} at the interval $[\phi_s, \phi_b)$ is the intersection curve between the end-mill and the top plane of the workpiece. Since the cutting tool moves a distance $p\theta/2\pi$ in the Z direction after rotating angle θ, d_{max} will be $p\theta/2\pi$ when $\phi \in [\phi_s, \phi_b)$. d_{max} at the interval $[\phi_b, \phi_e]$ is the intersection curve between the end mill and the helicoid. Figure 5.8 clearly shows the two intersection curves of the CWE for the first turn.

$$d_{\min} = 0, \qquad d_{\max} = \begin{cases} \dfrac{p}{2\pi}\theta & \phi_s \le \phi < \phi_b \\ \dfrac{p}{2\pi}(\theta - \alpha - \beta) & \phi_b \le \phi \le \phi_e \end{cases} \tag{5.11}$$

Middle Turns

For the middle turns, d_{min} is always zero, as with the first turn. d_{max} includes only one curve within the valid engagement range $[\phi_s, \phi_e]$, since the end mill only intersects with the helicoid. However, d_{max} at the interval $[\phi_s, \phi_b)$ needs to be shifted a pitch distance p upward to connect d_{max} at the interval $[\phi_b, \phi_e]$ in order to satisfy constraint 3. The equation for *ceF*s in the middle turns is described in Eq. (5.12). Figure 5-9 presents an example for these *ceF*s.

$$d_{\min} = 0, \qquad d_{\max} = \begin{cases} \dfrac{p}{2\pi}(\theta - \alpha - \beta) + p & \phi_s \le \phi < \phi_b \\ \dfrac{p}{2\pi}(\theta - \alpha - \beta) & \phi_b \le \phi \le \phi_e \end{cases} \tag{5.12}$$

Last Turn

For the last turn, d_{min} and d_{max} have two different representations, depending on the type of hole being machined.

Eq. (5.13) represents d_{min} and d_{max} for a blind hole. d_{min} and d_{max} for the interval $[\phi_b, \phi_e]$ are zero since the end-mill has transitioned from moving along the helix to a circular toolpath for generating the planar hole bottom. d_{max} for the interval $[\phi_s, \phi_b)$ is the intersection curve between the end-mill and the helicoid. Figure 5-10 shows the engagement boundary of the *ceF*s at different locations on the last turn for machining a blind hole.

$$d_{\min} = 0, \qquad d_{\max} = \begin{cases} -\dfrac{p}{2\pi}(\alpha+\beta)+p & \phi_s \le \phi < \phi_b \\ 0 & \phi_b \le \phi \le \phi_e \end{cases} \tag{5.13}$$

Eq. (5.14) represents d_{min} and d_{max} for a through hole. Both d_{min} and d_{max} over the interval $[\phi_b, \phi_e]$ are increasing when the cutting tool moves through the last turn. d_{max} over the interval $[\phi_s, \phi_b)$ is the intersection curve between the end-mill and the helicoid. Figure 5-11 shows the engagement boundary of the *ceF*s over the last turn for machining a through hole.

$$d_{\min} = \frac{p}{2\pi}\theta, \qquad d_{\max} = \begin{cases} \dfrac{p}{2\pi}(\theta-\alpha-\beta)+p & \phi_s \le \phi < \phi_b \\ \dfrac{p}{2\pi}\theta & \phi_b \le \phi \le \phi_e \end{cases} \tag{5.14}$$

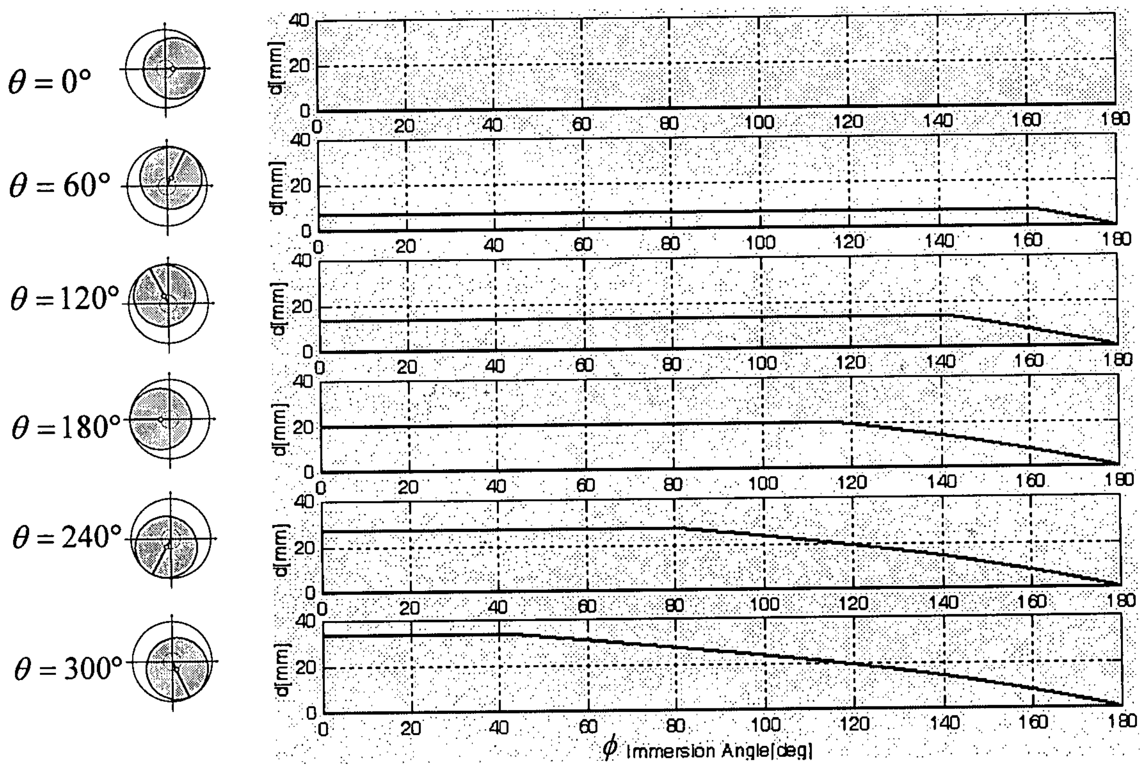


Figure 5-8: d_{min}, d_{max} over the 1st Turn from Eq. (5.11)

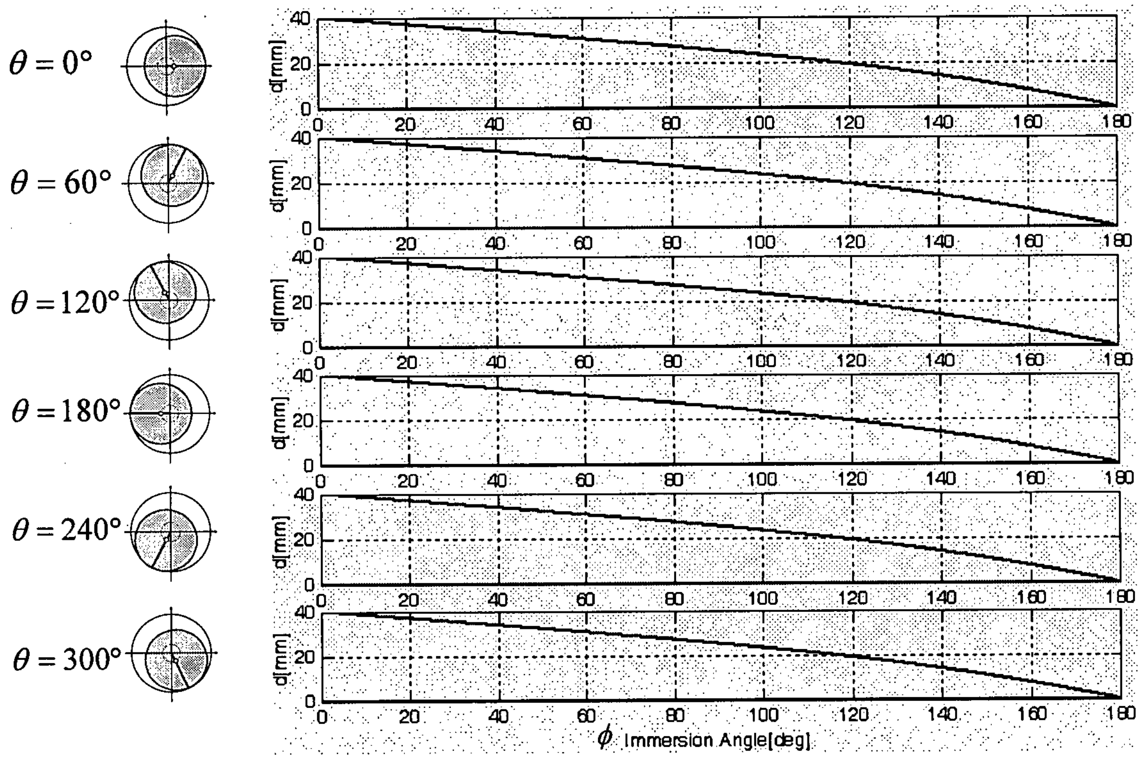


Figure 5-9: d_{min}, d_{max} over the Middle Turn from Eq. (5.12)

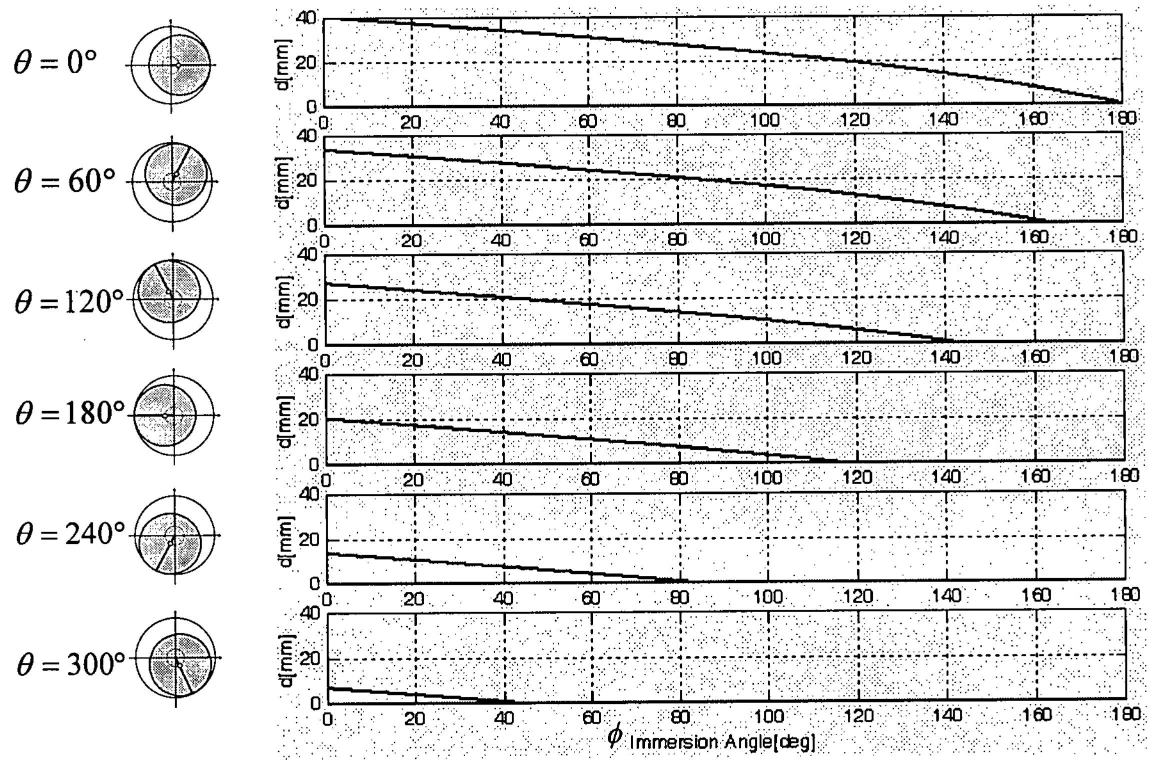


Figure 5-10: d_{min}, d_{max} over the Last Turn for a Blind Hole from Eq. (5.13)

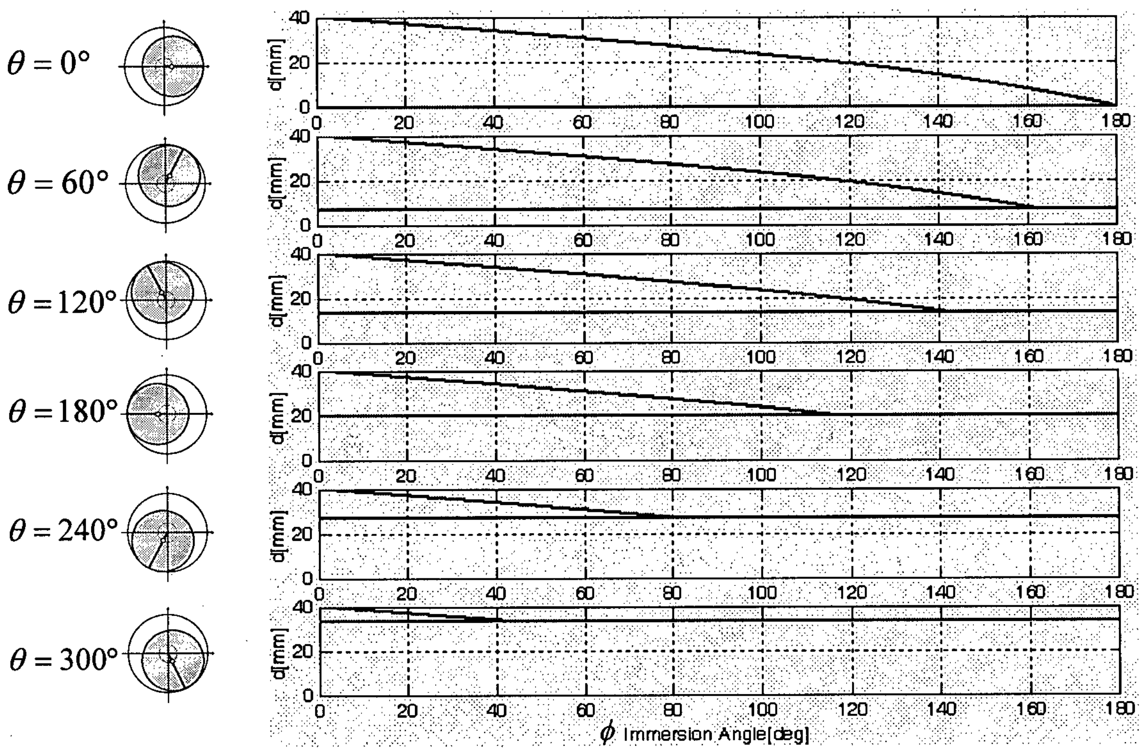


Figure 5-11: d_{min}, d_{max} over the Last Turn for a Through Hole from Eq. (5.14)

5.4 Implementation

The algorithms described in this chapter are implemented by using the commercial software Matlab. Both the *CWE* calculation and display are executed within the Matlab environment. In addition, a set of *CWE*s generated by the commercial machining simulation software VERICUT are compared with the *CWE*s generated analytically by the algorithms presented in this chapter. The computer used was a Pentium 4 processor, 3GHz/0.99GB.

5.4.1 Test Part

As illustrated in Figure 5-12, an aerospace gearbox cover is used as a test part to test the parametrization algorithm of the *ceFs* for the hole milling operation. In this test part, there are three *mhFs* with different parameters. *mhF*$_1$ and *mhF*$_3$ are through holes, and *mhF*$_2$ is a blind hole. Table 5.1 gives the details for each hole, as well as the tool parameters and toolpath parameters for the roughing stage. It can be seen from Table 5.1 that *mhF*$_2$ and *mhF*$_3$ have the same diameters. These three holes are machined by a flat end-mill with a diameter of 25 mm into three blind holes in the roughing stage.

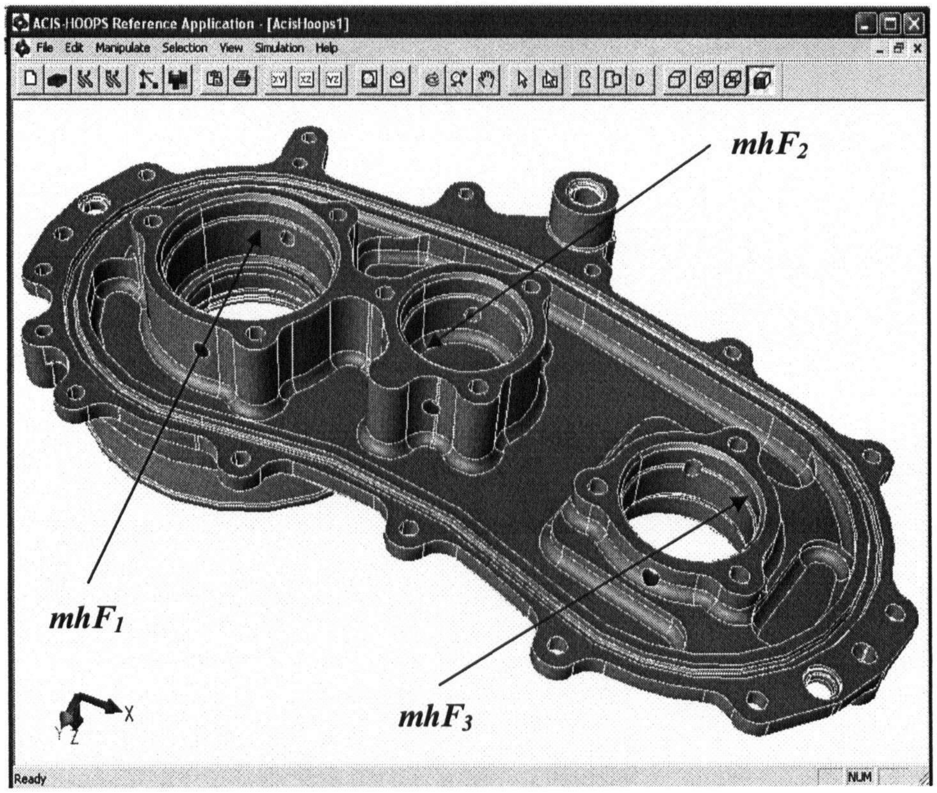

ACIS-HOOPS Reference Application - [AcisHoops1]
File Edit Manipulate Selection View Simulation Help
mhF_2
mhF_1
mhF_3
X
Y Z
Ready
NUM

Figure 5-12: A Test Part: Gearbox

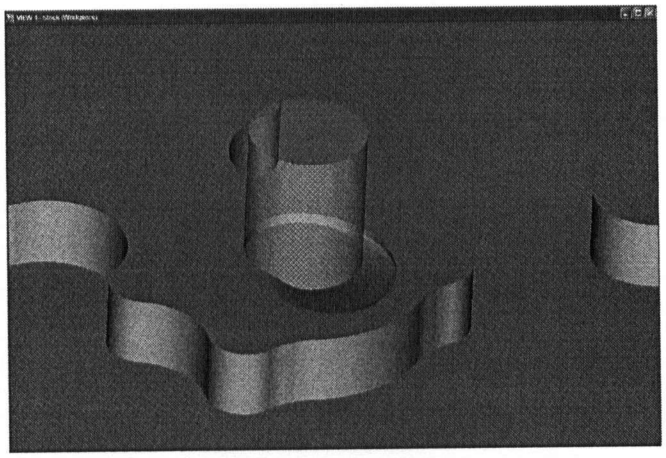
Machining Feature	Hole			Toolpath			Tool
	Diameter [mm]	Height [mm]	Type	Radius [mm]	Pitch [mm]	Helix Angle [°]	Diameter [mm]
mhF_1	49	24	Blind	12	7.92	6	25
mhF_2	40	24	Blind	7.5	6.62	8	25
mhF_3	40	21	Blind	7.5	6.62	8	25

Table 5-1: Parameters of *mhF*s for Roughing Stage

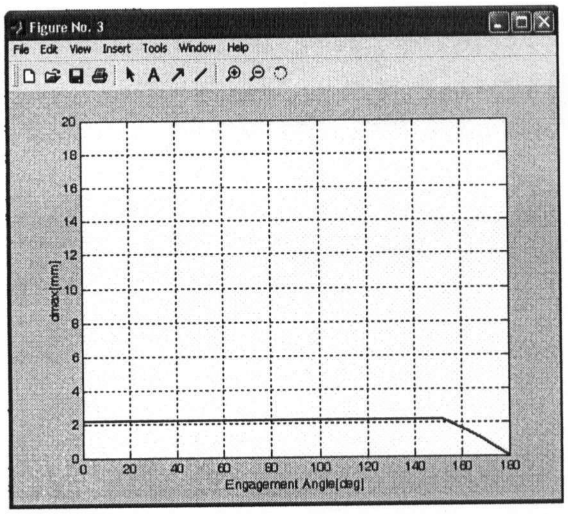
5.4.2 Results

Three test cases are designed to test the *ceF*s generated in the first, middle and last turns, respectively. These *ceF*s are compared with the *ceF*s generated by the commercial software VERICUT. VERICUT represents the *ceF*s using raster bitmaps. These raster bitmaps are generated using the Z-buffer method described in the literature review section, and therefore, produce an approximate solution to the *CWE*.

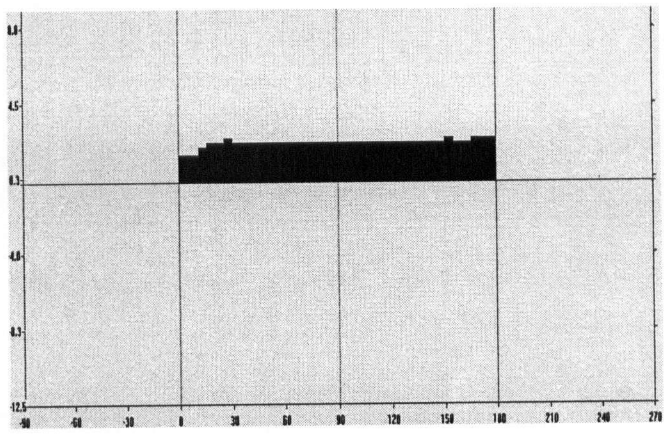
In Test Case 1, a *ceF* is generated when the cutting tool rotates 120° during its first turn to machine the hole *mhF*$_2$. The middle turn *ceF* is tested at the cutter location when the cutting tool has rotated 60° when machining the hole *mhF*$_1$ in Test Case 2. In Test Case 3, the *ceF* is presented where the cutting tool rotates 240° during its last turn to machine the blind hole *mhF*$_3$. Figures 5-13, 14, 15 (a) illustrate the cutter locations and in-process workpiece geometry where the *ceF*s are generated. Figures 5-13, 14, 15 (b) show *ceF*s generated by the analytical solution developed in this chapter, with the engagement angle ϕ of the *ceF*s arranged in an anticlockwise direction. Figures 5-13, 14, 15 (c) show *ceF*s generated by VERICUT, with ϕ arranged in a clockwise direction. It is obvious that VERICUT generates an approximation to the exact *ceF*s.



(a) Cutter Location

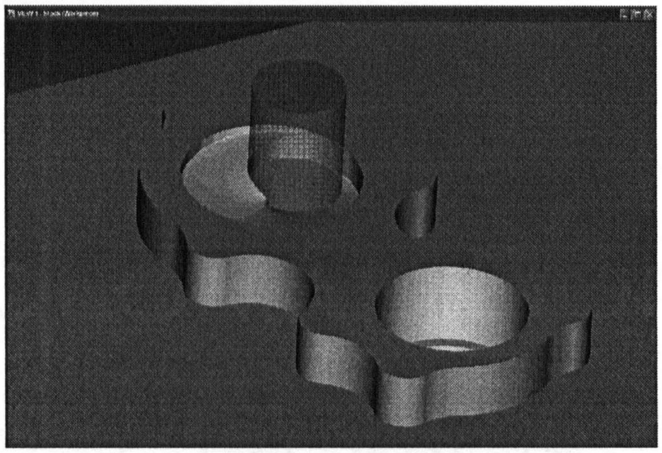


(b) *ceF* from Analytical Solution

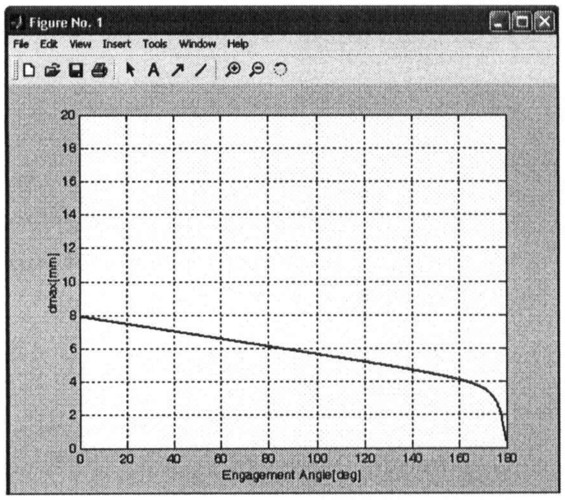


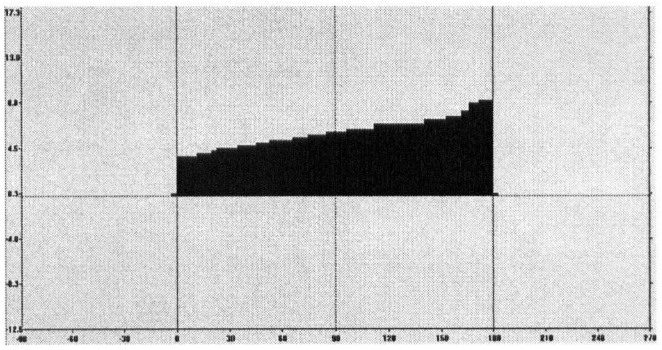
(c) *ceF* from VERICUT

Figure 5-13: Test Case 1: *ceF* on the *mhF*$_2$ at the First Turn (θ=120°)



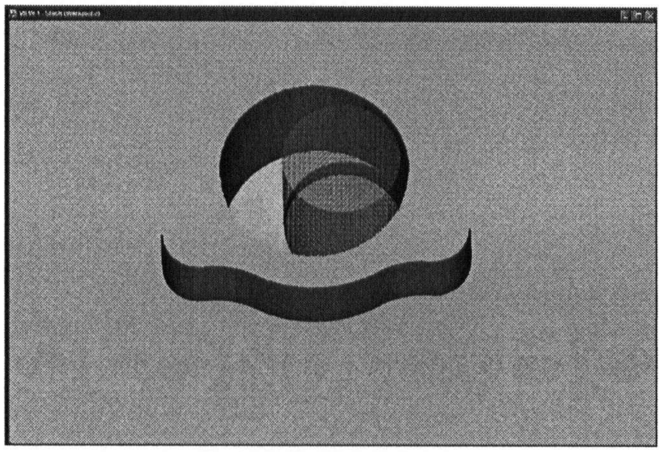
(a) Cutter Location

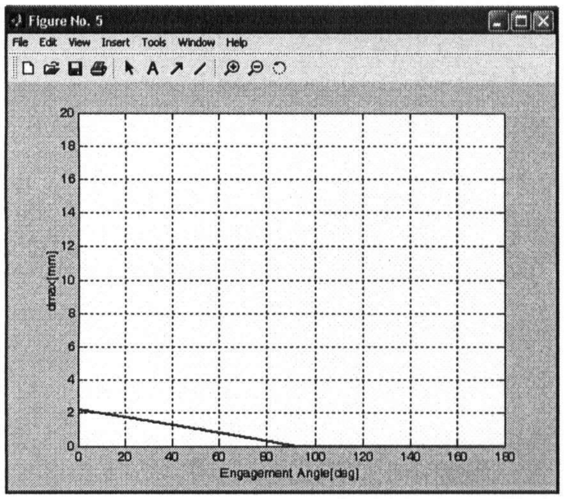


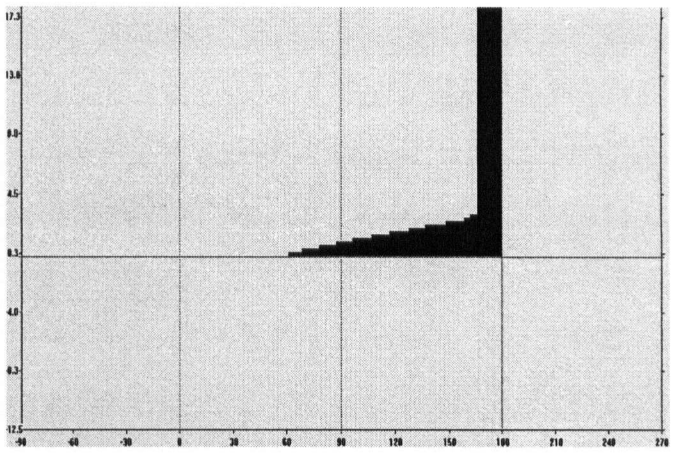


(b) *ceF* from Analytical Solution **(c) *ceF* from VERICUT**

Figure 5-14: Test Case 2: *ceF* on the *mhF$_1$* at the Middle Turn (θ=60°)


(a) Cutter Location





(b) *ceF* from Analytical Solution **(c) *ceF* from VERICUT**

Figure 5-15: Test Case 3: *ceF* on the *mhF*$_3$ at the Last Turn (θ=240°)

5.5 Discussion

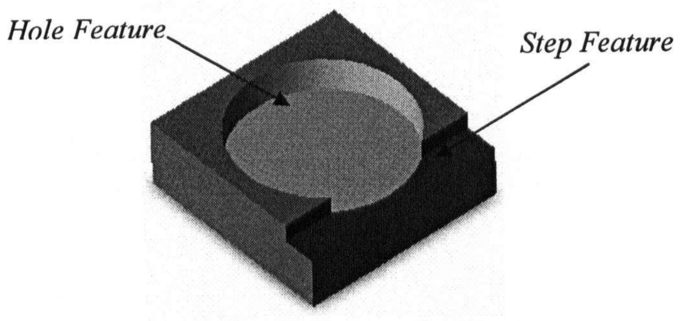
The *CWE* calculation approach described in this chapter provides a closed-form solution to process modeling for a hole milling operation. The results are compared with those generated from the commercial software VERICUT, as shown in Figures 5-13, 14, and 15. For hole milling process modeling, the analytical solution presented in this research has the following two advantages over the approach used in VERICUT:

- Computational Efficiency: VERICUT uses a discrete workpiece model (Z-Buffer) for geometric verification. *CWE*s are extracted as a raster bitmap by calculating the intersections between the cutting tool and a set of discrete normal vectors on the surface of the workpiece. A number of Line/Surface intersection calculations are involved during the *CWE* extraction. Moreover, the output raster bitmap needs to be processed into a standard format (described in Chapter 1) in order to satisfy the requirements of the force model. The analytical approach in this research provides a closed-form formula in the standard format. No further processing is needed.

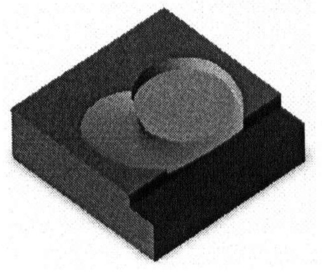
- Accuracy: the approach in VERICUT provides only an approximation to the *CWE*, since the calculation is based on a discrete workpiece model. In particular, the *CWE* region is wrongly calculated when the surface of the cutting tool has an edge contact with the wall of the hole where no forces exist between them (see Figures 5-15 (b) and (c)). This will introduce significant errors into the force prediction.

One advantage of using VERICUT is that it can be applied to *CWE* calculations across a broad set of machining domains, including complex 5-axis machining. As such, it is a generic methodology for a broad set of machining applications. The analytical approach developed in this research is limited to hole milling using a flat end-mill. An obvious extension of this research is to include the use of ball and bull nose end-mills in hole milling. The challenges of developing an analytical solution for these cutting tools involves finding closed-form solutions for intersections between surfaces of higher order than those investigated in this research; this in general is known to be a difficult problem. Numerical techniques will likely yield the only feasible solutions.

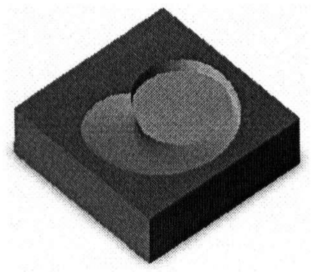
Another challenge to be addressed as future work is the influence on this closed-form technique of interacting features that may be present in a complicated part. As shown in Figure 5-16 (a), a step feature interacts with a hole feature in a part. It can be seen from Figure 5-16 (b) that some surfaces of the hole are trimmed by the step that has been machined previously. The *CWE* calculation presented in this chapter is affected when the cutting tool engages the surfaces of the machined interacting feature. This problem needs to be investigated to obtain the correct *CWE*s for the broadest range of hole machining to be properly extracted.



(a) Final Part with Interacting Features



(b) W_i with Interacting Features



(c) W_i without Interacting Features

Figure 5-16: Interacting Features

Chapter 6
In-Process Workpiece Modeling in Hole Milling

6.1 Introduction

In geometric verification, in-process workpiece modeling captures the in-process workpiece states during a machining operation. These models are important for verifying the correctness of NC toolpaths. As mention in Chapter 1, the in-process workpiece can be considered as a series of workpiece states, each generated by a machining operation. One or more operations can lead to the creation of a final part machining feature. For example, in Figure 6-1, a hole feature is created by a milling operation. The toolpath data consists of a single helical tool path. As can be seen, W_i is the workpiece geometry at any cutter location presented by the parametric value u_i along the helix during the hole machining operation. The initial workpiece is represented by W_0 prior to performing any cutting. After the tool motion along the helix is completed, a final part hole feature (H) is generated along with the corresponding in-process workpiece W_n (the entire model).

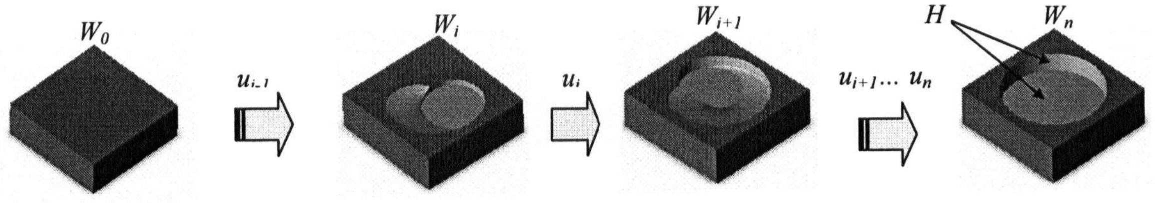


Figure 6-1: In-Process Workpiece in Hole Milling

In hole milling, the challenge for in-process workpiece modeling is in generating self-intersecting swept volumes that are created as the tool moves along the helix. In this chapter, a Non-Uniform Rational B-Spline (NURBS) based approach is developed to generate an exact solid model of the swept volume. Based on the exact B-rep representation of the swept volume, the in-process workpiece can be modeled by performing Boolean operations between the swept volume and the in-process workpiece generated by the previous tool motion. This is a novel capability, since self-intersecting solids cannot be generated by CAD systems.

NURBS curves and surfaces will be described in the next section, since they are used to construct the boundary surfaces of the swept volume. Following this, a NURBS based swept volume generation approach will be presented. This will be followed by in-process workpiece modeling using the Boolean operation between the swept volume and the in-process workpiece. Finally, examples will be presented to validate the proposed algorithm.

6.2 NURBS Curves and Surfaces

6.2.1 NURBS Curves

A NURBS curve is defined as follows:

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{n} h_i \mathbf{P}_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)} \tag{6.1}$$

where:

- $N_{i,k}(u)$ is a blending function, defined as:

$$N_{i,k}(u) = \frac{(u - t_i) N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u) N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}}$$

$$N_{i,1}(u) = \begin{cases} 1 & t_i \le u \le t_{i+1} \\ 0 & otherwise \end{cases}$$

- k-1 degree
- n+1 control points
- $[t_0, t_1, \ldots, t_{n+k}]$ is a knot vector, a knot vector has $n+k+1$ knot values
- $\mathbf{P}_i$ (x_i, y_i, z_i, h_i) is the i^{th} position vector or control point, h_i is a homogeneous coordinate or weight of the i^{th} control point

NURBS Representation of a Line:

As shown in Figure 6-2 (a), a straight line bounded with two points (**P**$_0$, **P**$_1$) can be represented by a first degree (k=2) NURBS curve with a knot vector [0, 0, 1, 1] and two control points, **P**$_0$ and **P**$_1$. The weights of the two control points are $h_0=h_1=1$.

NURBS Representation of an Arc:

An arc has several representations with NURBS curves. To achieve the best parametrization, a 2nd degree (k=3) rational Bezier curve is used to represent an arc with a central angle less than 90°. As shown in Figure 6-2 (b), a composite rational Bezier curve is used to represent an arc with central angle greater than 90°. The knot vector, control points and their weights are listed in Figure 6-2 (b).

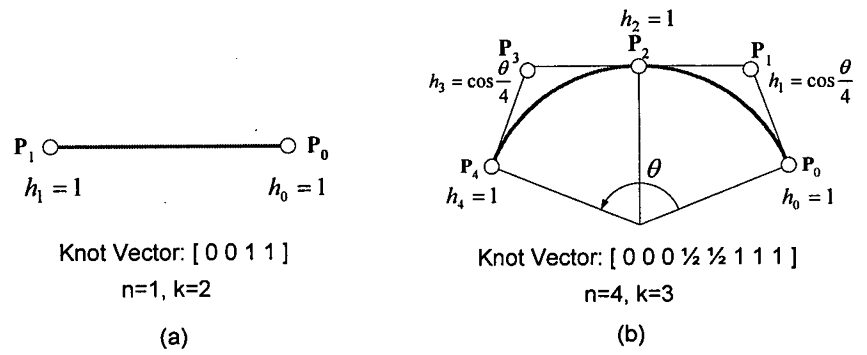


Figure 6-2: NURBS Representation of Line and Arc

6.2.2 NURBS Surfaces

A NURBS surface is defined as follows:

$$\mathbf{P}(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}h_{i,j}\mathbf{P}_{i,j}N_{i,p}(u)N_{j,q}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n}h_{i,j}N_{i,p}(u)N_{j,q}(v)} \tag{6.2}$$

where:

- $N_{i,p}(u)$ and $N_{j,q}(v)$ are blending functions in the u and v direction
- p-1 is the degree in the u direction, and q-1 the degree in the v direction

- m+1 control points are defined in the u direction and n+1 control points in the v direction
- $[u_0, u_1, \ldots, u_{p+m}]$ and $[v_0, v_1, \ldots, v_{q+n}]$ are the knot vectors in the u and v directions
- $\mathbf{P}_{i,j}$ $(x_{i,j}, y_{i,j}, z_{i,j}, h_{i,j})$ is the i^{th} row and j^{th} column control point in the control polyhedron; $h_{i,j}$ is the homogeneous coordinate (weight) of the control point $\mathbf{P}_{i,j}$

Figure 6-3 illustrates a NURBS representation of a semi-cylinder. The NURBS surface has the following parameters in the u and v directions:

- u direction: $m=3$, $p=3$, knot vector = [0, 0, 0, ½, ½, 1, 1, 1], homogenous coordinates=[1, 0.707, 1, 0.707, 1]
- v direction: $n=1$, $q=2$, knot vector = [0, 0, 1, 1], homogenous coordinates=[1, 1]

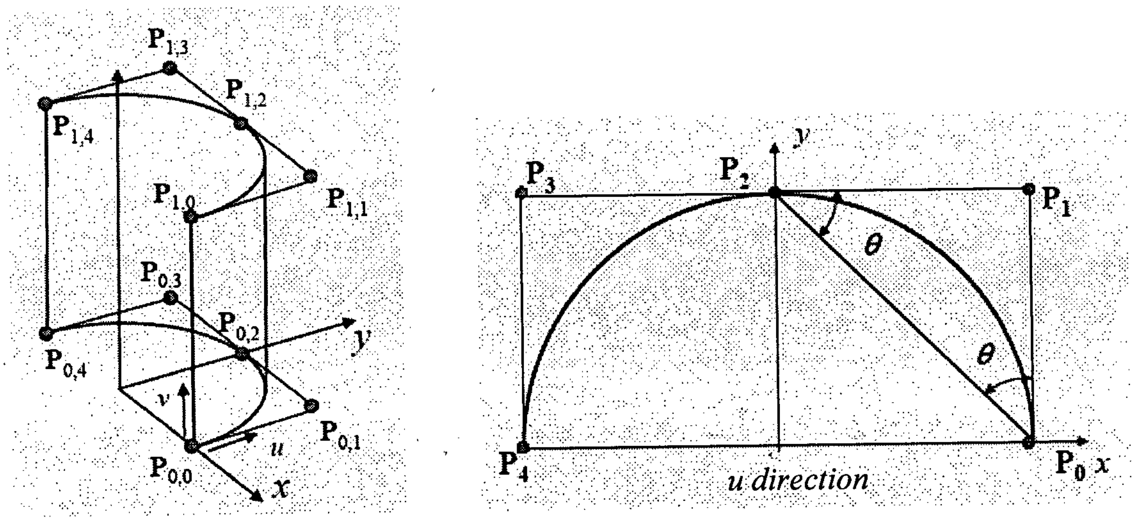


Figure 6-3: NURBS Representation of a Quadric Surface

6.3 Swept Volume Generation

A general swept surface is described in Figure 6-4. It is created by sweeping a parametric surface $\mathbf{S}(u,v)$ along an arbitrary path $\mathbf{\Psi}(t)$ with a rotation about axis $\mathbf{\Gamma}$. Mathematically, a swept volume is described by the sweep equation as:

$$\xi(u,v,t) = \mathbf{S}(u,v)\mathbf{R}(t) + \mathbf{\Psi}(t) \tag{6.3}$$

where:

$\xi(u,v,t)$: The set of all points inside and on the boundary

$\mathbf{R}(t)$: 3×3 rotation matrix

$\boldsymbol{\Psi}(t)$: Sweep path

$\mathbf{S}(u,v)$: Parametric equation of surface or solid

In a hole milling operation, the cutting tool moves along a helical toolpath. Therefore, the swept volume is generated by sweeping a quadric surface along a helix without rotations. In this section, a NURBS based approach is developed to generate an exact B-rep model for the swept volume to support in-process workpiece modeling for hole milling.

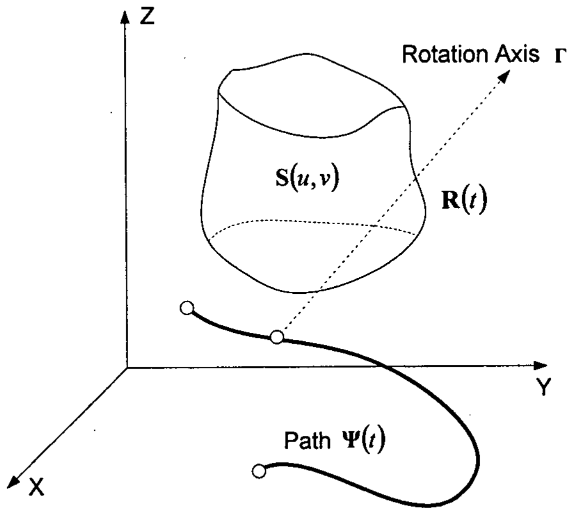


Figure 6-4: Sweeping a Surface

6.3.1 Overview

Figure 6-5 illustrates a NURBS based swept volume generation approach. It can be seen that the boundary surfaces of the swept volume can be classified into three categories: *Ingress*, *Egress* and *Envelope Surfaces*. Ingress and egress surfaces are created by splitting the cylinder along its silhouette curves. Envelope surfaces are generated by sweeping the silhouette curves along the helical toolpath. If there are intersections among any of these surfaces, a surface trimming procedure is needed to remove all of the patches located inside the swept volume. Finally, the ingress surfaces at the initial position, the egress surfaces at the final position, and the trimmed envelope surfaces are stitched together to form a solid model of the swept volume.

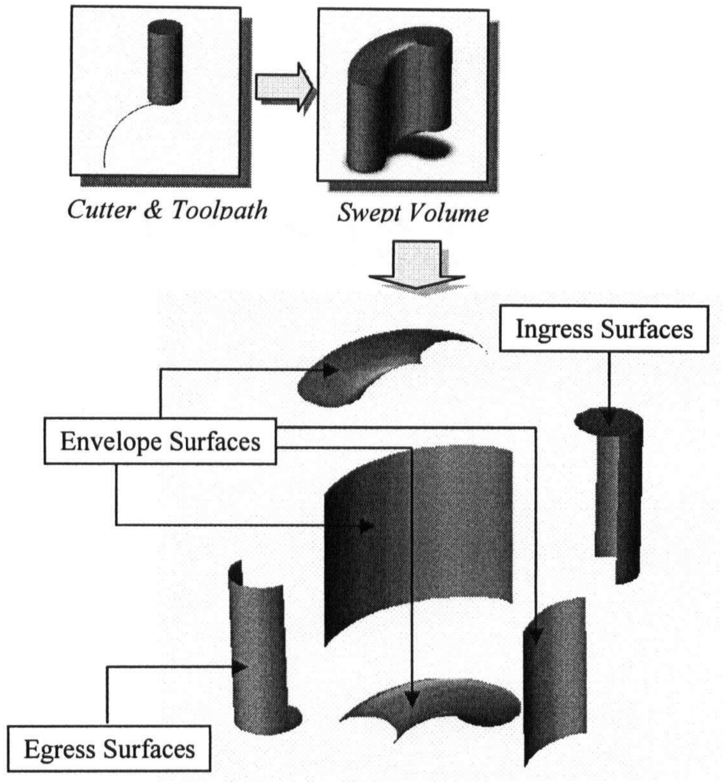


Figure 6-5: Swept Volume Generation

A flowchart of the NURBS based approach is presented in Figure 6-6. This approach includes the following steps:

1. ***Silhouette Curves Identification*:**

   To identify the silhouette curves of the tool geometry based on the helix angle of the toolpath, and to represent these curves with NURBS.

2. ***Envelope Surfaces Generation:***

   To generate envelope surfaces by sweeping the silhouette curves along the helical toolpath, and represent these envelope surfaces with NURBS.

3. ***Surfaces Trimming:***

   To identify the intersecting surfaces among the ingress, egress and envelope surfaces, and trim those patches that are within the boundary of the swept volume.

4. ***Surfaces Stitching:***

   To stitch all the boundary patches together to construct a B-rep solid model for the swept volume. The topological correctness is verified based on the Euler-Poincare formula.

The detail of the above steps will be provided in the following sections.

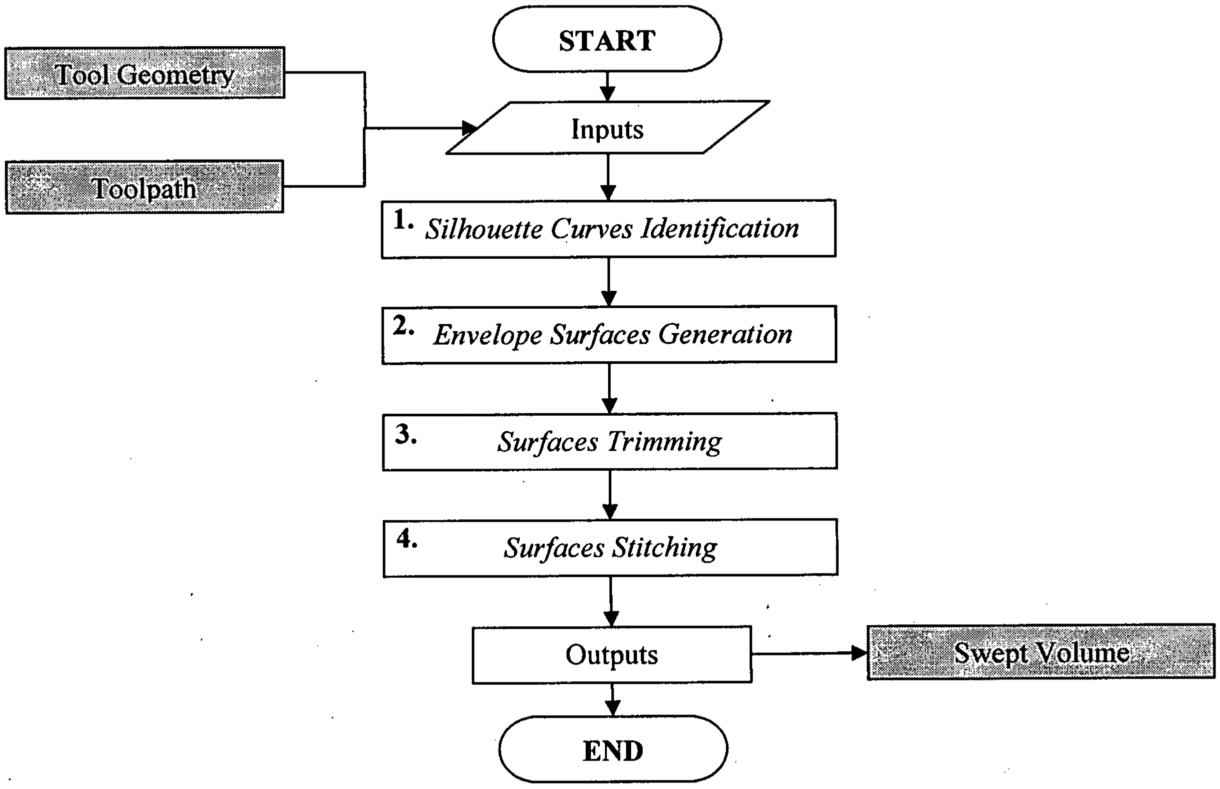


Figure 6-6: A Flowchart of the NURBS Based Approach

6.3.2 Silhouette Curves Identification

Figure 6-7 illustrates a silhouette curve on a general surface from a perspective direction **P**. A perspective direction is the tangent direction of the path curve during sweeping. A silhouette curve on the surface can be defined as a set of points on the surface where the normal of the surface is perpendicular to the perspective direction. Mathematically, a silhouette curve on a parametric surface $\mathbf{S}(u,v)$ can be represented as:

$$\mathbf{G}(u,v) = \mathbf{n}(u,v) \bullet \mathbf{P} = 0 \tag{6.4}$$

where:

Surface normal: $\mathbf{n}(u,v)=\dfrac{\partial \mathbf{S}}{\partial u}\times\dfrac{\partial \mathbf{S}}{\partial v}$

Perspective direction: **P**

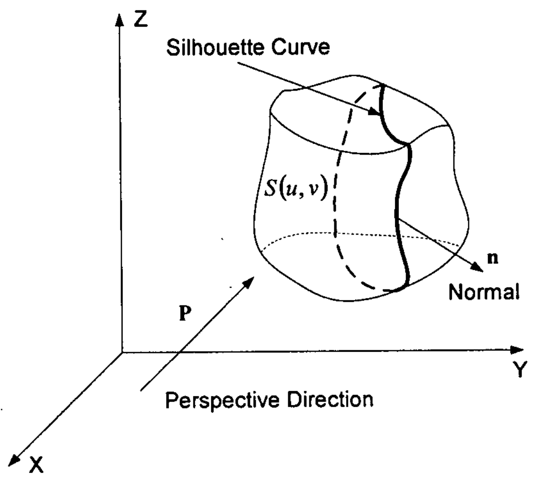


Figure 6-7: Silhouette Curves of a General Surface

In this section, the silhouette curves on the surfaces of flat, conical and ball nose end-mills are given. The perspective direction **P** has an angle over the XY plane that is equal to the helix angle α of the helical toolpath. Figure 6-8 illustrates silhouette curves for a ball end-mill that has a cylindrical and spherical surface. The NURBS parameters, such as the number of control points (*n*), degree (*k-1*) and knot vector, are given in Figure 6-8. The control points for each silhouette curve are listed in Table 6-1. The reader can refer to Appendix D for silhouette curves of the flat and conical end-mills.

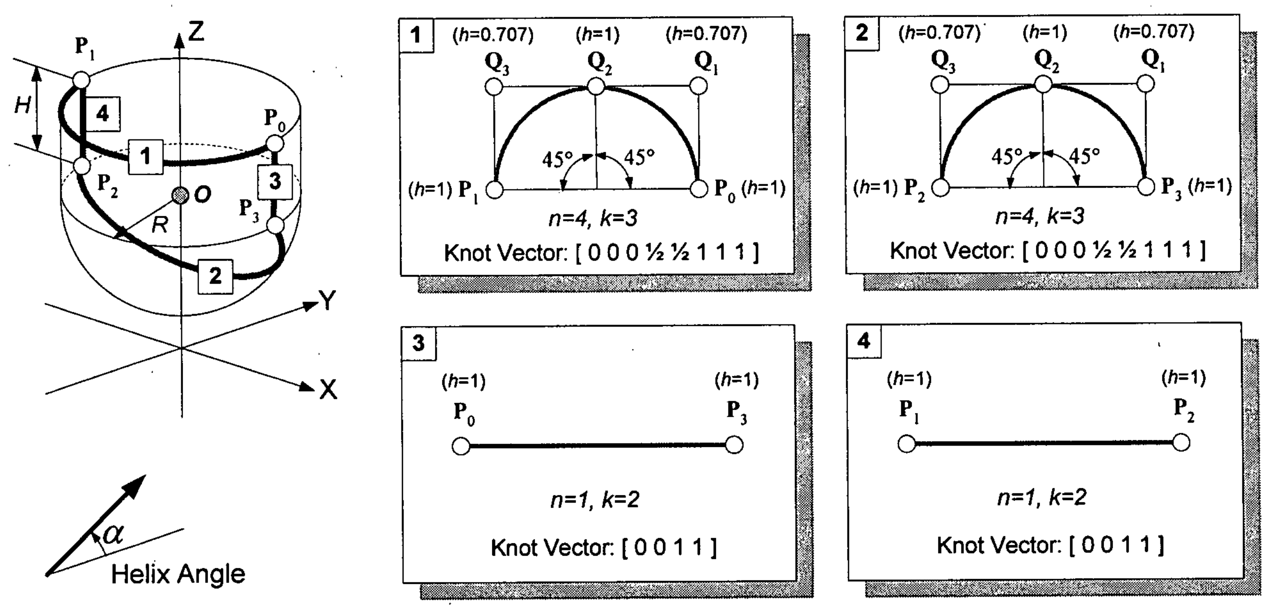


Figure 6-8: Silhouette Curves for a Ball End-Mill

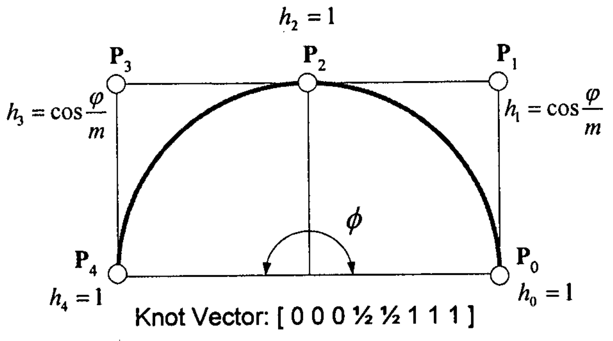
Curve #	Type	Control Points			
		Point #	X	Y	Z
1	Arc	P_0	x_0+R	y_0	z_0+H
		P_1	x_0-R	y_0	z_0+H
		Q_1	x_0+R	y_0-R	z_0+H
		Q_2	x_0	y_0-R	z_0+H
		Q_3	x_0-R	y_0-R	z_0+H
2	Arc	P_3	x_0+R	y_0	z_0
		P_2	x_0-R	y_0	z_0
		Q_1	x_0+R	$y_0+R\sin\alpha$	$z_0-R\cos\alpha$
		Q_2	x_0	$y_0+R\sin\alpha$	$z_0-R\cos\alpha$
		Q_3	x_0-R	$y_0+R\sin\alpha$	$z_0-R\cos\alpha$
3	Line	P_0	x_0+R	y_0	z_0+H
		P_3	x_0+R	y_0	z_0
4	Line	P_1	x_0-R	y_0	z_0+H
		P_2	x_0-R	y_0	z_0

Table 6-1: Control Points for Silhouette Curves of a Ball End-Mill

6.3.3 Envelope Surfaces Generation

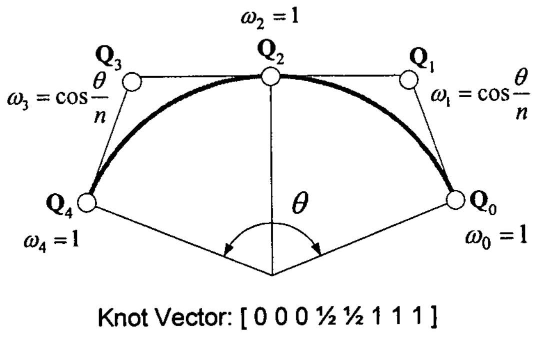
Envelope surfaces are generated by sweeping silhouette curves (profile curves) along the path curve. In today's CAD systems, the technique used to generate a swept surface by sweeping profile curves along a path curve is to interpolate a series of profile curves located at calculated intervals along the path curve. This is referred to as *skinning*. This interpolation is necessary to provide a general solution for a sweep regardless of the mathematical form of the path's geometry. The interpolation of the path curve leads to an approximation of the swept surface to a resolution specified in the modeler. In this thesis, both the profile curve (silhouette curve) and the path curve (helical curve) are represented exactly as NURBS curves. It is therefore possible to construct the NURBS surface to represent the swept surface exactly without the approximation created by interpolation. In order to provide an exact NURBS representation for envelope surfaces, the control polyhedron and associated weights have to be identified based on NURBS representation of the profile and path curves.

To assist in the description of the envelope surface generation approach, a NURBS representation of a silhouette curve and a helical curve is illustrated in Figure 6-9. Figure 6-9 (b) presents the top view of the helical curve.



m=4, p=3

(a) Silhouette Curve



n=4, q=3

(b) Helical Curve

Figure 6-9: NURBS Representation of a Silhouette Curve and a Helical Curve

Based on the profile and path curves described in Figure 6-9, a control polyhedron of a NURBS surface for the profile curve swept along the path curve is illustrated in Figure 6-10. The control points of the NURBS surface $\mathbf{P}_{i,j}$ are calculated by transforming the control polygon of the profile curve $\mathbf{P}_j$ along the control polygon of the path curve $\mathbf{Q}_i$. This transformation includes a rotation about the z-axis, a translation along the z-axis and a scaling in the x-axis. This calculation is described in Eq. (6.5).

$$\mathbf{P}_{i,j} = \mathbf{P}_j \bullet \mathbf{R}(\theta) \bullet \mathbf{T}(\theta) \bullet \mathbf{S}(\theta) \tag{6.5}$$

where:

$\mathbf{P}_{i,j}(x_{i,j}, y_{i,j}, z_{i,j}, 1)$: The homogeneous coordinates of the control point of the NURBS surface

$\mathbf{P}_j(x_j, y_j, z_j, 1)$: The homogeneous coordinates of the control point of the profile curve

$\mathbf{R}(\theta)$: The rotation matrix

$\mathbf{T}(\theta)$: The translation matrix

$\mathbf{S}(\theta)$: The scaling matrix

θ: The central angle of the helical curve

$\mathbf{R}(\theta)$,$\mathbf{T}(\theta)$ and $\mathbf{S}(\theta)$ are 4×4 matrices represented in Eq. (6.6)-(6.8).

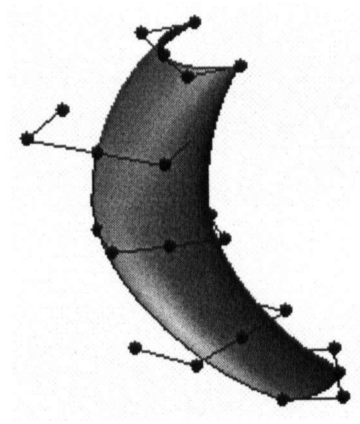
$$\mathbf{R} = \begin{bmatrix} \cos\left(i\frac{\theta}{n}\right) & \sin\left(i\frac{\theta}{n}\right) & 0 & 0 \\ -\sin\left(i\frac{\theta}{n}\right) & \cos\left(i\frac{\theta}{n}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -i\frac{p\theta}{2n\pi} & 1 \end{bmatrix} \tag{6.7}$$

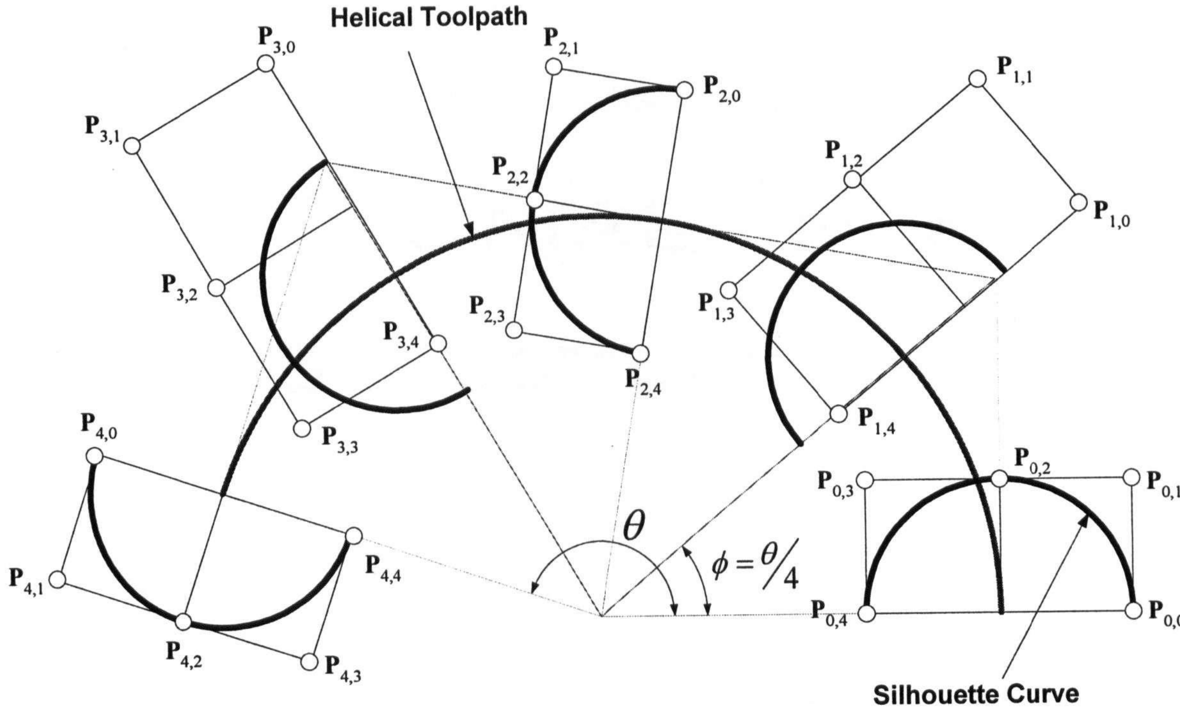
$$\mathbf{S} = \begin{bmatrix} k_i & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

where:

$$k_i = \begin{cases} \dfrac{1}{\cos\left(\dfrac{\theta}{n}\right)} & i = 1,3,5,\ldots \\ 1 & i = 0,2,4,\ldots \end{cases}$$



(a) An Envelope Surface and Control Polyhedron



(b) Top View of the Control Polyhedron

Figure 6-10: Control Polyhedron of the Envelope Surface

The weights of the control points of the NURBS surface can be calculatied using Eq (6.9).

$$h_{i,j} = \omega_i \cdot h_j \quad i = 0,\dots,m \quad j = 0,\dots,n \tag{6.9}$$

where h_j , ω_i are weights of the control point of the profile and path curve described in Figure 6-9.

Figure 6-11 shows an example of the control polyhedron of an envelope surface generated by the top edge of an end-mill based on the above equations. This approach provides an exact and more concise representation compared with the approximate approach used as part of the skinning technique.

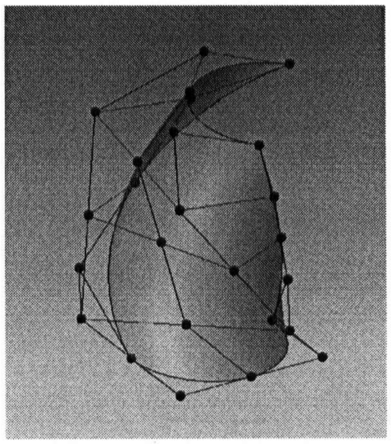


Figure 6-11: NURBS Surface and Control Polyhedron for an Envelope Surface

6.3.4 Surface Trimming

As mentioned in the section overview, the boundaries of a swept volume consist of ingress, egress and envelope surfaces. Ingress and egress surfaces are generated by splitting the boundary surfaces of the cutting tool along its silhouette curves. A surface trimming step will be applied if intersections exist among these surfaces. This step is accomplished by first intersecting the surfaces in question by applying surface/surface intersection algorithms within a solid modeler. Then the resulting patches inside the boundary of the swept volume are detected and removed. The remaining patches form the boundary of the swept volume.

Figure 6-12 illustrates how to identify the patched to be removed. The cutting tool sweeps from the initial position $\mathbf{P}_s(r, 0)$ to the final position $\mathbf{P}_e(rcos\theta, rsin\theta)$. The patches to be trimmed are always located inside the region defined by the intersection of the initial and final cylinders. Considering the circular projections of these cylinders given any point $\mathbf{P}(x, y)$

on a patch, if **P**(x, y) satisfies Eq. (6.10), the patch will be trimmed, otherwise the patch will be identified as a boundary surface of the swept volume.

$$\begin{cases}(x-r)^2+y^2 \le R^2 \\ (x-r\cos\theta)^2+(y-r\sin\theta)^2 \le R^2\end{cases} \tag{6.10}$$

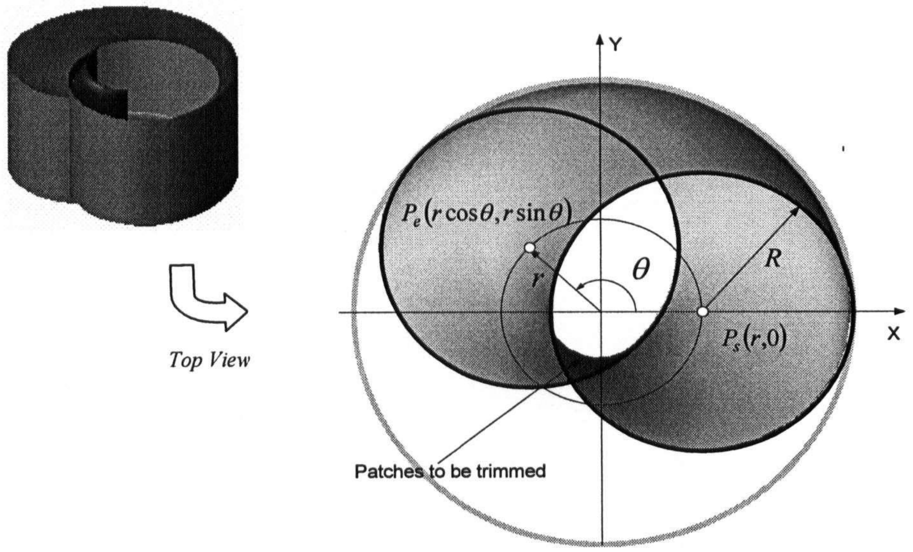


Figure 6-12: Surfaces Trimming

6.3.5 Surface Stitching

The stitching step constructs a B-rep solid model from the trimmed patches that make up the boundary of the swept volume. Solid modelers such as ACIS provide API routines to convert a set of surfaces forming a closed region in 3D space into a B-rep solid model. This operation will be successful only if a closed 3D space is surrounded by these patches.

The B-rep of the stitched volume can be verified by the Euler-Poincare formula as:

$$\mathrm{V}-\mathrm{E}+\mathrm{F}=2 \tag{6.11}$$

where:

V: Number of Vertices

E: Number of Edges

F: Number of Faces

Only a valid B-rep solid can be used for Boolean operations required by in-process workpiece modeling.

6.4 In-Process Workpiece Modeling

In-process workpiece modeling generates a series of in-process workpieces at any cutter location along the toolpath, As illustrated in Figure 6-13, the in-process workpiece W_{i+1} is generated by a regularized Boolean subtraction operation (-*) between the in-process workpiece W_i and the swept volume SV_i.

$$W_{i+1} = W_i - {}^{*}SV_i \tag{6.12}$$

where:

SV_i: Swept volume generated by the i-th tool motion

W_i: In-process workpiece before the i-th tool motion

W_{i+1}: In-process workpiece after the i-th tool motion

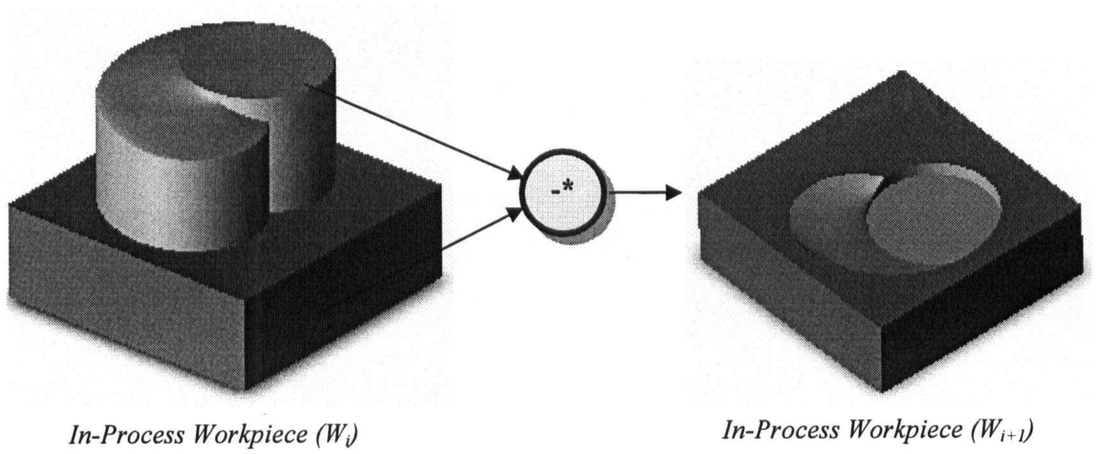

-*
In-Process Workpiece (W_i)
In-Process Workpiece (W_{i+1})

Figure 6-13: In-Process Workpiece Modeling

6.5 Implementation and Validation

The algorithms described in this chapter were implemented using the Microsoft Windows XP professional edition operating system. Visual C++ was used as the development tool. This implementation uses the ACIS 3D modeling kernel as the solid modeling engine, and the HOOP 3dGS as the graphics system. This program was tested using a computer with a Pentium 4 processor, with 3GHz/0.99GB.

Figure 6-14 illustrates data structures designed in this implementation. There are three types of geometry in the data structures. Each type corresponds to a geometric entity such as a volume, surface or curve. The class SweptVolume has a list of egress, ingress and envelop surfaces. The class EnvelopeSurface contains a list of silhouette curves. Each class points to a B-rep data structure of the corresponding geometric element.

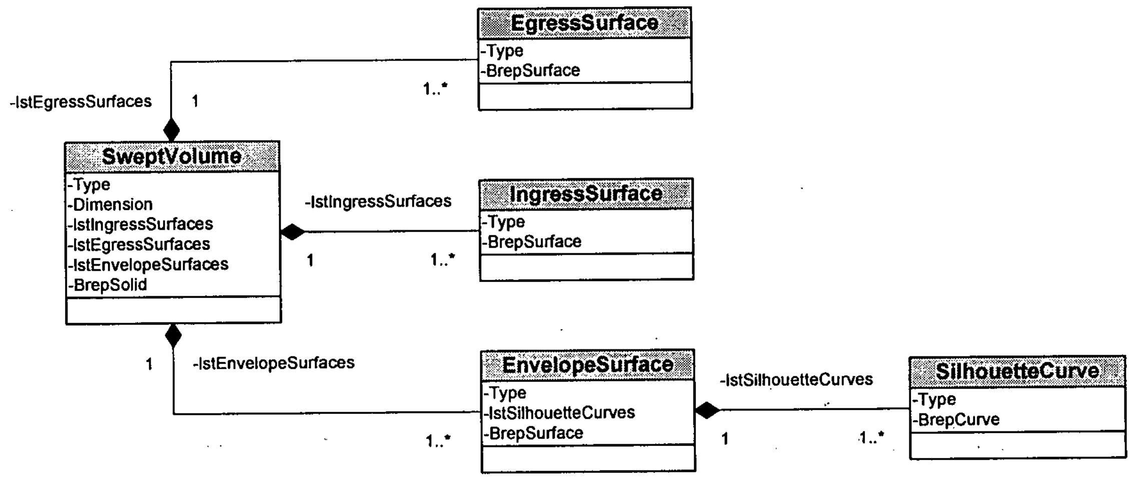


Figure 6-14: Data Structures

Table 6-2 presents an example of the swept volume for a flat end-mill. It can be seen that the swept volume is self-intersecting, since the radius of the cutter is greater than the radius of the toolpath. Table 6-3 shows a swept volume of a ball end-mill having the same radius as the helical toolpath. An example of the swept volume for a conical end-mill is presented in Table 6-4, where the radius of the cutter is smaller than that of the helical toolpath. This is also the case where the swept volume is self-intersecting.

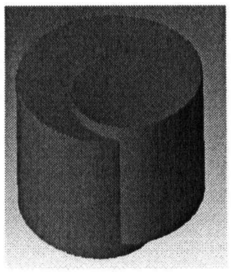
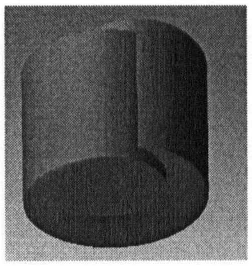
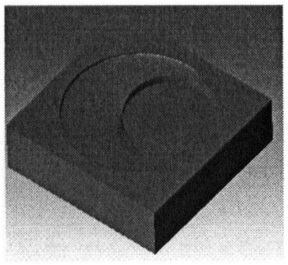
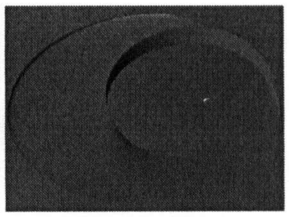
Helical Toolpath			
Radius [mm]	Pitch [mm]	Start Angle [deg]	End Angle [deg]
20	40	0	315
Flat End Mill			
Radius [mm]		Height [mm]	
30		80	
Swept Volume			
Swept Volume		Workpiece	
			

Table 6-2: Swept Volume Generation and Verification for a Flat End-Mill

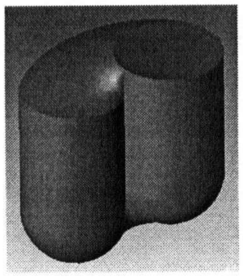
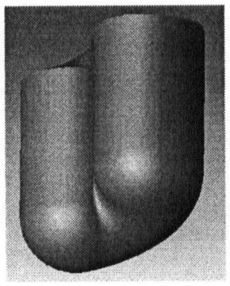
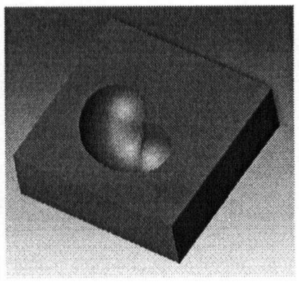
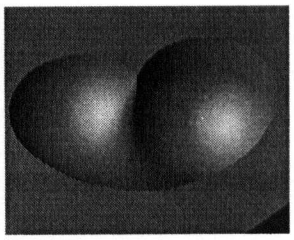
Helical Toolpath			
Radius [mm]	Pitch [mm]	Start Angle [deg]	End Angle [deg]
20	40	0	60
Ball End Mill			
Radius [mm]		Height [mm]	
20		50	
Swept Volume			
Swept Volume		Workpiece	
			

Table 6-3: Swept Volume Generation and Verification for a Ball End-Mill

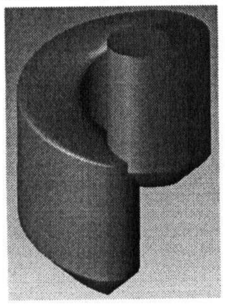
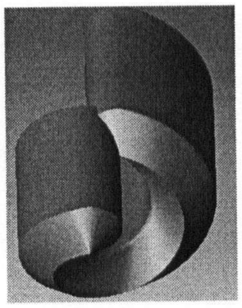
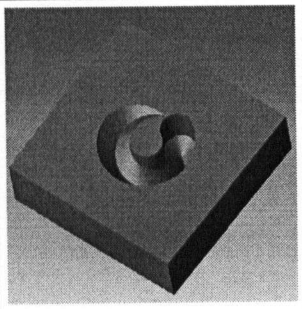
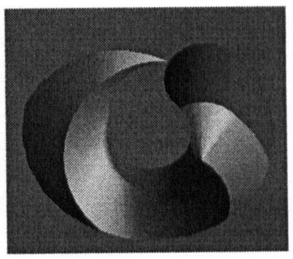
Helical Toolpath			
Radius [mm]	Pitch [mm]	Start Angle [deg]	End Angle [deg]
20	40	0	330
Conical End Mill			
Radius [mm]	Height [mm]	Cone Height [mm]	
10	40	10	
Swept Volume			
Swept Volume		Workpiece	
			

Table 6-4: Swept Volume Generation and Verification for a Conical End-Mill

6.6 Discussion

As shown in Tables 6-2, 6-3 and 6-4, the swept volumes and in-process workpieces generated by the three types of cutting tool at different cutter locations using the approach outlined in this chapter are verified. The advantage of the algorithms used is that the boundary surfaces of the swept volume are exactly represented with NURBS. This is because the ingress and egress surfaces are quadric surfaces based on the cutting tool geometry that can be exactly represented with NURBS. Also the envelope surfaces are accurately constructed with NURBS by calculating control points in the control polyhedron instead of applying the generic sweeping operations provided in solid modelers. In solid modelers, the sweeping operation generates a swept surface by sweeping profile curves along a path, which involves interpolating a series of profile curves located along the path. However, this technique provides only an approximate solution to the swept surfaces. The envelope surfaces generation approach presented in this research provides an exact solution to the envelope surfaces. These envelope surfaces have compact and exact representation. Appendix E-1 and E-2 are listed to compare the sat files (ACIS file) of an envelope surface generated by these two different approaches. It can be seen that the NURBS surface generated by the proposed approach has a significantly more compact data structure.

Future work will focus on the swept volumes generated by a bull end-mill. The challenge in this future research will be in finding an exact NURBS representation for the silhouette curves of a bull end-mill, since they are not quadric curves.

Chapter 7
A Multi-Agent System for Distributed, Internet-Enabled Cutter/Workpiece Engagement Extraction

7.1 Introduction

CWE calculations are an area of growing importance in process modeling. As B-rep models are widely used in the CAD/CAM field, interest in B-rep model based *CWE* calculations is likewise increasing. However, two difficulties are apparent in the B-rep model based approaches. First, the geometry of the workpiece becomes more and more complicated as the machining simulation progresses, leading to a large data structure with accompanying problems in computational efficiency that grind the simulation to a halt over time. Second, the user who runs the *CWE* calculation must have access to a solid modeler on their computing system to perform the simulation.

To address both of these problems, a distributed, Internet-enabled approach is proposed. Multi-Agent System (*MAS*) technology is playing an increasingly important role in developing intelligent, distributed and collaborative applications. A comprehensive survey and review of *MAS* applications in engineering design and manufacturing is provided in Chapter 2. The current chapter presents an approach that utilizes a *MAS* for the *CWE* calculations. The *MAS* based framework accepts requests from a process engineer for *CWE* calculations for a set of toolpaths (cutter location data). The framework generates one or more *CWE* agents that in turn initiate requests for removal and swept volume calculations from other agents that reside, or are created as needed, over the Internet. The *MAS* is managed to maximize the use of free resources in the framework, leading to greatly improved efficiency in the *CWE* calculation for the job at hand. In addition, because the computational effort is performed by distributed agents instead of on the computer at the user's location, process engineers wanting to perform *CWE* calculations do not need a resident B-rep solid modeler like ACIS. A user can start the calculation as long as they can find a free *CWE* agent somewhere on the Internet/intranet able to execute the request.

The rest of this chapter is organized as follows. Section 2 provides a general description of the key computational steps involved in *CWE* extraction. Section 3 introduces the architecture of the *MAS*, including agent specification and agent collaboration. Section 4 illustrates an example that demonstrates the application of the *CWE MAS*. Section 5 discusses future extensions to this work.

7.2 General Approach to *CWE* Calculations

The strategy for performing *CWE* calculations is designed to facilitate parallel processing where possible. As shown in Figure 7-1, the inputs to the problem include a B-rep representation of the initial workpiece, CLData generated by a CAM system for machining the final part from the workpiece, and cutting tool descriptions. Based on the latter two pieces of information, swept volumes are generated for the tool moving along each toolpath. These are represented as B-rep solid models. By performing Boolean intersections between each swept volume sequentially with the workpiece, removal volumes are generated. Each removal volume is then processed by a *CWE* extractor to identify the engagements at feed steps along the associated toolpath. This procedure involves intersecting a representation of the cutting tool with the removal volume, then manipulating the intersection graph to extract entry and exit angles as a function of depth of cut. It can be seen that steps (1) and (3) can be performed for creating each swept volume and processing each removal volume for its *CWE*s in parallel. Since the creation of removal volumes in step 2 requires an accurate in-process model of the workpiece after the previous tool motion, there are greater restrictions in the ability to perform these operations in parallel. Some research has addressed this issue. In their research, toolpaths are clustered into groups where the in-process workpiece generated by a given group does not affect engagement conditions in other groups, e.g., toolpaths at different Z-levels. Such groups can be processed in parallel.

The *MAS* framework described in the rest of this paper is designed to take advantage of being able to simultaneously perform multiple swept volume and *CWE* calculations per removal volume.

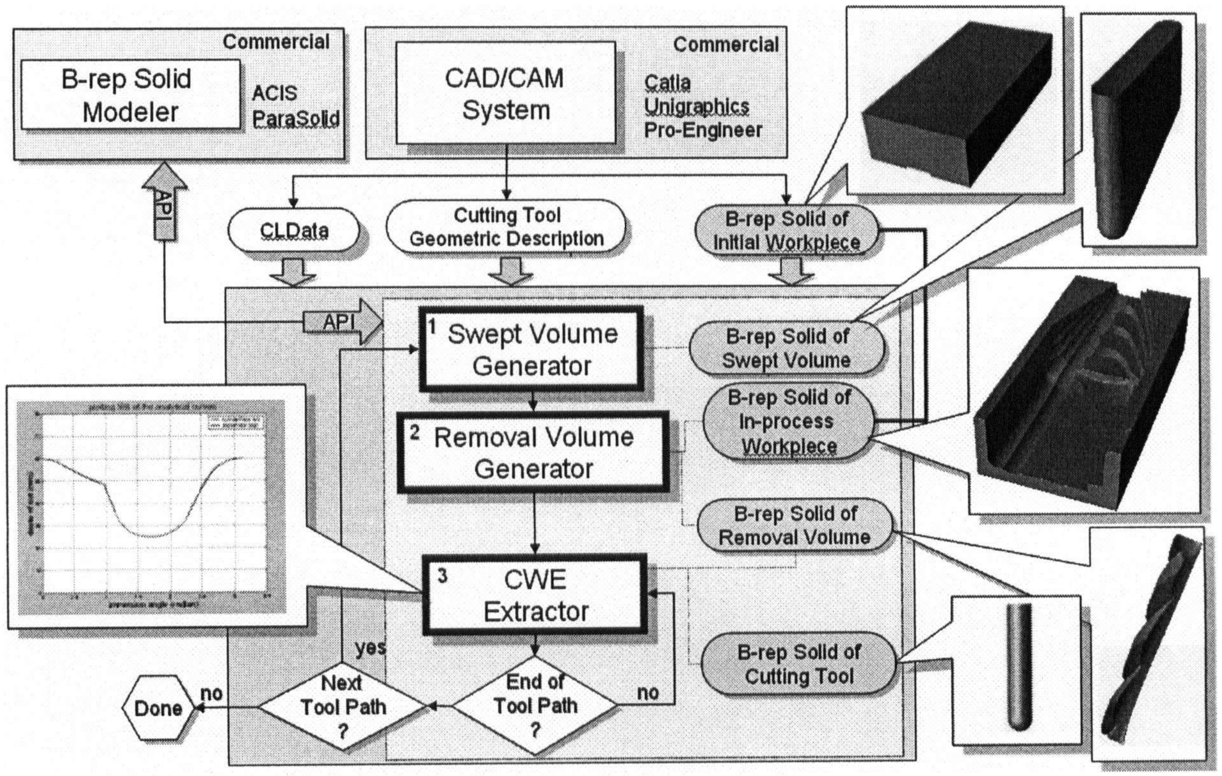

Commercial
B-rep Solid Modeler
ACIS
ParaSolid
Commercial
CAD/CAM System
Catia
Unigraphics
Pro-Engineer
API
CLData
Cutting Tool Geometric Description
B-rep Solid of Initial Workpiece
API
1 Swept Volume Generator
2 Removal Volume Generator
3 CWE Extractor
B-rep Solid of Swept Volume
B-rep Solid of In-process Workpiece
B-rep Solid of Removal Volume
B-rep Solid of Cutting Tool
yes
no
Done
Next Tool Path ?
End of Tool Path ?
no

Figure 7-1: Steps in *CWE* Calculation

7.3 Multi-Agent Framework for *CWE* Calculations

An agent is an intelligent, autonomous and self-adaptive entity that is able to reason, make decisions and respond accordingly within the application in question. In complicated applications, individual agents may lack the ability to solve a problem. *MAS* technology has been developed to make agents work together to deliver solutions to problems that are beyond the individual capabilities or knowledge of each agent. *MAS* technology can also manage agents to achieve parallel computation by breaking tasks into several independent subtasks that can be handled by separate agents.

To describe a Multi-Agent framework, it is important to understand the functionality of individual agents and how they interact with each other to reach their individual or shared goals. In this section, a specification is provided for each agent in the *CWE* calculation *MAS*. Following this, the collaborative strategy of agents for *CWE* calculation will be explained.

7.3.1 Agent Specification in *CWE* Calculation *MAS*

Figure 7-2 shows the agents that have been identified for the *CWE* calculation *MAS*. Eight agents have been defined. The three primary computational agents are:

- Cutter Workpiece Engagement Agent (*CWE*): This agent is responsible for performing the intersection between the cutting tool and the removal volume geometries to find the actual engagement geometry. To do this, it must coordinate with other *CWE* agents, since several of them may be created to handle computations in parallel for a large toolpath file. It is also responsible for coordinating with removal volume agents that must provide the solid model of the volume for processing.
- Removal Volume Agent (*RV*): This agent applies solid modeling intersection operations between the swept volumes of the cutter moving along selected toolpaths and the appropriate in-process workpiece state. To do this, it must be able to retrieve or create a model of the correct in-process workpiece state and models of all the selected swept volumes. The latter are generated by interactions with swept volume agents.
- Swept Volume Agent *(SV)*: Like the *RV* agents, these agents apply solid modeling operations to generate the geometry of the swept volume of the cutter along a toolpath

In addition to the computational agents, Interface Agents (*IA*) have been developed for each to facilitate communication with the user of the *MAS*. These agents are tailored to provide the functionality that the user needs to interact effectively with the computational agent it is associated with. Interface Agents act as a thin-client on the user's computer, collecting user inputs and visualizing the results. For visualizing solid models, the *IA*s render polyhedral models that are generated from the solid model by faceting functionality within the solid modeler used by the associated computational agent. This thin-client structure makes the user's computer independent of a heavy computational kernel. This feature supports use of the application by users who have not installed a solid modeler or complicated graphics engine on their system. For example, the *CWE* interface agent requires a graphical user interface to display the results of the engagement calculations. This interface is illustrated in Figure 7-12. As can be seen, through this agent the user can visualize in-process workpieces, swept volumes, removal volumes and the *CWE*s themselves.

The remaining two agents are responsible for data management (*DA*) and to provide registration/deregistration of the agent with on-line agent services (*DF*).

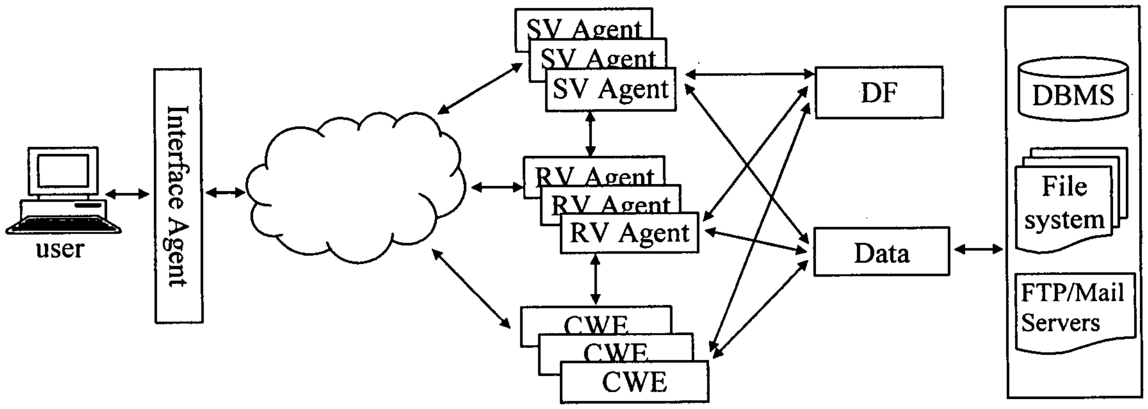


Figure 7-2: Agents in *CWE* Calculation *MAS*

7.3.2 Agent Collaboration in *CWE* Calculation *MAS*

In this section, we will describe how computational agents interact with each other to achieve parallelism. First of all a general description for collaboration strategies within the *CWE* calculation *MAS* is provided. Agent actions that will be performed during interaction and collaboration in the *MAS* need to be identified. Since each computational agent holds several different states, a Finite State Machine (*FSM*) model is introduced for agents to manage their states during collaboration. Finally, agent collaboration will be discussed by introducing three protocols to explain how agents cooperate with each other to complete tasks in parallel.

7.3.2.1 Overview

Requests to *CWE*, *RV* or *SV* agents may be for either a completely new or previously started job. This is significant, since machining is a sequential process. The existence of swept and removal volumes calculated during a previous job for parts of a tool path file can be used to reduce the computational effort needed when engagements for other tool motions within the same file are required. To simplify the discussion, it is assumed that the *CWE* calculation request is submitted for the first time for a given tool path file. Thus, there is no historical data stored in the data agent, and complete *CWE*, *RV* and *SV* calculations are required.

As shown in Figure 7-3, computational agents are grouped into several levels. From top to bottom, there are the *IA* group, *CWE* group, *RV* group and *SV* group. In this layered structure, the calculation requests flow downwards and results are submitted in the upward direction. There are two types of parallelism that can be achieved with this architecture:

- Parallelism within a group:

  To achieve parallelism within a group, agents are divided according to two different roles: one is the *master agent*, which interacts with one or more *slave agents*. The master agent is responsible for decomposing and allocating calculation tasks to multiple slave agents. The task scheduling activity that applies a strategy for allocating tasks among slave agents is performed by the master.

- Parallelism between adjacent groups:

  After sending calculation requests to an agent in a lower-level group, a simple strategy is for an upper-level agent to wait for each result before continuing its calculations. However, to achieve parallelism, agents at lower levels can instead be tasked to submit a set of results at a time so that agents in upper levels can process this result set in parallel. Master agents in each group take the responsibility for collecting results from their slave agents, grouping these into sets and submitting them to the master agent in the upper level. A *Result Passing Strategy* for determining when results should be transferred to an upper level is defined for each master agent operating on a lower level.

The master agent for each group is selected by the master agent that resides in the level immediately above, using a *Master Agent Selection* strategy.

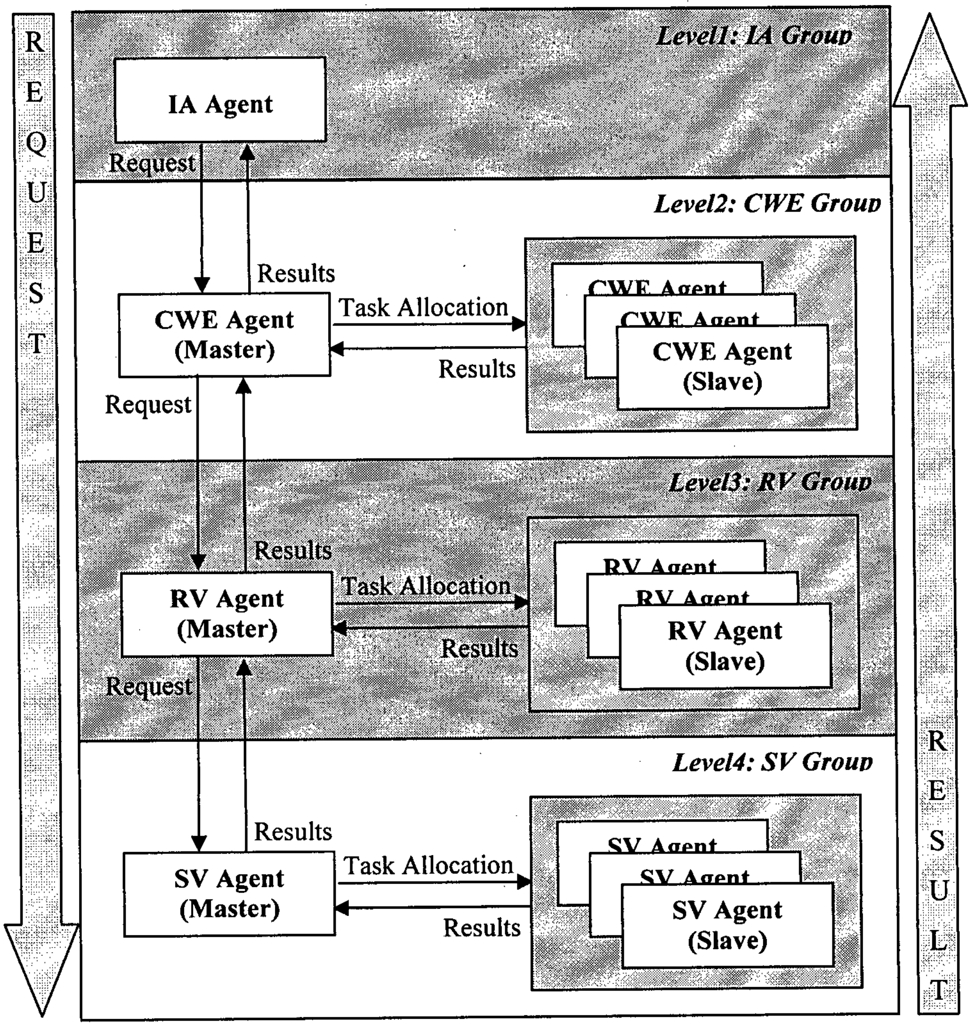


Figure 7-3: Agent Collaboration in *CWE* Calculation *MAS*

To better understand the nature of the collaborations within the *CWE* calculation *MAS*, the following scenario is described. First, the user finds a *CWE* calculation service through the Internet or on a company-wide intranet where the *MAS* is active. The user must then download a light-weight *IA* for use to input the calculation parameters and to visualize the calculation results. This *IA* allows the user to specify the workpiece and toolpaths to be processed with the *MAS*. The *IA* also facilitates uploading workpiece and toolpath files to the data agent from the user's computer. When the user broadcasts a calculation request, the interface agent selects a *CWE* agent that is identified by the *DF* agent as the master *CWE* agent and sends the request to this master agent. The master *CWE* agent accepts the request and extracts the calculation parameters, as well as the addresses of the workpiece and toolpath files, from the calculation request message. If the master *CWE* agent finds that *RV* calculations are necessary, it locates a master *RV* agent by using the *DF* agents and sends a request to it. The *RV* agent in turn checks with the *DA* to determine if the data it needs is available (i.e., the swept volume and in-process workpiece). If not, its task is delayed until a calculation request to an *SV* agent is satisfied. When the required geometry for the removal volumes is obtained, the master *CWE* agent decomposes the *CWE* calculation task and distributes it to other slave *CWE* agents in the *MAS*. Finally, the master *CWE* agent re-combines the calculation results returned by the slave *CWE* agents and returns them to the interface agent. The user can view the results using the *CWE* interface agent. The *RV* and *SV* agents follow a similar procedure when they receive a calculation request.

Figure 7-4 illustrates a comparison between sequential *CWE* calculation and the parallel *CWE* process implemented in the *MAS* framework. Calculation tasks are decomposed and distributed among agents in each group for parallel computation. The overlap between tasks for the different groups is achieved using the results passing strategy.

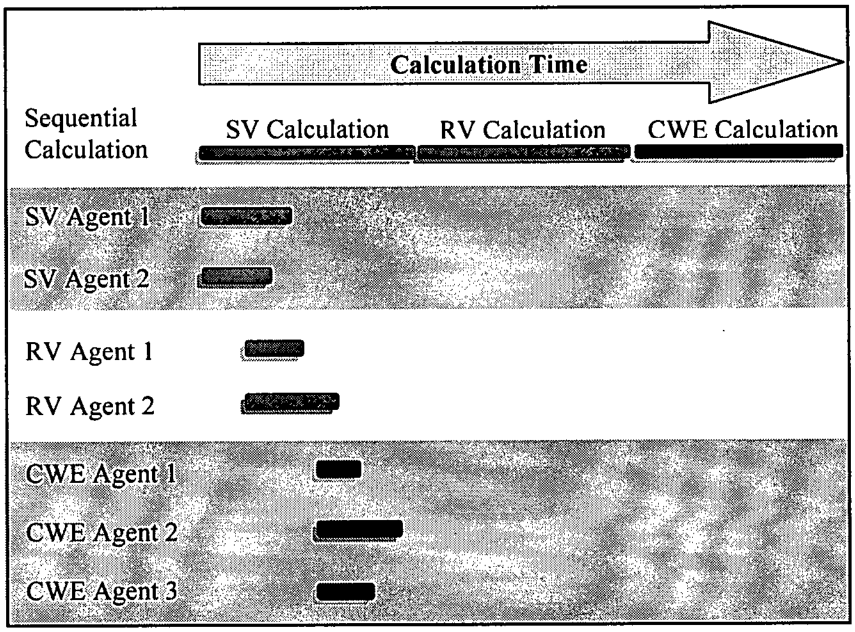


Figure 7-4: Parallelism in *CWE* Calculation *MAS*

From the above example, three actions can be identified for master agents to achieve these two types of parallelism during collaboration:

- Task Scheduling
- Master Agent Selection
- Results Passing

Detailed descriptions of these three agent actions will be provided in the next section.

7.3.2.2 Agent Actions

To indicate how busy the system is where the agent is located, an *idle factor (K)* is defined for each agent. Considering that CPU and memory usage rates are the two most important indications of busyness, they are used to calculate the idle factor as follows:

$$K = \frac{K_c + K_m}{2} \tag{7.1}$$

where $K_c = 1 - R_c$ is the idle factor for CPU usage, with R_c being the CPU usage rate. Similarly, $K_m = 1 - R_m$ is the idle factor for memory usage, while R_m is memory usage rate.

From Eq. (7.1), we can see that K is an average of K_c and K_m , and $K \in [0,1]$. The factor K will be determined for each agent to indicate its idleness. K plays an important role in the agent collaboration process, since most strategies (such as Task Scheduling, Master Agent Selection, etc.) are based on assigning tasks to agents that are the least active.

<u>Task Scheduling</u>

To schedule tasks among slave agents in the same group, the master agent broadcasts availability requests to slave agents that are identified by the *DF* agent based on their availabilities and idle factors. After receiving responses, a master agent maintains an agent list for all available slave agents. Assuming that slave agents $\{A_i \mid i = 1,2,\ldots, m\}$ are in an agent list, their tasks $\{T_i \mid i = 1,2,\ldots m\}$ can be allocated based on their idle factors $\{K_i \mid i = 1,2,\ldots m\}$ as described by Eq. (7.2),

$$T_i = T \times K_i / \sum_{i=1}^{m} K_i \tag{7.2}$$

where

T_i: Task allocated to the *i*-th slave agent

K_i: Idle factor of the *i*-th slave agent

T: Total tasks to be allocated

From Eq. (7.2), it can be seen that a master agent creates a schedule based on the percentage of total idle factors that a given slave agent has. This strategy guarantees that slave agents with larger idle factors will take on more computational tasks.

Master Agent Selection Strategy

The master agent in each group is determined by the master agent in the immediate upper-level group. The master agent requests availability from all agents in a lower-level group. Based on their idle factors, the agent with the largest idle factor is selected as the master agent. The master agent in the upper-level group then sends all of the calculation requests to the newly designated master agent in the lower-level group. This master agent then decomposes and distributes the tasks to the slave agents in its group. In the *CWE* calculation *MAS*, the master *CWE* agent is selected by an interface agent, the master *RV* agent by the master *CWE* agent and the master *SV* agent by the master *RV* agent.

Results Passing Strategy

The results passing strategy is designed to achieve parallelism between adjacent levels. This strategy utilizes a *Slide Window* to coordinate results submission from a lower level to an upper level. The master agent in a lower level maintains a task queue after receiving calculation requests from the master agent in the upper level. The slide window acts as a 1-dimension window sliding along this queue, starting from the first task. An integer parameter called *Slide Window Size* (*SWS*) specifies how many tasks the window spans at any position along the task queue. When the calculations for all the tasks located within the slide window are completed and submitted by slave agents, the results are submitted from the master agent in the lower level to the master agent in the upper level that requested the service. The slide window then slides *SWS* tasks along the task queue to a new position, where the master agent waits for a new set of results from its slave agents for the next submission.

An example of this approach is shown in Figure 7-5. A master *RV* agent receives six toolpath segments for calculation from the master *CWE* agent and distributes these tasks to several slave *RV* agents. Given that a *SWS* of 3 is used, the slide window first spans from segment 1 to segment 3. After receiving removal volumes 1 and 2 from slave agent 1, the master *RV* remains in a waiting state, since removal volume 3 is still not available. Once removal volumes 3, 4 and 5 are submitted by slave agent 2, all three removal volumes are now available and can be submitted to the master *CWE* agent by the master *RV*. After the submission is complete, the window slides to a span of the task queue starting at segment 4.

The parameter *SWS* has significant influence on the efficiency of the results passing strategy. The smaller its value, the greater the parallelism is between adjacent groups. However, this comes with a heavier network load. In the *CWE* calculation *MAS*, results passing using this approach is performed by the master *CWE*, *RV* and *SV* agents.

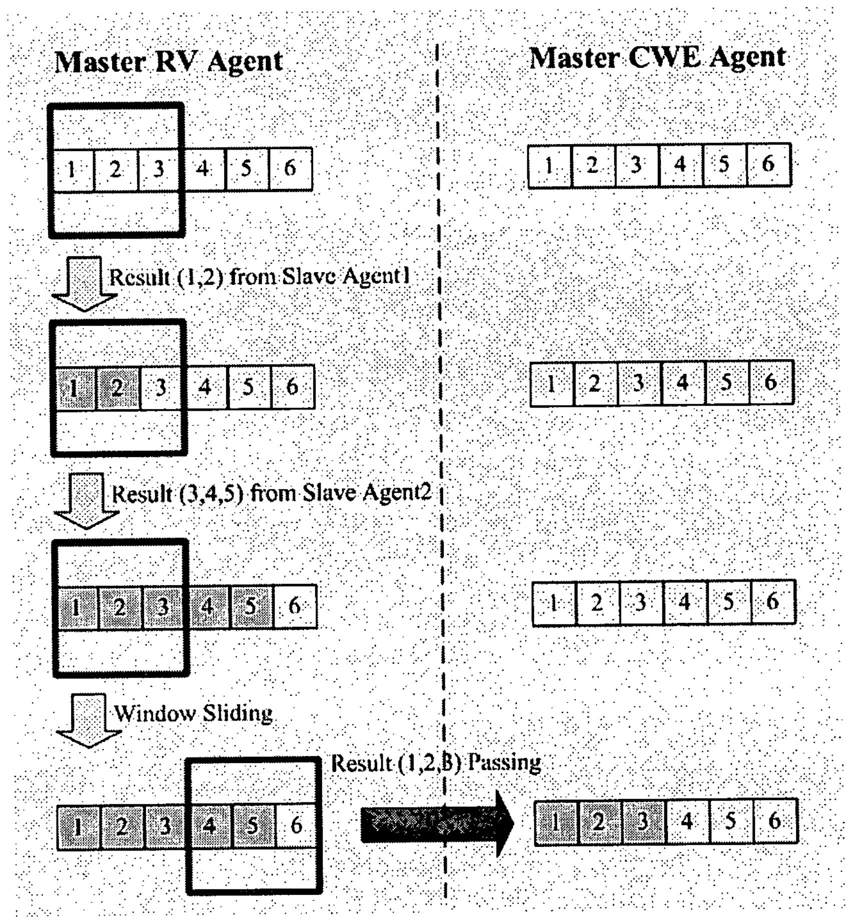


Figure 7-5: Slide Window for Results Passing

7.3.2.3 Agent States

Each agent manages its state using a Finite State Machine (*FSM*) model. There are in all five states for an agent: IDLE, WAIT, CALCULATION, SUCCESS and ERROR. In each state, only the expected messages are handled; unexpected ones are simply discarded. These states are described as follows:

- IDLE: This is the initial status of an agent. An agent in this state is free and available for queries and calculation requests.
- WAIT: When an agent receives a query from another agent, it responds with RES_AVAILABLE (Available) and switches to a WAIT state if it is currently IDLE. Otherwise, the query message is simply discarded or a RES_UNAVAILABLE response is given. An agent in this state waits for a calculation request from the querying agent. If the request arrives, it switches to a CALCULATION state. If the request does not arrive in a specified time, a timeout occurs and it returns to the IDLE state.
- CALCULATION: When an agent in an IDLE or WAIT state receives a calculation request, it responds with RES_AVAILABLE and switches to the CALCULATION state. At this point, a new session is created. A session that is attached to an agent starts when the calculation request arrives and ends when the calculation finishes. As mentioned above, in the framework there is one agent selected as a master, with the others performing as slaves. For slaves, a session ends when its own job is finished, while for a master a session does not end until all calculations it has distributed over slave agents are finished in addition to any master specific tasks, such as recombining the results from slaves. After a session is created, an agent begins task scheduling, which results in it doing the job by itself, sending calculation requests to other agents or distributing sub-tasks over slave agents.
- SUCCESS: When a session finishes successfully, the agent transfers to a SUCCESS state. Once it reaches this state, it informs the requesting agent of the calculation result and returns to an IDLE state.
- ERROR: When there is any error during the calculation or a timeout is caused by a native calculation process or a delayed response from any slave agent, the agent transfers to an

ERROR state. After reporting the error type and description, it automatically transfers to an IDLE state.

Figure 7-6 shows the *FSM* model of a receiver agent. When the agent is in a CALCULATION state and sends requests to other agents, it also doubles as a sender agent. To simplify the state management, its state does not change but remains in the CALCULATION state. This means that in the current architecture an agent handles only one session at a time.

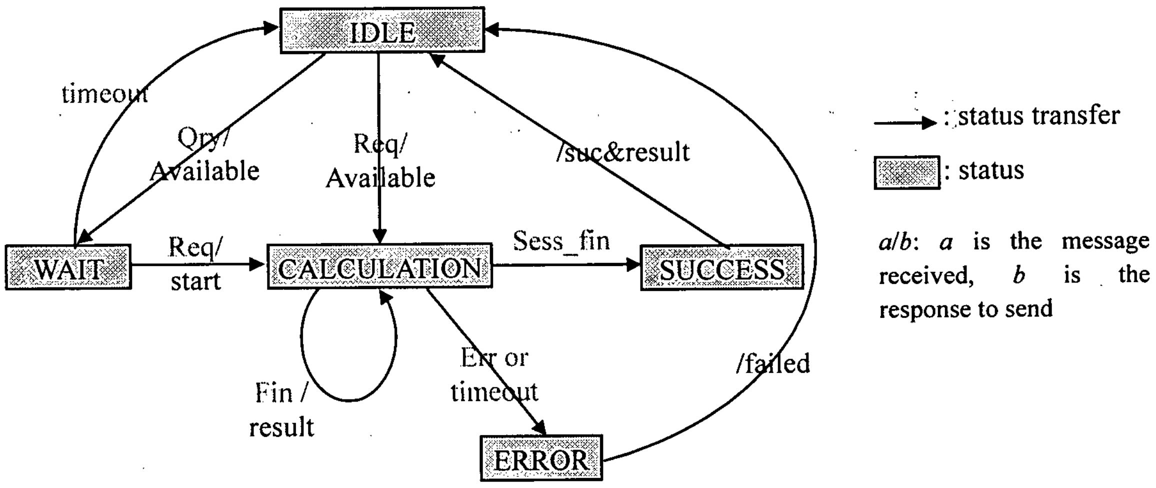


Figure 7-6: *FSM* Model of a Receiver Agent

7.3.2.4 Agent Collaborations

Based on the agent actions and states discussed above, this section describes how agents collaborate with each other to realize parallel computation. In *CWE* calculation *MAS*, three collaborations, *Selecting Master Agent*, *Allocating Tasks* and *Coordinating Results Submission*, are performed to achieve parallelism. Each of these collaborations will be explained with an interaction protocol.

Collaboration 1: Selecting Master Agent

As mentioned previously, the master agent in each group is selected by the master agent in the immediate upper level group. Each of the three levels in the *MAS* has a master agent. Initially, the master *CWE* agent is selected by the interface agent interacting with the end user. Following this, the master *CWE* agent selects the master *RV* agent, and the master *RV* agent selects the master *SV* agent.

Figure 7-7 illustrates a scenario that shows how a master *RV* agent identifies a master *SV* agent. A master *RV* agent broadcasts RES_AVAILABLE messages to all *SV* agents for their availability. At the time when the *SV* agents receive this message, *SV* agent 1 and 2 are in the IDLE state, and *SV* agent 3 is in the WAIT or CALCULATION state. *SV* agents 1 and 2 respond to the master *RV* agent with RES_AVAILABLE messages along with their idle factors, 0.4 and 0.8, respectively. *SV* agent 3 responds with the RES_UNAVAILABLE message. Master *RV* agent maintains an available *SV* agent list that contains *SV* agent 1 and 2 and their associated idle factors. Master *RV* agent performs Master Agent Selection (described in previous section) to select *SV* agent 2 as the master *SV* agent for the *SV* agent group. Upon completion of the selection process, the master *RV* agent sends calculation tasks 1, 2, 3, 4, 5 and 6 to the master *SV* agent.

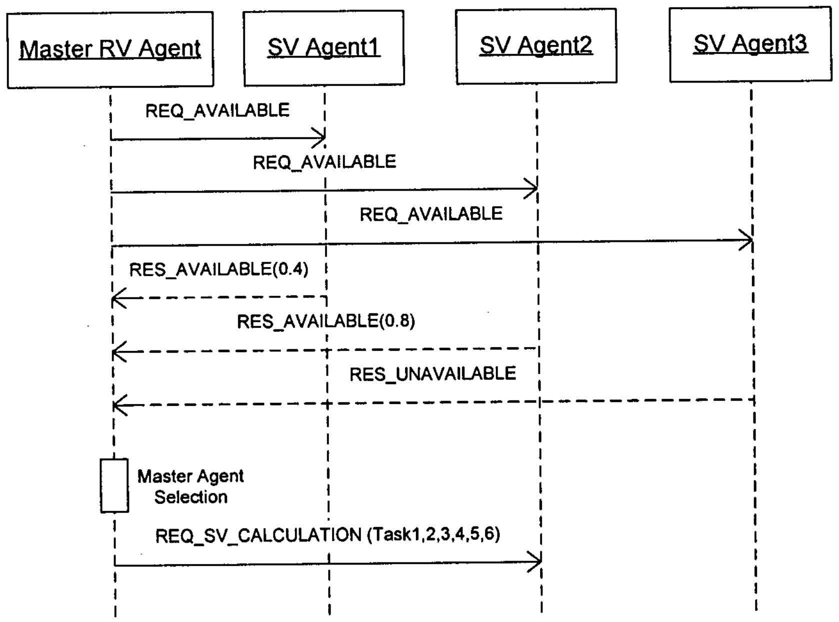


Figure 7-7: Protocol for Collaboration 1 (Selecting Master Agent)

Collaboration 2: Allocating Tasks

The master agent in each group decomposes tasks and distributes them to slave agents according to their idle factors. The master agent at each level broadcasts to all slave agents in the same group for their available state. The master agent forms a list of available slave agents after receiving responses. As shown in Figure 7-8, the agent list includes slave agent 1 with idle factor 0.4 and slave agent 2 with idle factor 0.8. The master agent utilizes the task scheduling strategy discussed in the last section to allocate the six tasks, 1, 2, 3, 4, 5 and 6. Tasks 1 and 2 are assigned to slave agent 1, and tasks 3, 4, 5, 6 are assigned to slave agent 2.

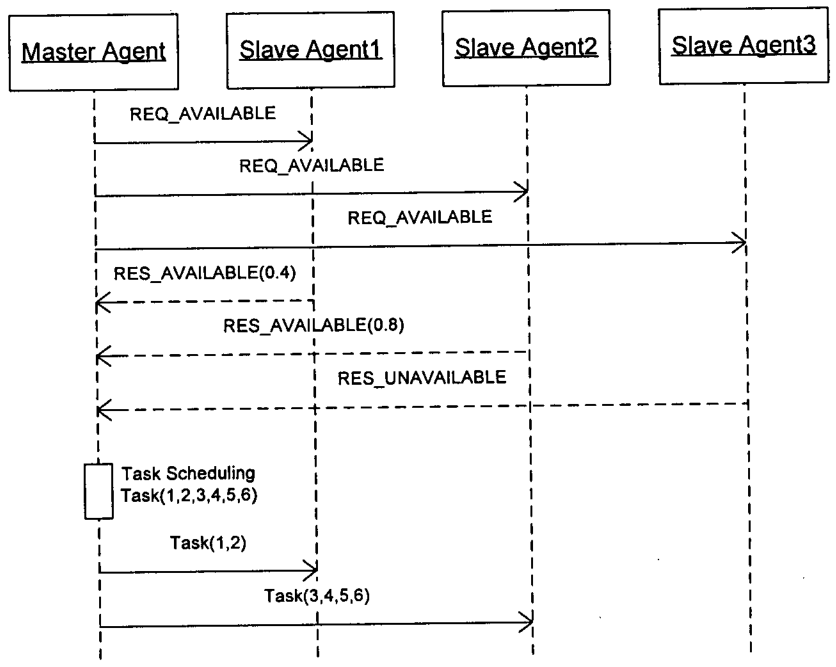


Figure 7-8: Protocol for Collaboration 2 (Allocating Tasks)

Collaboration 3: Coordinating Results Submission

The master agent in each group submits calculation results collected from the slave agents to the master agent in the immediate upper-level group. Starting with the master *SV* agent, swept volumes are submitted to the master *RV* agent. The master *RV* agent sends removal volumes collected from the slave *RV* agents to the master *CWE* agent. Finally, the *CWE*s are submitted by the master *CWE* agent to the interface agent for visualization.

Figure 7-9 shows how a master *RV* agent, 3 slave *RV* agents and a master *CWE* interact with each other for this collaboration. In this example, the master *RV* agent will use the Results Passing Strategy introduced in the last section to determine when the removal volumes are to be submitted to the master *CWE* agent. After receiving tasks (1,2,3,4,5,6) from the master *CWE* agent, the master *RV* agent distributes tasks 1 and 2 to the slave *RV* agent 1, tasks 3, 4 and 5 to slave *RV* agent 2 and task 6 to slave *RV* agent 3. Since the Slide Window Size for results passing is set to 3, the master *RV* agent sends results 1,2,3 to the master *CWE* agent only after slave *RV* agent 1 completes and submits its tasks (results 1,2) and slave *RV* agent 1 completes and submits its tasks (results 3,4,5). Once slave *RV* agent 3 sends its calculation result 6, the master *RV* agent submits results 4,5,6 to the master *CWE* agent.

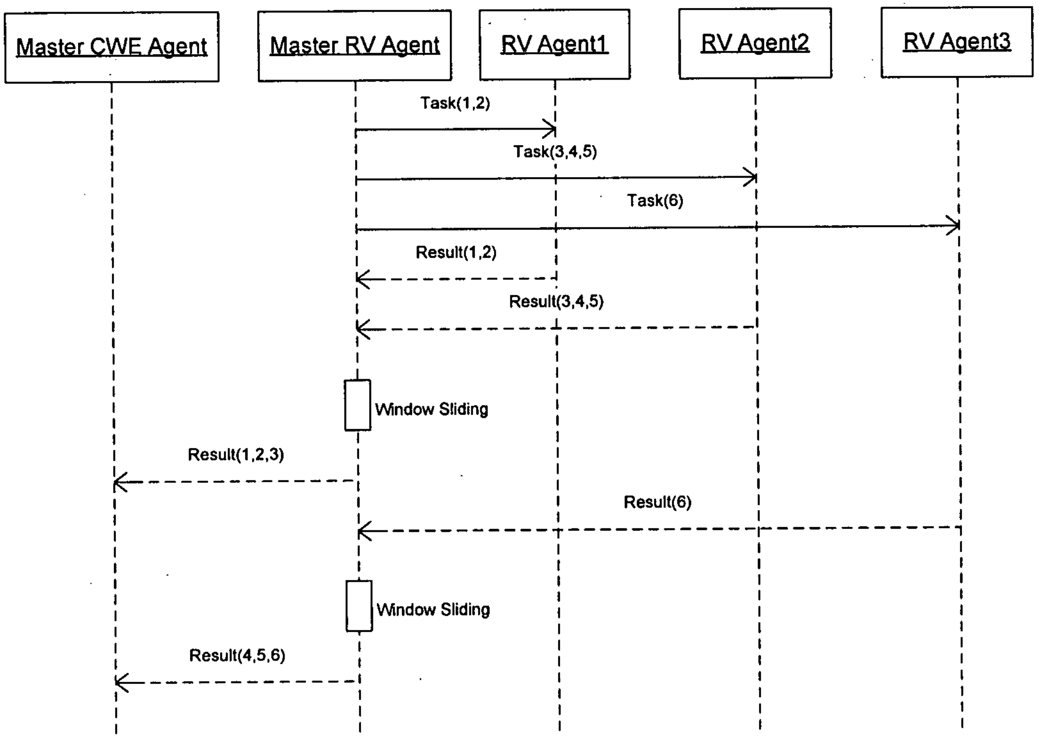


Figure 7-9: Protocol for Collaboration 3 (Coordinating Results Submission)

7.4 Implementation and Example

To standardize the development of *MAS*, the Foundation for Intelligent Physical Agents (FIPA) [18] was founded in 1996. The main goal of FIPA is to specify guidelines for agent management and communication. JADE (Java Agent Development Framework) [18] [19] is a FIPA-compliant middle-ware for facilitating FIPA compliant *MAS* development. Agents developed by JADE can be easily distributed across machines and different operating systems. The *CWE* calculation framework is implemented using the JADE platform.

As shown in Figure 7-10, all of the agents in the *CWE* calculation *MAS* are implemented using JADE. Ontologies are defined to facilitate the communication among agents. Java Swing and Java 3D are used in interface agent development for user interface and 3D visualization. Computational modules such as those for *CWE, RV* and *SV* calculations are built using ACIS R13.0 and implemented as DLLs (Dynamic Link Library) by Visual C++. The corresponding computational agents implemented with Java on the JADE platform call these C++ modules through JNI (Java Native Interface), the Java interface for calling software modules written in languages other than Java.

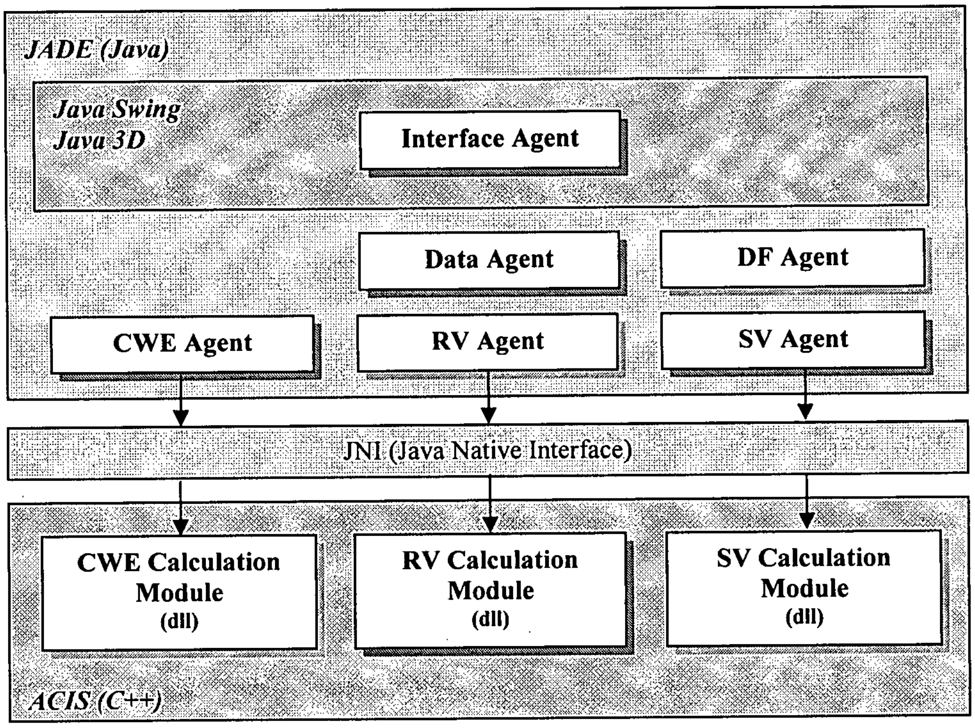
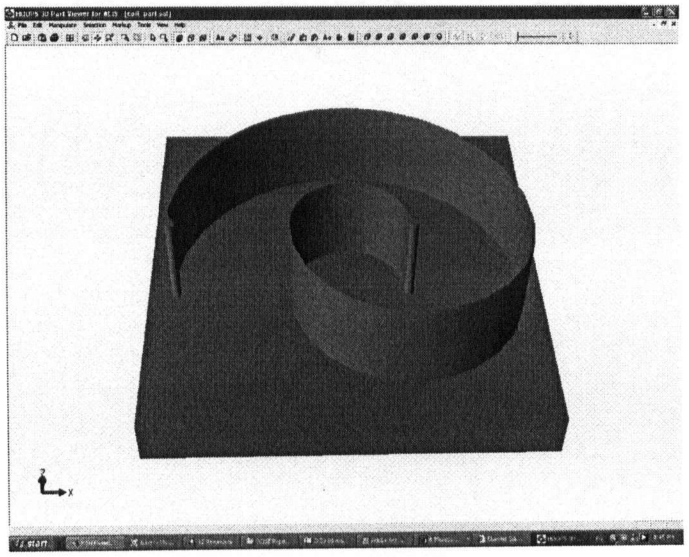

JADE (Java)
Java Swing
Java 3D
Interface Agent
Data Agent
DF Agent
CWE Agent
RV Agent
SV Agent
JNI (Java Native Interface)
CWE Calculation Module (dll)
RV Calculation Module (dll)
SV Calculation Module (dll)
ACIS (C++)

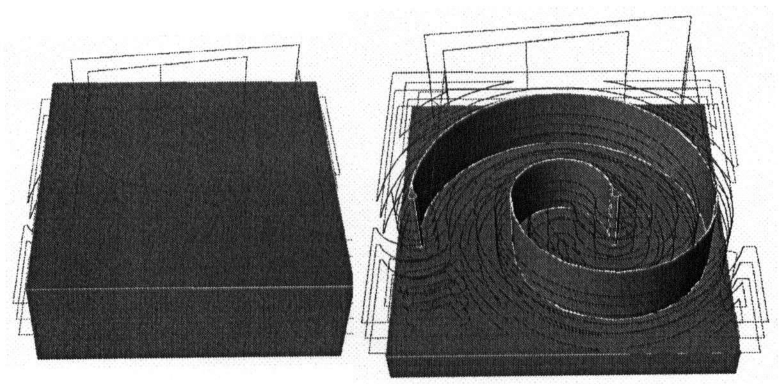
Figure 7-10: Implementation of *CWE* Calculation *MAS*

To test the effectiveness of the *CWE MAS*, a milling process for a spiral part is simulated, as shown in Figure 7-11. Figure 7-11 (a) shows the final part, (b) the initial workpiece, and (c) the toolpaths. There are in total 410 toolpath segments in this milling process. The test was conducted over an Intranet where three workstations (Pentium 4 CPU 1.8GHz and 512MB RAM, 100Mb Network Adaptor) are located. Five test cases were performed on this test part with all of the 410 toolpath segments. The calculation times are shown in Table 7-1. Test case 1 is a sequential *CWE* calculation where all three calculations are executed on one workstation. In test case 2, six agents (2 *CWE*s, 2 *RV*s and 2 *SV*s) are generated in workstation 1. Each agent has its own process. Compared to test case 1, calculation time is reduced significantly, since parallel computation is performed in test case 2. In test case 3, agents are distributed to different workstations (2 *CWE*s in workstation 1, 2 *RV*s in workstation 2 and 2 *SV*s in workstation 3). The calculation time is further improved since agents have more computational resources to utilize. By increasing the agent numbers in each workstation, additional improvements in the calculation time can be observed.

Figures 7-12 (a), (b), (c) and (d) show the visualization within a *CWE* Interface Agent of the *SV*, *RV*, in-process workpiece and the *CWE* results from the 398th toolpath segment, respectively.



(a) Final Part

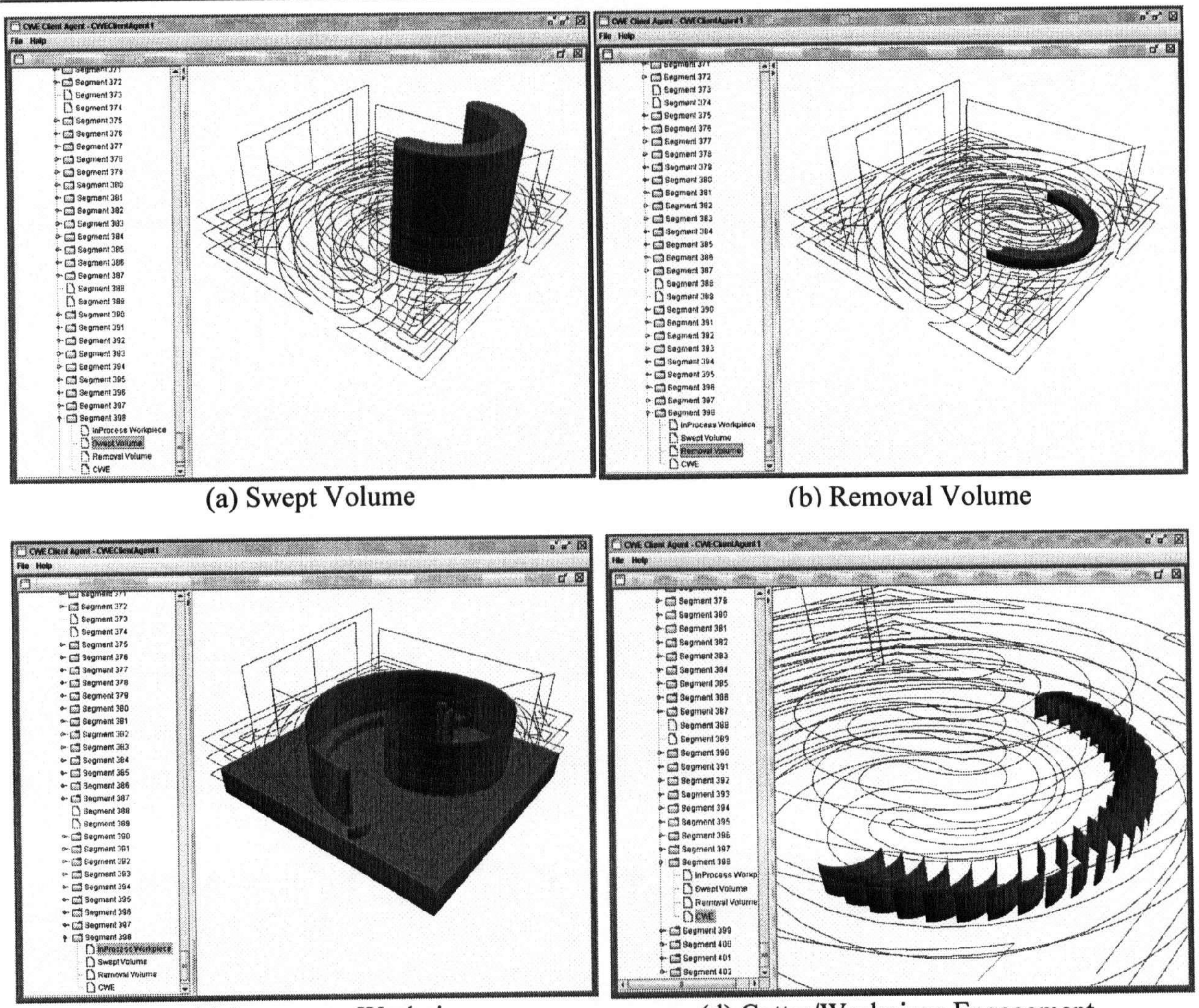


(b) Initial Workpiece (c) Toolpaths

Figure 7-11: Test Part

Test #	Workstation 1			Workstation 2			Workstation 3			Calculation Time (Second)
	CWE	RV	SV	CWE	RV	SV	CWE	RV	SV	
1	1	1	1	0	0	0	0	0	0	68
2	2	2	2	0	0	0	0	0	0	37
3	2	0	0	0	2	0	0	0	2	31
4	3	0	0	0	3	0	0	0	3	27
5	4	0	0	0	4	0	0	0	4	24

Table 7-1: Calculation Time in *CWE* Calculation *MAS*


(a) Swept Volume (b) Removal Volume

(c) In-process Workpiece (d) Cutter/Workpiece Engagement

Figure 7-12: *CWE* Calculation for the 398th Toolpath Segment

(from *CWE* Interface Agent)

7.5 Discussion

One of the challenges not addressed in the *CWE MAS* research to-date is the management of data transfer for the in-process workpiece by the data agent to the *RV* agent. To correctly generate the removal volume for a toolpath, the swept volume must be intersected with this B-rep model and the model itself updated. The size of this B-rep model and the corresponding memory and data transfer time increase significantly as the simulation progresses. Since several slave *RV* agents (plus the master), each requiring access to the in-process B-rep model, can exist at any given time, the ability to transfer the model efficiently will greatly impact the overall efficiency of the *MAS*. A decomposition of the in-process workpiece model is one option for distributing the B-rep model in pieces as needed to different *RV* agents, reducing the size of the data transferred and further increasing the parallelism of processing. As part of future work, the data agent will be expanded to include this functionality.

While the current approach is B-rep based, other computational kernels can be utilized (e.g., Z-buffer or Polyhedral Modeling) for agents in a manner that make them transparent to other agents. Thus, a user can initiate a calculation once a *CWE* agent employing any kernel can be located on the Internet or an Intranet. A hybrid system of this type can be further leveraged to enhance processing efficiency.

As mentioned in the introduction, *CWE* extraction is part of a Virtual Machining methodology. The other components of *VM* involve process modeling and optimization. As part of future research, agents for performing these steps will be implemented and integrated into the current *MAS*. These will include force calculation agents, tool-life prediction agents, chatter and vibration agents and process parameter optimization agents. Many of these process modeling activities are computationally intensive and can benefit greatly from implementation within the current framework.

Chapter 8
Conclusion

8.1 Contribution

In this thesis, new methodologies are proposed to facilitate the extraction of *CWE* geometry in milling process modeling. This includes a feature-based approach for *CWE* calculation in 2½D end milling and hole milling processes, and a *MAS* and B-rep model based *CWE* calculation framework. The feature-based approaches are verified experimentally by conducting a geometric simulation on a test part requiring 2½D end milling and hole milling processes. The *MAS* and B-rep model based framework is implemented with a multi-agent development toolkit, JADE, and Java. *CWE* extraction in the framework is tested on several workstations located in the Intranet. The computational efficiency is verified by observing the processing times with different agent configurations in the framework.

The contributions of this thesis are summarized as follows:

- Introduced the concept of an in-process machining feature that can be used to address the computational challenges in process modeling. Geometric Invariant Machining Features (*giF*) and Form Invariant Machining Features (*fiF*) are defined to represent engagement conditions analytically in 2½D end milling. A *giF* characterizes regions of the removal volume where the *CWE* is identical, and a *fiF* defines regions where the *CWE* can be represented in a predictable way. The introduction of *giF*s and *fiF*s facilitates *CWE* extraction and representation.
- Developed a feature extraction algorithm for recognizing *giF*s and *fiF*s in the removal volumes by applying volume decomposition.
- Proposed an analytical approach to extract *CWE* from *giF*s and *fiF*s. This approach provides a closed-form solution. The robustness and computational efficiency of *CWE* extraction are significantly improved.
- Developed a NURBS based swept volume generation method for helical toolpaths with multiple cutting tools to machine hole type features. This approach generates NURBS

represented boundary surfaces of the swept volume. A surface trimming procedure is applied to the self-intersected surfaces. This approach can provide an exact solid model for the swept volume, and the swept volume is used for in-process workpiece modeling for the purposes of NC verification.

- Developed an analytical approach to *CWE* calculation for hole milling. This approach provides a closed-form representation for *ceF* based on the parameters of the hole geometry and milling operation.
- Proposed a *MAS* and B-rep model based *CWE* calculation framework. This framework can improve the efficiency of the *CWE* calculations by parallelizing computational activities. The *MAS* is managed to maximize the use of free resources in the framework, leading to greatly improved efficiency in the *CWE* calculation for the job at hand. In addition, because the computational effort is performed by distributed agents instead of on the computer at the user's location, users without a resident B-rep solid modeler can access remote *CWE* calculations services over the Internet or an Intranet. The proposed framework is also designed to facilitate integration of other non-B-rep based *CWE* calculation methods.

8.2 Future Work

The future research work outlined in this thesis is summarized as taking the following directions:

- Extending feature-based methodologies to end-mills with more complicated geometry for 2½D end milling:

  The feature-based methodologies can be extended to various tool types, such as ball, cone and bull end-mills.
- Exploring the possibility of introducing the concept of an in-process machining feature into 3 and 5-axis machining:

  The feature-based methodologies need to be extended to broader application domains, such as 3 and 5-axis machining.

- Expanding feature-based methodologies to different machining operations, such as turning and boring.

  The feature-based approach detailed in this thesis focuses on process modeling for milling process. Future work will focus on wider types of machining operations, including turning and boring.

- Developing a simplified workpiece model to reduce data transfer time in the *MAS*:

  Complicated workpiece models in *MAS* require enormous memory and transfer time in the computer network. Research efforts are needed to develop a simplified in-process workpiece model to improve the efficiency of data exchange.

- Developing and integrating new agents into the current *MAS* for facilitating the calculations in Virtual Machining:

  More time-consuming calculations are involved in Virtual Machining than the *CWE* extraction discussed in this research. New agents for performing these computationally intensive activities need to be designed and integrated into *MAS*.

Bibliography

1. Abdel-Malek, K., Yeh, H.J., *Geometric representation of the swept volume using Jacobian rank-deficiency conditions*, Computer-Aided Design, Vol.29 No.6, 1997, pp. 457-468.
2. Abdel-Malek, K., Blackmore, D., Joy, K., *Swept volumes: foundations, perspectives and applications*, International Journal of Shape Modeling, 2004.
3. Altintas, Y., Spence, A.D., *End milling force algorithms for CAD systems*, Manufacturing Technology CIRP Annals, Vol.40, 1991, pp. 31-34.
4. Altintas, Y., Spence, A.D., *A solid modeler based milling process simulation and planning system*, Transactions of the ASME, Journal of Engineering for Industry, Vol.116, 1994, pp. 61-69.
5. Blackmore, D., Leu, M.C., Shih, F., *Analysis and modeling of deformed swept volumes*, Computer-Aided Design, 1994, Vol.26, 1994, pp. 315-326.
6. Blackmore, D., Leu, M.C., Wang, L.P., *The sweep-envelope differential equation algorithm and its application to NC machining verification*, Computer-Aided Design, Vol.29 No.9, 1997, pp. 629-637.
7. Blackmore, D., Samulyak, R., Leu, M.C., *Trimming swept volumes*, Computer-Aided Design, Vol.31 No.3, 1999, pp. 215-224.
8. Chang, K., Goodman, E., *A method for NC toolpath interference detection for a multi-axis milling system*, ASME Control of Manufacturing Process, DSC 28/PED 52, pp. 23-30.
9. Chuang, S.H., Henderson, M.R., *Three-dimensional pattern recognition using vertex classification and vertex-edge graphs*, Computer-Aided Design, Vol.22 No.6, 1990, pp. 377-387.
10. Chung, Y.C., Park, J.W., Shin, H., Choi, B.K., *Modeling the surface swept by a generalized cutter for NC Verification*, Computer-Aided Design, Vol.30 No.8, 1998, pp. 581-593.
11. Coles, J., Crawford, R., Wood, K., *Form feature recognition using base volume decomposition*, Proceedings of ASME Design Automation Conference, 1994, pp. 281-297.
12. Corney, R.C., *Graph-based feature recognition*, Ph.D. Dissertation, Heriot-Watt University, 1993.
13. Cutkosky, M.R., Tenenbaum, J.M., Glicksman, J., *Madefast: Collaborative Engineering over the Internet*, Communications of the ACM, Vol.39 No.9, 1996, pp 78-87.
14. Dave, P., Sakurai, H., *Maximal volume decomposition and its application to feature recognition*, Computers in Engineering ASME Database Symposium, Vol.29, 1995, pp. 553-568.

15. De Floriani, L., *Feature extraction from boundary models of three-dimensional objects*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.11 No.8, 1989, pp. 785-798.

16. DeVor, R.E., Kline, W.A., Zdeblick, W.J., *A mechanistic model for the force system in end milling with application to machining airframe structures*, Proceedings of the 8th North American Manufacturing Research Conference, 297-303, 1980, Rolla, Missouri, USA.

17. Drysdale, R.L., Jerard, R.B., Schaudt, B.,Hauck, K., *Discrete simulation of NC machining*, Algorithmica, Vol. 4, 1989, pp. 33-60.

18. Falcidieno, B., Giannini, F., *Automatic recognition and representation of shape-based features in a geometric modeling system*. Computer Vision, Graphics, and Image Processing, Vol. 48, 1989, pp. 93-123.

19. Fussell, B.K., *Modeling and adaptive force control pf end milling operations*, PhD Dissertation, Dept. of Mechanical Engineering, Ohio State University, 1987.

20. Fussell, B.K., Hemmett, J.G., Jerard, R.B., *Geometric and mechanistic modeling integration for five-axis milling force prediction*, Proceedings of the Japan-USA Symposium on Flexible Automation. Kobe, Japan, Vol. 2, 1994, pp. 747-750.

21. Fussell, B.K., Jerard, R.B.,Hemmett, J.G., *Modeling of cutting geometry and forces for 5-axis sculptured surface machining*, Computer-Aided Design, Vol. 35, 2003, pp. 333-346.

22. Han J., Requicha A.A.G., *Hint generation and completion for feature recognition*, Proceedings of International Symposium on Automotive Technology and Automation (ISATA), 1996, pp. 89-96.

23. Han, J., Pratt, M.,Regli, W.C., *Manufacturing feature recognition from solid models: A status report*, IEEE Transactions on Robotics and Automation, Vol.16 No.6, 2000, pp. 782-796.

24. Hao, Q., Shen, W., Park, S. W., Lee, J. K., Zhang, Z. and Shin, B. C., *An agent-based e-engineering services framework for engineering design and optimization*, Proceedings of the 17th International Conference on Industrial and Engineering Application of AI and Expert Systems, 2004, Ottawa, Canada.

25. Hemmett, J.G., Fussell, B.K.,Jerard, R.B. *Automatic five-axis CNC feedrate selection via discrete mechanistic, geometric, and machine model integration*, Proceedings of the Sculptured Surface Machining Conference: Machining Impossible Shapes, Detroit, MI, USA, Vol.1, 1998, pp. 138-146.

26. Huang, H., Oliver, J.H., *NC milling error assessment and tool path correction*, Computer Graphics Proceedings, Annual Conference Services, 28(4), pp. 287-294.

27. Huang, G.Q., Huang, J., Mak, K.L., *Agent-based workflow management in collaborative product development on the internet*, Computer-Aided Design, Vol.32, 2003, pp. 133-144.

28. Joshi, S.B., *CAD interface for automated process planning*, Ph.D. Dissertation, Purdue University, 1987.

29. Jerard, R.B., Drysdale, R.L., *Geometric simulation of numerical control machining*, Computers in Engineering, Vol. 2, 1988, pp. 129-136.

30. Jerard, R.B., Drysdale, R.L., Hauck, K., Schaudt, B., Magewick, J., *Methods for detecting errors in sculptured surface machining*, IEEE Computer Graphics and Applications, Vol.9 No.1, 1989, pp. 26-39.

31. Jerard, R.B., Drysdale, R.L., Hauck, K., Schaudt, B., *Approximate methods for simulation and verification of numerically controlled machining programs*, The Visual Computer, Vol.5 No.6, 1989.

32. Jerard, R.B., Angleton, J.M., Drysdale, R.L., Hauck, K., Su, P., *The use of surface points sets for generation, simulation, verification and automatic correction of NC machining programs*, Proceedings of NSF Design and Manufacturing System Conference, 1998, SME, USA, pp. 143-148.

33. Kim, Y.S., *Recognition of form features using convex decomposition*, Computer-Aided Design, Vol.24 No.9, 1992, pp. 461-476.

34. Kimura, K., *Development of a high performance end mill based on the analysis of chip flow generated by curved rake face*, Cirp Annals, Vol.52 No.1, 2003 pp. 57-60.

35. Kline, W.A., DeVor, R.E., *The prediction of cutting forces in end milling with application to concerning cuts*, International Journal of Machine Tools and Manufacture, Vol.22 No.1, pp. 7-22.

36. Kline, W.A., DeVor, R.E., *The effect of runout on cutting geometry and forces in end milling,* International Journal of Machine Tool Design and Manufacture, Vol.23, No.2, pp. 123-140.

37. Kyprianou L.K., *Shape classification in computer aided design*, Ph.D. Dissertation, Cambridge University, 1980.

38. Lacourse, D., *Alibre Design 8.0 - Take 3D mechanical design online*, Cadalyst, Vol.21 No.11, November, 2004, pp. 36-39.

39. Li, W.D., Lu, W.F., *Development of a web-based process planning optimization system*, Proceedings of ASME 2004 Design Engineering Technical Conferences and Computer in Engineering Conference, Salt lake City, Utah, September 28-October 2, 2004, DETC2004-57657.

40. Marefat, M., Kashyap, R.L., *Geometric reasoning for recognition of 3-D object features*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.12 No.10, 1990, pp. 949-965.

41. Oliver, J.H.,Goodman, E.D., *Graphical verification on N/C milling programs for sculptured surface parts*, Proceedings of the ASME Winter Annual Meeting, Anaheim, CA, Vol.21, 1986, pp. 247-263.

42. Oliver, J.H.,Goodman, E.D., *Direct dimensional NC verification*, Computer-Aided Design, Vol.22 No.1, 1990, pp. 3-9.

43. Peng, X.B., Yip-Hoi, D., *Polyhedral model based cutter workpiece engagements calculation for 2.5 and 3 axis milling*, Technical Report, University of British Columbia, 2005.

44. Roth, D., Dedi, S., Ismail, F., Mann, S., *Surface swept by a toroidal cutter during 5-axis machining*, Computer-Aided Design, Vol. 33 No.1, 2001, pp. 57-63.

45. Sakurai, H., Chin, C., *Definition and recognition of volume features for process planning*, Advances in Feature Based Manufacturing, Edited by J.J. Shah, M. Mantyla and D.S. Nau, Elsevier, 1994, pp. 65-79.

46. Sakurai, H., Gossard, D.C., *Recognizing shape features in solid models*, IEEE Computer Graphics and Applications, Vol.10 No.5, 1990, pp. 22-32.

47. Sambandan, K.,Wang, K.K., *Five-axis swept volumes for graphic NC simulation and verification*, Proceedings of the 15th ASME Design Engineering Technical Conferences, Montreal, QE, Canada, Vol. 19, 1989, pp. 143-150.

48. Shah, J.J., Mantyla, M. and Nau, D.S., *Advances in feature based manufacturing*, Elsevier, Amsterdam, Netherland, 1994.

49. Sheltami, K., Bedi, S., Ismail, F., *Swept volumes of toroidal cutters using generating curves*, International Journal of Machine Tools and Manufacture, Vol.38, 1998, pp. 855-870.

50. Shen, W., Norrie, D. H., Barthès, J. P., *Multi-agent systems for concurrent intelligent design and manufacturing*, Taylor and Francis, London, UK, 2001.

51. Shpitalni, M., Fischer, A., *CSG representation as a basis for extraction of machining features*, CIRP Annals, Vol.40, 1991, pp. 157-160.

52. Spence, A.D., Altintas, Y., Kirkpatrick, D., *Direct calculation of machining parameters from a solid model*, Computers in Industry, Vol.14, 1990, pp. 271-280.

53. Spence, A.D., Abrari, F., Elbestawi, M.A., *Integrated solid modeler based solutions for machining*, Computer-Aided Design, Vol.32 No.8, 2000, pp. 553-568.

54. Spence, A.D., Li, Z., *Parallel processing for 2-1/2 D machining simulation*, Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI, 2001, pp. 140-148.

55. Stori, J.A., Wright, P.K., *A knowledge-based system for machining operation planning in feature based open-architecture manufacturing*, Proceedings of ASME Design Engineering Technical Conferences and Computer in Engineering Conference, Irvine, CA, August 18-22, 1996, DETC1996/DFM-1286.

56. Subrahmanyam, S., Wozny, M., *Overview of automatic feature recognition techniques for computer-aided process planning*, Computers in Industry, Vol.26 No.1, 1995, pp. 1-21.

57. Sungurtekin, U.A., Voelcker, H.B., *Graphical simulation and automatic verification of NC machining programs*, Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, Vol.2, 1986, pp. 156-165.

58. Takata, S., *A cutting simulation system for machinability evaluation using a workpiece model*, CIRP Annals, Vol.38 No.1, 1989, pp. 417-420.

59. Tlusty, J., MacNeil, P., *Dynamics of cutting forces in end milling*, Annals of CIRP, Vol.24 No.1, 1975.

60. Trika S.N., Kashyap, R.L., *Geometric reasoning for extraction of manufacturing features in iso-oriented polyhedrons*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.16 No.11, 1994, pp. 1087-1100.

61. Van Hook, T., *Real-time shaded NC milling display*, Computer Graphics, Vol.20 No.4, 1986, pp. 15-20.

62. Vandenbrande J.H., Requicha A.A.G., *Spatial reasoning for the automatic recognition of machinable features in solid models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15, 1993, pp. 1-17.

63. Vandenbrande J.H., Requicha A.A.G., *Geometric computation for the recognition of spatially interacting machinable features*, Advances in Feature Based Manufacturing, Edited by J.J. Shah, M. Mantyla and D.S. Nau, Elsevier, 1994, pp. 83-106.

64. Voelcker, H.B., Hunt, W.A., *Role of solid modelling in machining-process modelling and NC verification*, SAE Preprints, Vol. 810195, 1981.

65. Wang, W.P., Wang, K.K., *Real-time verification of multiaxis NC programs with raster graphics*, Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, Vol.2, 1986, pp. 166-171.

66. Wang Y., Ajoku, P.N., Nnaji, B.O., *Distributed data access control for lean product information sharing in collaborative design*, Proceedings of ASME 2004 Design Engineering Technical Conferences and Computer in Engineering Conference, Salt lake City, Utah, September 28-October 2, 2004, DETC2004-57748.

67. Wang, L., Shen, W., Xie, H., Neelamkavil, J., Pardasani, A., *Collaborative conceptual design: a state-of-the-art survey*, Computer-Aided Design, Vol.34 No.13, 2002, pp. 981-996.

68. Weinert, K., Du, S., Damm, P., Stautner, M., *Swept volume generation for the simulation of machining processes*, International Journal of Machine Tools and Manufacture, Vol.44, 2004, pp. 617-628.

69. Woo, T., *Feature extraction by volume decomposition*, Proceedings of Conference CAD/CAM Technology in Mechanical Engineering, 1982.

70. Yip-Hoi, D., Huang, X., *Cutter engagement feature extraction from solid models for end milling*, Proceedings of IMECE: ASME International Mechanical Engineering Congress Novermber 13-19, 2004, Anaheim, California, USA.

71. http://www.fipa.org

72. http://jade.tilab.com

Appendix A
Entry and Exit Position Calculations

A.1 Linear Tool Motion/Linear Edge

Figure A-1 illustrates a cutter with radius r moving along a linear tool motion $\mathbf{P}_s\mathbf{P}_e$ for cutting a linear edge $\mathbf{P}_1\mathbf{P}_2$. The entry ($\mathbf{P}_{entry}$) and exit ($\mathbf{P}_{exit}$) position need to be determined based on these parameters. Points $\mathbf{P}_s(x_s, y_s)$, $\mathbf{P}_e(x_e, y_e)$, $\mathbf{P}_1(x_1, y_1)$ and $\mathbf{P}_2(x_2, y_2)$ are shown in Figure A-1.

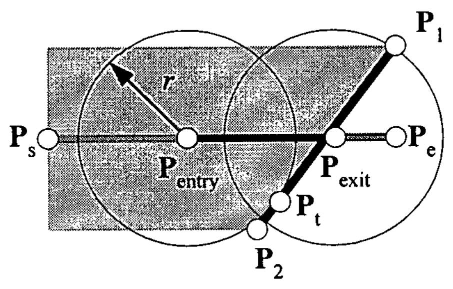


Figure A-1: Linear Tool Motion/Linear Edge

The equation of a circle with center at $\mathbf{P}_0$ and radius r is given by

$$|\mathbf{P}-\mathbf{P}_0| = r \tag{A.1}$$

The equation of a line $\mathbf{P}_s\mathbf{P}_e$ is given by

$$\mathbf{P} = \mathbf{P}_s + u(\mathbf{P}_e - \mathbf{P}_s) \tag{A.2}$$

To represent a circle whose center is located on the line $\mathbf{P}_s\mathbf{P}_e$, replace $\mathbf{P}_0$ in Eq. (A.1) with $\mathbf{P}$ in Eq. (A.2). The equation of the circle is represented in implicit form as:

$$[x-(x_s+u(x_e-x_s))]^2 + [y-(y_s+u(y_e-y_s))]^2 = r^2 \tag{A.3}$$

The equation of a line $\mathbf{P}_1\mathbf{P}_2$ in parametric form is given by

$$\begin{cases} x = x_1 + v(x_2 - x_1) \\ y = y_1 + v(y_2 - y_1) \end{cases} \tag{A.4}$$

Substituting Eq. (A.4) into Eq. (A.3),

$$[(x_1-x_s)+(x_2-x_1)v-(x_e-x_s)u]^2+[(y_1-y_s)+(y_2-y_1)v-(y_e-y_s)u]^2=r^2 \quad (A.5)$$

Introducing the following expressions

$$X_{1s}=x_1-x_s,\ X_{21}=x_2-x_1,\ X_{es}=x_e-x_s,\ Y_{1s}=y_1-y_s,\ Y_{21}=y_2-y_1,\ Y_{es}=y_e-y_s \quad (A.6)$$

Substituting these expressions into Eq. (A.5),

$$(X_{1s}+X_{21}v-X_{es}u)^2+(Y_{1s}+Y_{21}v-Y_{es}u)^2=r^2 \quad (A.7)$$

Expanding Eq. (A.7),

$$\begin{aligned}&(X_{es}^2+Y_{es}^2)u^2-2u[X_{es}(X_{1s}+X_{21}v)+Y_{es}(Y_{1s}+Y_{21}v)]\\&+(X_{1s}+X_{21}v)^2+(Y_{1s}+Y_{21}v)^2-r^2=0\end{aligned} \quad (A.8)$$

The roots of Eq. (A.8) is given by the quadratic formula

$$u=\frac{-b\pm\sqrt{b^2-4ac}}{2a} \quad (A.9)$$

where:

$$a=X_{es}^2+Y_{es}^2$$

$$b=-2[X_{es}(X_{1s}+X_{21}v)+Y_{es}(Y_{1s}+Y_{21}v)]$$

$$c=(X_{1s}+X_{21}v)^2+(Y_{1s}+Y_{21}v)^2-r^2$$

To determine the entry and exit positions for cutting the linear edge $\mathbf{P}_1\mathbf{P}_2$, the following three cutter positions along the linear tool motion $\mathbf{P}_s\mathbf{P}_e$ need to be calculated.

Position 1: The cutter position where the cutter has tangential contact with the linear edge.

To calculate this position, only one root of Eq. (A.8) exists. In other words, the following equation should be satisfied:

$$\begin{aligned}\Delta&=b^2-4ac\\&=4[X_{es}(X_{1s}+X_{21}v)+Y_{es}(Y_{1s}+Y_{21}v)]^2-4(X_{es}^2+Y_{es}^2)[(X_{1s}+X_{21}v)^2+(Y_{1s}+Y_{21}v)^2-r^2]\\&=4\left[r^2(X_{es}^2+Y_{es}^2)-(X_{es}(Y_{1s}+Y_{21}v)-Y_{es}(X_{1s}+X_{21}v))^2\right]\\&=4\left[r^2(X_{es}^2+Y_{es}^2)-((X_{es}Y_{21}-X_{21}Y_{es})v+(X_{es}Y_{1s}-X_{1s}X_{1s}))^2\right]\\&=0\end{aligned}$$

The above equation can be rewritten as:

$$v = \frac{(X_{es}Y_{1s} - X_{1s}Y_{es}) \pm r\sqrt{X_{es}^2 + Y_{es}^2}}{X_{es}Y_{21} - X_{21}Y_{es}} \tag{A.10}$$

The parameter v is calculated using Eq. (A.10). If $v \in [0, 1]$ then there is a contact point $\mathbf{P}_t$ within the linear edge $\mathbf{P}_1\mathbf{P}_2$, otherwise, the cutter does not have a tangential contact with $\mathbf{P}_1\mathbf{P}_2$. If the contact point $\mathbf{P}_t$ exists, the parameter of the cutter location is calculated by

$$u_t = \frac{-b}{2a} \tag{A.11}$$

Position 2: The cutter position where the cutter intersects with point $\mathbf{P}_1$.

To calculate this position, set v=0 in Eq. (A.9), and choose minimum root. The parameter of the cutter position is calculated by

$$u_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{A.12}$$

where:

$$a = X_{es}^2 + Y_{es}^2$$

$$b = -2(X_{es}X_{1s} + Y_{es}Y_{1s})$$

$$c = X_{1s}^2 + Y_{1s}^2 - r^2$$

Position 3: The cutter position where the cutter intersects point $\mathbf{P}_2$.

To calculate this position, set v=1 in Eq. (A.9), and choose minimum root. The parameter of the cutter position is calculated by

$$u_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{A.13}$$

where:

$$a = X_{es}^2 + Y_{es}^2$$

$$b = -2[X_{es}(X_{1s} + X_{21}) + Y_{es}(Y_{1s} + Y_{21})]$$

$$c = (X_{1s} + X_{21})^2 + (Y_{1s} + Y_{21})^2 - r^2$$

Based on the positions calculated using the above equations (u_1, u_2 and u_t), the parameters of the entry and exit positions (u_{entry}, u_{exit}) can be determined by following two rules:

Rule 1: If the cutter has tangential contact with the edge, then

$u_{entry} = u_t$ and $u_{exit} = \max(u_1, u_2)$

Rule 2: If the cutter does not have tangential contact with the edge, then

$u_{entry} = \min(u_1, u_2)$ and $u_{exit} = \max(u_1, u_2)$

A.2 Linear Tool Motion/Circular Edge

Figure A-2 illustrates a cutter with radius r moving along a linear tool motion $\mathbf{P}_s\mathbf{P}_e$ for cutting a circular edge $\mathbf{P}_1\mathbf{P}_2$ with center at $\mathbf{O}(x_0, y_0)$ and radius R.

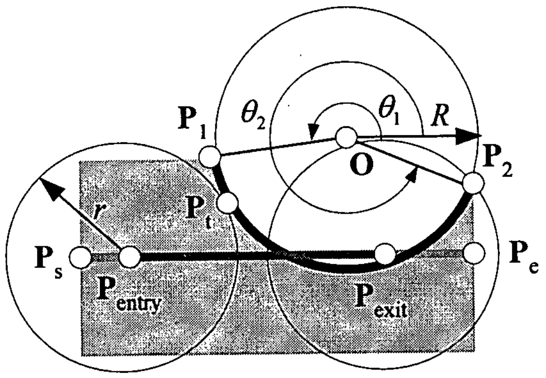


Figure A-2: Linear Tool Motion/Circular Edge

The equation of a circle with center at a line $\mathbf{P}_s\mathbf{P}_e$ and radius r is given by

$$[x - (x_s + u(x_e - x_s))]^2 + [y - (y_s + u(y_e - y_s))]^2 = r^2 \tag{A.14}$$

The equation of arc $\mathbf{P}_1\mathbf{P}_2$ in parametric form is given by

$$\begin{cases} x = x_0 + R\cos\theta \\ y = y_0 + R\sin\theta \end{cases} \tag{A.15}$$

where $\theta \in [\theta_1, \theta_2]$. Substituting Eq. (A.15) into Eq. (A.14),

$$[(x_0 - x_s) + R\cos\theta - (x_e - x_s)u]^2 + [(y_0 - y_s) + R\sin\theta - (y_e - y_s)u]^2 = r^2 \quad \text{(A.16)}$$

Introducing the following expressions

$$X_{0s}=x_0-x_s,\ X_{es}=x_e-x_s,\ Y_{0s}=y_0-y_s,\ Y_{es}=y_e-y_s \quad \text{(A.17)}$$

Substituting these expressions into Eq. (A.16),

$$(X_{0s} + R\cos\theta - X_{es}u)^2 + (Y_{0s} + R\sin\theta - Y_{es}u)^2 = r^2 \quad \text{(A.18)}$$

Expanding Eq. (A.18),

$$\begin{aligned}&(X_{es}^2 + Y_{es}^2)u^2 - 2[X_{es}(X_{0s} + R\cos\theta) + Y_{es}(Y_{0s} + R\sin\theta)]u \\ &+ (X_{0s} + R\cos\theta)^2 + (Y_{0s} + R\sin\theta)^2 - r^2 = 0\end{aligned} \quad \text{(A.19)}$$

The roots of Eq. (A.8) is given by the quadratic formula

$$u = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \text{(A.20)}$$

where:

$$a = X_{es}^2 + Y_{es}^2$$

$$b = -2[X_{es}(X_{0s} + R\cos\theta) + Y_{es}(Y_{0s} + R\sin\theta)]$$

$$c = (X_{0s} + R\cos\theta)^2 + (Y_{0s} + R\sin\theta)^2 - r^2$$

<u>Position 1</u>: The cutter position where the cutter has a tangential contact with the circular edge. Similar to the calculations described in Section **A.1**, the following equation should be satisfied:

$$\begin{aligned}\Delta &= b^2 - 4ac \\ &= 4[X_{es}(X_{0s} + R\cos\theta) + Y_{es}(Y_{0s} + R\sin\theta)]^2 - 4(X_{es}^2 + Y_{es}^2)[(X_{0s} + R\cos\theta)^2 + (Y_{0s} + R\sin\theta)^2 - r^2] \\ &= 4\left[r^2(X_{es}^2 + Y_{es}^2) - (X_{es}(Y_{0s} + R\sin\theta) - Y_{es}(X_{0s} + R\cos\theta))^2\right] \\ &= 4\left[r^2(X_{es}^2 + Y_{es}^2) - ((X_{es}Y_{0s} - X_{0s}Y_{es}) + X_{es}R\sin\theta - Y_{es}R\cos\theta)^2\right] \\ &= 0\end{aligned}$$

The above equation can be rewritten as:

$$\theta = \pm\cos^{-1}\left(\frac{(X_{es}Y_{0s} - X_{0s}Y_{es}) - r^2(X_{es}^2 + Y_{es}^2)}{R\sqrt{X_{es}^2 + Y_{es}^2}}\right) - \tan^{-1}\frac{Y_{es}}{X_{es}} \tag{A.21}$$

If $\theta \in [\theta_1, \theta_2]$ then the parameter of this position is calculated by

$$u_t = \frac{-b}{2a} \tag{A.22}$$

Position 2: The cutter position where the cutter intersects with point $\mathbf{P}_1$.

To calculate this position, set θ=θ_1 in Eq. (A.20), and choose minimum root. The parameter of the cutter position is calculated by

$$u_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{A.23}$$

where:

$$a = X_{es}^2 + Y_{es}^2$$

$$b = -2[X_{es}(X_{0s} + R\cos\theta_1) + Y_{es}(Y_{0s} + R\sin\theta_1)]$$

$$c = (X_{0s} + R\cos\theta_1)^2 + (Y_{0s} + R\sin\theta_1)^2 - r^2$$

Position 3: The cutter position where the cutter intersects point $\mathbf{P}_2$.

To calculate this position, set θ=θ_2 in Eq. (A.20), and choose minimum root. The parameter of the cutter position is calculated by

$$u_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{A.24}$$

where:

$$a = X_{es}^2 + Y_{es}^2$$

$$b = -2[X_{es}(X_{0s} + R\cos\theta_2) + Y_{es}(Y_{0s} + R\sin\theta_2)]$$

$$c = (X_{0s} + R\cos\theta_2)^2 + (Y_{0s} + R\sin\theta_2)^2 - r^2$$

Based on the positions calculated using the above equations (u_1, u_2 and u_t), the parameters of the entry and exit positions (u_{entry}, u_{exit}) can be determined by following the same rules as described in Section **A.1**.

A.3 Circular Tool Motion/Linear Edge

Figure A-3 illustrates a cutter with radius r moving along a circular tool motion $\mathbf{P}_s\mathbf{P}_e$ with center at $\mathbf{O}(x_0, y_0)$ and radius R for cutting a linear edge $\mathbf{P}_1\mathbf{P}_2$.

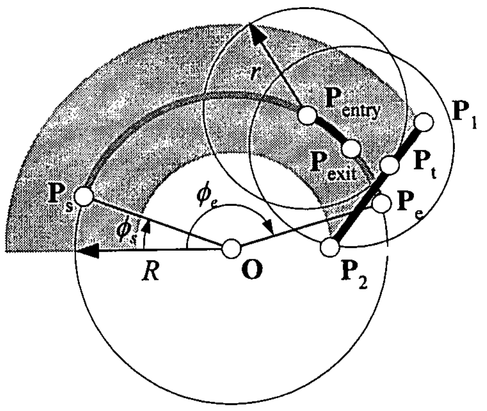


Figure A-3: Circular Tool Motion/Linear Edge

The equation of a circle with center at the circular tool motion $\mathbf{P}_s\mathbf{P}_e$ and radius r is given by

$$[x-(x_0+R\cos\phi)]^2+[y-(y_0+R\sin\phi)]^2=r^2 \tag{A.25}$$

where $\phi \in [\phi_s, \phi_e]$. The equation of a line $\mathbf{P}_1\mathbf{P}_2$ in parametric form is given by

$$\begin{cases} x = x_1 + v(x_2 - x_1) \\ y = y_1 + v(y_2 - y_1) \end{cases} \tag{A.26}$$

Substituting Eq. (A.26) into Eq. (A.25),

$$[(x_1-x_0)+(x_2-x_1)v-R\cos\phi]^2+[(y_1-y_0)+(y_2-y_1)v-R\sin\phi]^2=r^2 \tag{A.27}$$

Introducing the following expressions

$$X_{10}=x_1-x_0,\ X_{21}=x_2-x_1,\ Y_{10}=y_1-y_0,\ Y_{21}=y_2-y_1 \tag{A.28}$$

Substituting these expressions into Eq. (A.27),

$$(X_{10}+X_{21}v-R\cos\phi)^2+(Y_{10}+Y_{21}v-R\sin\phi)^2=r^2 \tag{A.29}$$

Expanding Eq. (A.29),

$$(X_{10}+X_{21}v)\cos\phi+(Y_{10}+Y_{21}v)\sin\phi=\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R} \tag{A.30}$$

Dividing Eq. (A.30) by $\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}$.

$$\frac{(X_{10}+X_{21}v)}{\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\cos\phi+\frac{(Y_{10}+Y_{21}v)}{\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\sin\phi=\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}} \tag{A.31}$$

Introducing angle α as:

$$\begin{cases}\sin\alpha=\dfrac{(Y_{10}+Y_{21}v)}{\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\\ \cos\alpha=\dfrac{(X_{10}+X_{21}v)}{\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\end{cases},\ \alpha=\tan^{-1}\left(\frac{Y_{10}+Y_{21}v}{X_{10}+X_{21}v}\right) \tag{A.32}$$

Substituting Eq. (A.32) into Eq. (A.31),

$$\cos(\phi-\alpha)=\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}} \tag{A.33}$$

The angle ϕ can be calculation using Eq. (A.34).

$$\begin{aligned}\phi&=\alpha\pm\cos^{-1}\left(\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\right)\\ &=\tan^{-1}\left(\frac{Y_{10}+Y_{21}v}{X_{10}+X_{21}v}\right)\pm\cos^{-1}\left(\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}\right)\end{aligned} \tag{A.34}$$

Position 1: The cutter position where the cutter has a tangential contact with the linear edge.

To calculate this position, the following equation should be satisfied:

$$\frac{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2}}=1 \tag{A.35}$$

Rewriting Eq. (A.35),

$$(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2+R^2-r^2=2R\sqrt{(X_{10}+X_{21}v)^2+(Y_{10}+Y_{21}v)^2} \qquad \text{(A.36)}$$

Solving Eq.(A.36), the roots of Eq. (A.36) can be calculated using Eq. (A.37).

$$v=\frac{-(X_{10}X_{21}+Y_{10}Y_{21})\pm\sqrt{(R\pm r)^2(X_{21}^2+Y_{21}^2)-(X_{21}Y_{10}-X_{10}Y_{21})^2}}{X_{21}^2+Y_{21}^2} \qquad \text{(A.37)}$$

If $v\in[0, 1]$ then there is a contact point within the linear edge **P**$_1$**P**$_2$, and the parameter of the cutter location ϕ_t can be obtained by substituting Eq. (A.37) into Eq. (A.34). Otherwise, the cutter does not have a tangent contact with **P**$_1$**P**$_2$.

Position 2: The cutter position where the cutter intersects with point **P**$_1$.

To calculate this position, set v=0 in Eq. (A.34), and choose minimum root. The parameter of the cutter position is calculated by

$$\phi_1=\min\left(\tan^{-1}\left(\frac{Y_{10}}{X_{10}}\right)\pm\cos^{-1}\left(\frac{X_{10}^2+Y_{10}^2+R^2-r^2}{2R\sqrt{X_{10}^2+Y_{10}^2}}\right)\right) \qquad \text{(A.38)}$$

Position 3: The cutter position where the cutter intersects point **P**$_2$.

To calculate this position, set v=1 in Eq. (A.34), and choose minimum root. The parameter of the cutter position is calculated by

$$\phi_2=\min\left(\tan^{-1}\left(\frac{Y_{10}+Y_{21}}{X_{10}+X_{21}}\right)\pm\cos^{-1}\left(\frac{(X_{10}+X_{21})^2+(Y_{10}+Y_{21})^2+R^2-r^2}{2R\sqrt{(X_{10}+X_{21})^2+(Y_{10}+Y_{21})^2}}\right)\right) \qquad \text{(A.39)}$$

Based on the positions calculated using the above equations (ϕ_1, ϕ_2 and ϕ_t), the parameters of the entry and exit positions (ϕ_{entry}, ϕ_{exit}) can be determined by following the same rules as described in Section **A.1**.

A.4 Circular Tool Motion/Circular Edge

Figure A-4 illustrates a cutter with radius r moving along a circular tool motion **P**$_s$**P**$_e$ with center at **O**$_0$(x_0, y_0) and radius R_0 for cutting a linear edge **P**$_1$**P**$_2$ with center at **O**$_1$(x_1, y_1) and radius R_1

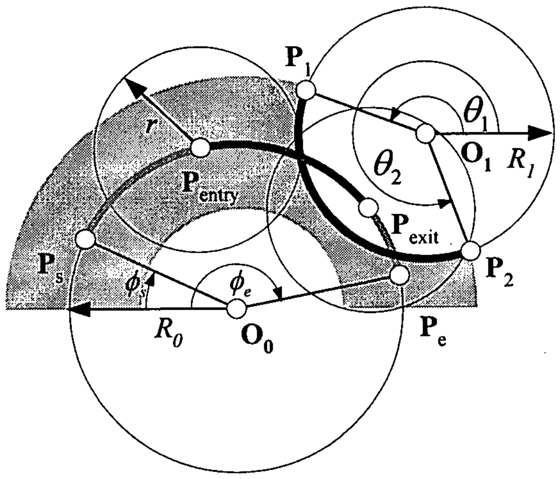


Figure A-4: Circular Tool Motion/Circular Edge

The equation of a circle with center at the circular tool motion $\mathbf{P}_s\mathbf{P}_e$ and radius r is given by

$$[x-(x_0+R_0\cos\phi)]^2+[y-(y_0+R_0\sin\phi)]^2=r^2 \qquad \text{(A.40)}$$

where $\phi \in [\phi_s, \phi_e]$. The equation of arc $\mathbf{P}_1\mathbf{P}_2$ in parametric form is given by

$$\begin{cases} x = x_1 + R_1\cos\theta \\ y = y_1 + R_1\sin\theta \end{cases} \qquad \text{(A.41)}$$

where $\theta \in [\theta_1, \theta_2]$. Substituting Eq. (A.41) into Eq. (A.40),

$$[(x_1-x_0)+R_1\cos\theta-R_0\cos\phi]^2+[(y_1-y_0)+R_1\sin\theta-R_0\sin\phi]^2=r^2 \qquad \text{(A.42)}$$

Introducing the following expressions

$$X_{10}=x_1-x_0,\ Y_{10}=y_1-y_0 \qquad \text{(A.43)}$$

Substituting these expressions into Eq. (A.42),

$$(X_{10}+R_1\cos\theta-R_0\cos\phi)^2+(Y_{10}+R_1\sin\theta-R_0\sin\phi)^2=r^2 \qquad \text{(A.44)}$$

Expanding Eq. (A.44),

$$(X_{10}+R_1\cos\theta)\cos\phi+(Y_{10}+R_1\sin\theta)\sin\phi=\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0}$$

(A.45)

Dividing Eq. (A.45) by $\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}$.

$$\frac{(X_{10}+R_1\cos\theta)}{\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\cos\phi+\frac{(Y_{10}+R_1\sin\theta)}{\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\sin\phi=\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}} \tag{A.46}$$

Introducing angle α as:

$$\begin{cases}\sin\alpha=\dfrac{(Y_{10}+R_1\sin\theta)}{\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\\ \cos\alpha=\dfrac{(X_{10}+R_1\cos\theta)}{\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\end{cases},\ \alpha=\tan^{-1}\left(\frac{Y_{10}+R_1\sin\theta}{X_{10}+R_1\cos\theta}\right) \tag{A.47}$$

Substituting Eq. (A.47) into Eq. (A.46),

$$\cos(\phi-\alpha)=\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}} \tag{A.48}$$

The angle ϕ can be calculated using Eq. (A.49).

$$\begin{aligned}\phi&=\alpha\pm\cos^{-1}\left(\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\right)\\&=\tan^{-1}\left(\frac{Y_{10}+R_1\sin\theta}{X_{10}+R_1\cos\theta}\right)\pm\cos^{-1}\left(\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}\right)\end{aligned} \tag{A.49}$$

Position 1: The cutter position where the cutter has a tangential contact with the circular edge.

To calculate this position, the following equation should be satisfied:

$$\frac{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2+R_0^2-r^2}{2R_0\sqrt{(X_{10}+R_1\cos\theta)^2+(Y_{10}+R_1\sin\theta)^2}}=1 \tag{A.50}$$

Rewriting Eq. (A.50),

$$X_{10}\cos\theta+Y_{10}\sin\theta=\frac{(R_0\pm r)^2-(X_{10}^2+Y_{10}^2)-R_1^2}{2R_1} \tag{A.51}$$

Solving Eq.(A.51), the roots can be calculated using Eq. (A.52).

$$\theta = \tan^{-1}\left(\frac{Y_{10}}{X_{10}}\right) \pm \cos^{-1}\left(\frac{(R_0 \pm r)^2 - \left(X_{10}^2 + Y_{10}^2\right) - R_1^2}{2R_1\sqrt{X_{10}^2 + Y_{10}^2}}\right) \tag{A.52}$$

If $\theta \in [\theta_1, \theta_2]$ then there is a contact point within the circular edge $\mathbf{P}_1\mathbf{P}_2$, and the parameter of the cutter location ϕ_t can be obtained by substituting Eq. (A.52) into Eq. (A.49). Otherwise, the cutter does not have a tangent contact with $\mathbf{P}_1\mathbf{P}_2$.

Position 2: The cutter position where the cutter intersects with point $\mathbf{P}_1$.

To calculate this position, set $\theta=\theta_1$ in Eq. (A.49), and choose minimum root. The parameter of the cutter position is calculated by

$$\phi_1 = \min\left(\tan^{-1}\left(\frac{Y_{10} + R_1\sin\theta_1}{X_{10} + R_1\cos\theta_1}\right) \pm \cos^{-1}\left(\frac{(X_{10} + R_1\cos\theta_1)^2 + (Y_{10} + R_1\sin\theta_1)^2 + R_0^2 - r^2}{2R_0\sqrt{(X_{10} + R_1\cos\theta_1)^2 + (Y_{10} + R_1\sin\theta_1)^2}}\right)\right) \tag{A.53}$$

Position 3: The cutter position where the cutter intersects point $\mathbf{P}_2$.

To calculate this position, set $\theta=\theta_2$ in Eq. (A.49), and choose minimum root. The parameter of the cutter position is calculated by

$$\phi_2 = \min\left(\tan^{-1}\left(\frac{Y_{10} + R_1\sin\theta_2}{X_{10} + R_1\cos\theta_2}\right) \pm \cos^{-1}\left(\frac{(X_{10} + R_1\cos\theta_2)^2 + (Y_{10} + R_1\sin\theta_2)^2 + R_0^2 - r^2}{2R_0\sqrt{(X_{10} + R_1\cos\theta_2)^2 + (Y_{10} + R_1\sin\theta_2)^2}}\right)\right) \tag{A.54}$$

Based on the positions calculated using the above equations (ϕ_1, ϕ_2 and ϕ_t), the parameters of the entry and exit positions (ϕ_{entry}, ϕ_{exit}) can be determined by following the same rules as described in Section **A.1**.

Appendix B
Helicoid Equation (5.6)

Figure B-1 illustrates a helicoid with a semi-circle profile curve and its parameters.

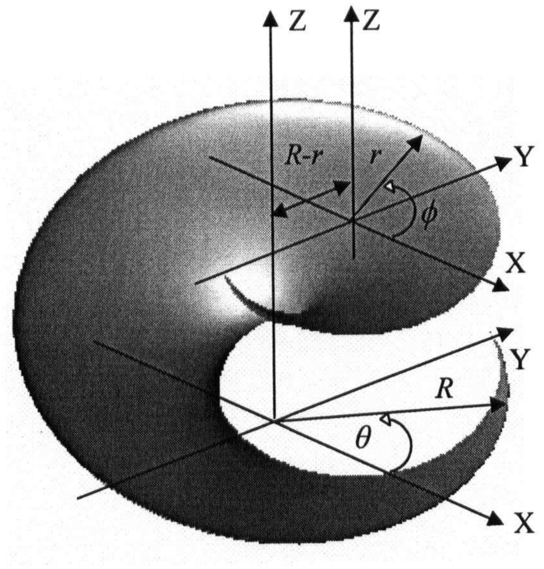


Figure B-1: A Helicod

The equation of the helicoid in parametric form is

$$\begin{cases} x = (R-r)\cos\theta + r\cos\phi \\ y = (R-r)\sin\theta + r\sin\phi \\ z = -\dfrac{p}{2\pi}\theta \end{cases} \tag{B.1}$$

where $\theta \in [0, 2\pi]$ and $\phi \in [-\pi, 0]$. Combine x and y coordinate expressions as follows:

$$[x-(R-r)\cos\theta]^2 + [y-(R-r)\sin\theta]^2 = r^2 \tag{B.2}$$

Rewriting Eq. (B.2),

$$x\cos\theta + y\sin\theta = \frac{x^2 + y^2 + (R-r)^2 - r^2}{2(R-r)} \tag{B.3}$$

Dividing Eq. (B.3) by $\sqrt{x^2 + y^2}$,

$$\frac{x}{\sqrt{x^2+y^2}}\cos\theta+\frac{y}{\sqrt{x^2+y^2}}\sin\theta=\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}} \tag{B.4}$$

Introducing angle α in Eq. (B.5) as follows:

$$\begin{cases}\sin\alpha=\dfrac{y}{\sqrt{x^2+y^2}}\\[2ex]\cos\alpha=\dfrac{x}{\sqrt{x^2+y^2}}\end{cases},\ \alpha=\tan^{-1}\left(\frac{y}{x}\right) \tag{B.5}$$

Substituting Eq. (B.5) into Eq. (B.4),

$$\cos\alpha\cos\theta+\sin\alpha\sin\theta=\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}} \tag{B.6}$$

Rewriting Eq. (B.6),

$$\cos(\theta-\alpha)=\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}} \tag{B.7}$$

The angle θ can be calculation using Eq. (B.8).

$$\begin{aligned}\theta&=\alpha+\cos^{-1}\left(\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}}\right)\\&=\tan^{-1}\left(\frac{y}{x}\right)+\cos^{-1}\left(\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}}\right)\end{aligned} \tag{B.8}$$

Substituting Eq. (B.8) into the z coordinate expression in Eq. (B.1), the equation of the helicoid in explicit form can be represented as follows:

$$z=-\frac{p}{2\pi}\left(\tan^{-1}\left(\frac{y}{x}\right)+\cos^{-1}\left(\frac{x^2+y^2+(R-r)^2-r^2}{2(R-r)\sqrt{x^2+y^2}}\right)\right) \tag{B.9}$$

Appendix C

Intersection and Connection Point Calculation Equation (5.10)

As shown in Figure C-1, a cutting tool rotates angle θ along a helical toolpath from its initial position $\mathbf{O}_1$ to its final position $\mathbf{O}_2$. The intersection and connection point can be determined by calculating the intersection point between Circle $\mathbf{O}_1$ (initial position) and Circle $\mathbf{O}_2$ (final positon).

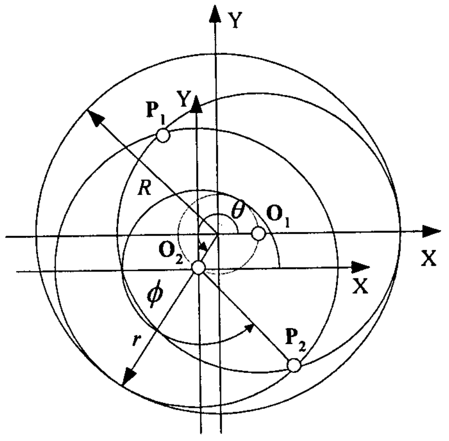


Figure C-1: Intersection and Connection Point Calculation

The equation of Circle $\mathbf{O}_2$ with center at ($(R-r)\cos\theta$, $(R-r)\sin\theta$) and radius r in parametric form is

$$\begin{cases} x = (R-r)\cos\theta + r\cos\phi \\ y = (R-r)\sin\theta + r\sin\phi \end{cases} \tag{C.1}$$

The equation of Circle $\mathbf{O}_1$ with center at ($(R-r)$, 0) is

$$[x-(R-r)]^2 + y^2 = r^2 \tag{C.2}$$

Substituting Eq. (C-1) into Eq. (C-2),

$$[(R-r)\cos\theta + r\cos\phi - (R-r)]^2 + [(R-r)\sin\theta + r\sin\phi]^2 = r^2 \tag{C.3}$$

Rewriting Eq. (C.3),

$$r[\sin\theta\sin\phi - (1-\cos\theta)\cos\phi] = (R-r)(1-\cos\theta) \tag{C.4}$$

Dividing Eq. (C.4) by $r\sqrt{2(1-\cos\theta)}$,

$$\sqrt{\frac{1+\cos\theta}{2}}\sin\phi-\sqrt{\frac{1-\cos\theta}{2}}\cos\phi=\frac{R-r}{r}\sqrt{\frac{1-\cos\theta}{2}} \tag{C.5}$$

Replacing $\sqrt{\frac{1+\cos\theta}{2}}$ and $\sqrt{\frac{1-\cos\theta}{2}}$ in Eq. (C.5) with $\cos\frac{\theta}{2}$ and $\sin\frac{\theta}{2}$ respectively,

$$\cos\frac{\theta}{2}\sin\phi-\sin\frac{\theta}{2}\cos\phi=\frac{R-r}{r}\sin\frac{\theta}{2} \tag{C.6}$$

Rewriting Eq. (C.6),

$$\sin\left(\phi-\frac{\theta}{2}\right)=\frac{R-r}{r}\sin\frac{\theta}{2} \tag{C.7}$$

From Eq. (C.7), two intersection points are calculated as follows:

$$\phi=\frac{\theta}{2}+\sin^{-1}\left(\frac{R-r}{r}\sin\frac{\theta}{2}\right) \tag{C.8}$$

$$\phi=\frac{\theta}{2}+\pi-\sin^{-1}\left(\frac{R-r}{r}\sin\frac{\theta}{2}\right) \tag{C.9}$$

Eq. (C.8) is for calculating intersection point $\mathbf{P}_1$ (see Figure C-1), and Eq. (C.9) is for $\mathbf{P}_2$. Since $\mathbf{P}_2$ is within the front semi-circle of Circle $\mathbf{O}_2$, Eq. (C.9) is used to calculate the intersection and connection points as Eq. (5.10) in Chapter 5.

Appendix D
Silhouette Curves of End-Mills

D.1 Silhouette Curves of a Flat End-Mill

Figure D-1 illustrates a cylinder representing a flat end-mill and its silhouette curves. Given the center **O**(x_0, y_0, z_0) of this cylindrical surface, radius R and height H, the control points of the silhouette curves are listed in Table D-1.

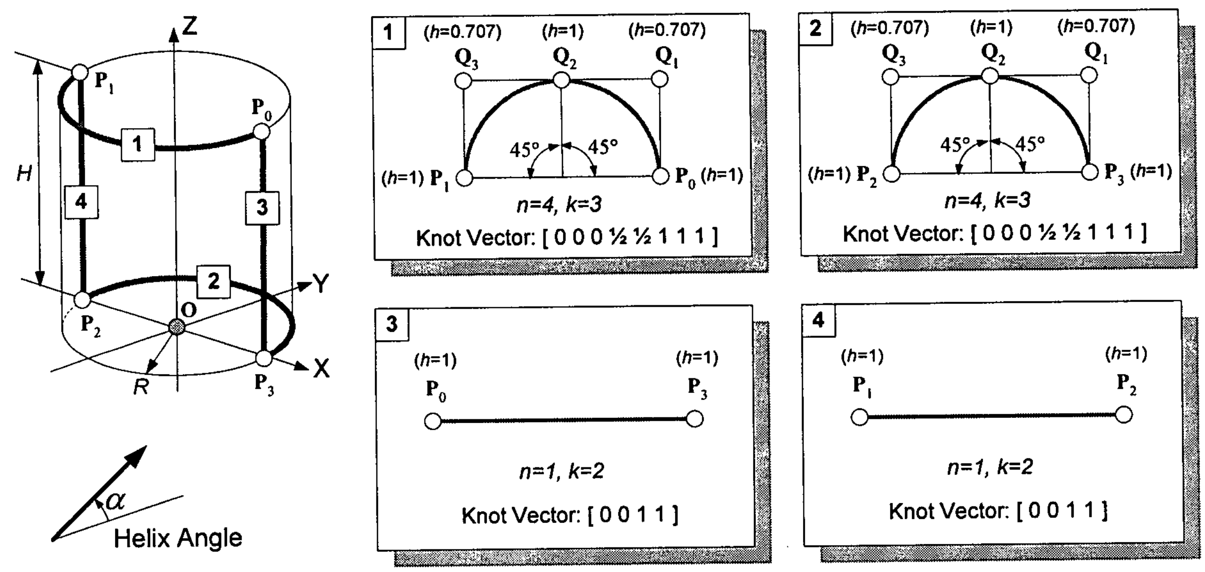


Figure D-1: Silhouette Curves for a Flat End-Mill

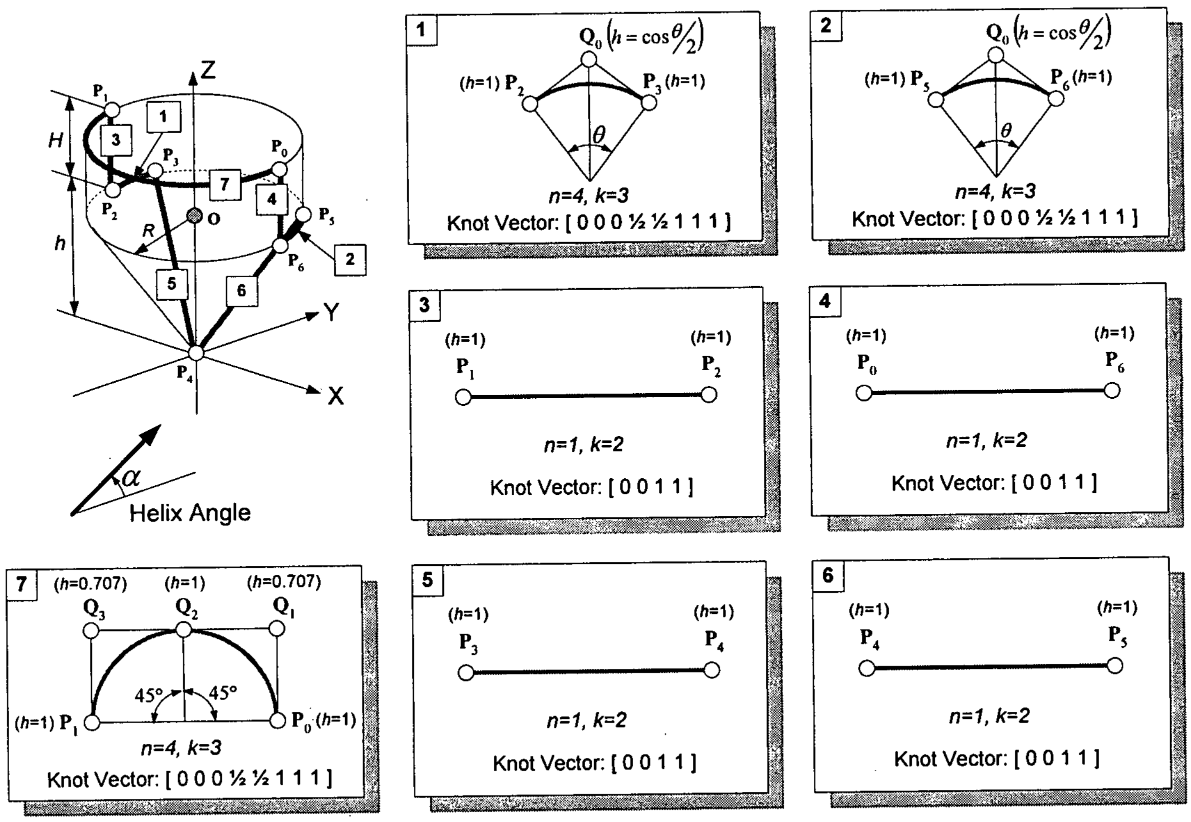
Curve #	Type	Control Points			
		Point #	X	Y	Z
1	Arc	P_0	x_0+R	y_0	z_0+H
		P_1	x_0-R	y_0	z_0+H
		Q_1	x_0+R	y_0-R	z_0+H
		Q_2	x_0	y_0-R	z_0+H
		Q_3	x_0-R	y_0-R	z_0+H
2	Arc	P_3	x_0+R	y_0	z_0
		P_2	x_0-R	y_0	z_0
		Q_1	x_0+R	y_0-R	z_0
		Q_2	x_0	y_0-R	z_0

		Q$_3$	x_0-R	y_0-R	z_0
3	Line	**P**$_0$	x_0+R	y_0	z_0+H
		P$_3$	x_0+R	y_0	z_0
4	Line	**P**$_1$	x_0-R	y_0	z_0+H
		P$_2$	x_0-R	y_0	z_0

Table D-1: Control Points for Silhouette Curves of a Flat End-Mill

D.2 Silhouette Curves of a Conical End-Mill

A conical end-mill with center at **O**(x_0, y_0, z_0), radius R, height H and cone height h and its silhouette curves are illustrated in Figure D-2. The control points for each silhouette curve are listed in Table D-2.



Z
Y
X
H
h
R
O
P1 P2 P3 P4 P5 P6 P0
Helix Angle
α
1: Q0 (h = cos θ/2); (h=1) P2; P3 (h=1); θ; n=4, k=3; Knot Vector: [0 0 0 ½ ½ 1 1 1]
2: Q0 (h = cos θ/2); (h=1) P5; P6 (h=1); θ; n=4, k=3; Knot Vector: [0 0 0 ½ ½ 1 1 1]
3: (h=1) P1; (h=1) P2; n=1, k=2; Knot Vector: [0 0 1 1]
4: (h=1) P0; (h=1) P6; n=1, k=2; Knot Vector: [0 0 1 1]
7: (h=0.707) Q3; (h=1) Q2; (h=0.707) Q1; 45°; 45°; (h=1) P1; P0 (h=1); n=4, k=3; Knot Vector: [0 0 0 ½ ½ 1 1 1]
5: (h=1) P3; (h=1) P4; n=1, k=2; Knot Vector: [0 0 1 1]
6: (h=1) P4; (h=1) P5; n=1, k=2; Knot Vector: [0 0 1 1]

Figure D-2: Silhouette Curves for a Conical End-Mill

Curve #	Type	Control Points			
		Point #	X	Y	Z
1	Arc	P_2	x_0-R	Y_0	z_0
		P_3	*	*	*
		Q_1	†	†	†
2	Arc	P_6	x_0+R	Y_0	z_0
		P_5	*	*	*
		Q_1	†	†	†
3	Line	P_1	x_0-R	y_0	z_0+H
		P_2	x_0-R	y_0	z_0
4	Line	P_0	x_0+R	y_0	z_0+H
		P_6	x_0+R	y_0	z_0
5	Line	P_3	*	*	*
		P_4	x_0	y_0	z_0-h
6	Line	P_4	x_0	y_0	z_0-h
		P_5	*	*	*
7	Arc	P_0	x_0+R	y_0	z_0+H
		P_1	x_0-R	y_0	z_0+H
		Q_1	x_0+R	y_0-R	z_0+H
		Q_2	x_0	y_0-R	z_0+H
		Q_3	x_0-R	y_0-R	z_0+H

* see D.2.1 for details.

† see D.2.2 for details.

Table D-2: Control Points for Silhouette Curves of a Conical End-Mill

D.2.1 Calculating P_3 and P_5

A conical end-mill with center at **O**(x_0, y_0, z_0), radius R, height H and cone height h and its silhouette curves are illustrated in Figure D-3. The control points for each silhouette curve are listed in Table D-2.

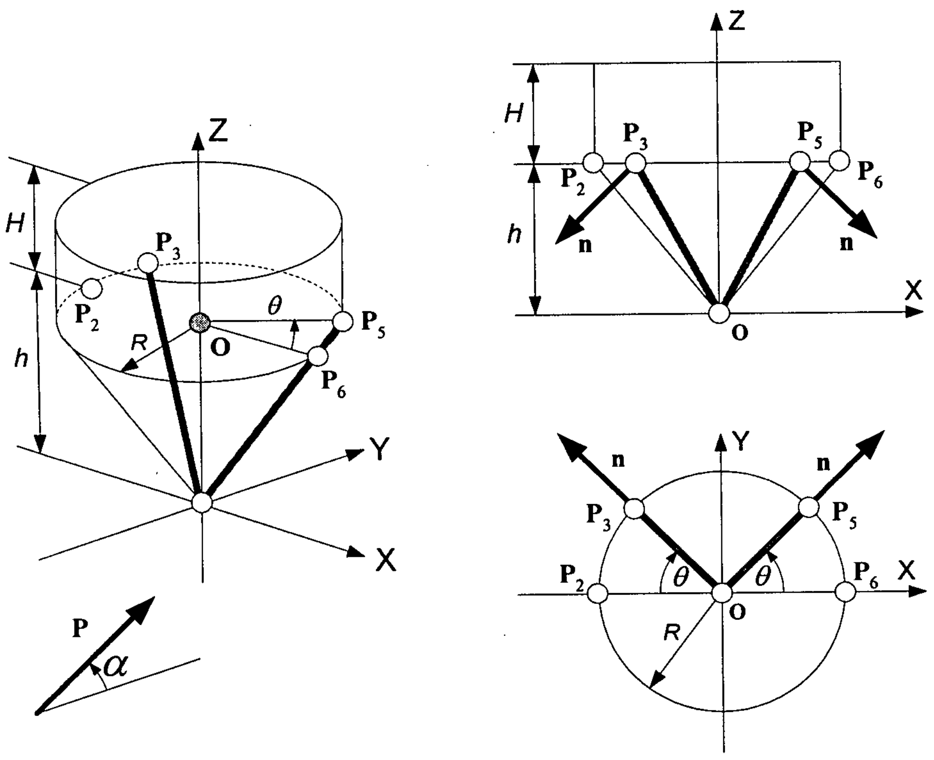


Figure D-3: Calculating P₃ and P₅

The perspective direction **P** is given as:

$$\mathbf{P}(0, \cos\alpha, \sin\alpha) \tag{D.1}$$

The normal **n** of the conical surface at angle θ can be calculated as:

$$\mathbf{n}\left(\cos\theta, \sin\theta, -\frac{R}{\sqrt{R^2+h^2}}\right) \tag{D.2}$$

The silhouette curve is composed of the points on the surface whose normal **n** are perpendicular to the perspective direction **P**. This can be represented in Eq. (D.3).

$$\mathbf{P} \bullet \mathbf{n} = 0 \tag{D.3}$$

Substituting Eq. (D.1) and (D.2) into (D.3),

$$\cos\alpha \sin\theta - \sin\alpha \frac{R}{\sqrt{R^2+h^2}} = 0 \tag{D.4}$$

Rewriting Eq. (D.4),

$$\sin\theta = \frac{R\tan\alpha}{\sqrt{R^2+h^2}} \tag{D.5}$$

The $\cos\theta$ can be obtained from Eq. (D.5) as:

$$\cos\theta = \pm\sqrt{1-\sin^2\theta} = \pm\sqrt{\frac{R^2\left(1-\tan^2\alpha\right)+h^2}{R^2+h^2}} \tag{D.6}$$

Given the center point **O**(x_0, y_0, z_0) of the conical end mill, $\mathbf{P}_3$ and $\mathbf{P}_5$ can be calculated easily by using Eq. (D.5) and Eq (D.6),

$$\begin{aligned}&\mathbf{P}_3\left(x_0 - R\cos\theta, y_0 + R\sin\theta, z_0\right)\\ &= \mathbf{P}_3\left(x_0 - R\sqrt{\frac{R^2\left(1-\tan^2\alpha\right)+h^2}{R^2+h^2}}, y_0 + \frac{R^2\tan\alpha}{\sqrt{R^2+h^2}}, z_0\right)\end{aligned} \tag{D.7}$$

$$\begin{aligned}&\mathbf{P}_5\left(x_0 + R\cos\theta, y_0 + R\sin\theta, z_0\right)\\ &= \mathbf{P}_5\left(x_0 + R\sqrt{\frac{R^2\left(1-\tan^2\alpha\right)+h^2}{R^2+h^2}}, y_0 + \frac{R^2\tan\alpha}{\sqrt{R^2+h^2}}, z_0\right)\end{aligned} \tag{D.8}$$

D.2.2 Calculating Control Point $\mathbf{Q}_0$ for Silhouette Curve 1 and 2

Figure D-4 illustrates the calculation of the middle control point $\mathbf{Q}_0$ for Curve1 and Curve2. $\mathbf{Q}_1$=$\mathbf{P}_2$, $\mathbf{Q}_2$=$\mathbf{P}_3$ in Curve1, $\mathbf{Q}_1$=$\mathbf{P}_5$, $\mathbf{Q}_2$=$\mathbf{P}_6$ in Curve2. **O** is the center position of the end-mill. The angle θ, $\mathbf{P}_3$ and $\mathbf{P}_5$ are calculated in D.2.1 (see Eq. (D.5)-(D.8)), and $\mathbf{P}_2$ and $\mathbf{P}_6$ are calculated in Table D-2.

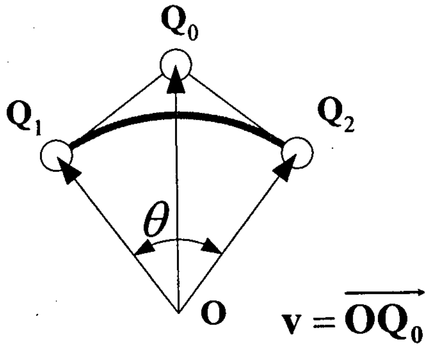


Figure D-4: Calculating the Control Point $\mathbf{Q}_0$

The unit vector of **v** can be calculated by adding $\overrightarrow{\mathbf{OQ}_1}$ and $\overrightarrow{\mathbf{OQ}_2}$.

$$\hat{\mathbf{v}} = \frac{\overrightarrow{\mathbf{OQ}_1} + \overrightarrow{\mathbf{OQ}_2}}{\left|\overrightarrow{\mathbf{OQ}_1} + \overrightarrow{\mathbf{OQ}_2}\right|} \tag{D.9}$$

The vector $\mathbf{v} = \overrightarrow{\mathbf{OQ}_0}$ can be calculated by multiplying the length of $\overrightarrow{\mathbf{OQ}_0}$.

$$\mathbf{v} = \overrightarrow{\mathbf{OQ}_0} = \frac{\left|\overrightarrow{\mathbf{OQ}_1}\right|}{\left|\overrightarrow{\mathbf{OQ}_1} + \overrightarrow{\mathbf{OQ}_2}\right| \cos\frac{\theta}{2}} \left(\overrightarrow{\mathbf{OQ}_1} + \overrightarrow{\mathbf{OQ}_2}\right) \tag{D.10}$$

Since **O** is given, the control point $\mathbf{Q}_0$ can be calculated using Eq. (D.10).

Appendix E
The Sat Files of an Envelope Surface

A swept surface generated by sweeping a circle along a helical curve using both the skinning and exact NURBS representation methods is shown in Figure E-1. The parameters of the profile and path curves are list in Table E-1. The sat files of this surface generated by these two different methods are listed in E.1 and E.2. The sat file generated by the skinning method has size 20KB, and the file generated by the exact NURBS representation method has size 3KB. It can be seen that the swept surface by the exact NURBS representation method has a more compact data structures than the skinning method.

Path Curve (Helix)				**Profile Curve**
Radius [mm]	**Pitch [mm]**	**Start Angle [°]**	**End Angle [°]**	**Radius [mm]**
30	80	0	90	10

Table E-1: Control Points for Silhouette Curves of a Conical End-Mill

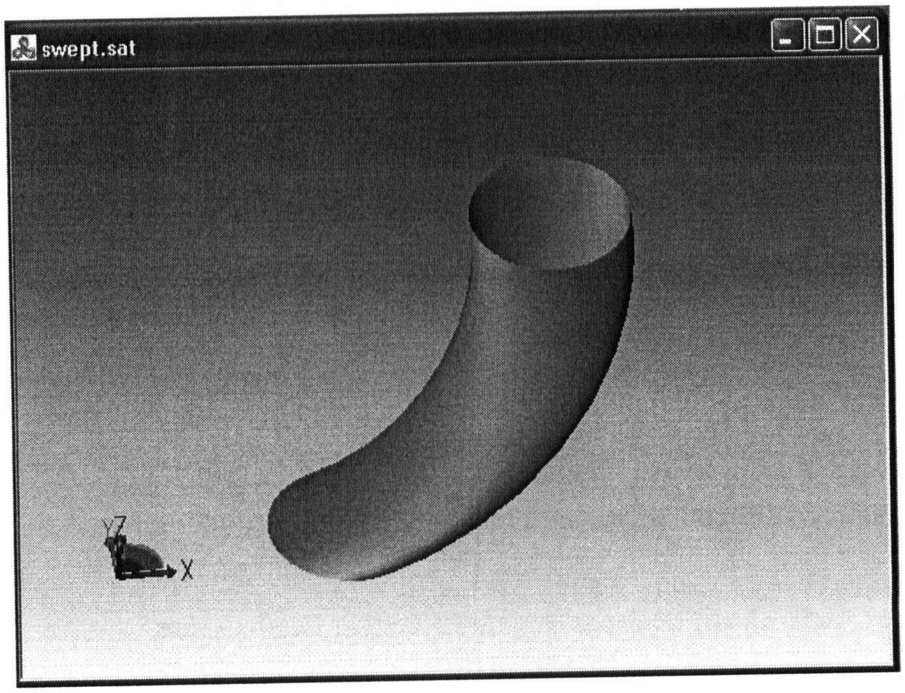


Figure E-1: A Swept Surface

E.1 The Sat File by Exact NURBS Representation Method

```
1000 0 1 0
24 ACIS Test Harness - 10.0 12 ACIS 10.0 NT 24 Thu Apr 07 21:34:53 2005
1 9.9999999999999995e-007 1e-010
face $-1 -1 $-1 $-1 $1 $-1 $-1 $2 forward single F F #
loop $-1 -1 $-1 $-1 $3 $0 F unknown #
spline-surface $-1 -1 $-1 forward { exactsur full nurbs 2 2 both open closed none none 2 5
          0 2 1 2
          0 2 0.25 2 0.5 2 0.75 2 1 2
          40 0 0 1
          40 -40 -10 0.70709999999999995
          0 -40 -20 1
          40 10 0 0.70709999999999995
          50 -30 -10 0.5
          10 -40 -20 0.70709999999999995
          30 10 0 1
          40 -20 -10 0.70709999999999995
          10 -30 -20 1
          20 10 0 0.70709999999999995
          30 -10 -10 0.5
          10 -20 -20 0.70709999999999995
          20 0 0 1
          20 -20 -10 0.70709999999999995
          0 -20 -20 1
          20 -10 0 0.70709999999999995
          10 -30 -10 0.5
          -10 -20 -20 0.70709999999999995
          30 -10 0 1
          20 -40 -10 0.70709999999999995
          -10 -30 -20 1
          40 -10 0 0.70709999999999995
          30 -50 -10 0.5
          -10 -40 -20 0.70709999999999995
          40 0 0 1
          40 -40 -10 0.70709999999999995
          0 -40 -20 1
          0
          0
          0
          0
          3 0.25 0.5 0.75
          0
          0
          F 1 F 0 F 1 F 0 } I I I I #
coedge $-1 -1 $-1 $4 $5 $6 $7 forward $1 $8 #
coedge $-1 -1 $-1 $6 $3 $-1 $9 forward $1 $10 #
coedge $-1 -1 $-1 $3 $6 $-1 $11 reversed $1 $12 #
coedge $-1 -1 $-1 $5 $4 $3 $7 reversed $1 $13 #
edge $-1 -1 $-1 $14 0 $15 1 $3 $16 forward @7 unknown F #
pcurve $-1 -1 $-1 1 $16 0 0 #
edge $-1 -1 $-1 $15 0 $15 1 $4 $17 forward @7 unknown F #
pcurve $-1 -1 $-1 1 $17 0 0 #
edge $-1 -1 $-1 $14 0 $14 1 $5 $18 forward @7 unknown F #
pcurve $-1 -1 $-1 -1 $18 0 0 #
pcurve $-1 -1 $-1 -1 $16 0 1 #
vertex $-1 -1 $-1 $7 $19 #
vertex $-1 -1 $-1 $7 $20 #
intcurve-curve $-1 -1 $-1 forward { parcur full nurbs 2 open 2
        0 2 1 2
        40 0 0 1
        40 -40 -10 0.70709999999999995
        0 -40 -20 1
        0
        spline forward { ref 0 } I I I I
        null_surface
        nubs 1 open 2
        0 1 1 1
        0 0
        1 0

        nullbs
        I I
        0
        0
        0
        surfl } I I #
intcurve-curve $-1 -1 $-1 forward { parcur full nurbs 2 closed 5
        0 2 0.25 2 0.5 2 0.75 2 1 2
        0 -40 -20 1
        10 -40 -20 0.70709999999999995
        10 -30 -20 1
        10 -20 -20 0.70709999999999995
        0 -20 -20 1
        -10 -20 -20 0.70709999999999995
        -10 -30 -20 1
        -10 -40 -20 0.70709999999999995
        0 -40 -20 1
        0
        spline forward { ref 0 } I I I I
        null_surface
        nubs 1 closed 2
        0 1 1 1
        1 0
        1 1

        nullbs
        I I
        3 0.25 0.5 0.75
        0
        0
        surfl } I I #
intcurve-curve $-1 -1 $-1 forward { parcur full nurbs 2 closed 5
        0 2 0.25 2 0.5 2 0.75 2 1 2
        40 0 0 1
        40 10 0 0.70709999999999995
        30 10 0 1
        20 10 0 0.70709999999999995
        20 0 0 1
        20 -10 0 0.70709999999999995
        20 -10 0 0.70709999999999995
        30 -10 0 1
        40 -10 0 0.70709999999999995
        40 0 0 1
        0
        spline forward { ref 0 } I I I I
        null_surface
        nubs 1 closed 2
        0 1 1 1
        0 0
        0 1

        nullbs
        I I
        3 0.25 0.5 0.75
        0
        0
        surf1 } I I #
point $-1 -1 $-1 40 0 0 #
point $-1 -1 $-1 0 -40 -20 #
End-of-ACIS-data
```

E.2 The Sat File by Skinning Method

```
1200 0 1 0
37 SolidWorks(2004231)-Sat-Convertor-2.0 12 ACIS 13.0 NT 24 Thu Apr 07 13:06:10 2005
1 9.9999999999999995e-007 1e-010
-0 body $1 -1 -1 $-1 $2 $-1 $-1 F #
-1 name_attrib-gen-attrib $-1 -1 $-1 $-1 $0 keep keep_kept ignore copy @5 Part2 #
-2 lump $3 -1 -1 $-1 $-1 $4 $0 F #
-3 rgb_color-st-attrib $-1 -1 $-1 $-1 $2 0.75294117647058822 0.75294117647058822 0.75294117647058822 #
-4 shell $-1 -1 -1 $-1 $-1 $-1 $5 $-1 $2 F #
-5 face $-1 -1 -1 $-1 $6 $7 $4 $-1 $8 reversed single F F #
-6 face $-1 -1 -1 $-1 $9 $10 $4 $-1 $11 forward single F F #
-7 loop $-1 -1 -1 $-1 $-1 $12 $5 F unknown #
-8 spline-surface $-1 -1 -1 $-1 forward { exactsur full nurbs 3 3 both open open none none 2 5
        0.5 3 1 3
        0 3 0.071428571428571425 3 0.14285714285714285 3 0.21428571428571427 3
0.25000000000000061 3
        19.999999999999996 1.2246467991473531e-015 0 1
        19.999999999999993 -19.999999999999996 0 0.33333333333333331
        39.999999999999993 -19.999999999999996 0 0.33333333333333331
        40 0 0 1
        19.999999999999996 -3.0173804715261601 -1.4164293882669949 0.9832852747878823
        16.98261952847383 -23.017380471526153 -1.4164293882669949 0.32776175826262749
        36.982619528473819 -26.034760943052316 -1.4164293882669949 0.32776175826262749
        40 -6.0347609430523219 -1.4164293882669949 0.9832852747878823
        19.32856967937445 -5.9591089148893666 -2.86928489744728 0.9832852747878823
        13.369460764485078 -25.287678594263813 -2.8692848974472795 0.32776175826262749
        32.69803044385953 -31.24678750915319 -2.8692848974472795 0.32776175826262749
        38.657139358748914 -11.918217829778738 -2.86928489744728 0.9832852747878823
        18.019377358048377 -8.6776747823511595 -4.2857142857142749 1
        9.3417025756972176 -26.697052140399538 -4.2857142857142749 0.33333333333333331
        27.361079933745593 -35.374726922750703 -4.2857142857142749 0.33333333333333331
        36.038754716096761 -17.355349564702326 -4.2857142857142749 1
        16.710185036722311 -11.396240649812952 -5.7021436739812703 0.9832852747878823
        5.3139443869093537 -28.10642568653526 -5.7021436739812703 0.32776175826262749
        22.024129423631663 -39.502666336348213 -5.7021436739812703 0.32776175826262749
        33.420370073444623 -22.792481299625912 -5.7021436739812703 0.9832852747878823
        14.828879084398052 -13.755323697036332 -7.1549991831615554 0.9832852747878823
        1.0735553873617152 -28.584202781434378 -7.1549991831615545 0.32776175826262749
        15.90243447175977 -42.339526478470717 -7.1549991831615545 0.32776175826262749
        29.657758168796111 -27.510647394072674 -7.1549991831615554 0.9832852747878823
        12.469796037174671 -15.636629649360593 -8.5714285714285499 1
        -3.1668336121859242 -28.106425686535264 -8.5714285714285499 0.33333333333333331
        9.3029624249887473 -43.743055335895846 -8.5714285714285499 0.33333333333333331
        24.939592074349342 -31.273259298721193 -8.5714285714285499 1
        10.11071298995129 -17.517935601684854 -9.9878579596955426 0.9832852747878823
        -7.4072226117335642 -27.628648591636132 -9.9878579596955444 0.32776175826262749
        2.7034903782177229 -45.14658419332099 -9.9878579596955444 0.32776175826262749
        20.22142597990258 -35.035871203369709 -9.9878579596955426 0.9832852747878823
        7.3921471224894955 -18.827127923010924 -11.440713468875831 0.9832852747878823
        -11.434980800521428 -26.21927504550041 -11.440713468875829 0.32776175826262749
        -4.0428336780319327 -45.04640296851133 -11.440713468875829 0.32776175826262749
        14.784294244978991 -37.654255846021847 -11.440713468875831 0.9832852747878823
        4.4504186791262894 -19.498558243636467 -12.857142857142824 1
        -15.04813956451018 -23.948976922762753 -12.857142857142824 0.33333333333333331
        -10.59772088538389 -43.44753516639922 -12.857142857142824 0.33333333333333331
        8.9008373582525788 -38.99711648727294 -12.857142857142824 1
        2.9919506021658848 -19.831444064809034 -13.559388861772048 0.99164263739394098
        -16.839493462643151 -22.823394666974913 -13.559388861772053 0.33054754579798035
        -13.847542860477263 -42.654838731783954 -13.559388861772053 0.33054754579798035
        5.9839012043317679 -39.662888129618075 -13.559388861772048 0.99164263739394098
        1.5023058653347638 -20.000000000000004 -14.276646291867323 0.98746395609091164
        -18.497694134665242 -21.502305865334762 -14.276646291867323 0.32915465203030386
        -16.99538826933048 -41.502305865334769 -14.276646291867323 0.32915465203030386
        3.004611730669525 -40.000000000000007 -14.276646291867323 0.98746395609091164
        -7.6638049594309347e-014 -19.999999999999996 -15.000000000000002 0.98746395609091175
        -20.000000000000078 -19.999999999999911 -15.000000000000002 0.32915465203030392
        -20.000000000000153 -39.999999999999915 -15.000000000000002 0.32915465203030392
        -1.5591121837237719e-013 -39.999999999999993 -15.000000000000002 0.98746395609091175
        0
        0
        0
        0
        0
        3 0.071428571428571425 0.14285714285714285 0.21428571428571427
        0
        F 1 F 0 F 1 F 0 } I I I I #
-9 face $-1 -1 -1 $-1 $13 $14 $4 $-1 $15 reversed single F F #
-10 loop $-1 -1 -1 $-1 $-1 $16 $6 F unknown #
-11 plane-surface $-1 -1 -1 $-1 30 0 0 0 0 1 1000 0 0 forward_v I I I I #
-12 coedge $-1 -1 -1 $-1 $17 $18 $19 $20 reversed $7 $21 #
-13 face $-1 -1 -1 $-1 $-1 $22 $4 $-1 $23 reversed single F F #
-14 loop $-1 -1 -1 $-1 $-1 $24 $9 F unknown #
-15 plane-surface $-1 -1 -1 $-1 -1.1140577831304493e-013 -30 -15 0 0 1 -3.713525943768165e-012 -1000 0
forward_v I I I I #
-16 coedge $-1 -1 -1 $-1 $19 $19 $25 $26 forward $10 $-1 #
-17 coedge $-1 -1 -1 $-1 $27 $12 $28 $29 forward $7 $30 #
-18 coedge $-1 -1 -1 $-1 $12 $27 $31 $32 reversed $7 $33 #
-19 coedge $-1 -1 -1 $-1 $16 $16 $12 $20 forward $10 $-1 #
-20 edge $-1 -1 -1 $-1 $34 -3.1415926535897931 $35 0 $19 $36 forward @7 unknown F #
-21 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 2 open 13
        0 2 0.26179938779914941 2 0.52359877559829882 2 0.78539816339744828 2 1.0471975511965976
2
        1.308996938995747 2 1.5707963267948966 2 1.8325957145940459 2 2.0943951023931953 2
2.3561944901923448 2
        2.6179938779914944 2 2.879793265790644 2 3.1415926535897931 2
        1 0
        0.94611254164616254 0
        0.94183174699473993 0
        0.93677493932354483 0
        0.8943375672974061 0
        0.85684756116418259 0
        0.85355339059327373 0
        0.84995023689285931 2.1337867821239758e-018
        0.8169872981077807 2.1654431452783809e-017
        0.78586701224058342 1.8772212167380323e-018
        0.78291312439676708 0
        0.7798739506635517 2.5908894082864976e-018
        0.75000000000000022 2.8058371412343862e-017
        0.72012604933638202 2.5908894082875245e-018
```

```
          0.71708687560315842 0
          0.71413298775934742 3.1880491120183416e-018
          0.6830127018922193 3.67753093501905e-017
          0.65004976310714069 3.62376953516684e-018
          0.64644660940672627 0
          0.64315243883581741 2.8286329116603327e-018
          0.60566243270259346 3.5020489078212388e-017
          0.56322506067645517 3.7287069478462213e-018
          0.5581682530052603 0
          0.55388745835383768 0
          0.50000000000000011 0
          0.001
          0.0107299081764064
          spline reversed { ref 0 } I I I I
          } 0 0 #
-22 loop $-1 -1 -1 $-1 $-1 $28 $13 F unknown #
-23 spline-surface $-1 -1 -1 $-1 forward { exactsur full nurbs 3 3 both open open none none 2 5
          0 3 0.5 3
          0 3 0.071428571428571425 3 0.14285714285714285 3 0.21428571428571427 3
0.25000000000000178 3
          40 0 0 1
          39.999999999999993 19.999999999999996 0 0.33333333333333331
          19.999999999999996 19.999999999999996 0 0.33333333333333331
          19.999999999999996 1.2246467991473531e-015 0 1
          40 -6.0347609430523219 -1.4164293882669949 0.9832852747878823
          43.017380471526153 13.965239056947674 -1.4164293882669949 0.32776175826262749
          23.017380471526153 16.982619528473833 -1.4164293882669949 0.32776175826262749
          19.999999999999996 -3.0173804715261601 -1.4164293882669949 0.9832852747878823
          38.657139358748914 -11.918217829778738 -2.86928489744728 0.9832852747878823
          44.616248273638263 7.4103518495957141 -2.8692848974472795 0.32776175826262749
          25.287678594263813 13.369460764485083 -2.8692848974472795 0.32776175826262749
          19.32856967937445 -5.9591089148893666 -2.86928489744728 0.9832852747878823
          36.038754716096761 -17.355349564702326 -4.2857142857142749 1
          44.716429498447908 0.66402779334605655 -4.2857142857142749 0.33333333333333331
          26.697052140399538 9.3417025756972176 -4.2857142857142749 0.33333333333333331
          18.019377358048377 -8.6776747823511595 -4.2857142857142749 1
          33.420370073444623 -22.792481299625912 -5.7021436739812703 0.9832852747878823
          44.816610723257561 -6.0822962629035953 -5.7021436739812703 0.32776175826262749
          28.10642568653526 5.3139443869093563 -5.7021436739812703 0.32776175826262749
          16.710185036722311 -11.396240649812952 -5.7021436739812703 0.9832852747878823
          29.657758168796111 -27.510647394072674 -7.1549991831615554 0.9832852747878823
          43.413081865832432 -12.681768309674615 -7.1549991831615545 0.32776175826262749
          28:584202781434378 1.0735553873617178 -7.1549991831615545 0.32776175826262749
          14.828879084398052 -13.755323697036332 -7.1549991831615554 0.9832852747878823
          24.939592074349342 -31.273259298721193 -8.5714285714285499 1
          40.576221723709935 -18.803463261546511 -8.5714285714285499 0.33333333333333331
          28.106425686535264 -3.166833612185922 -8.5714285714285499 0.33333333333333331
          12.469796037174671 -15.636629649360593 -8.5714285714285499 1
          20.22142597990258 -35.035871203369709 -9.9878579596955426 0.9832852747878823
          37.739361581587431 -24.925158213418406 -9.9878579596955444 0.32776175826262749
          27.628648591636143 -7.4072226117335633 -9.9878579596955444 0.32776175826262749
          10.11071298995129 -17.517935601684854 -9.9878579596955426 0.9832852747878823
          14.784294244978991 -37.654255846021847 -11.440713468875831 0.9832852747878823
          33.611422167989907 -30.262108723532343 -11.440713468875829 0.32776175826262749
          26.219275045500417 -11.434980800521428 -11.440713468875829 0.32776175826262749
```

```
        7.3921471224894955 -18.827127923010924 -11.440713468875831 0.9832852747878823
        8.9008373582525788 -38.997116487272294 -12.857142857142824 1
        28.399395601889047 -34.546697808146646 -12.857142857142824 0.33333333333333331
        23.948976922762757 -15.04813956451018 -12.857142857142824 0.33333333333333331
        4.4504186791262894 -19.498558243636467 -12.857142857142824 1
        5.9839012043316719 -39.662888129618089 -13.559388861772076 0.99164263739394076
        25.815345269140721 -36.670937527452253 -13.559388861772076 0.33054754579798024
        22.823394666974878 -16.839493462643212 -13.559388861772076 0.33054754579798024
        2.9919506021658369 -19.831444064809045 -13.559388861772076 0.99164263739394076
        3.0046117306693301 -40.000000000000036 -14.276646291867369 0.98746395609091131
        23.00461173066935 -38.497694134665366 -14.276646291867369 0.32915465203030375
        21.502305865334673 -18.497694134665352 -14.276646291867369 0.32915465203030375
        1.5023058653346664 -20.000000000000014 -14.276646291867369 0.98746395609091131
        -4.4928782083082226e-013 -40 -15.000000000000073 0.98746395609091175
        19.999999999999549 -40.00000000000022 -15.000000000000071 0.32915465203030392
        19.999999999999773 -20.000000000000227 -15.000000000000071 0.32915465203030392
        -2.2332635082353182e-013 -19.999999999999993 -15.000000000000073 0.98746395609091175
        0
        0
        0
        0
        0
        3 0.071428571428571425 0.14285714285714285 0.21428571428571427
        0
        F 1 F 0 F 1 F 0 } I I I I #
-24 coedge $-1 -1 -1 $-1 $37 $37 $38 $39 reversed $14 $-1 #
-25 coedge $-1 -1 -1 $-1 $31 $28 $16 $26 reversed $22 $40 #
-26 edge $-1 -1 -1 $-1 $35 0 $34 3.1415926535897931 $16 $41 forward @7 unknown F #
-27 coedge $-1 -1 -1 $-1 $18 $17 $37 $42 forward $7 $43 #
-28 coedge $-1 -1 -1 $-1 $25 $38 $17 $29 reversed $22 $44 #
-29 edge $-1 -1 -1 $-1 $34 0 $45 0.25000000000000061 $17 $46 forward @7 unknown F #
-30 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 1 open 2
        0 1 0.25000000000000061 1
        0.5 0
        0.50000000000000011 0.25000000000000061
        0.001
        -1
        spline reversed { ref 0 } I I I I
        } 0 0 #
-31 coedge $-1 -1 -1 $-1 $38 $25 $18 $32 forward $22 $47 #
-32 edge $-1 -1 -1 $-1 $35 0 $48 0.25000000000000061 $31 $49 forward @7 unknown F #
-33 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 1 open 2
        -0.25000000000000061 1 0 1
        1 0.25000000000000061
        1 0
        0.001
        -1
        spline reversed { ref 0 } I I I I
        } 0 0 #
-34 vertex $-1 -1 -1 $-1 $20 $50 #
-35 vertex $-1 -1 -1 $-1 $20 $51 #
-36 ellipse-curve $-1 -1 -1 $-1 30 0 0 0 0 1 10.000000000000002 0 0 1 F -3.1415926535897931 F 0 #
-37 coedge $-1 -1 -1 $-1 $24 $24 $27 $42 reversed $14 $-1 #
-38 coedge $-1 -1 -1 $-1 $28 $31 $24 $39 forward $22 $52 #
-39 edge $-1 -1 -1 $-1 $48 -2.0194839173657899e-030 $45 3.1415926535897931 $38 $53 forward
```

```
        unknown F #
-40 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 2 open 13
        -3.1415926535897931 2 -2.879793265790644 2 -2.6179938779914944 2 -2.3561944901923448 2 -
2.0943951023931957 2
        -1.8325957145940461 2 -1.5707963267948966 2 -1.3089969389957472 2 -1.0471975511965979 2 -
0.78539816339744828 2
        -0.5235987755982987 2 -0.26179938779914935 2 0 2
        0.5 0
        0.44611254164616165 3.0391826216922957e-016
        0.44183174699473904 3.280613876924436e-016
        0.43677493932354422 2.9313198471132126e-016
        0.39433756729740671 0
        0.35684756116418265 3.7385529923538775e-019
        0.35355339059327379 4.0670520382444336e-019
        0.34995023689285948 3.6662924108056502e-019
        0.31698729810778081 0
        0.2858670122406527 4.3022305350325433e-014
        0.28291312439684152 4.7105913879169037e-014
        0.27987395066361792 4.275618838927289e-014
        0.25 0
        0.22012604933644889 4.2731648028064627e-014
        0.21708687560323295 4.7078876947551781e-014
        0.21413298775941622 4.299761224850589e-014
        0.18301270189221919 0
        0.15004976310714063 0
        0.14644660940672627 0
        0.14315243883581746 4.9183516869039065e-018
        0.10566243270259383 6.089269513339682e-017
        0.063225060676456807 2.6800737900841621e-016
        0.058168253005262127 2.9268702037714195e-016
        0.053887458353839519 2.7114721186238487e-016
        0 0
        0.001
        0.010729908174256897
        spline reversed { ref 2 } I I I I
        } 0 0 #
-41 ellipse-curve $-1 -1 -1 $-1 30 0 0 0 0 1 10.000000000000002 0 0 1 F 0 F 3.1415926535897931 #
-42 edge $-1 -1 -1 $-1 $45 -3.1415926535897931 $48 -2.0194839173657899e-030 $27 $54 forward @7
unknown F #
-43 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 2 open 13
        -3.1415926535897931 2 -2.879793265790644 2 -2.6179938779914944 2 -2.3561944901923448 2 -
2.0943951023931957 2
        -1.8325957145940461 2 -1.5707963267948966 2 -1.3089969389957472 2 -1.0471975511965979 2 -
0.78539816339744828 2
        -0.5235987755982987 2 -0.26179938779914935 2 -2.0194839173657899e-030 2
        0.5 0.25000000000000061
        0.55388745835383824 0.25000000000000039
        0.55816825300526085 0.25000000000000033
        0.56322506067645561 0.25000000000000033
        0.60566243270259346 0.25000000000000061
        0.64315243883581752 0.25000000000000056
        0.64644660940672638 0.25000000000000056
        0.6500497631071408 0.25000000000000056
        0.68301270189221941 0.25000000000000056
        0.71413298775934742 0.24999999999995826
        0.71708687560315854 0.24999999999995426
        0.72012604933638213 0.24999999999995853
        0.75000000000000011 0.25000000000000061
        0.77987395066355136 0.24999999999995859
        0.7829131243967673 0.24999999999995429
        0.78586701224058408 0.24999999999995831
        0.81698729810778081 0.25000000000000061
        0.84995023689285976 0.25000000000000061
        0.85355339059327417 0.25000000000000061
        0.85684756116418304 0.25000000000000061
        0.89433756729740677 0.25000000000000061
        0.93677493932354361 0.25000000000000044
        0.94183174699473826 0.25000000000000039
        0.94611254164616077 0.25000000000000039
        1 0.25000000000000061
        0.001
        0.010729908174639924
        spline reversed { ref 0 } I I I I
        } 0 0 #
-44 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 1 open 2
        -0.25000000000000061 1 0 1
        0.49999999999999989 0.25000000000000067
        0.5 0
        0.001
        -1
        spline reversed { ref 2 } I I I I
        } 0 0 #
-45 vertex $-1 -1 -1 $-1 $39 $55 #
-46 intcurve-curve $-1 -1 -1 $-1 forward { exactcur full nurbs 2 open 5
        0 2 0.071428571428571425 2 0.14285714285714285 2 0.21428571428571427 2
0.25000000000000061 2
        19.999999999999996 1.2246467991473531e-015 0 1
        19.999999999999996 -4.564869487802997 -2.1428571428571379 0.97492791218182362
        18.019377358048377 -8.6776747823511595 -4.2857142857142749 1
        16.038754716096765 -12.790480076899319 -6.4285714285714119 0.97492791218182362
        12.469796037174671 -15.636629649360593 -8.5714285714285499 1
        8.900837358252577 -18.482779221821868 -10.714285714285687 0.97492791218182362
        4.4504186791262894 -19.498558243636467 -12.857142857142824 1
        2.2534587980021841 -20.000000000000007 -13.914969437800982 0.98746395609091153
        -7.6198863063682964e-014 -19.999999999999996 -15 0.98746395609091175
        0
        null_surface
        null_surface
        nullbs
        nullbs
        -1
        -1
        I I
        0
        3 0.071428571428571425 0.14285714285714285 0.21428571428571427
        0

        -1
        nullbs F 1 F 0 } I I #
 -47 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 1 open 2
```

```
        0 1 0.25000000000000061 1
        0 0
        1.445876070142189e-016 0.25000000000000061
        0.001
        -1
        spline reversed { ref 2 } I I I I
        } 0 0 #
-48 vertex $-1 -1 -1 $-1 $39 $56 #
-49 intcurve-curve $-1 -1 -1 $-1 forward { exactcur full nurbs 2 open 5
        0 2 0.071428571428571425 2 0.14285714285714285 2 0.21428571428571427 2
0.25000000000000061 2
        40 0 0 1
        40 -9.1297389756059975 -2.1428571428571379 0.97492791218182362
        36.038754716096761 -17.355349564702326 -4.2857142857142749 1
        32.077509432193537 -25.58096015379865 -6.4285714285714119 0.97492791218182362
        24.939592074349342 -31.273259298721193 -8.5714285714285499 1
        17.801674716505154 -36.965558443643744 -10.714285714285687 0.97492791218182362
        8.9008373582525788 -38.99711648727294 -12.857142857142824 1
        4.5069175960043655 -40.000000000000014 -13.914969437800982 0.98746395609091153
        -1.4854103775072659e-013 -40 -15 0.98746395609091175
        0
        null_surface
        null_surface
        nullbs
        nullbs
        -1
        -1
        I I
        0
        3 0.071428571428571425 0.14285714285714285 0.21428571428571427
        0

        -1
        nullbs F 1 F 0 } I I #
-50 point $-1 -1 -1 $-1 19.999999999999996 1.2246467991473531e-015 0 #
-51 point $-1 -1 -1 $-1 40 0 0 #
-52 pcurve $-1 -1 -1 $-1 0 forward { exppc nubs 2 open 13
        -2.0194839173657899e-030 2 0.26179938779914941 2 0.52359877559829882 2
0.78539816339744828 2 1.0471975511965976 2
        1.308996938995747 2 1.5707963267948966 2 1.8325957145940459 2 2.0943951023931953 2
2.3561944901923448 2
        2.6179938779914944 2 2.879793265790644 2 3.1415926535897931 2
        1.445876070142189e-016 0.25000000000000061
        0.053887458353840234 0.25000000000000105
        0.05816825300526287 0.25000000000000105
        0.063225060676457376 0.250000000000001
        0.10566243270259346 0.25000000000000056
        0.1431524388358173 0.25000000000000056
        0.14644660940672607 0.25000000000000056
        0.15004976310714049 0.25000000000000056
        0.18301270189221935 0.25000000000000061
        0.21413298775941625 0.25000000000000167
        0.21708687560323253 0.25000000000000178
        0.22012604933644808 0.25000000000000167
        0.24999999999999994 0.25000000000000056
        0.27987395066361787 0.25000000000000167
        0.28291312439684146 0.25000000000000178
        0.28586701224065253 0.25000000000000167
        0.31698729810778065 0.25000000000000061
        0.34995023689285931 0.25000000000000061
        0.35355339059327368 0.25000000000000061
        0.35684756116418248 0.25000000000000061
        0.39433756729740643 0.25000000000000061
        0.43677493932354428 0.25000000000000089
        0.44183174699473909 0.25000000000000089
        0.44611254164616176 0.25000000000000089
        0.49999999999999994 0.25000000000000061
        0.001
        0.010729908176226099
        spline reversed { ref 2 } I I I I
        } 0 0 #
-53 ellipse-curve $-1 -1 -1 $-1 -1.1140577831304493e-013 -30 -15 0 0 1 -3.7135259437681647e-014 -
10.000000000000002 0 1 F -2.0194839173657899e-030 F 3.1415926535897931 #
-54 ellipse-curve $-1 -1 -1 $-1 -1.1140577831304493e-013 -30 -15 0 0 1 -3.7135259437681647e-014 -
10.000000000000002 0 1 F -3.1415926535897931 F -2.0194839173657899e-030 #
-55 point $-1 -1 -1 $-1 -7.6198863063682964e-014 -19.999999999999996 -15 #
-56 point $-1 -1 -1 $-1 -1.4854103775072659e-013 -40 -15 #
End-of-ACIS-data
```