Time-optimal CNC Tool Paths—A Mathematical Model of Machining

BY

Taejung Kim

B.S., Seoul National University (summa cum laude)
(1993)
M.S.M.E., Massachusetts Institute of Technology
(1995)

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Mechanical Engineering

at the
Massachusetts Institute of Technology

February 2001

Signature of Author ______________________

Department of Mechanical Engineering
January 12, 2001

Certified by [illegible]

Sanjay E. Sarma
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by [illegible]

Ain A. Sonin
Chairman, Department Committee for Graduate Student

Time-optimal CNC Tool Paths——A Mathematical Model of Machining

Taejung Kim

Submitted to the Department of Mechanical Engineering
on January 12, 2001 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

Abstract

Free-form surface machining is a fundamental but time-consuming process in modern manufacturing. The central question we ask in this thesis is how to reduce the time that it takes for a 5-axis CNC (Computer Numerical Control) milling machine to sweep an entire free-form surface in its finishing stage.

We formulate a non-classical variational time-optimization problem defined on a 2-dimensional manifold subject to both equality and inequality constraints. The machining time is the cost functional in this optimization problem. We seek for a preferable vector field on a surface to obtain skeletal information on the toolpaths. This framework is more amenable to the techniques of continuum mechanics and differential geometry rather than to path generation and conventional CAD/CAM (Computer Aided Design and Manufacturing) theory.

After the formulation, this thesis derives the necessary conditions for optimality. We decompose the problem into a series of optimization problems defined on 1-dimensional streamlines of the vector field and, as a result, simplify the problem significantly.

The anisotropy in kinematic performance has a practical importance in high-speed machining. The greedy scheme, which this thesis implements for a parallel hexapod machine tool, uses the anisotropy for finding a preferable vector field. Numerical integration places tool paths along its integral curves. The gaps between two neighboring toolpaths are controlled so that the surface can be machined within a specified tolerance. A conservation law together with the characteristic theory for partial differential equations comes into play in finding appropriately-spaced toolpaths, avoiding unnecessarily-overlapping areas. Since the greedy scheme is based on a local approximation and does not search for the global optimum, it is necessary to judge how well the greedy paths perform. We develop an approximation theory and use it to economically evaluate the performance advantage of the greedy paths over other standard schemes.

In this thesis, we achieved the following two objectives: laying down the theoretical basis for surface machining and finding a practical solution for the machining problem. Future work will address solving the optimization problem in a stricter sense.

Thesis Committee: **Professor Sanjay E. Sarma**, Chairman, Mechanical Engineering, MIT
Professor Samir Nayfeh, Mechanical Engineering, MIT
Professor Nicholas Patrikalakis, Ocean Engineering, MIT

ACKNOWLEDGMENT

This work was supported by the National Science Foundation Career Award and the Ford Motor Company. I thank them for their financial support.

I feel very fortunate to have had Professor Sanjay E. Sarma as my thesis advisor and mentor. He picked this topic for me, supported me financially, and allowed me great freedom in research; he provided me with the luxury to ponder over two great bodies of human knowledge, geometry and mechanics. His critical advice and enthusiastic encouragement were invaluable in my making any progress. I also appreciate his patience to improve the way I use English, Graphic, Mathematical and Programming languages, still none of which I am fluent in. Frequently, I was surprised at his capturing my hazy reasoning and converting it to either English or mathematical statements. In spite of some personal circumstances, I could enjoy the last millennium as his student. I am grateful to the other members of my thesis committee for their precious time and assistance. Professor Samir Nayfeh never forgot to encourage me. Professor Nicholas Patrikalakis introduced computational geometry to me through his exciting course. All my committee members are patient listeners, logical thinkers, passionate engineers and my role models in the academic world.

I am grateful to all the enthusiastic members of RAMLab. Prasad and Jaehyun helped me to settle in this new lab. I appreciate Elmer and Edmund's effort in keeping our lab being organized. Stephen and Mahadevan did not hesitate to proofread this thesis. Seungkil raised many problems of machine tool design and broadened my perspective on the matter. I am also grateful to my office-mates. Justin was available whenever I had difficulty in expressing myself in English. Pablo and I will never forget the coldest winter that we have passed through together.

Boston might have been a dull place without my friends. I remember such romantic times, spent together with Junbeom, Sunghwan, Junhee and Sanghoon at Tang Hall. I am grateful to Kay, Donghyun, Hyungsoo, Yoon and Youyon for staying in Boston.

***

This thesis is dedicated to my parents,

Jong Doo Kim and Jayoak Cho,

for always giving me the best in the world.

The advantage of confining attention to a definite group of abstractions, is that you confine your thoughts to clear-cut definite things, with clear-cut definite relations. ... ... It was an ambitious enterprise, and they were completely successful. ... ... Maupertuis' success in this particular case shows that almost any idea which jogs you out of your current abstractions may be better than nothing. ... . But this would be a mere tautology. ... ... Thus nature is a structure of evolving processes. The reality is the process.

—Alfred North Whitehead in Science and the Modern World—

The process of solution is correct, but our answer is $a = a$, *or* $x = x$, *or* $0 = 0$.

—Leo Tolstoy in My Confession—

So there is nothing new under the sun.

—Ecclesiastes 1:9—

But new wine must be put into fresh wineskins.

—Luke 5:38—

Table of Contents

1 INTRODUCTION

In this thesis, we attempt to mathematically understand the *Numerical Control (NC) milling process* of solid parts bounded by *free-form surfaces* in the **finishing stages**. Unlike conventional CAM (Computer-Aided Manufacturing)[†] systems, which usually concentrate on the geometric aspects of machining, the model we construct in this thesis includes the **kinematics** of a specific machine tool. There is another significant deviation from the conventional approach: we model tool paths as a finite selection from the streamlines of a **vector field** on the surface. We deal with tool paths as a whole rather than as individual entities. Based on the established framework, the path-planning task is formulated as a constrained (non-classical) variational time-optimization problem. In other words, given geometric tolerance, we seek to finish, as quickly as possible, machining a given surface while respecting the kinematic limits of machine tools.

Tool and die manufacture remains a bottleneck in product development today. In the automotive industry, for example, dies often require tens of hours to machine. Reducing machining time is therefore of considerable economic importance because the time to market is a crucial factor in today's competitive economic environment. We propose a subtle but effective way to increase the productivity of NC machine tools. We describe how the global geometry (as opposed to local geometries or feed rates) of tool paths can be optimized for a specific machine tool just as computer programs are today compiled for specific computers. Of all of tool paths possible, we will seek the one that minimizes a very important performance metric: total cutting time.

Solving this problem is numerically-intensive but finding the quickest path would have been much more intractable in the absence of our framework. We simplify the problem and suggest an approximate solution that can be found utilizing only the local information on surface geometry and machine kinematics. This approach shows promise for machining a surface with high-speed machine tools; in this

[†] We use the terminology, CAM, in its narrow sense—computer-aided NC part-programming or tool path generation.

case, such local information carries a strong implication on the best directions of cuts. The algorithms and the concepts originating from this local approach provide a good basis of attacking more general problems. Additionally, the algorithm shows promise for being highly automated while not sacrificing machining efficiency.

We claim three main contributions in this dissertation work. First, we formulate a fairly general optimization problem that captures the reality of machining. The formulated problem is amenable to numerical analysis. This not only lays down a theoretical basis for the field of surface machining but also potentially conveys the implications of future practical implementations. Second, we enlarge the scope of path planning. The kinematics of a machine tool is now reflected in the formulation. This consideration has a practical importance in the context of *high-speed machining* and in recent trends in parallel mechanisms. Third, we develop an algorithm that finds approximate solutions to the formulation presented.

The rest of this chapter briefly explains related work, the motivation for this work, and the scope of the study. We introduce the problem of interest gradually while the next chapter takes care of more rigorous definitions. Finally, we present an outline of this thesis.

1.1 NC Tool Path Generation: State and Scope

In this section, we provide a brief overview of CNC machining and tool path generation. We then describe the problem that we deal with in this thesis.

HARDWARE

Machining has been one of the most important means of manufacturing processes in history. Based on the century-old machine tool technology, together with the electronics and the emerging notion of computers, the first NC machine tool was built in the 1950s. In the following decades, various types of NC machine tools were successfully commercialized. With the advance in computer-technology, NC machines evolved in the stage of DNC (Direct/Distributed NC) and CNC (Computerized NC). *Milling machines*, which we consider in this study, were among the earliest NC machine tools [13]. They perform wide range of operations. Milling machines equipped with a *tool changer* are called *machining centers* and are widely utilized in everyday manufacturing practice. Today, besides being a backbone of the automotive and aerospace industry, they play a crucial role in the manufacture of billions of dollars of consumer products, made of metal or plastic, that require dies and molds.

Recently, the advantage of parallel mechanisms for machines/robots over serial mechanisms has been widely investigated, even though it is an open question whether this new type of machine tools can easily outperform the traditional structures that have been optimized for two centuries [11]. An innovation—the metamorphosis of mechanical skeletons—has begun from the very bottom of this technology. We have used a parallel mechanism called a hexapod for examples in this work. However, the theory is equally applicable for all of mechanisms.

In recent years, the insatiable need for performance of machine tools has spawned the area of *high speed machining*. The local approach developed in this work has special significance in this context. Today, high speed spindles cut at *30,000 - 50,000 rpm* and can sustain very high feed rates, on the order of *500 inch/min - 1000 inch/min*. Unfortunately, advances in actuation technology have not kept pace with advances in spindle technology. The result is that motion actuation is often the bottleneck in machine performance. Our research addresses an opportunity to squeeze better performance out of machine tools and enables higher speed machining.

SOFTWARE

In the early days of NC, users communicated with the controller (MCU) directly in machine languages consisting of *G-codes* and *M-codes*. In the 1970s, high-level part-programming languages such as APT and ADAPT emerged, which were compiled into the machine languages. The advantage is similar to the one achievable from high-level computer languages such as C over *assembly languages*, but the necessary part-geometry had to be typed in the program files manually. Advances in CAD (Computer-Aided Design) made it possible to use the CAD data in generating machine code, and the direct link between the design task and the manufacturing task was established. Such systems are called CAD/CAM systems.

The ultimate goal of CAD/CAM systems, in its narrow sense, is seemingly simple: given a shape of a part, generate "efficient" tool paths that produce the part within tolerance. However, this task turns out to be challenging if we require high quality and efficiency of generated paths for complex part-geometry. The CAD/CAM problem is quite inter-disciplinary and has attracted many researchers from diverse fields, including mathematics, industrial engineering, mechanical engineering, and computer science. In recent years, a tremendous amount of programming effort has been devoted to solving this problem, and several CAD/CAM systems have been commercialized. However, CAD/CAM systems are still user-intensive, and users are responsible for recognizing and harnessing any advantages from a particular strategy of path-generation. There is still room for improvement in systems available today, especially when the users push the envelope of performance.

Free-form Surfaces

For engineering and aesthetic reasons, non-prismatic parts, bounded by free-form surfaces, are becoming increasingly common in engineering design. Examples include fuselages of planes, turbine blades, bodies of automobiles and surfaces in molds. Manually designing and machining these parts require a tremendous amount of human effort. Surface machining is the area where CAD/CAM is vital.

In today's CAD/CAM systems, free-form surfaces of parts are represented mathematically in parametric form [1 and 2]. Examples of parametric surfaces include Bezier surfaces, B-Spline surfaces and NURBS surfaces. The theory that is developed in this work is applicable to any parametric form, but the implementation was made for the simplest form, Bezier tensor product surfaces.

A surface often consists of several patches [*e.g.* 3, 4 and 5]. It is also possible that a surface is trimmed off for various reasons [*e.g.* 6 and 7]. We deal with a surface consisting of only one patch. Additional handling for either trimmed or multi-patched surfaces is not covered in the current work, which is more involved but possible by extension of this work.

Geometric Tolerances

Tool paths can be thought of as mathematical curves in the Euclidean 3-space. In physics, a time-parameterized curve of a distinguished point is called a trajectory. In finishing, the cutter moves in tangential contact with the surface. In path-planning literature [*e.g.* 19], such a point of contact is called a cutter contact (CC) point. A trajectory of the CC-point can be thought of as a surface curve. A certain nominal point irrespective of the surface, for example the tool tip, is selected as a cutter location (CL) point for the purpose of commanding a machine tool. Proper discrete data of the trajectories of the CC-point or the CL-point are needed for communicating with the controller in a finite period. They are called CC-data and CL-data, respectively. Generating the discrete data from a trajectory is a relatively easy task achieved through the consideration on the *chordal deviation* as shown in Figure 1-(*a*), [*e.g.*, 17 and 18]. We do not cover this issue; fast *block processing rate* and the availability of higher order *interpolations* makes this a minor issue in modern controllers.

In addition to the chordal deviation, the heights of *cusps/scallops* are a measure of the (macro) roughness of a machined surface. Generally, cusps are not avoidable as shown in Figure 1-(*b*), but their heights are reducible with fine path-intervals. Cusp heights vary from place to place. It is usual to prevent the cusp heights from exceeding a certain value in order to achieve the required finish. The maximum cusp height is called a *cusp-height-limit*. It is surprising that this requirement is not guaranteed by every CAD/CAM

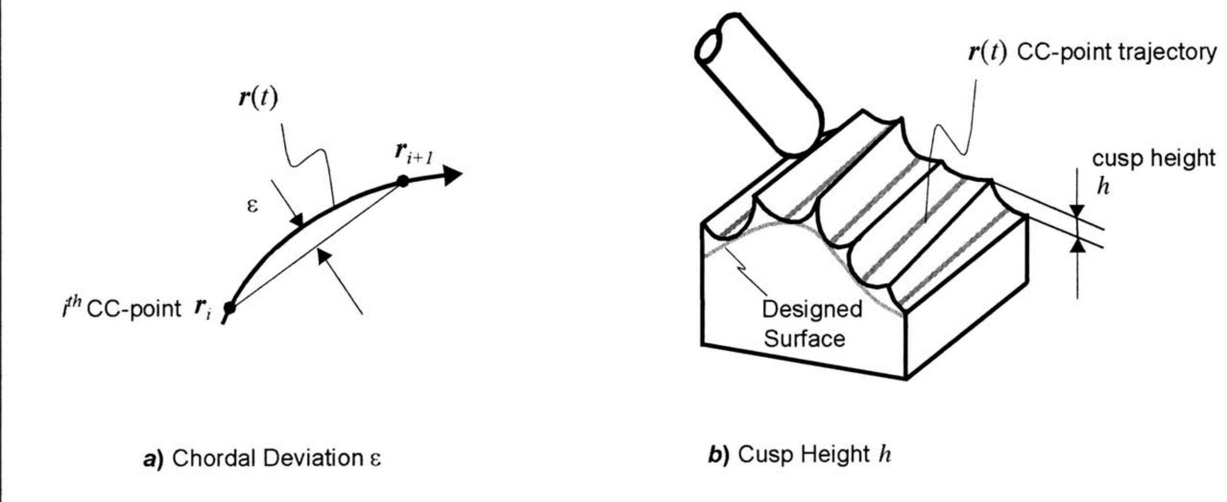


Figure 1 Two kinds of discrepancy

system. Sometimes, even when systems allow users to specify the cusp-height-limit, the condition is not guaranteed [9 and 10].

The (micro) roughness of machined surfaces, the cutter marks [14] and the vibration of the machine tool are related to various dynamic factors: machine tool power, cutting dissipation, machine inertia, machine stiffness, feed rate, *etc*. We do not consider high order dynamics in the path-planning stage. As a rule, the negative effect on both the roughness and the vibration is proportional to the cutting force. We simply assume that a machine tool is *rigid* until the cutting force reaches a threshold value. In finishing stages, the cutting force is quite negligible because a relatively small amount of material is removed. It is not unreasonable that a (well-designed) machine tool can be considered to be an ideal rigid machine during finishing.

FINISHING MODULE SPECIFICATION

The goal of this work is to find "efficient" trajectories. There are several other tasks that need to be planned so that proper milling operation can be assured. Examples are spindle speed, tool selection, tool change sequence, fixturing, tool wear, depth of cut and so on. We do not cover these topics in this work. We assume that appropriate selection of some process variables is done in a higher level CAD/CAM system or in a CAPP (Computer-aided Process Planning) system [8].

In the current work, we are interested only in a **finishing**. It is assumed that a roughing process generates a near-net shape part that has only a small amount of uncut material. The objective of finishing is to remove these remnants as quickly as possible and to sculpture the fine detail of designed surfaces. The desired level of surface finish—the *cusp-height-limit*—is specified by users either in the design stage or in the planning stage. Machine kinematics and actuator limits are specified to capture the anisotropy of

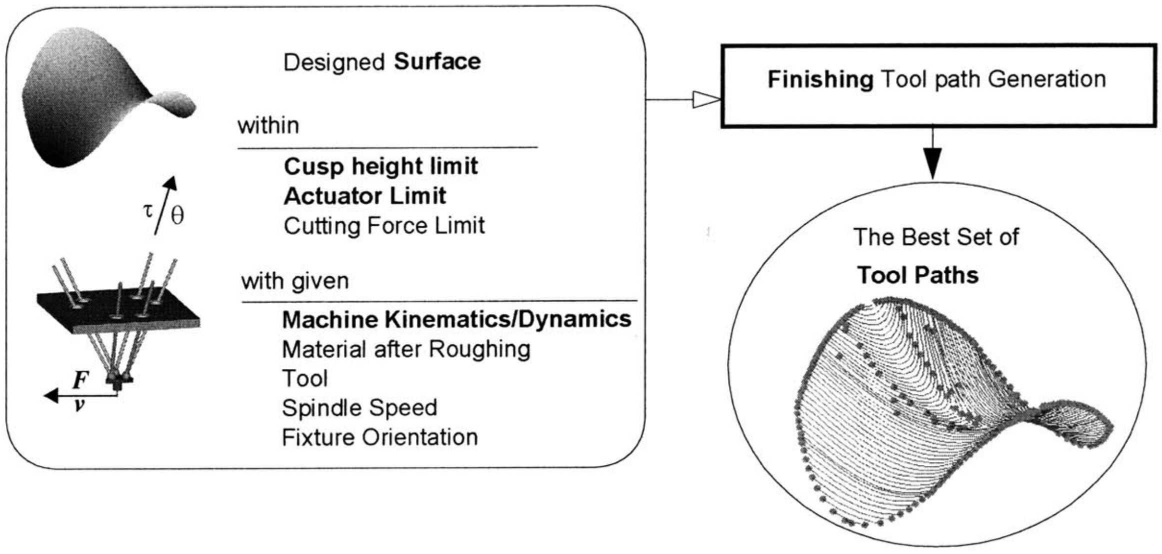


Figure 2 Finishing Tool path Generation

machine tool performance. This can be integrated into the system just as printing devices are specified in today's word-processors or operating systems. Our system is asked to generate an efficient tool path for the specified machine once other higher level details have been determined. This is shown in Figure 2.

1.2 Motivation

We now show a few simple examples to explain how the geometry of surfaces and the kinematics of machine tools come into play in machining performance. The consideration of the kinematics is a substantial departure from the conventional geometric approach in today's CAM systems.

Length vs. Speed and Direction vs. Overlap

There are two intuitive approaches for reducing the machining time: one is to reduce the length of paths and the other is to increase the feed rate. The two approaches have never been considered in a single framework.

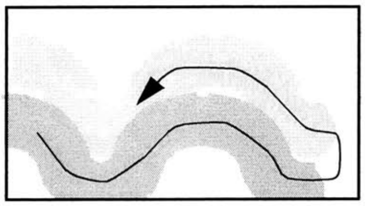
First, consider reducing the length of paths; to achieve this, it has been attempted either to move in the directions of the widest cut [37, 36 and 38] or to avoid overlapping paths [18, 22 and 28]. These are quite different strategies. For example, along Path (*c*) in Figure 3, the tool conforms to the cylindrical surface better than along Path (*d*). Given a tool size and a cusp-height-limit, Path (*c*) allows wider cuts and therefore will be shorter. Another example concerns the importance of avoiding the overlaps. When an apple (a sphere) is peeled, it makes no difference locally whether one peels vertically or horizontally, in other words, whether one chooses Path (*e*) or Path (*f*). However, Path (*f*) is shorter not because the independent cuts are wider but because it does not overlap. Path (*e*) overlaps highly at the top of the sphere. The question is then, whether we can achieve both objectives—both avoiding the overlaps and moving in the most efficient directions—at the same time for general surfaces. The answer is "no" and the underlying geometrical problem is subtle. In addition, neither of the two strategies, in general, leads to the shortest path. It should also be pointed out that the pathological case, "spotting,"[†] is in a naive sense the shortest because its length is zero. Some mathematical rigor is needed for constructing a problem that yields practically acceptable solutions.

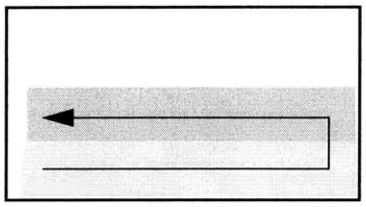
Second, the speed with which a surface can be machined is also of considerable importance. To improve the speed, feed rate adjustment based on cutting physics has been studied in the past. The strategy is basically to allow high feed rates when the cutting force is light and to reduce them when the cut is aggressive. This approach is more applicable to high-force situations like a rough machining. In finishing, especially in high-speed machining, it is more likely that the motion actuation, rather than the cutting force, is the bottleneck. We, therefore, ignore cutting force related limits in this work.

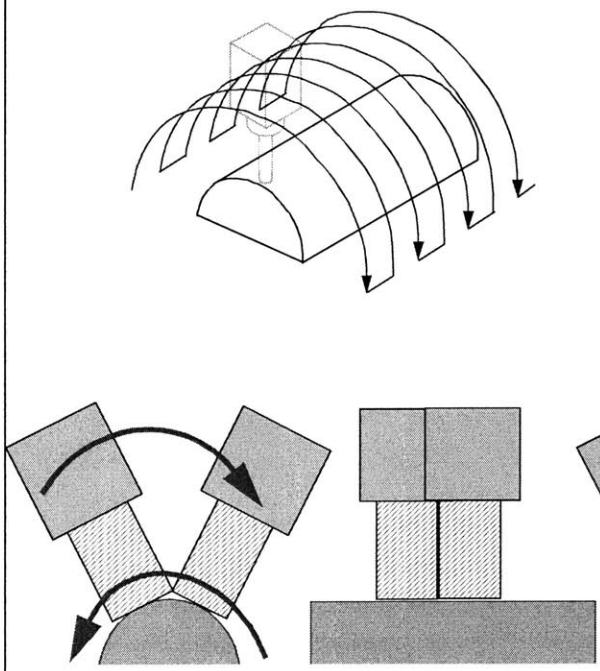
Kinematic Limits

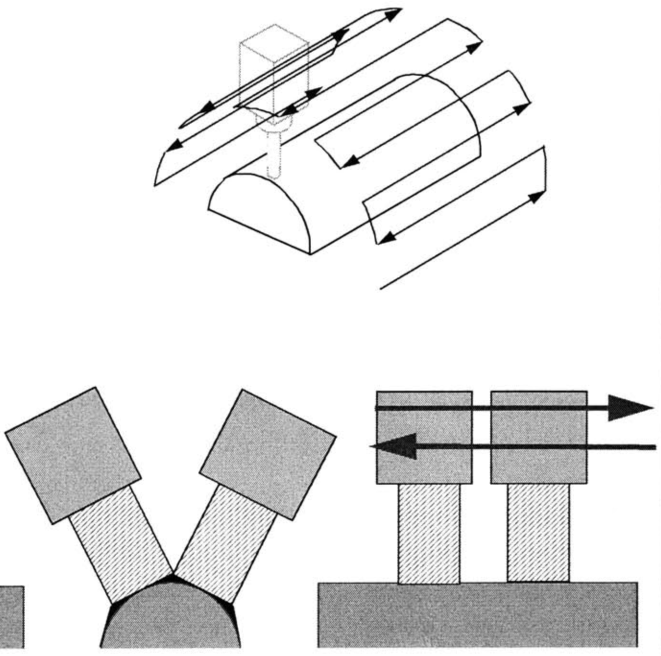
The motors in a machine tool have certain performance limits in terms of velocity and acceleration

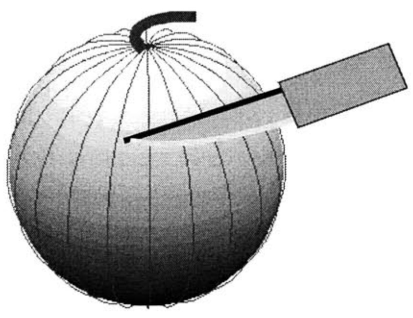
[†] Spotting is to locate a cutting tool on the surface finitely-many times and cover a surface with its imprints. The majority of time is spent on moving the cutter from a location to another location. The actual "in-cut" time is zero; most of the time is spent out of the cut. This is analogous to removing a potato peel by small (infinitesimal) spots as apposed to peeling it. The first approach yields a zero length, but is clearly impractical.


a) A mower path that has unnecessary acceleration


b) This mower turns only at the boundary.


c) The spindle swings.


d) These paths leave large cusps.


e) Highly overlapped at the top.

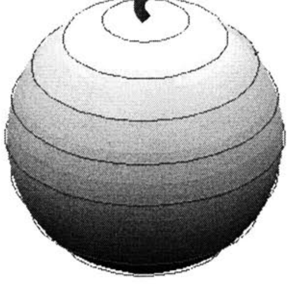
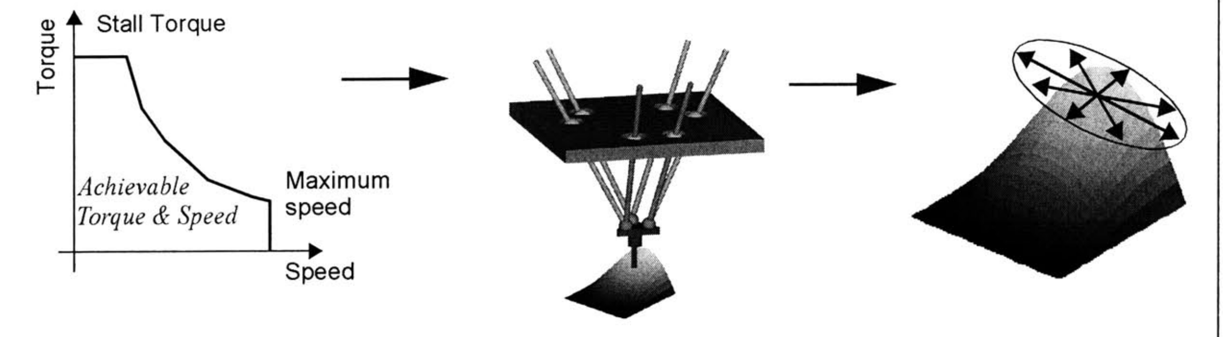

f) The paths without overlaps

Figure 3 Trade-offs in Tool Paths



(a) Torque characteristics of a motor **(b)** Machine Tool Mechanism **(c)** Maximum Velocity on a Surface

Figure 4 Kinematic transformations skew the performance envelope.

capabilities. As shown in Figure 4-(*a*), every motor has a torque-speed curve that determines the maximum torque as a function of velocity. Velocity limits occur when the motors that drive the axes of a machine reach their maximum velocities. Acceleration limits are a consequence of motor torque limits and machine inertia at any posture in the workspace. The motors drive the cutting tool through a kinematic mechanism such as a Cartesian 3-axis structure or a hexapod 6-axis structure as shown in Figure 4. The performance limits of the motors are manifested in cutting performance through these kinematic relationships. This has two implications. First, as one would expect, there are limits on the maximum feed rate and acceleration achievable in any configuration. Second, and perhaps more important, the kinematic transformations skew the velocity and acceleration limits in Cartesian space. This makes the performance envelope of the machine tool extremely direction dependent. In the presence of this anisotropy, some tool paths are significantly better than others. This fact, which has traditionally been ignored in geometry-based path generation techniques, can be taken advantage of to produce time optimal tool paths. We will show that the gains can be significant. The situation is especially pronounced in 5-axis machines, in which there is a coupling between translation and rotation. In fact, the problem is especially acute in machines which have a singularity in their workspace. C-axis machines exhibit this problem.

TRADE-OFFS

The strategy is therefore to avoid slower tool paths and to move in the most effective direction. The slow tool paths are those that figuratively "take the winding road" and hence force the NC machine to move much more slowly. Two simple examples of the trade-offs in tool paths are presented in Figure 3. Cases (*a*) and (*b*) show the importance of acceleration limits. When mowing a lawn for example, the mower has limited traction, which limits the acceleration sustainable. Case (*a*) unnecessarily taxes this limit while case (*b*) is efficient. We perform this type of optimization intuitively and continuously. Case (*c*) and (*d*) show a slightly more subtle trade-off. Along Path (*d*), the tool conforms less to the surface and therefore leaves a worse surface finish. As mentioned earlier, to improve the surface finish, finer path-intervals are needed and, as a result, the length of the path increases. However, for most Cartesian B-axis machine tools, motion along Path (*c*) will tend to be sluggish due to kinematic limits. In addition, in a hexapod shown in Figure 4-(*b*), Path (*c*) generates higher inertial force because the heavy spindle head swings, which is an unnecessary burden. Therefore, the analysis of which path is preferable is not trivial; the trade-off must be examined and framed mathematically.

1.3 Previous Work by other Researchers and Some related Optimal Path Problems

There is a massive body of previous work in the area of surface machining. The most basic role of a CAD/CAM system is to produce geometrically feasible tool paths, where bulk of the emphasis lies. There is also some work in geometric performance improvement. Separately, there is also considerable amount of work on cutting forces and their considerations in path generation. Comprehensive lists of references are available in [15, 16, 20 and 21]. We also explore the characteristics of the machining problem through comparison with path problems in other fields such as robotics. It is shown that none of the previous frameworks are suited for the machining problem.

Basic Feasibility

Much previous work has dealt with basic issues of geometric feasibility such as: techniques for path generation including iso-parametric, iso-offset, iso-planar, iso-cusp-height, iso-phote, iso-engagement and projection schemes [20, 21, 49, 23, 18, 22, 24, 25 and 26]; issues of system integration [39, 40 and 41]; gouge-free tool paths and gouge correction [43, 48 and 49]; issues of accessibility and global interference [45, 47, 46 and 50]; tool shape, tool orientation and surface finish prediction [52 and 53].

Geometric Performance Improvement

Once feasibility is ensured, it is possible to improve the efficiency of machining, and there has been work in this area as well. Elber and Cohen, for example, try to reduce unnecessary tool path overlaps following iso-parametric curves, where only tangential distance was regulated below a critical overlap value instead of cusp height [27]. In iso-cusp-height schemes, one of the families of the curves, instead of iso-parametric curves, is selected to avoid overlaps [18, 22 and 28]. With the assumption of constant feed rate, such tool paths are "near optimal" although strict optimality is ensured only for flat surfaces. In [29], the authors specify varying cusp-height-limits based on heuristics. The opposite approach is to move in the most convex direction, which is sometimes called (optimal) principal-axis-machining (PAM) [37; 38 and 36]. Even though the widest area is swept from the viewpoint of an individual path, the neighboring paths overlap and some regions are redundantly machined. However, it should be mentioned that the focus of the authors' attention is not the path length or the finishing time but the subsequent grinding and polishing period which is performed after surface finishing. These types of global issues are considered by Choi *et al.* and Mullins *et al.* [19 and 32]. They adjust tool orientations to reduce cusp heights for a given set of CC-points and, consequently, cut down the cost of manual polishing. Spatial global optimization has also been studied. Elber divides a surface into convex, concave and parabolic regions and suggests the use of a flat-end cutter for convex regions in order to sweep much larger area during the same time [30]. This idea can be extended to gearing up a sequence of cutters of gradually diminishing size [31]. For concave regions, a new, and interesting, scheme called multi-point machining (MPM), developed by Warketin *et al.*, shows promise [33, 34 and 35; 38]. They place a toroidal or flat cutting tool on a surface such that multiple contacts are allowed. Global intersection problems are solved to implement this idea. With this method, much larger area is swept because the effect is similar to using multiple ball-end mills. For concave spherical surfaces, this method works extremely well because infinitely many contact points can be found. The effect is similar to using a formed-tool.

Cutting Physics

The idea of *adjusting the feed rate* to optimize tool paths for considerations related to cutting physics has been studied by some researchers. A typical application is the reduction of tool deflection or vibration by reducing the feed rate at points on a given path where the cut is too aggressive [55, 56, 57 and 58]. Parameters other than feed rates are also controlled to reduce the cutting force. In the iso-engagement scheme, cutting force variation is reduced, which eliminates the most aggressive point itself [25], in other words, the cutting force is almost constant along a tool path. The implication of the tool orientation on cutting forces is discussed by Kruth and Klewais [38]. The most productive roughing path pattern is identified for a given designed part based on fuzzy pattern analysis in [77]. Unlike other researchers, they regard the cutting time, instead of the path length as the criterion of the optimality and consider the global path pattern [78]. The idea of choosing preferable **directions** is discussed by Lim *et. al.* [59]. The authors suggest following the directions that cause the minimum cutting force and tool deflection in order to avoid machining error and chatter. Their work is very pertinent in high-force situations, where machine deflection is an important issue. We seek tool paths for surface finishing, where motion actuation is usually the bottleneck. The idea, however, is similar.

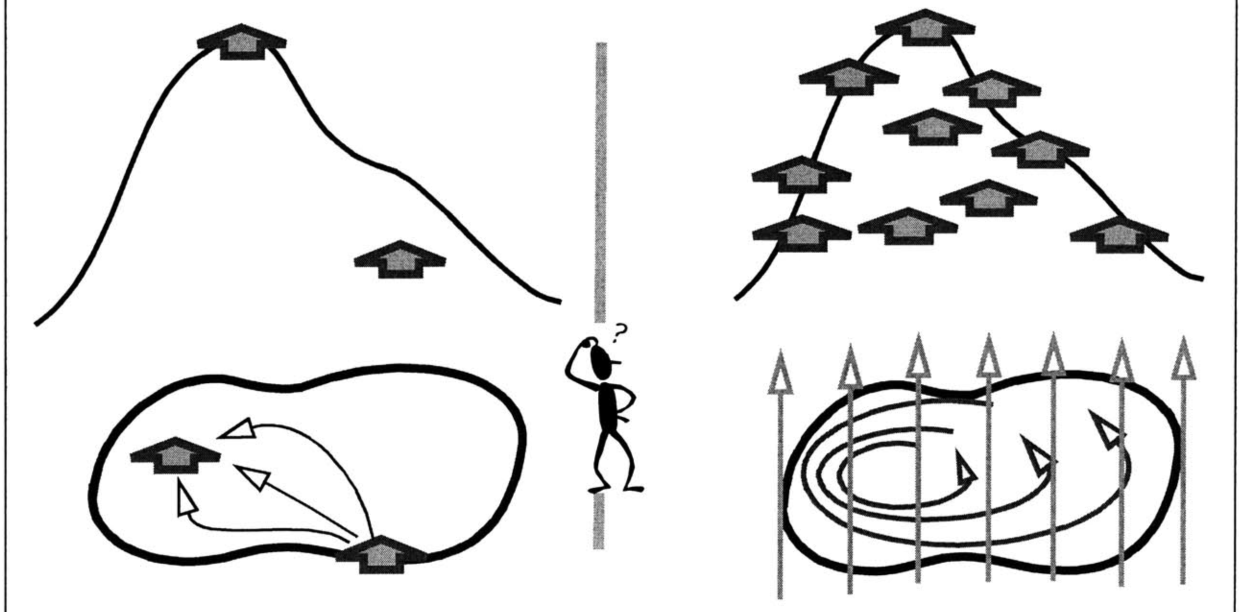
Staircase Face Milling

The staircase face milling for flat convex polygonal surfaces is one of the most fundamental and simple milling processes [73 and 74]. In this case, paths are mutually parallel except at the edges of the polygon and the length is a function of only the direction of the paths. In [73], the authors developed an algorithm to find the shortest staircase path. The length is procedurally evaluated through the construction of actual tool paths for a given cutting direction, and then optimized. Even though one-dimensional optimization is a relatively easy task, we reformulate the problem from the viewpoint presented in this dissertation as a thought experiment. In our framework, the movements inside the polygon are called *effective movements* and the movements along the edges are called *non-effective movements*. Instead of procedurally evaluating the total length, we take an approximation assuming the "regularity" of paths. That is, we assume that the space between the tool paths was set small enough to neglect boundary effects. Then, the length of effective movements is irrelevant to the cutting direction because the length can be approximated by $(Area)/(Tool\ Space)$. It is also realized that the length of the non-effective movements is approximated by half length of the perimeter, which is also a constant, unless the cutting direction coincides with the direction of any edges. If so, the length of the non-effective movements is reduced by the half of the length of the corresponding edge. Therefore, the path length is only piecewise continuous (and, in fact, *lower semi-continuous*) staying in constant value except at the discontinuity. The consequence is that the minimum occurs for the directions of the longest edge; this was presented as an observation after intensive numerical analysis in [73]. Of course, for general surfaces, this kind of simple analysis fails because of the complications mentioned in the previous sections. For sculptured surfaces, we are largely involved in the situation in which the effective movements are dominant whereas in the planar problem it was the non-effective movements that dominated. Besides, we need to add an additional penalty term to the non-effective movements considering the inertia effect; at the end of each effective movement, the tool needs to be decelerated, moved to the next path and accelerated again [75 and 76]. This additional penalty term is not proportional to the perimeter but rather to the number of tool paths. In [75], it is proved that the number of tool paths are minimized by the tool paths that are parallel to one of the edges of the polygon.

Optimal Control

The problem of time-optimal path planning under actuator limits has been studied intensively by the robotics community [83, 84, 85, 86, 87, 88, 89 and 91]. Difficulties in finding time-optimal paths are often carefully described by authors as the words, from the articles of the participants, as "the curse of dimensionality," "numerically-intensive," and "numerically-sensitive." Therefore, less ambitious goals are set if real time control is required [83 and 86]. Nevertheless, *optimal control* is a well-established theory, which is crystallized in *Pontriagin's maximum principle* that was proved by Boltianskii [80, 81 and 82]. This theory extends the applicability of the *classical (one-dimensional) calculus of variations* for systems that are (1) constrained by the *state equations* and (2) actuated by piecewise continuous control signals confined in closed sets. Classical examples include finding the shortest path on a smooth surface. Given two points on a surface, it is proved that a *geodesic* curve connects the two points with minimal length [110]. This geometric problem is relatively easy to grasp, and helps us to intuitively understand the key features of one-dimensional variational problems. Both the calculus of variations and Pontriagin's maximum principle lead to what are known as 2-point boundary value problems (BVPs) of ODEs (Ordinary Differential Equations), which are necessary conditions for optimality. On the other hand, reasoning based on *Bellman's dynamic programming* results in a PDE (Partial Differential Equation), called the *Hamilton-Jacobi-Bellman equation* [92, 97, 95 and 94]. Stronger smoothness is required for the PDE and its applicability is, consequently, somewhat narrower [95]. The main strategy of all the previously mentioned approaches is finding the solutions of the equations which are necessary, or even "weaker", conditions for optimality. If we directly discretize the state equation, it is possible to apply *dynamic programming* techniques for difference equations [93 and 96], which is known to be intractable in high dimensional situations such as robot manipulator control. However, dynamic programming is general reasoning and we can apply it in various ways if we alleviate the "curse of dimensionality" [90 and 98]. The *direct (transcription) method* is based on the fact that the unknown functions that we seek are represented as points in a certain *function space* [98, 99, 100 and 130]. Thus, we attempt to treat the functions as if they were points in a "usual" space. Approximating some unknown functions—usually, the control signals—and the cost functional in terms of certain finitely-many pre-selected functions, we reduce the dimensions of the problem from the infinity. Now, various standard or static optimization techniques such as the *steepest descent method* can be applied in the reduced dimensional spaces. Instead of searching discrete control signals as in the usual direct method, the *space search method* [85 and 87] makes use of a well-established algorithm, called the *Bobrow-Dubowsky-Shin algorithm* (BDSA) [88 and 89]. The BDSA finds the minimum time tracking control for a pre-defined path connecting the two end points. We search the optimal path by discretizing and varying the predefined path while the BDSA evaluates the optimal tracking time of the given path at each stage. By this approach, the state-dependent constraints on the control signals are conveniently dealt with, which is not straightforward with the maximum principle. In general, the compromise between the search dimension, cost evaluation, stability and convergence should be considered when a particular method is chosen. A brief history is found in [101 and 102].

Unfortunately, the well-studied result of the optimal control theory cannot be directly carried over to the realm of machining. This is due to one fundamental difference between path planning for machining and for robots: while robotic paths seek at most to follow a given path, or go from point to point, machining paths almost always seek to fill an area. The two problems differ qualitatively. For example, a villager



(a) A villager's problem: Optimal Control/Calculus of Variations/Geodesics/2-point BVP/Dynamic programming **(b)** A postman's problem: Traveling Salesman Problem. Machining problem?

Figure 5 Qualitatively different 2 path problems

who intends to visit a friend's house may have to solve the geodesic problem while a postman, whose everyday task is to visit all the houses in a village, has to solve the traveling salesman problem (TSP), which is a combinatorial optimization problem. This is shown in Figure 5. Planning robotic paths is similar to the villager's problem while the TSP captures a crucial aspect of the machining problem in that machining paths need to cover a whole surface properly. (This does not imply that the machining problem is equivalent to the TSP in every aspect.)

Graph-Theoretic Approach

The problem could be posed graph-theoretically. An intuitive strategy is to find a distribution of points on a surface, whose mean distance is as small as the square root of the surface area covered by a tool, and to formulate the machining problem as a TSP, by requiring the tool to visit all the points in the distribution. Unfortunately, this does not solve the problem in a practical sense. First, the formulated problem is conjectured to be highly intractable; the TSP is known to be an *NP-complete problem* [124]. Second, the distribution biases the problem towards some directions in which the tool should move. In other words, we cannot achieve arbitrary freedom/precision for the directions. If we add more points in the distribution to allow more directions, the requirement that the tool should visit all the points will cause redundant machining. In fact, the redundancy itself is determined by the distribution. Even though the TSP captures an aspect of machining, the distribution commits too much before the problem is solved. That is, for the machining problem, there is no obvious choice of the distribution of points like the locations of the houses which are known to the postman. It seems that the problem was discretized at too early a stage by being formulated as a TSP even though, after all, we need to introduce certain discretization to solve the problem using digital computers. The discretization is deferred in this thesis until a proper continuum model is established. Under the continuum model, we are, after all, required to solve problems similar to the TSP especially when we construct the non-effective movements, but the total cutting time will not be sensitive to the construction as long as "reasonable" movements are made. That is, it is of no practical importance to find the exact solution at great computational expense because the performance of cheap approximate solutions is very close to the one of the exact solutions.

It should be mentioned that Chou [66, *c.f.* 67, 68, 69, 70, 71 and 72] considered similar kinds of complexity of machining problems. He constructed an "adjacency graph" with the tessellation of a surface. He formulated the machining problem as a series of known problems in graph theory: the Hamiltonian path problem, the TSP, the Euler walk problem and the Chinese Postman Problem (CPP).[†] In the context of the formulation, the tessellation was dominated by "large" polygons. He machined each polygon with zigzag patterned paths and his concern was how to find movements that connect the polygons. Since the requirement that either an edge or a vertex on the graph should be visited exactly once is too restrictive, the Hamiltonian path problem and the Euler walk problem are not suitable for general machining. Thus, the TSP and the CPP are considered. While the TSP is an NP-complete problem, there is a known *4th* order polynomial time algorithm for the CPP [126 and 127]. He converted the TSP into the CPP by inverting the role of the vertices and the edges. Even though the formulation of the machining problem as a CPP is reasonable in his context, the CPP loses some characteristics of the machining problem largely because the tool is required to visit all the edges, and such paths will not form streamlines but a grid. Thus, the TSP is closer to the reality of machining than the CPP; the TSP, however, also loses some of the reality of machining due to the directional bias.

Spray Coating

A series of variational optimization problems related to spray-coating was formulated by Antonio and Ramabhadran [62, 63, 64 and 65]. In their framework, either minimum painting time or uniform film thickness can be pursued subject to various constraints. The authors sought curves in one piece. The spray coating problem is different from the machining problem in that paint propagates on the surface and, as a result, it accumulates at each point. The distribution of the accumulation rate is typically set to be "bell-shaped" centered at the end-effector. An extreme case of this model is when the end-effector is held still. By propagation, the whole surface will be covered eventually even though paint is not uniformly distributed. On the other hand, only constant area of the surface is covered by a standing cutter. In this sense, there is a fundamental difference between spray painting and machining.

Uniform Surface Scanning

The question of how to uniformly cover a surface with scanning curves is asked by Tam [79]. This question will also be answered in this thesis. However, Tam's measure of uniformity is dependent on his *a priori* choice of surface partition based on parametric lines and lacks intrinsic meaning. We will resolve this problem by employing a "conservation law" derived from *differential manifolds*.

[†] The CPP is to find the paths that visit all the edges with a minimum cost allowing repeated visits; the other three problems are well known [124 and 127].

Concluding Remarks

In general, there is little literature in the field of computational geometry on the consideration of cutting time as a condition for optimality. Work in process physics is usually limited in focus to local feed rate adjustment, and lacks a global view. Tool path generation is typically about suggesting a particular scheme for feasibility. It is the users of a CAM system who determine the particular scheme to be used. Well-developed time-optimal control does not apply in machining. Often, the objectives of algorithms for surfaces are not intrinsic and depend on the parameterization of the surfaces, which is selected only for its convenience. It is here that this work contributes: we seek to generate, automatically, "optimal" tool paths based on a comprehensive mathematical model.

1.4 Outline

In Chapter 2, we formulate the machining problem as a mathematical optimization problem. In Chapter 3, we discuss the general consequences under this framework. In Chapter 4, we simplify the problem and explain the way to find an approximate solution. In Chapter 5, we present some examples with the necessary analysis. Finally, we conclude with the discussion on the future improvements of current work.

1.5 Conventions

Many details are presented in the appendices in order to carry the essential ideas without confusion. If a section (or a REMARK) is introduced just for completeness, its title is accompanied by a star sign. Footnotes can be skipped safely. They are either for some references or for clarifying some minor subtleties. Instead of presenting the most general case, we restrict the scope of the problem for simplicity. Simple notations are chosen unless there is ambiguity. This is achieved mainly by avoiding the strict distinction between tensors, matrices, transformations, vectors, column vectors (matrices) and points in multi-dimensional spaces. Matrices and the column vectors, which are possibly tensors and vectors, are all typed in bold letters. All vectors are treated as column vectors and, as a result, row vectors are denoted with the transpose signature, which is the superscript T. The distinction between "usual" functions and functionals is made. Functionals are indicated with brackets. For example, $F[x, y, z]$ means that F is a functional that depends on the functions, x, y and z, which possibly depends on the derivatives of x, y and z, while $F(x, y, z)$ is a function of the independent variables, x, y and z. A function and its value are not distinguished in the strict sense. Thus, we allow somewhat risky expressions such as $f = f(x, y)$ and $g = g(x, y)$, to which many of us are accustomed. We state "a point f" or "a function f" to avoid the confusion. Also, if two functions of different variables always have the same value, we sometimes do not distinguish the two functions as long as it is not critical, *e.g.* we sometimes write $f(x, y) = f(u, v) = f$. The first f is a function, the second f is another function that should have been denoted by a different symbol, and the last f is a value or the dependent variable of the functions. We risk these hazards because we only have 26 alphabetic characters in Roman alphabet [*c.f.* 136].

2 Formulation of the Machining Problem

In this chapter, we formulate a mathematical optimization problem that captures the machining process. The final set of machining paths is a finite selection from the infinitely-many **streamlines** of a **vector field** derived from the **kinematics** of the particular machine tool under consideration. The derived vector field plays a role as a skeletal representation of machining paths; tool paths are chosen from its streamlines through the consideration on the **geometry** of the surface and the tool. This viewpoint constitutes the basis of the framework developed in this chapter: a "good" vector field, satisfying a certain criterion, is found and tool paths are placed along its streamlines.

In Section 2.1, we briefly review the local geometry of a surface with respect to a tool, introduce the general goal, and provide an overview of surface finishing. In particular, we introduce two geometric notions, *side step* and *its limit*. From Section 2.2, we start to formulate the machining problem in a formal way. That is, we mathematically state the objective and the constraints of the machining process. We begin by constructing a comprehensive kinematic model of surface-cutter-machine interaction. We also develop a differential equality constraint based on the "regularity" or "continuity" of tool paths. This equality constraint must be respected to ensure proper coverage of area of a surface. We then develop an expression for the cutting time, which we consider the cost function (or functional) in our optimization problem. Finally, we describe the constraints of the process, including actuator limits and geometric cusp height requirements.

2.1 The Geometry of Tool Paths: a Brief Overview

The goal of surface finishing is to sculpt a part, or a component, to a required degree of surface finish as quickly as possible. We briefly describe how this is typically achieved. Terms are loosely defined in this section and rigorous definitions are provided in later sections.

Cusp height and Side step

If we are to prevent "over-cutting" or "under-cutting," a cutting tool must be placed on a surface in such a way that the tool surface and the designed surface share a tangent plane. Furthermore, a ball-end mill that has a smaller radius than the radii of (principal normal) curvatures of a designed surface does not cause gouging as it prescribes its osculating path. For other tool geometries, more sophisticated but similar tool placement strategies are necessary. In path planning literature, such a point of intersection of a surface with a tool at an instant is referred to as the *cutter contact point* (*CC-point*). The CC-point moves on the surface during a finishing process. A trajectory of the CC-point is called a *surface tool path* or simply a *tool path*, which is a *surface curve* (or a curve on the surface) in mathematical terms [111].

Because a cutting tool cannot conform to every surface, *scallops* or *cusps* of finished parts are unavoidable in general surface machining as shown in Figure 6-(*a*). Their heights can, however, be reduced if we place more tool paths at finer intervals. The *cusp height* of a machined surface, which is denoted by h in the same figure, generally varies from place to place. It measures how close a machined surface and its designed surface are; it is necessary in machining to make sure that the cusp height is below a certain limit value over an entire machined surface. We call such a limit value h_o a *cusp-height-limit* and we require that $h \le h_o$.

To control cusps, we study the local geometry of a surface with respect to a tool—for example, a ball-end mill. The conformity of the tool to a surface is best observed along a *normal section* at a surface point as shown in Figure 6-(*b*); this particular normal section is perpendicular to the "averaged" tangent direction of the neighboring two tool paths at the two CC-points shown in the figure. Loosely defined, the gap between the two CC-points along the normal section, is called the *side step* and denoted by w. It is observed that the cusp height h increases as the side step w becomes wider. That is, given the "averaged"†

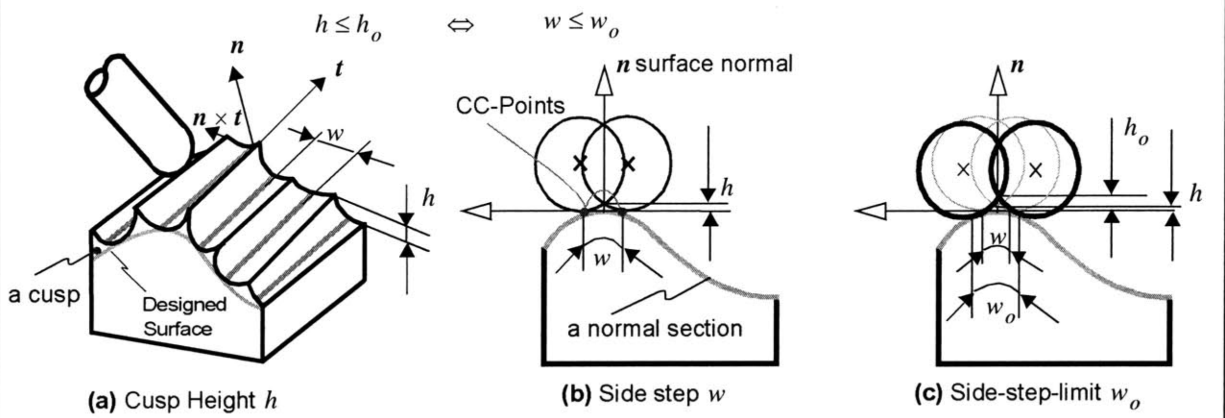


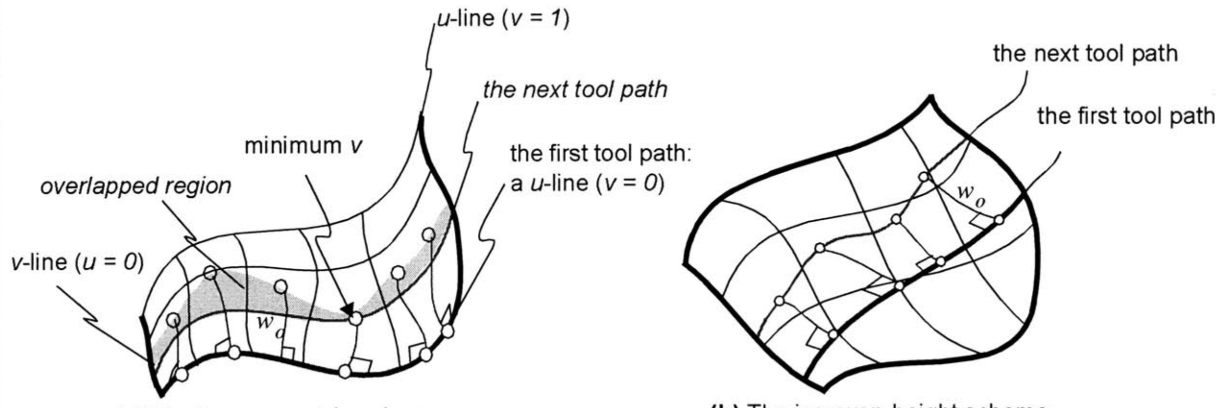
Figure 6 Cusps and side-step-limit corresponding a cusp-height-limit

tangent direction at a point on the surface, there is established a one-to-one correspondence between the side step w and the cusp height h (of course, for "small" cusps). Consequently, a side step corresponds to the cusp-height-limit h_o as shown in Figure 6-(*c*), which is referred to as the *side-step-limit*, and denoted by w_o. The cusp height constraint, $h \leq h_o$, is now "equivalently" stated as the following inequality: $w \leq w_o$. This equivalent form turns out to be more convenient than its direct form. Note that the side-step-limit depends on in which direction the cutting tool proceeds—the most convex direction is the most preferable one from the viewpoint of an individual tool path in that it allows widest cut, in other words, the largest side-step-limit.

Most CAD/CAM systems account for cusp-height-limits in their algorithms. We introduce two well-known schemes for tool path generation and describe roughly how the cusp-height-limit constraint is respected in them.

The iso-parametric scheme

The iso-parametric scheme selects tool paths from the infinitely many *iso-parametric lines* on a surface parameterized by a map $\boldsymbol{r}(u, v)$. For example, we choose a *u-line* as the first tool path; we choose "enough" points on this first tool path for its discrete representation; at each point, into the direction perpendicular to the first tool path, we mark a new point, whose distance from the original point is the side-step-limit w_o; we evaluate the v-value of each marked point; from the marked points, we select the point which furnishes the minimum of the v-values; the u-line whose v-value corresponds to this



(a) The iso-parametric scheme. **(b)** The iso-cusp-height scheme.

Figure 7 The two well-known schemes of tool path generation

[†] *Normal sections* at a point on a surface are the intersections of the surface with the planes that contain the *surface normal*. A *normal plane* is defined as the one that contains both the *principal normal* and the *binormal* of a general curve. *Normal sections*, *normal planes*, *principal normal lines*, *binormal lines*, and *surface normal lines* are frequently referred to in this work. The definitions can be found in most text books on *differential geometry* [*e.g.* 112]. There are infinitely-many normal sections at a point on a surface because there are infinitely-many planes that contain the surface normal line. A plane can be specified by its own normal vector. This plane normal vector is aligned to the "averaged," tangent vector of the two tool paths. We used somewhat vague term, "averaged" because there are no tool paths on the surface in the gap between the two tool paths. This notion will be clarified in the next section, where we formulate the problem with more rigor.

minimum is the next tool path; the steps are repeated until the surface is filled with tool paths. These steps are shown in Figure 7-(a). In this scheme, tool paths generally suffer from redundant machining. Apart from the simplicity of the procedure, there are no firm grounds for claiming any "efficiency" in iso-parametric tool paths.

THE ISO-CUSP-HEIGHT SCHEME

The iso-cusp-height scheme maintains the height of each cusp **equal to** the cusp-height-limit, changing the limit, $h \leq h_o$ or $w \leq w_o$, into an equality requirement, $h = h_o$ or $w = w_o$ [18, 22 and 28]: a surface curve is chosen as the first tool path; new points are found, being shifted from the first tool path by the distance of the side-step-limit w_o as before; the interpolation of such a new set of points defines the next tool path; the steps are repeated. This procedure is shown in Figure 7-(b). This scheme is an extreme case that minimizes the amount of overlap of machined area. In this scheme, however, tool paths can be aligned in inefficient ways. This happens, for example, in concave directions.

2.2 Field Description of Tool Paths

From this section onwards, we start to formulate the time-optimal machining problem mathematically. The problem we seek to solve is very general, and it is important to clearly state the objective and the constraints with appropriately defined variables, parameters and constants. In terms of such descriptors, we study the kinematics of a machine tool with respect to a surface. The essential idea that we propose in this section is to postulate certain functions on a designed surface that contain enough information to construct tool paths. We refer to this method as the *field description method*. By this way of description, a convenient mathematical structure is imposed on the problem, without which stating the problem itself would have been difficult. Based on the framework developed in this section, we complete a variational formulation of the machining problem through the rest of this chapter. To explain the essential ideas quickly, we consider simple cases only.

1. Pre-assigned Orientation Vector Field

The inputs to this optimization problem are as follows: (*1*) a definition of a designed surface in parametric form with the cusp-height-limit being specified, (*2*) a definition of a machine and cutting tool and (*3*) an orientation (or access) vector field. Parametric surfaces are well known, and examples include Bezier surfaces, B-spline surfaces and NURBS surfaces as mentioned in the introduction. The definition of a machine tool includes the kinematics, and although we do not necessarily consider them in practice, the dynamics. In 5-axis machining, cutters (or cutting tools) such as *flat endmills*, *filleted endmills* and *ball-end mills* are used widely. In the current formulation, only **ball-end mills** are considered for simplicity; other tools are easily incorporated. The third point needs more elaboration. The orientation of a cutting tool with respect to a surface is critical for two reasons: for controlling cutting conditions, and for eliminating both local interference (or gouging) and global interference (or collision) [38, 43, 44, 45, 46, 47, 48, 49, 50, 51 and 52]. We assume that the **orientation is pre-defined** at every point on a designed surface in such a way that interference is avoided. Thus, we treat the orientation as a known/specified vector field on a surface. In theory, though, it is possible to pose an even more general problem and ask what is the orientation field that minimizes the cutting time; of course, the expression for interference avoidance is quite complicated.[†]

2. Designed Surface

A designed surface $\mathcal{S}$ is assumed to be given as a *regular* parametric form, $\boldsymbol{r}(u, v)$. In other words, the map $\boldsymbol{r}$ is a *smooth*[‡] *one-to-one map s.t.* $\boldsymbol{r}:(\mathcal{P} \subset \mathcal{R}^2) \to (\mathcal{W} \subset \mathcal{R}^3)$, $\mathcal{S} = \boldsymbol{r}(\mathcal{P}) \subset \mathcal{W}$ and $\boldsymbol{r}_u \times \boldsymbol{r}_v \neq 0$ for a simply-connected compact set $\mathcal{P}$ and an open set $\mathcal{W}$. We call the domain $\mathcal{P}$ the *parameter space* and the co-domain $\mathcal{W}$ the *workpiece space*. The map $\boldsymbol{r}$ is called a *parameterization* of the surface $\mathcal{S}$ and the independent variables, u and v, are called the *parameters* of the surface. We sometimes say "a surface $\boldsymbol{r}(u, v)$" and "a curve $\boldsymbol{r}(u(t), v(t))$." We denote partial derivatives with respect to the parameters by subscription, *e.g.* $\boldsymbol{r}_u = \partial \boldsymbol{r}/\partial u$. A point in the parameter space, or a pair of the parameters, is denoted by a

[†] Note that our consideration of a known orientation field ignores feed-direction-dependent orientation adjustment. For example, *forward tilting* is generally recommended [16].

[‡] A map is smooth if it is differentiable infinitely-many times or of differential class C^∞. The required degree of the smoothness will be clear from the context where the map $\boldsymbol{r}$ is involved; we avoid mentioning the required degree of differentiability explicitly.

bold letter $\boldsymbol{u}$, *i.e.*, $\boldsymbol{u} = (u, v) \in \mathcal{P}$. We write $\boldsymbol{u} \leftrightarrow \boldsymbol{r}(\boldsymbol{u}) \in \mathcal{S}$ for the correspondence between the parameter space $\mathcal{P}$ and the surface $\mathcal{S}$. Because of this one-to-one correspondence, we do not distinguish the two sets strictly. That is, a map on the surface $\mathcal{S}$ is considered a map in the parameter space $\mathcal{P}$. For example, both $f(\boldsymbol{u})$ and $f(\boldsymbol{r})$ are written with the same symbol f; strictly, the first $f(\boldsymbol{u})$ is the composition map $(f \circ \boldsymbol{r})(\boldsymbol{u})$. The **unit outside** *normal vector* of the surface at (u, v) (or $\boldsymbol{u} \in \mathcal{P}$) is denoted by $\boldsymbol{n}(u, v)$, $\boldsymbol{n}(\boldsymbol{u})$ or even $\boldsymbol{n}(\boldsymbol{r})$. Without loss of generality, we assume that the surface is parameterized such that $det[\boldsymbol{r}_u\ \boldsymbol{r}_v\ \boldsymbol{n}] > 0$ or

$$\boldsymbol{n} = (+)\frac{(\boldsymbol{r}_u \times \boldsymbol{r}_v)}{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\|}.$$

While the designed surface remains stationary in $\mathcal{W}$ during an entire machining process, the workpiece itself can be in movement with respect to an *inertial reference frame*. A *change of frame*[†] between the workpiece space $\mathcal{W}$ and the inertial reference frame needs to be established for mechanical analysis—in other words, to establish the *equation of motion* of the mechanical system. We choose an *orthonormal basis* $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$ which is stationary in $\mathcal{W}$ as shown in Figure 9. This is a basis moving with the surface. When we represent a (position) vector as a matrix, the vector will be treated as a column vector and it will be projected into the basis $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$, *e.g.* $\boldsymbol{r} = [\boldsymbol{r} \bullet \boldsymbol{i}\ \ \boldsymbol{r} \bullet \boldsymbol{j}\ \ \boldsymbol{r} \bullet \boldsymbol{k}]^T$.

3. Tangent Spaces and Normal Curvatures of a Surface: Necessary Linear Algebra and Geometry*

This subsection provides the definitions for the *tangent space* and the *normal curvatures* at a surface point. A tangent space is essentially a *linear space*. We introduce its coordinate systems, which we frequently refer to. The definitions are well-known but we clarify the sign conventions and the notations that we use in this thesis. They are also general; that is, there is no physics involved in this subsection.

The Tangent Space

At each point $\boldsymbol{r}$ on the designed surface $\mathcal{S}$, a *linear space* $T_r\mathcal{S}$ is spanned by the basis $\{\boldsymbol{r}_u, \boldsymbol{r}_v\}$, which and referred to as the *tangent space* at the point $\boldsymbol{r}$ on $\mathcal{S}$. This is a generalization of the classic notion, *tangent plane*: $\{\boldsymbol{r}(u, v) + a \cdot \boldsymbol{r}_u(u, v) + b \cdot \boldsymbol{r}_v(u, v) \mid (a, b) \in \mathcal{R}^2\}$ for a point $(u, v) \in \mathcal{P}$ or $\boldsymbol{r}(u, v) \in \mathcal{S}$. Similarly, the *tangent space* $T_r\mathcal{W}$ at a point $\boldsymbol{r}$ on the workpiece space $\mathcal{W}$ is the linear space spanned by (a copy of) the basis $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$. We treat a tangent space $T_r\mathcal{S}$ of the surface as a *subspace* of the tangent space $T_r\mathcal{W}$ of the workpiece space, namely $T_r\mathcal{S} \subset T_r\mathcal{W}$ for a point $\boldsymbol{r} \in \mathcal{S} \subset \mathcal{W}$.[‡]

A vector $\boldsymbol{v} \in T_r\mathcal{S}$ in the tangent space is identified by two real numbers $\dot{u}$ and $\dot{v}$ *s.t.*

$$\boldsymbol{v} = \dot{u} \cdot \boldsymbol{r}_u + \dot{v} \cdot \boldsymbol{r}_v \qquad \text{or in matrix form,} \qquad \boldsymbol{v} = \boldsymbol{A}\dot{\boldsymbol{u}} \tag{1}$$

[†] We refer to [131] for the definition of the *change of frame* in mechanics.

[‡] In modern language, the workpiece space $\mathcal{W}$ is a *manifold* and the designed surface $\mathcal{S}$ is one of its *submanifolds* (with its boundary) parametrized by the *immersion* $\boldsymbol{r}$. To each point p on a manifold $\mathcal{M}$, a linear space $T_p\mathcal{M}$ is assigned, which is called the *tangent space* at $p \in \mathcal{M}$. The details of this topic can be found in most text books on *differential geometry* and *differential forms* such as [104 and 107].

where $\dot{u} \equiv [\dot{u} \;\; \dot{v}]^T$ and $A \equiv [r_u \;\; r_v]$; the matrix A can be thought of as the *Jacobian matrix* of the map r. This one-to-one correspondence is symbolized in the following way: $\dot{u} \leftrightarrow v \in T_r S$. We point out that the dot symbols that appear in this definition do not have any relation to time derivatives upto this point; $\dot{u}$ and $\dot{v}$ are simply certain real numbers that should be interpreted literally. We, however, used the dot symbol because, in Subsection 7, the vectors in the tangent space are related to the velocities of tool paths.

Normal Curvatures

The *normal curvature* $\kappa_u(\dot{u})$ (or $\kappa_r(v)$) in a direction $\dot{u} \leftrightarrow v \in T_r S$ at a point $u \leftrightarrow r \in S$ is:

$$\kappa_u(\dot{u}) \equiv \kappa_r(v) \equiv \frac{\dot{u}^T D \dot{u}}{\dot{u}^T G \dot{u}} \qquad (2)$$

where

$$G \equiv \begin{bmatrix} r_u^T r_u & r_u^T r_v \\ r_u^T r_v & r_v^T r_v \end{bmatrix} = [g_{ij}] \qquad \text{and} \qquad D \equiv \begin{bmatrix} r_{uu}^T n & r_{uv}^T n \\ r_{uv}^T n & r_{vv}^T n \end{bmatrix} = [d_{ij}] ;$$

both matrices are evaluated at the given point $u \in \mathcal{P}$. In differential geometry, the matrices, G and D, are referred to as the *1st* and *2nd fundamental* (*form coefficient*) *matrix*, respectively. Both are symmetric and the *1st* fundamental matrix G is *positive definite* by the regularity of the surface. In general, the sign convention of normal curvatures is not fixed because of the ambiguity of the sign of a surface normal vector n used for the definition of the *2nd* fundamental matrix D. In our convention, the surface normal curvature in a direction is positive for the "concavity" of solid parts, in other words, if a surface curve tangential to the given direction turns towards the **outside** surface normal vector n. A way to see the normal curvature $\kappa_r(v)$ is to construct a plane at the given point r that is parallel to both the surface normal vector n and the direction v. The intersection of the surface with the plane forms a surface curve, which is a plane curve at the same time by construction. (The intersection is referred to as a *normal section* in differential geometry.) The curvature of this intersection curve at the point r is the normal curvature in the direction v.

It can be proved that the surface normal curvatures, as defined in Equation (2), are bounded and their maximum κ_1 and minimum κ_2 are respectively the maximum and minimum *eigenvalue* of the following generalized *eigensystem*:

$$\kappa G \dot{u} = D \dot{u} . \qquad (3)$$

Let its maximum eigenvector[†] be denoted by $\dot{u}_1$ and the minimum eigenvector by $\dot{u}_2$. Let them be normalized *s.t.*

$$\dot{u}_1^T G \dot{u}_1 = 1 \qquad \text{and} \qquad \dot{u}_2^T G \dot{u}_2 = 1 .$$

[†]Here, we somewhat misused the term, maximum/minimum. Correct expression for "the maximum eigenvector" is *the eigenvector that corresponds to the maximum eigenvalue.*

From the correspondence (Equation 1), a **unit** vector $\boldsymbol{i}_U$ in the tangent space corresponds to the eigenvector $\dot{\boldsymbol{u}}_1$, namely $\boldsymbol{i}_U = \boldsymbol{A}\dot{\boldsymbol{u}}_1 \in T_r S$ where $\boldsymbol{A}$ is the Jacobian matrix of the map $\boldsymbol{r}$ at the point under consideration. This is the most "concave" direction. The most convex direction corresponds to the minimum eigenvector and it can be proved that the most concave direction is perpendicular to the most convex direction, *i.e.*, $\boldsymbol{A}\dot{\boldsymbol{u}}_2 = \boldsymbol{n} \times \boldsymbol{i}_U \in T_r S$.† The two directions, $\boldsymbol{i}_U$ and $\boldsymbol{n} \times \boldsymbol{i}_U$, are referred to as the *principal directions* at the given point and the two eigenvalues, κ_1 and κ_2, are referred to as the *principal curvatures* therein.

COORDINATE SYSTEMS FOR TANGENT SPACES

A tangent space of a surface is essentially a 2 dimensional linear space. At each point $\boldsymbol{r} \in S$, a pair of numbers can be used to specify a vector in its tangent space $T_r S$ in various ways. Each such pair constitutes a particular coordinate system of the tangent space. Here, we define coordinate systems of the tangent space that we use in this thesis and show the transformations between the coordinates.

Each coordinate system is based on a particular basis that it refers to. We introduce three bases of the tangent space $T_r S$ at a point $\boldsymbol{r}$. The basis $\{\boldsymbol{r}_u, \boldsymbol{r}_v\}$ is a skewed one by a parameterization. It is more convenient to perform analysis in the tangent space using **orthonormal** bases. The basis $\{\boldsymbol{r}_u/\|\boldsymbol{r}_u\|, \boldsymbol{n} \times \boldsymbol{r}_u/\|\boldsymbol{r}_u\|\}$ and the basis $\{\boldsymbol{i}_U, \boldsymbol{n} \times \boldsymbol{i}_U\}$ are the most frequently referred to, where $\pm\boldsymbol{i}_U$ is the most concave direction at a point. It is a routine in differential geometry to find the *principal directions*, $\boldsymbol{i}_U$ and $\boldsymbol{n} \times \boldsymbol{i}_U$ except, of course, at *umbilical points*. Using these three bases, a vector $\boldsymbol{v} \in T_r S$ is identified by certain pairs $(\dot{u}, \dot{v})$, $(\underline{\dot{u}}, \underline{\dot{v}})$ and $(\underset{\sim}{\dot{u}}, \underset{\sim}{\dot{v}})$ of real numbers *s.t.*

$$\boldsymbol{v} = \dot{u} \cdot \boldsymbol{r}_u + \dot{v} \cdot \boldsymbol{r}_v = \underline{\dot{u}} \cdot \boldsymbol{r}_u/\|\boldsymbol{r}_u\| + \underline{\dot{v}} \cdot \boldsymbol{n} \times \boldsymbol{r}_u/\|\boldsymbol{r}_u\| = \underset{\sim}{\dot{u}} \cdot \boldsymbol{i}_U + \underset{\sim}{\dot{v}} \cdot \boldsymbol{n} \times \boldsymbol{i}_U .$$

Finding the transformation of $(\dot{u}, \dot{v})$ to $(\underline{\dot{u}}, \underline{\dot{v}})$ and $(\underset{\sim}{\dot{u}}, \underset{\sim}{\dot{v}})$ from the above equation is a matter of linear algebra. The relation is defined by certain linear transformations, $\boldsymbol{P}$ and $\underset{\sim}{\boldsymbol{P}}$:

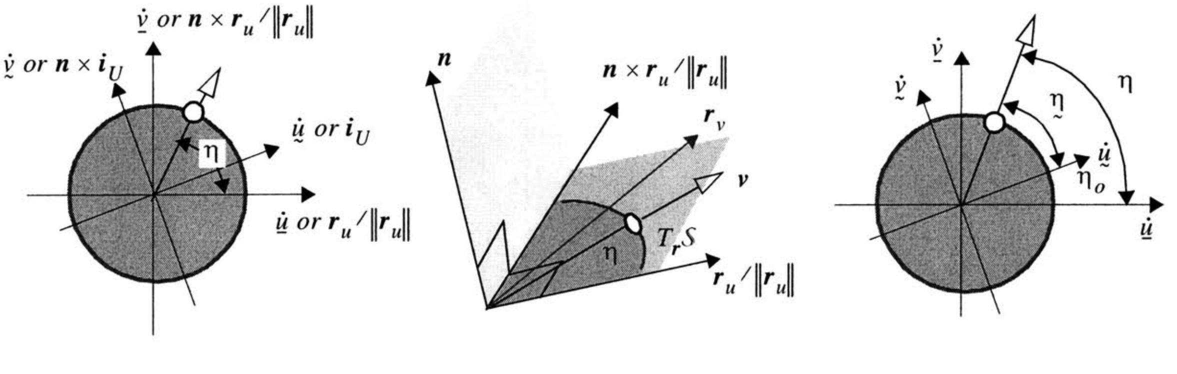
$$\dot{\boldsymbol{u}} = \boldsymbol{P}\underline{\dot{\boldsymbol{u}}} = \underset{\sim}{\boldsymbol{P}}\underset{\sim}{\dot{\boldsymbol{u}}} \tag{4}$$

where $\dot{\boldsymbol{u}} \equiv [\dot{u} \;\; \dot{v}]^T$, $\underline{\dot{\boldsymbol{u}}} \equiv [\underline{\dot{u}} \;\; \underline{\dot{v}}]^T$, $\underset{\sim}{\dot{\boldsymbol{u}}} \equiv [\underset{\sim}{\dot{u}} \;\; \underset{\sim}{\dot{v}}]^T$,

$$\boldsymbol{P} \equiv \frac{1}{\sqrt{(g_{11} \cdot g_{22} - g_{12}^2) \cdot g_{11}}} \begin{bmatrix} \sqrt{(g_{11} \cdot g_{22} - g_{12}^2)} & -g_{12} \\ 0 & g_{11} \end{bmatrix} \text{ and } \underset{\sim}{\boldsymbol{P}} \equiv [\dot{\boldsymbol{u}}_1 \;\; \dot{\boldsymbol{u}}_2] .$$

Note that $\dot{\boldsymbol{u}}_1$ and $\dot{\boldsymbol{u}}_2$ are the **normalized** eigenvectors from Equation (3) and g_{ij} are the *1st fundamental form coefficients* as defined in Equation (2). Following the tradition of discrete dynamics, we call the

†The orthogonality in a symmetric eigensystem is well known. We only consider the case when the two eigenvalues are different, *i.e.* $\kappa_1 \neq \kappa_2$. By definition, (*a*) $\kappa_1 \boldsymbol{G}\dot{\boldsymbol{u}}_1 = \boldsymbol{D}\dot{\boldsymbol{u}}_1$ and (*b*) $\kappa_2 \boldsymbol{G}\dot{\boldsymbol{u}}_2 = \boldsymbol{D}\dot{\boldsymbol{u}}_2$. If we multiply (*a*) by $\dot{\boldsymbol{u}}_2^T$ and (*b*) by $\dot{\boldsymbol{u}}_1^T$, we get (*c*) $\kappa_1 \dot{\boldsymbol{u}}_2^T \boldsymbol{G}\dot{\boldsymbol{u}}_1 = \dot{\boldsymbol{u}}_2^T \boldsymbol{D}\dot{\boldsymbol{u}}_1$ and (*d*) $\kappa_2 \dot{\boldsymbol{u}}_1^T \boldsymbol{G}\dot{\boldsymbol{u}}_2 = \dot{\boldsymbol{u}}_1^T \boldsymbol{D}\dot{\boldsymbol{u}}_2$. Since $\boldsymbol{G}$ and $\boldsymbol{D}$ are symmetric, the following equations hold: $\dot{\boldsymbol{u}}_2^T \boldsymbol{G}\dot{\boldsymbol{u}}_1 = \dot{\boldsymbol{u}}_1^T \boldsymbol{G}\dot{\boldsymbol{u}}_2$ and $\dot{\boldsymbol{u}}_2^T \boldsymbol{D}\dot{\boldsymbol{u}}_1 = \dot{\boldsymbol{u}}_1^T \boldsymbol{D}\dot{\boldsymbol{u}}_2$. By the subtraction $(c)-(d)$, we have $(\kappa_1 - \kappa_2)\dot{\boldsymbol{u}}_2^T \boldsymbol{G}\dot{\boldsymbol{u}}_1 = 0$. If $\kappa_1 \neq \kappa_2$, (*e*) $\dot{\boldsymbol{u}}_2^T \boldsymbol{G}\dot{\boldsymbol{u}}_1 = 0$. Note that $\boldsymbol{G} = \boldsymbol{A}^T\boldsymbol{A}$. Then, (*e*) is reduced to $\dot{\boldsymbol{u}}_2^T \boldsymbol{G}\dot{\boldsymbol{u}}_1 = (\boldsymbol{A}\dot{\boldsymbol{u}}_2)^T(\boldsymbol{A}\dot{\boldsymbol{u}}_1) = 0$. Therefore, $\boldsymbol{A}\dot{\boldsymbol{u}}_2 \perp \boldsymbol{A}\dot{\boldsymbol{u}}_1$.



$\underline{\dot v}$ or $n \times r_u / \|r_u\|$
$\underset{\sim}{\dot v}$ or $n \times i_U$
$\underset{\sim}{\dot u}$ or i_U
$\underline{\dot u}$ or $r_u / \|r_u\|$
η
n
$n \times r_u / \|r_u\|$
r_v
v
T_rS
η
$r_u / \|r_u\|$
$\underline{\dot v}$
$\underset{\sim}{\dot v}$
η
$\underset{\sim}{\eta}$
$\underset{\sim}{\dot u}$
η_o
$\underline{\dot u}$

Figure 8 Coordinate Systems of a Tangent Space

matrix $\boldsymbol{P}$ the *(rotated) modal matrix* and the matrix $\underset{\sim}{\boldsymbol{P}}$ the *(principal) modal matrix* at a given point on the surface. The following relation can be confirmed through routine manipulation:

$$\boldsymbol{P}^T\boldsymbol{G}\boldsymbol{P} = \boldsymbol{1} \qquad \text{and} \qquad \underset{\sim}{\boldsymbol{P}}^T\boldsymbol{G}\underset{\sim}{\boldsymbol{P}} = \boldsymbol{1}. \tag{5}$$

The pair $(\dot u, \dot v)$ is considered as a point in a Cartesian plane which we call the $(\dot u, \dot v)$-space. In the same manner, we define the $(\underline{\dot u}, \underline{\dot v})$-space and the $(\underset{\sim}{\dot u}, \underset{\sim}{\dot v})$-space. The angle $\underset{\sim}{\eta}$ of a vector $\boldsymbol{v} \in T_rS$ from $\boldsymbol{i}_U$ is called its *principal direction angle*, *i.e.* $\underset{\sim}{\dot{\boldsymbol{u}}} = \vartheta \cdot [cos\underset{\sim}{\eta} \;\; sin\underset{\sim}{\eta}]^T$, where $\vartheta \equiv \|\boldsymbol{v}\|$ is its magnitude. The angle η from $\boldsymbol{r}_u$ is referred to as the *direction angle* of the vector, *i.e.* $\underline{\dot{\boldsymbol{u}}} = \vartheta \cdot [cos\eta \;\; sin\eta]^T$. There is a relation between them: $\eta = \underset{\sim}{\eta} + \eta_o$, where η_o is the angle of $\boldsymbol{i}_U$ from $\boldsymbol{r}_u$.[†] The angle η and the magnitude ϑ can be thought of as the coordinates of a *polar coordinate system* for the $(\underline{\dot u}, \underline{\dot v})$-space. These are shown in Figure 8.

Finally, the relation between the pairs, $(\dot u, \dot v)$ and (ϑ, η), is

$$\vartheta = \sqrt{\boldsymbol{v}^T\boldsymbol{v}} = \sqrt{\dot u^2 \cdot \boldsymbol{r}_u^T\boldsymbol{r}_u + 2\dot u\dot v \cdot \boldsymbol{r}_u^T\boldsymbol{r}_v + \dot v^2 \cdot \boldsymbol{r}_v^T\boldsymbol{r}_v} = \sqrt{\dot{\boldsymbol{u}}^T\boldsymbol{G}\dot{\boldsymbol{u}}}$$

$$\eta = atan_2[\{\dot v \cdot \boldsymbol{r}_v^T(\boldsymbol{n} \times \boldsymbol{r}_u)\}, \{\dot u \cdot \boldsymbol{r}_u^T\boldsymbol{r}_u + \dot v \cdot \boldsymbol{r}_v^T\boldsymbol{r}_u\}]$$

where $\boldsymbol{G}$ is the *1st fundamental matrix* of the surface in differential geometry. Inverted at each point on the surface, the pair $(\dot u, \dot v)$ is known in terms of the direction angle η and the magnitude ϑ as:

$$\dot{\boldsymbol{u}} = \vartheta \cdot \boldsymbol{h}(\eta, u, v) \text{ or in component form,} \tag{6}$$

$$\dot u = \vartheta \cdot h_1(\eta, u, v) \text{ and } \dot v = \vartheta \cdot h_2(\eta, u, v)$$

where $\boldsymbol{h}: \mathcal{R} \times \mathcal{P} \to \mathcal{R}^2$ can be derived to be:

$$\boldsymbol{h}(\eta, u, v) \equiv [h_1(\eta, u, v) \;\; h_2(\eta, u, v)]^T = \boldsymbol{P}(u, v) \cdot \boldsymbol{a}(\eta).$$

[†] The angle η_o is found by the following system of equations: $cos\eta_o = \boldsymbol{i}_U^T\boldsymbol{r}_u / \|\boldsymbol{r}_u\|$ and $sin\eta_o = \boldsymbol{n}^T(\boldsymbol{r}_u \times \boldsymbol{i}_U) / \|\boldsymbol{r}_u\|$.

Note that $\boldsymbol{a}(\eta) \equiv [cos\eta \quad sin\eta]^T$ and $\boldsymbol{P}(u, v)$ is the modal matrix at the point $(u, v) \in \mathcal{P}$ as defined in Equation (4).

We summarize the transformations between the coordinate systems that were introduced in this subsection:

$$\dot{\boldsymbol{u}} = \boldsymbol{P}\dot{\underline{\boldsymbol{u}}} = \utilde{\boldsymbol{P}}\utilde{\dot{\boldsymbol{u}}} = \vartheta \cdot \boldsymbol{h}(\eta, u, v) = \vartheta \cdot \boldsymbol{h}(\utilde{\eta} + \eta_o, u, v)$$

$$\dot{\underline{\boldsymbol{u}}} = \vartheta \cdot [cos\eta \quad sin\eta]^T \text{ and } \utilde{\dot{\boldsymbol{u}}} = \vartheta \cdot [cos\utilde{\eta} \quad sin\utilde{\eta}]^T.$$

4. Typical CNC code and Kinematics of a Machine Tool

A machine tool is essentially a robot. We need 3 translational and 2 rotational coordinates to describe the configuration (or posture) of a machine tool in *5-axis machining*. A classic way to describe a *rigid body motion* is to apply a translation and a rotation about a specified point on the rigid body, which is called its *base point* [132]. It should be pointed out that in 5-axis machining, the motion of a cutting tool worth analysis is not a rigid body motion. There are two reasons for this. First, the spinning rotation of the spindle base does not contribute to a milling process to a considerable extent because the last (spinning) rotation provided by a "powerful" spindle motor is the far more dominant factor for the cutting process. Second, the spinning motion is a minor factor for interference problems, too, because of the (pseudo) symmetry of a cutting tool. Therefore, the motion that needs to be described is not the rigid body motion of a cutting tool, but rather the motion of its centerline (or the line of symmetry).

We align a **unit** vector $\boldsymbol{q}$ $(= q_x \cdot \boldsymbol{i} + q_y \cdot \boldsymbol{j} + q_z \cdot \boldsymbol{k} \in T_{\boldsymbol{r}}\mathcal{W},\ \|\boldsymbol{q}\| = 1)$ with the centerline from the tool tip towards the spindle base, and refer to it as the *orientation* (or *access*) *vector* as shown in Figure 9. The orientation vector accounts for the rotational motion of the centerline. On the other hand, to account for the translation of the centerline, we attach a point *M* to it, the position vector of which in the workpiece space $\mathcal{W}$ is denoted by $\boldsymbol{r}^M$ $(= r_x^M \cdot \boldsymbol{i} + r_y^M \cdot \boldsymbol{j} + r_z^M \cdot \boldsymbol{k})$. This nominal point plays a similar role as the base point of a rigid body motion. In short, the *work space* in interest for 5-axis machining is $\mathcal{R}^3 \times \bar{S}^2$ where $\bar{S}^2$ is a *unit 2 sphere*. In path planning literature [*e.g.* 16 and 19], the point *M* is called the *cutter location point* (*CL-point*), and the (time) series of the 6-tuple $(\boldsymbol{r}^M, \boldsymbol{q})$ is called the *cutter location data* (*CL-data*). It is possible that the orientation vector $\boldsymbol{q}$ is specified by a pair (φ, ϕ) of two real numbers, for example, the two angular coordinates in a spherical coordinate system. The details of this parameterization are discussed in **APPENDIX B**.

The 5-tuple $(r_x^M, r_y^M, r_z^M, \varphi, \phi)$ determines the configuration of a cutting tool as well as the *joint angles* θ_i $(i = 1, \ldots, N)$, where $N \geq 5$ is the number of actuators.† Typically, CNC *G-codes* of a machine tool for 5-axis machining consist of a series of such 5-tuple location descriptors and feedrates. In robotics, the map from the joint angles to the corresponding posture $(r_x^M, r_y^M, r_z^M, \varphi, \phi)$ is called the *forward kinematics* of a

† 5-axis machining is not necessarily performed by 5-axis machine tools. For example, a hexapod machine tool has 6 axes. With such redundant machine tools, the redundancy can be resolved by a control law of "holonomic" type [12]. If the control law is of "non-holonomic" type, the problem becomes fairly complicated. For 3-axis machining, the two orientation parameters, φ and ϕ, are constants over the surface and we require that $N \geq 3$.

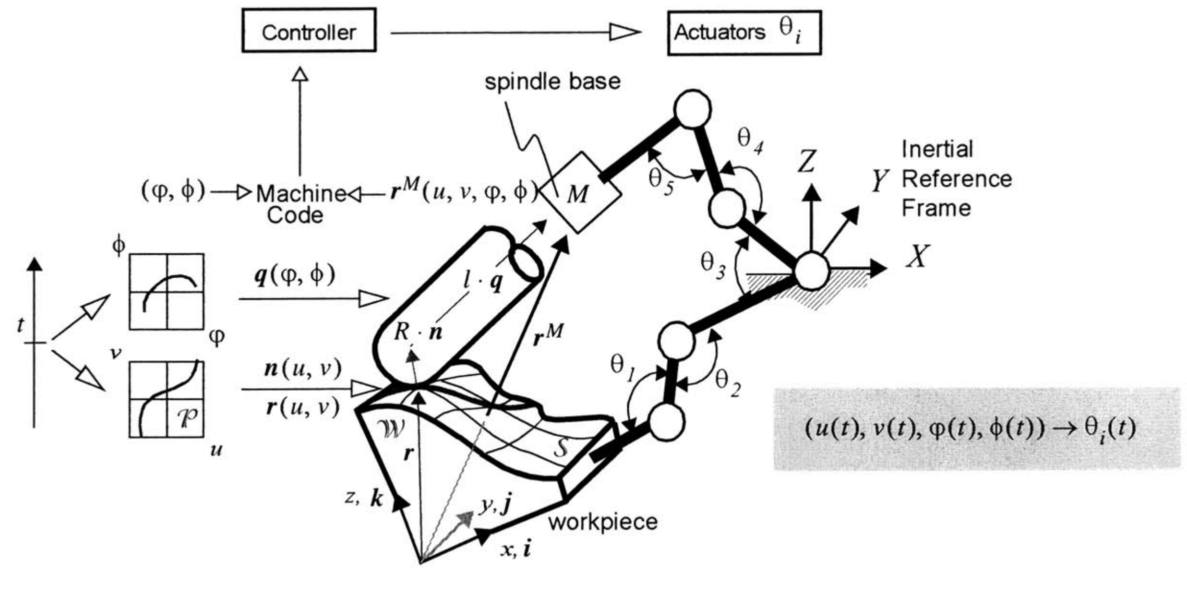


Figure 9 Surface Inverse Kinematics

machine tool. It is inverted either by the controller of the machine tool or by a planner, which we call the "(kinematic) *control law*" of a machine tool. For a non-redundant machine tool, this control law can be thought of as its *inverse kinematics*.

5. Motion Restricted by the Designed Surface and the Surface Inverse Kinematics

After defining proper descriptors for surfaces and machine tools, we now study their interactions and define a new concept, *surface inverse kinematics*.

To prevent the cutting tool from gouging the designed surface, we require the envelope of the cutting tool to touch the designed surface tangentially at every instant. The trajectory of such a point of contact (namely, *CC-point*) is referred to as the *(surface) tool path*, which is a surface curve. The time-series of the CC-point and the corresponding access vector is called the *CC-data*. One way to program a CNC machine tool is to first generate the CC-data in the parameter space and to convert it into the CL-data or machine code. This conversion is made by the requirement that the cutting tool should osculate the designed surface.

In the case of a ball-end mill, this *requirement of tangency* results in the following relation:

$$\boldsymbol{r}^M = \boldsymbol{r} + R \cdot \boldsymbol{n} + l \cdot \boldsymbol{q}$$

where R is the radius of the tool, l is the length from the ball center to the CL-point M, $\boldsymbol{r} \in \mathcal{S}$ is a CC-point, $\boldsymbol{r}^M$ is the position vector to the CL-point, $\boldsymbol{q} \in T_r\mathcal{W}$ is the access vector and $\boldsymbol{n}$ is the outside unit surface normal vector at $\boldsymbol{r} \in \mathcal{S}$ as shown in Figure 9.

Then, the configuration $(r_x^M, r_y^M, r_z^M, \varphi, \phi)$ of a machine tool, restricted by this condition, is described by 4 parameters (u, v, φ, ϕ), which define the location and orientation of the cutting tool on the surface. The 4-tuple, (u, v, φ, ϕ), is transformed into a machine code by a planning system using the *requirement of tangency*. The machine code is further transformed into the joint angles θ_i by the controller based on the *inverse kinematics* of the mechanical system. In other words, we can establish a mapping *f* between the parameters and the joint angles, *i.e.*

$$\theta_i = f_i(u, v, \varphi, \phi), \qquad (i = 1, \ldots, N) \tag{7}$$

which we call a *surface inverse kinematics map*.

At the outset of this formulation, we assumed that the orientation vector is given by a fixed vector field on the surface, which implies that the two orientation parameters, φ and ϕ, are known as functions of the surface parameters u and v, *i.e.*, $\varphi = \varphi_o(u, v)$ and $\phi = \phi_o(u, v)$. Inserting these restrictions into the surface inverse kinematics, we have a new map f^o *s.t.*

$$\theta_i = f_i(u, v, \varphi_o(u, v), \phi_o(u, v)) \equiv f_i^o(u, v)$$

which we call a *restricted surface inverse kinematics map* or simply a *surface inverse kinematics*. An example for deriving a surface inverse kinematics is shown in **Appendix B**.

6. Dynamics of a Machine Tool

It is possible to develop the *equations of motion* of a machine tool, taking its *joint angles* θ_i as *generalized coordinates* of a *rigid body system*, the machine tool under consideration. In general, the equation of motion can be derived in the following form of equations:

$$\tau_i = \tau_i\left(\theta_1, \ldots, \theta_N, \frac{d\theta_1}{dt}, \ldots, \frac{d^2\theta_1}{dt^2}, \ldots\right), \quad (i = 1, \ldots, N) \tag{8}$$

where τ_i are the *joint torques*, $d\theta_i/dt$ are the generalized velocities, $d^2\theta_i/dt^2$ are the generalized accelerations. Of course, we would have to assume that the cutting forces could be derived in an analytical form.† Especially, in high speed machining, the cutting forces are negligible. We point this out here for completeness; we do not directly address forces and dynamics in this work.

†For deriving the equation of motion of a mechanical system, readers are referred to standard texts on multi-body dynamics or robotics [133 and 134]. Generally, a CL-data point or the 5-tuple $(r_x^M, r_y^M, r_z^M, \varphi, \phi)$ is not a *complete* set of generalized coordinates [135]. A CL-data point is a complete set of generalized coordinates only of a 5-axis machine tool. There are a massive body of research on cutting force prediction [*e.g.* 60].

7. Field Description

A surface tool path is essentially a (time) parametrized surface curve $\boldsymbol{r}(u(t), v(t)) \equiv \boldsymbol{r}(t)$ (or $\boldsymbol{r}(\boldsymbol{u}(t))$),[†] which corresponds to a curve $(u(t), v(t)) \equiv \boldsymbol{u}(t)$ in the parameter space (given a map $\boldsymbol{u}:(\mathcal{T} \subset \mathcal{R}) \to \mathcal{P}$ in an interval $\mathcal{T}$). The curve $\boldsymbol{u}(t)$ in the parameter space is called a *(parameter) tool path*. We construct a parameter tool path $\boldsymbol{u}(t)$ and map it to a *surface tool path* $\boldsymbol{r}(t)$ through the surface parameterization $\boldsymbol{r}$. Of course, we can choose different parameters for the curves, for example, using the arc length s of the surface tool paths.

The (time) series of CC-points, either in $\mathcal{S}$ or $\mathcal{P}$, is a straightforward descriptor for tool paths. We explain, however, another way to represent tool paths in this subsection. This indirect method conveniently deals with families of tool paths rather than an individual tool path.

The Velocity Vector Field

An important abstraction that we introduce in this work is the use of a vector field to capture the skeletal information of a family of tool paths. The velocity of a tool path $\boldsymbol{r}(t)$ at an instant is evaluated by the time derivative $d\boldsymbol{r}/dt$ at the given instant. Conversely, if we were to postulate a vector field $\boldsymbol{v}(\boldsymbol{r})$ on the surface and to require that $\boldsymbol{v}(\boldsymbol{r}) = d\boldsymbol{r}/dt$ on tool paths, then it would be possible to integrate the vector field (or the differential equation) and to generate tool paths, which are *streamlines* or *integral curves* of the vector field. This is the procedure we will use in this thesis; we will find a "good" vector field and then generate tool paths by integrating the vector field.

Streamlines

A streamline of the velocity vector field $\boldsymbol{v}(\boldsymbol{r})$ on the designed surface $\mathcal{S}$ is called a *surface streamline* and the corresponding curve on the parameter space $\mathcal{P}$ is called a *parameter streamline*. Henceforth, the symbols, t and s, are reserved exclusively for the time and arc length parameter of a surface streamline, respectively. The unit tangent vector of a surface streamline of the velocity field is denoted by $\boldsymbol{t}$, *i.e.*, $\boldsymbol{t} \equiv \boldsymbol{v}/\|\boldsymbol{v}\|$.

The Velocity Vector Field in the Parameter Space

Because a surface is a 2-dimensional entity (or a manifold), it is more convenient to express the velocity vector field with 2 components rather than using 3 components. We define a vector field $\dot{\boldsymbol{u}}(\boldsymbol{u})$ on the parameter space $\mathcal{P}$ that is related to the velocity vector field $\boldsymbol{v}(\boldsymbol{r})$ in the following way:

$$\boldsymbol{v} = \boldsymbol{r}_u \dot{u} + \boldsymbol{r}_v \dot{v} = \boldsymbol{A}\dot{\boldsymbol{u}}$$

where $\dot{\boldsymbol{u}}(\boldsymbol{u}) \equiv [\dot{u}(u, v) \;\; \dot{v}(u, v)]^T$ and $\boldsymbol{A}$ is the Jacobian matrix of the surface map $\boldsymbol{r}$.[‡]

[†] The statement $\boldsymbol{r}(u(t), v(t)) \equiv \boldsymbol{r}(t)$ is not without confusion. The 2nd $\boldsymbol{r}$ is in fact the composition map $\boldsymbol{r} \circ \boldsymbol{u}$.
[‡] The terms $\boldsymbol{v}$, $\dot{\boldsymbol{u}}$, $\underline{\dot{u}}$, $\underline{\dot{u}}$, η and ϑ that are defined in Subsection 3 are general mathematical entities and do not have any physical connection. From now on, those notations are used exclusively for the velocity vectors of a tool path.

Then this field value $\dot{\boldsymbol{u}}(\boldsymbol{u})$ is proved to correspond to the time-derivative of the parameter streamlines which we call the *parameter velocity vector*, namely,

$$\dot{\boldsymbol{u}}(\boldsymbol{u}) = d\boldsymbol{u}/dt$$

where $\boldsymbol{u}(t)$ is a parameter streamline. Therefore, if we specify the (parameter) velocity vector field $\dot{\boldsymbol{u}}(\boldsymbol{u})$, the above differential equation can be integrated to generate a parameter tool path. The parameter tool path is finally mapped to the surface tool path $\boldsymbol{r}(\boldsymbol{u}(t))$. The transformation, $\dot{\boldsymbol{u}} = \boldsymbol{P}\dot{\underline{\boldsymbol{u}}} = \underset{\sim}{\boldsymbol{P}}\underset{\sim}{\dot{\boldsymbol{u}}}$ (as defined in Equation 4), also shows two other coordinate systems for velocity vectors, which were covered in a general setting in Subsection 2.2.3.

THE DIRECTION ANGLE

The velocity vector field $\boldsymbol{v}(\boldsymbol{r})$ or $\dot{\boldsymbol{u}}(\boldsymbol{u})$ can be specified also in terms of polar components as well as its Cartesian components. The magnitude component is the cutting speed ϑ and the angular component is the angle η with the u-lines (or iso-v-parametric lines) as shown in Figure 10-(a):

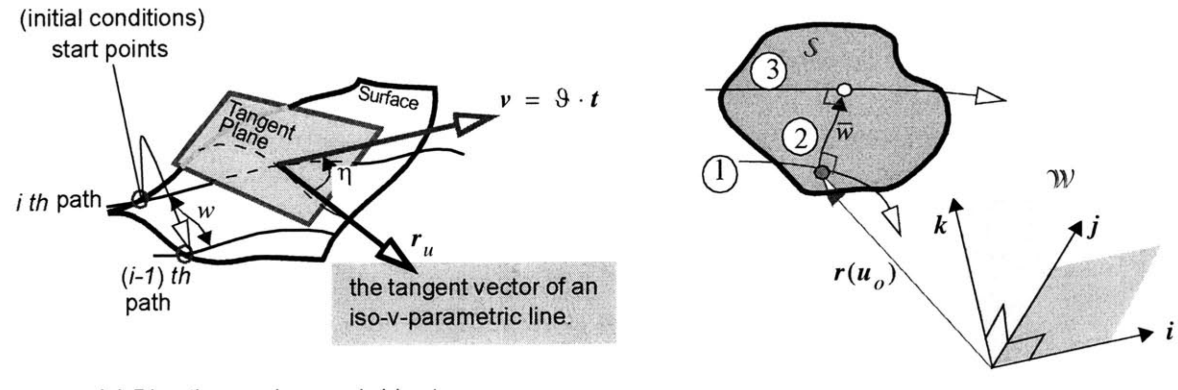
$$\boldsymbol{v} = \vartheta \cdot (\cos\eta \cdot \boldsymbol{r}_u / \|\boldsymbol{r}_u\| + \sin\eta \cdot \boldsymbol{n} \times \boldsymbol{r}_u / \|\boldsymbol{r}_u\|)$$

where $\boldsymbol{n}$ is the unit outside surface normal vector. The field η is called the *direction angle* or *direction field*. The speed $\vartheta(\boldsymbol{u})$ and the direction angle $\eta(\boldsymbol{u})$ are related to the vector field $\dot{\boldsymbol{u}}(\boldsymbol{u})$ on $\mathcal{P}$ by the following relation: $\dot{\boldsymbol{u}}(u, v) = \vartheta(u, v) \cdot \boldsymbol{h}(\eta(u, v), u, v) = \vartheta(u, v) \cdot \boldsymbol{P}(u, v) \cdot [\cos\eta(u, v) \;\; \sin\eta(u, v)]^T$ as explained in Equation (6) in a general setting, (where $\boldsymbol{P}(u, v)$ is the modal matrix at a point $(u, v) \in \mathcal{P}$ and $\boldsymbol{h}: \mathcal{R} \times \mathcal{P} \rightarrow \mathcal{R}^2$ is a map $s.t.$ $\boldsymbol{h}(\eta, u, v) \equiv \boldsymbol{P}(u, v) \cdot [\cos\eta \;\; \sin\eta]^T$). Therefore, if we specify the two functions, $\vartheta(u, v)$ and $\eta(u, v)$, it is possible to integrate the following differential equation from a "start point" to get a parameter tool path $\boldsymbol{u}(t)$ and its corresponding surface tool path $\boldsymbol{r}(\boldsymbol{u}(t))$:

$$\frac{d\boldsymbol{u}}{dt} = \vartheta(u, v) \cdot \boldsymbol{h}(\eta(u, v), u, v) \tag{9}$$

(or equivalently, $\frac{d\boldsymbol{u}}{ds} = \boldsymbol{h}(\eta(u, v), u, v)$ and $\frac{ds}{dt} = \vartheta(u, v)$).

In practice, we will adopt Cartesian components for specifying the vector field. In theory, we assume that the vector field is specified by the polar components because the polar system allows us to state the formulation rather conveniently.



(a) Direction angle η and side step w (b) Shifting Procedure

Figure 10 Field description and Tool paths

Side Step Function

The vector field, $\boldsymbol{v}$ or $\dot{\boldsymbol{u}}$, which is specified by the speed $\vartheta(u, v)$ and the direction angle $\eta(u, v)$, carries only the skeletal information on the family of tool paths. This vector field has infinitely-many *streamlines*, or *integral curves*, each of which could potentially be used as a tool path. In addition, tool paths must be spaced properly—after all, tool paths are separate swathes not a continuum. The missing element is how to choose the tool paths from the streamlines, which is effectively described by the "start points" or initial conditions (together with the final conditions or arc length) of these paths as shown in Figure 10-(*a*). It is more convenient to devise a function on the surface that contains the information on how wide the gap between two neighboring tool paths is. By treating the "gap width" as a (piecewise) continuous function, we make the path planning problem amenable to convenient mathematical tools such as calculus and differential geometry. This is a crucial mechanism in our attempt to tackle this unstructured problem, which is analogous to *continuum hypothesis* in continuum mechanics. The devised function $w(u, v)$ is referred to as the *side step* (*function*) and the gap is "measured" in the direction perpendicular to the tool paths as roughly shown in Figure 10-(*a*). The immediately following part shows how we interpret the side step function "physically" and Section 2.3 defines a condition that the side step function must meet with respect to a direction field.

Construction of Tool Paths from the Field Description

Suppose we have found a "good" triad $(\eta(u, v), w(u, v), \vartheta(u, v))$ of functions on the parameter space. From Equation (9 or 6), the vector field $\dot{\boldsymbol{u}}$ is known. Then, we take a streamline of the velocity vector field as the first tool path and follow the subsequent procedure:

(Shifting Procedure)

(1) On the first tool path, we pick a point, say $\boldsymbol{r}(\boldsymbol{u}_o)$ ($\boldsymbol{u}_o \in \mathcal{P}$) as shown in Figure 10-(*b*). We construct an **arc-length** parameterized surface curve $\boldsymbol{r}(\boldsymbol{u}_\perp(\alpha)) \equiv \boldsymbol{r}_\perp(\alpha)$ which is both centered at the point $\boldsymbol{r}(\boldsymbol{u}_o)$ and perpendicular to the streamlines of the velocity vector field $\boldsymbol{v}$.† In other words, the curve $\boldsymbol{r}_\perp(\alpha)$ is a streamline of the vector field $\boldsymbol{n} \times \boldsymbol{t}$.

(2) The side step along this **arc length** parameterized **orthogonal** *transverse curve* $\boldsymbol{r}(\boldsymbol{u}_\perp(\alpha))$ is known by $w(\boldsymbol{u}_\perp(\alpha)) \equiv w_\perp(\alpha)$.

(3) For the variable $\bar{w}$, we solve the following nonlinear equation:

$$\int_0^{\bar{w}} \frac{d\alpha}{w_\perp(\alpha)} = 1 \tag{10}$$

(4) We choose the streamline of the velocity vector field $\boldsymbol{v}$ that passes through the point $\boldsymbol{r}(\boldsymbol{u}_\perp(\bar{w}))$ as the next tool path.

† In equations, $\boldsymbol{v}^T \cdot \frac{d\boldsymbol{r}_\perp}{d\alpha} = \dot{\boldsymbol{u}}^T \cdot \boldsymbol{G} \cdot \frac{d\boldsymbol{u}_\perp}{d\alpha} = 0$ $\left\|\frac{d\boldsymbol{r}_\perp}{d\alpha}\right\|^2 = \left(\frac{d\boldsymbol{u}_\perp}{d\alpha}\right)^T \boldsymbol{G}\left(\frac{d\boldsymbol{u}_\perp}{d\alpha}\right) = 1$ and $\boldsymbol{u}_\perp(0) = \boldsymbol{u}_o$.

We say then that we *shifted* the point $\boldsymbol{r}(\boldsymbol{u}_o)$ to the point $\boldsymbol{r}(\boldsymbol{u}_\perp(\bar{w}))$. Iteration of the same steps will fill the surface with tool paths. (Of course, we can partition the surface and fill the surface by performing the above shifting procedure in each region—along the partition line, "intermittent cuts" can happen.)

This procedure shows that from the triad of functions ($\eta(u, v)$, $w(u, v)$, $\vartheta(u, v)$), it is possible to place tool paths. These functions capture the essential information of reasonable, contiguous paths. Of course, the side step function is an artificial device that imposes "**regularity**" on the problem. Roughly speaking, Equation (10) declares that the "**harmonic mean**" of the side step in the gap between two neighboring tool paths coincides with the arc length $\bar{w}$ of the transverse curve $\boldsymbol{r}_\perp(\alpha)$ because Equation (10) can be rewritten in the following form:

$$\frac{1}{\bar{w}} = \frac{1}{\bar{w}} \cdot \int_0^{\bar{w}} \frac{d\alpha}{w_\perp(\alpha)}.$$

This equation shows how the "gap width" $\bar{w}$ is measured with the side step function $w(u, v)$.

Classes of the Participating Functions

It is general enough to require such participating functions, $\vartheta(u, v)$, $\eta(u, v)$ and $w(u, v)$, to be **piece-wise smooth** and **piece-wise continuous**. In fact, we will refine this statement further in the next section.

8. Summary

Typically, tool paths have been specified as a time series of a certain nominal point. The field description achieves the same objective by specifying the triad $(\vartheta(u, v), \eta(u, v), w(u, v))$ of functions, which are the speed, the direction angle and the side step. They are the unknowns of our optimization problem. The questions now we ask are what defines good triads and how we would find it.

2.3 Compatibility: an Equality Constraint

In the preceding section, we have seen a procedure to pick tool paths from the infinitely-many streamlines of a velocity vector field based on the field description. With the same piece of information, we can take a different procedure. At first, we decide the first tool path as before.

(1) We shift "enough" points on the first tool path solving Equation (10).

(2) We "interpolate" the shifted points, and declare the newly interpolated curve as the next tool path.

This procedure is shown in Figure 11-(*b*). If the side step function is defined in an arbitrary way, there is no guaranteeing that the interpolated (or shifted) curve is a streamline of the velocity vector field. This shows that

the side step function $w(u, v)$ and the direction field $\eta(u, v)$ are not independent.

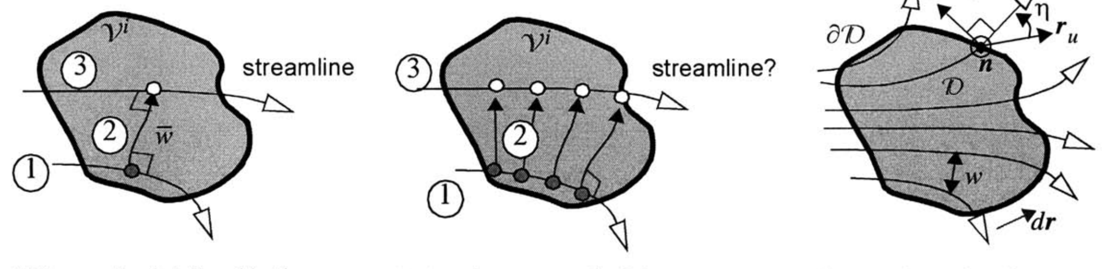
A condition that is imposed on the side step function with respect to a direction field is defined in this section. In our optimization problem, this relation plays a role as an equality constraint. We refer to it as the *compatibility* (*equation*).

In short, the compatibility requires that

CONDITION I:

given a velocity vector field (or direction field) and a side step function, there are finitely-many **mutually-disjoint simply-connected** open sets $\mathcal{V}^i \subset \mathcal{S}$ $(i = 1, \ldots, n)$ *s.t.* $\bigcup_i closure(\mathcal{V}^i) = \mathcal{S}$ and

in each region $\mathcal{V}^i$, from any streamline in the region $\mathcal{V}^i$, there is another streamline in the same region that passes through all the shifted points. (We do not account for the points which are shifted out of the region $\mathcal{V}^i$ but we require that at least one point on a streamline in $\mathcal{V}^i$ is shifted to a point in $\mathcal{V}^i$.)



a) The method defined in the previous section

b) Another way to find the next tool path

c) The number of tool paths is conserved.

Figure 11 Compatibility Equation

We start by deriving a differential relation which is sufficient for above CONDITION I. For non-smooth direction fields, jump conditions are specified along boundaries between two regions in each of which tool paths are smooth. Finally, a formal structure is constructed to effectively describe this rather complicated constraint of the optimization problem under consideration.

1. Compatibility via a Stream Function

The following CONDITION II is sufficient for CONDITION I.

CONDITION II:

given a direction field (or velocity vector field) and a side step function, there are finitely-many mutually-disjoint simply-connected open sets $\mathcal{V}^i \subset \mathcal{S}$ $(i = 1, \ldots, n)$ *s.t.* $\bigcup_i closure(\mathcal{V}^i) = \mathcal{S}$ and

in each region $\mathcal{V}^i$, the *differential 1-form*

$\psi \equiv \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}$ is **exact**, in other words,

$$\text{there is a } \textbf{continuous} \text{ function } \Psi \ s.t. \ d\Psi = \psi \ (\text{or } \Psi = \int\psi) \tag{11}$$

where $\boldsymbol{t}$ is the unit tangent vector (field) of the given velocity vector field, $\boldsymbol{n}$ is the unit outside surface normal vector (field), $\mathcal{S}$ is the designed surface, w is the side step function and $d\boldsymbol{r}$ is a differential 1-form on the designed surface $\mathcal{S}$, *i.e.* $d\boldsymbol{r} = \boldsymbol{r}_u du + \boldsymbol{r}_v dv$.

We refer to the function Ψ that appears in CONDITION II as the *stream function* of the given direction field, borrowing a fluid mechanics term.

To see if CONDITION II guarantees CONDITION I, we first observe that, by definition,

$$\psi(\boldsymbol{t}) \equiv \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T \boldsymbol{t} = 0.^{\dagger}$$

This implies that the integral $\int\psi \equiv \int\psi(\boldsymbol{t}) \cdot ds$ along every streamline vanishes. Therefore, the stream function Ψ is constant along a streamline of the velocity vector field. In other words,

level sets of the stream function Ψ are streamlines.

Second, we observe that

† A differential 1-form can be regarded as a function of tangent vectors. One way to define the integration of differential 1-forms on a manifold is to regard the integration as a "usual" one of the *pull-back* of the differential forms. In our case, $\psi(\boldsymbol{t})$ is the value of the differential 1-form at the vector $\boldsymbol{t}$ and $\psi(\boldsymbol{t}) \cdot ds$ is the pull-back of the differential form along the streamline. The general definitions and discussion on this issue can be found in most text books on differential forms such as [103].

$$\psi(\boldsymbol{n} \times \boldsymbol{t}) \equiv \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T (\boldsymbol{n} \times \boldsymbol{t}) = \frac{1}{w} \text{ or}$$

$$\int_{C_\perp} \psi \equiv \int \psi(\boldsymbol{n} \times \boldsymbol{t}) \cdot d\alpha = \int_0^{\bar{w}} \frac{d\alpha}{w_\perp(\alpha)} \text{ where } \mathcal{C}_\perp = \{ \boldsymbol{r}_\perp(\alpha) : 0 \le \alpha \le \bar{w} \} . \tag{12}$$

That is, the integration $\int \psi$ along an arc length parameterized orthogonal transverse curve $\boldsymbol{r}_\perp(\alpha)$ is identical to the integration defined in the shifting procedure, Equation (10). Now, if we set the stream function Ψ along a first tool path to be zero, then, by definition, $\Psi = 1$ at all the shifted points. This allows us to conclude that a streamline (or a level set of Ψ) contains all the shifted points.

2. Compatibility as a Conservation Law

It is well known that if a differential 1-form ψ is **exact** then its integral $\int \psi$ is **path-independent** or its cyclic integration vanishes, namely, $\oint \psi = 0$. [103]. To most engineers or physicists, this equivalent statement is probably the most appealing form. We state that

$$\oint_{\partial \mathcal{D}} \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} = 0 \tag{13}$$

for all the open subsets $\mathcal{D}$ of each region $\mathcal{V}^i$ (as defined in Condition **I** and **II**). This integration can be thought of as a *flux* integration and a physical interpretation of the equation is that "the number of tool paths" (or the flux of the vector field $\boldsymbol{t}/w$) into the region $\mathcal{D}$ must be "the number of tool paths" out of the region as shown in Figure 11-(*c*). This compatibility only applies in regions where there are **no intermittent cuts.** It is an analog of the principle of conservation of mass assuming no sources or sinks in fluid mechanics terms.

3. Compatibility as a Partial Differential Relation: Generalized Stokes' Theorem (or Poincaré Lemma)

We invoke *generalized Stokes' theorem* for Equation (13) and say that $\int_{\mathcal{D}} d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}/w] = 0$, where by the symbol d we denote the *exterior derivative* of a differential form.† By the arbitrariness of the domain $\mathcal{D}$, the differential 2-form, $d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}/w]$, itself should vanish, *i.e.*

$$d\psi \equiv d\left[\frac{(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}}{w}\right] = 0 \qquad (a.e.). \tag{14}$$

This derivation is analogous to deriving the *continuity equation* in fluid mechanics using the *divergence theorem*. (We can reach the same conclusion from Condition **II** using *Poincaré lemma* [114], $d\psi \equiv d(d\Psi) = 0$.) By applying the rules for the *exterior derivative* and the *wedge product* on a manifold, the following, what we refer to as the *differential continuity* or *type-0 compatibility*, can be derived:

† We use *exterior calculus* here using *differential forms* and the *generalized Stokes' theorem*. This approach is more natural for a curved space than elementary vector and tensor analysis [106]. They are described in several books including [103~108].

$$h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v) \cdot w \qquad (a.\,e.) \qquad (15)$$

where

$h = [h_1 \ h_2]^T = Pa$, (as defined earlier in Equation 6)

$$h_3 \equiv \left[[\eta_u \ \ \eta_v]P + \left([0 \ \ 1]\frac{\partial}{\partial u}(GP) - [1 \ \ 0]\frac{\partial}{\partial v}(GP) \right) / \sqrt{\det G} \right] Ra, \text{ and } R \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Note that $a \equiv [\cos\eta \ \ \sin\eta]^T$, $G = [g_{ij}]$ is the *1st fundamental matrix* of the surface as shown in Equation (2), P was defined in Equation (4) and R is the $90°$-rotation matrix. A step-by-step derivation is given in APPENDIX **A.1**.

4. Jump Conditions

Just for convenience, we introduce the following term: for a direction field and a side step function,

> A **GRAIN** is a simply-connected open set in the designed surface $\mathcal{S}$, at every point of which the **differential continuity** holds. (A formal definition is given in Subsection 6.)

If the direction field $\eta(u, v)$ is not C^1-continuous, the differential continuity as shown in Equation (15), does not hold. This is one of the reasons why the modifier (*a.e.*), which stands for *almost everywhere*, appears in the equation and also why we included somewhat technical conditions about the sets $\mathcal{V}^i$ in CONDITION **I** and **II**. Along *grain boundaries*, another form of compatibility, which must be a "non-differential" relation, should be specified (if we require the continuity of tool paths across the grain boundaries).

THE CONTINUITY JUMP CONDITION

Consider the two neighboring grains, $\mathcal{G}_1$ and $\mathcal{G}_2$, as shown in Figure 12-(*a*). Let a **unit** tangent vector of the grain boundary be denoted by $l \in T_r\mathcal{S}$. Of course, we assume that the boundary curve is at least piecewise smooth. We resolve the ambiguity in the sign of the vector l by setting

$$l^T \cdot (n \times t) > 0$$

The same condition can be stated in terms of vectors in $(\dot{\underline{u}}, \dot{\underline{v}})$-*space*:

$$b^T Ra > 0$$

where $t = APa$, $l = APb$ and R is the $90°$-rotation matrix. The limit values of the side step approaching from $\mathcal{G}_1$ and $\mathcal{G}_2$ to a point in the boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$ are denoted by w_1 and w_2, respectively. We say that the compatibility is satisfied between the two grains if a side step function, defined in the combined region $\mathcal{G}_1 \cup \mathcal{G}_2$, satisfies the following condition for a given direction field:

if a tool path in $\mathcal{G}_1$ and a tool path in $\mathcal{G}_2$ are "connected" at a point in a boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$ (for example, at the point A in Figure 12), the shifting procedure (Equation 10) in each grain generates the next tool paths that meet at a point in the boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$ (for example, at the point B in the figure).

It is sufficient for this type of compatibility to require that a stream function in $\mathcal{G}_1$ and a stream function in $\mathcal{G}_2$ have the same (limit) value along the boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$. Therefore, the following convergence should be satisfied:

$$\lim_{(r \in \mathcal{G}_1) \to r_o} \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T \cdot \boldsymbol{l} = \lim_{(r \in \mathcal{G}_2) \to r_o} \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T \cdot \boldsymbol{l} \quad \left(= \lim_{r \to r_o} d\Psi(\boldsymbol{l})\right),\ \forall \boldsymbol{r}_o \in \partial\mathcal{G}_1 \cap \partial\mathcal{G}_2.$$

Abbreviated,

$$\frac{w_2}{w_1} = \frac{\boldsymbol{l}^T \cdot (\boldsymbol{n} \times \boldsymbol{t}_2)}{\boldsymbol{l}^T \cdot (\boldsymbol{n} \times \boldsymbol{t}_1)} = \frac{\boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_2}{\boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_1} \quad \text{or "equivalently,"} \quad \left(\frac{1}{w_1}\right) \cdot \boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_1 = \left(\frac{1}{w_2}\right) \cdot \boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_2. \qquad (16)$$

Note that the subscriptions indicate the domains of the limit processes, *e.g.*

$$\boldsymbol{a}_1 \equiv \lim_{(r \in \mathcal{G}_1) \to r_o} [\cos\eta \quad \sin\eta]^T \qquad \text{where } \boldsymbol{r}_o \in \partial\mathcal{G}_1 \cap \partial\mathcal{G}_2.$$

This condition is referred to as the *continuity jump condition* or *type-I compatibility.* We can also make a "flux argument" to reach the same conclusion.

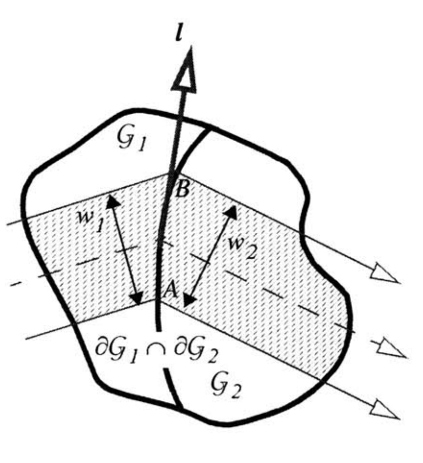
> **Remark:** The 2nd expression in Equation (16) is more appropriate in that it is bounded. By the same token, it is better to treat the reciprocal $1/w(u, v)$ of the side step function as a fundamental quantity rather than the side step function itself but we did not do in that way because length is understood more intuitively.

The Coverage Jump Condition*

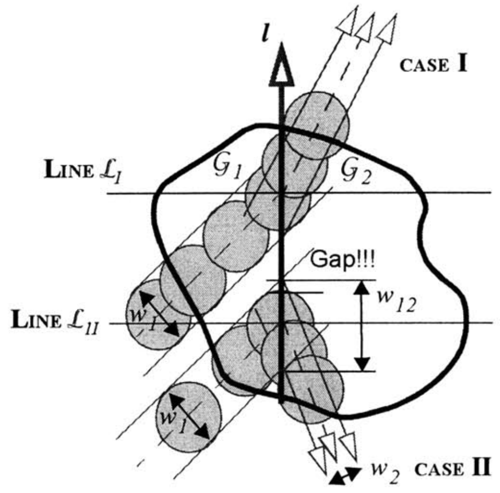
We now ask what the value of the side step w_{12} and the direction angle η_{12} at a point on the boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$ is (if a cutting tool moves from a grain $\mathcal{G}_1$ to another grain $\mathcal{G}_2$ continuously). We regard a surface as its tangent space and a vector field as a parallel one and we require the condition derived under this special circumstance to hold locally for general curved surfaces with non-parallel vector fields—this is what we meant by "flux argument."

We refer to the area bounded by two neighboring tool paths as a *tool path strip*, for example, the hatched area in Figure 12-(*a*). We require the following condition for proper coverage of a surface:

> if we sweep the (approximated flat) surface along a streamline (which is a center line of two neighboring tool paths) with a disk whose diameter is the maximum side step in the streamline, the swept area [*c.f.*, 118, *Minkowski's sausage*] covers the tool path strip.



(a) Continuity Jump Condition



(b) Coverage Jump Condition

Figure 12 Jump Conditions along a Grain Boundary

Let us tentatively assume that the value w_{12} of the side step on the boundary $\partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$ is less than or equal to the maximum (for example, w_1 in Figure 12-*b*) of the two limit values approaching from both grains. There are two cases shown in Figure 12-(*b*). In the first case, when the streamline is deflected through Line $\mathcal{L}_I$ which is orthogonal to the boundary, it is observed that the "sausage" properly covers the tool path strip. However, in the second case, when the streamline is reflected by Line $\mathcal{L}_{II}$, it is observed that there remains a missed gap near the boundary and the coverage condition fails. [This is well known, *c.f.* 42]. To fill this gap, the diameter of the disk need be increased. Doing some plane geometry, we can find the necessary size of the disk. The result is generalized on the surface as planned and we state what we refer to as the *coverage jump condition* or *type-II compatibility* as follows:

$$\left(\frac{1}{w_{12}}\right) = \begin{cases} \left(\frac{1}{w_1}\right) \cdot \boldsymbol{b}^T\boldsymbol{R}\boldsymbol{a}_1 = \left(\frac{1}{w_2}\right) \cdot \boldsymbol{b}^T\boldsymbol{R}\boldsymbol{a}_2 & \text{if } (\boldsymbol{b}^T\boldsymbol{a}_1)\cdot(\boldsymbol{b}^T\boldsymbol{a}_2) < 0 \\ 1/max\{w_1, w_2\} & \text{otherwise} \end{cases} \tag{17}$$

where the symbols are as defined in Equation (16).

Just for convenience which will be evident in the next chapter, we require that the direction angle on the boundary to satisfy the following condition:

$$\boldsymbol{a}_{12} = \begin{cases} \pm\boldsymbol{R}\boldsymbol{b} & \text{if } (\boldsymbol{b}^T\boldsymbol{a}_1)\cdot(\boldsymbol{b}^T\boldsymbol{a}_2) < 0 \\ \boldsymbol{a}_i & \text{otherwise and if } w_i = max\{w_1, w_2\} \end{cases}$$

where $\boldsymbol{a}_{12} \equiv [cos\eta(\boldsymbol{u}_o) \quad sin\eta(\boldsymbol{u}_o)]^T$ for $\boldsymbol{r}(\boldsymbol{u}_o) \in \partial\mathcal{G}_1 \cap \partial\mathcal{G}_2$. For example, in Figure 12-(*b*), the direction angle of Line $\mathcal{L}_{11}$ is assigned on the boundary.

Remark I*: A more complete model must allow multiple values for the side step w_{12} on the boundary. This requires more general mathematical structure than we developed for this thesis work. We do not consider the remaining task further.

Remark II: We can reach the same conclusion with a limit process of the type-0 compatibility, Equation (15). That is, we construct a sequence of C^1 direction fields that converges at a discontinuous direction field and observe what happens on the equation.

Remark III: In the most outside grains or along the boundary of a surface, we need to specify a similar condition. Along a part of the boundary of a boundary-grain $\mathcal{G}$, we could require that

$$\left(\frac{1}{w_b}\right) = \left(\frac{1}{w_{lim}}\right) \cdot \boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}$$

where w_b is the side step at a point on the boundary of the surface and w_{lim} is its limit value at the boundary point. It is possible to neglect this condition if we make a cut along the boundary of a surface in the beginning of any finishing stage.

5. Pseudo-continuity, Matching Pairs and the Pseudo-coverage-condition

Consider Figure 13-(*a*). Intermittent cuts happen because the two grains, $\mathcal{G}_1$ and $\mathcal{G}_2$, are incompatible. The distinction between the intermittent cuts and the "continuous" cuts is important in our modeling the cutting time, our cost function. Unfortunately, the distinction is not that simplistic.

Pseudo-continuity

Consider Figure 13-(*a*) again. It is observed that two tool paths can be connected at the center. This phenomenon is observed better in Case (*b*), where tool paths are shifted slightly. We refer to this phenomenon as the *path matching* or *pseudo-continuity.*

Modifying the continuity jump condition (Equation 16), we capture the path matching by

$$\left(\frac{1}{\Gamma_1}\right) \cdot \left(\frac{1}{w_1}\right) \cdot \boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_1 = \left(\frac{1}{\Gamma_2}\right) \cdot \left(\frac{1}{w_2}\right) \cdot \boldsymbol{b}^T \boldsymbol{R} \boldsymbol{a}_2$$

for mutually **prime** positive integers Γ_1 and Γ_2. In the example, $\Gamma_1 = 5$ and $\Gamma_2 = 7$. The pair, $(1/\Gamma_1, 1/\Gamma_2) \equiv (\gamma_1, \gamma_2)$, of the reciprocals of these two numbers are referred to as a *matching pair.* By changing the conventions for the tangent vector of the boundary as shown in Figure 13-(*b*), we establish a symmetric setting:

$$\gamma_1 \cdot \left(\frac{1}{w_1}\right) \cdot \boldsymbol{b}_1^T \boldsymbol{R} \boldsymbol{a}_1 + \gamma_2 \cdot \left(\frac{1}{w_2}\right) \cdot \boldsymbol{b}_2^T \boldsymbol{R} \boldsymbol{a}_2 = 0$$

where $\boldsymbol{b} = \boldsymbol{b}_1 = -\boldsymbol{b}_2$. (The boundary curve is oriented anti-clockwise viewed from outside.)

A topic of the next section is to model "the amount of intermittent cuts," which represents the inefficiency of the discontinuity. Unfortunately, modeling "the amount of intermittent cuts" is quite complicated if we consider all the possible matching pairs.

We decide to take account of very small portion of matching pairs in our optimization problem. Henceforth, by **path matching** or **pseudo-continuity**, we mean that

$$\left(\frac{\gamma_1}{w_1}\right) \cdot \boldsymbol{b}_1^T \boldsymbol{R} \boldsymbol{a}_1 + \left(\frac{\gamma_2}{w_2}\right) \cdot \boldsymbol{b}_2^T \boldsymbol{R} \boldsymbol{a}_2 = 0, \ \forall \boldsymbol{r} \in (\partial \mathcal{G}_1 \cap \partial \mathcal{G}_2)$$

where the matching pair (γ_1, γ_2) is thought of as a **constant** map on the boundary to the following set:

$$\mathcal{Q}' \equiv \left\{(0, 0), (1, 1), \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right)\right\}$$

i.e. $(\gamma_1, \gamma_2) : (\partial \mathcal{G}_1 \cap \partial \mathcal{G}_2) \to \mathcal{Q}'$. Note that the case, $(\gamma_1, \gamma_2) = (0, 0)$, corresponds to the incompatibility while the case, $(\gamma_1, \gamma_2) = (1, 1)$, represents the "true continuity." This setting allows us to express the whole problem in a more compact way.

Remark: Our choice of the set $\mathcal{Q}'$ of the feasible matching pairs is a "front" part of the following denumerable set:

$$\{(0, 0)\} \cup \{(1, 1)\} \cup \left\{\left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right)\right\} \cup \left\{\left(1, \frac{1}{3}\right), \left(\frac{1}{3}, 1\right)\right\} \cup$$
$$\left\{\left(1, \frac{1}{4}\right), \left(\frac{1}{2}, \frac{1}{3}\right), \left(\frac{1}{3}, \frac{1}{2}\right), \left(\frac{1}{4}, 1\right)\right\} \cup \left\{\left(1, \frac{1}{5}\right), \left(\frac{1}{5}, \frac{1}{1}\right)\right\} \cup \ldots\ldots$$

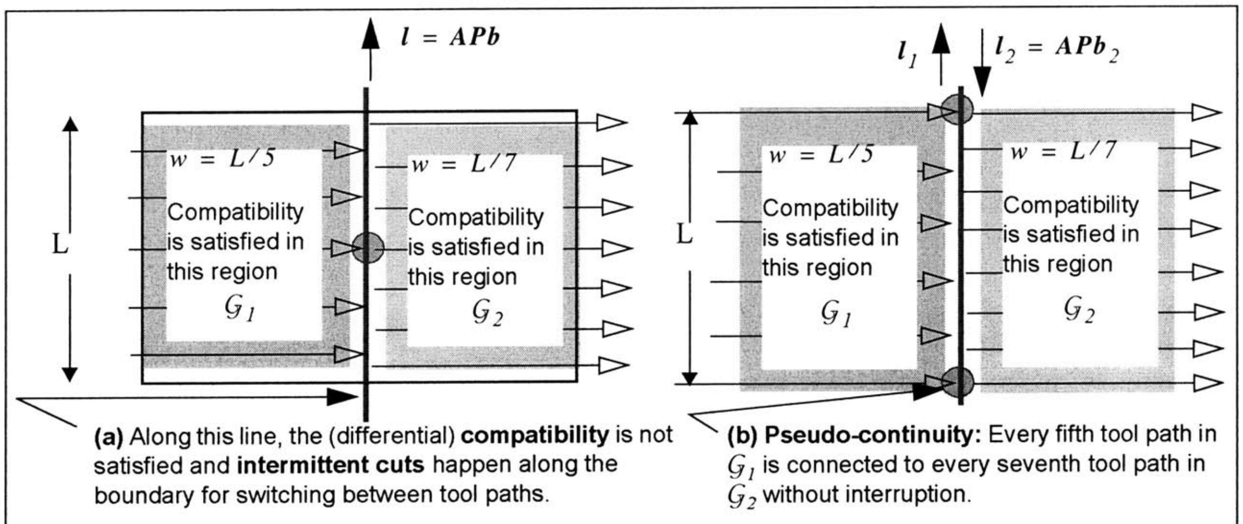

$l = APb$
l_1
$l_2 = APb_2$
L
$w = L/5$
$w = L/7$
Compatibility is satisfied in this region
$\mathcal{G}_1$
$\mathcal{G}_2$
(a) Along this line, the (differential) compatibility is not satisfied and intermittent cuts happen along the boundary for switching between tool paths.
(b) Pseudo-continuity: Every fifth tool path in $\mathcal{G}_1$ is connected to every seventh tool path in $\mathcal{G}_2$ without interruption.

Figure 13 Grains and the Pseudo-continuity

In fact, this set is bijective to the set of rational numbers. As we include more pairs in our consideration, the problem becomes more intractable.

The Pseudo-coverage-condition*

Like the coverage jump condition, we need to specify the value between the two neighboring grains which satisfy the pseudo-continuity. We refer to this jump condition as the *pseudo-coverage-condition*. We do not present the equation because of its similarity to the coverage jump condition.

6. A Path Structure, Path Continua and Grains: Compatibility Summarized

In the previous section, we stated that

the triad $(\vartheta(u, v), \eta(u, v), w(u, v))$ of functions defines tool paths,

which forms our basis approaching the machining problem. In this section, we have seen

the compatibility conditions that relate the direction field $\eta(u, v)$ and the side step function $w(u, v)$.

This subsection constructs an additional mathematical structure that has more descriptive power. The construction is necessary in order to state our optimization problem formally. The compatibility is embedded in the new structure. In a sense, this subsection summarizes the compatibility (and even the optimization problem itself). Therefore, all the terms, which were defined previously, are defined again in this new setting.

Grains will be defined as basic building blocks of tool paths. Compatible grains are collected and form what is referred to as a *path continuum*. We consider the path continuum independent of other path continua in the sense of path generation. A collection of non-overlapping path continua is defined as a *path structure*. Our optimization problem is then reduced to finding an **optimal path structure**. In a grain, the compatibility in its strict sense should hold. In a path continuum, neither tool paths nor participating functions are smooth but a cutting tool can move continuously from one grain to another.

The following series of definitions fulfill the mathematical construction as outlined above. The definitions are repetitive in that the definition for a term contains some replicas of the preceding definitions. Readers are advised not to be distracted by some technical provisos. The motivations and the context for the following definitions have already been given in the previous subsections. The definitions are simply generalizations from the 2-grain situation.

(GEOMETRIC) GRAIN:

A GRAIN is a 4-tuple $(\mathcal{G}, \eta, w, \Psi)$ of a non-empty simply-connected open set $\mathcal{G} \subset \mathcal{S}$ (or $\mathcal{F} \subset \mathcal{P}$) in the designed surface $\mathcal{S}$ (or the parameter space $\mathcal{P}$) together with **smooth** functions, η, w and Ψ, on $\mathcal{F}$ *s.t.*

$$\nabla\Psi \equiv \begin{Bmatrix} \Psi_u \\ \Psi_v \end{Bmatrix} = \frac{\boldsymbol{GPR}}{w}\begin{Bmatrix} cos\eta \\ sin\eta \end{Bmatrix} \neq \boldsymbol{0},\ \forall(\boldsymbol{u} \in \mathcal{F} \subset \mathcal{P})$$

where $\boldsymbol{G}$ is the first fundamental matrix, $\boldsymbol{P}$ is the modal matrix and $\boldsymbol{R}$ is the $90°$-rotation matrix. We refer to the function w as the *side step function*, the function η as the *direction field* and the function Ψ as the *stream function* in $\mathcal{G}$ or $\mathcal{F}$. Usually, we shall denote a grain $(\mathcal{G},\ \eta,\ w,\ \Psi)$ simply by $\mathcal{G}$ with the understanding that we are also considering the underlying 3 functions when we speak of a grain.

An immediate consequence of this definition is that the *differential continuity* (Equation 15)

$$h_1(\eta, u, v)\cdot\frac{\partial w}{\partial u} + h_2(\eta, u, v)\cdot\frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v)\cdot w$$

holds at every point in a grain.The above definition is redundant in that the direction field and the side step function are uniquely determined by the stream function in a grain *s.t.*

$$\begin{Bmatrix} cos\eta \\ sin\eta \end{Bmatrix} = \frac{\boldsymbol{R}^T\boldsymbol{P}^T\nabla\Psi}{\sqrt{(\nabla\Psi)^T\boldsymbol{PP}^T(\nabla\Psi)}} \qquad \text{and} \qquad \left(\frac{1}{w}\right) = \sqrt{(\nabla\Psi)^T\boldsymbol{PP}^T(\nabla\Psi)}\,. \qquad (18)$$

We keep this redundant expression because the direction angle and the side step are more easily grasped while the stream function is a more essential notion.

From grains, a path continuum is constructed as follows:

(GEOMETRIC) PATH CONTINUUM:

A (Geometric Path) Continuum is a 6-tuple $(\mathcal{V},\ \{\mathcal{G}_i\},\ \langle\gamma_{ij}\rangle,\ \eta,\ w,\ \Phi)$ of a simply-connected open set $\mathcal{V}= \boldsymbol{r}(\mathcal{U}) \subset \mathcal{S}$ together with **(I)** a collection of finitely many **mutually-disjoint** open sets $\mathcal{G}_i= \boldsymbol{r}(\mathcal{F}_i) \subset \mathcal{V}$ $(i = 1, \ldots n)$ that properly tessellate the set $\mathcal{V}$ (in that $cl(\mathcal{V}) = \bigcup_i cl(\mathcal{G}_i)$ and $\partial\mathcal{F}_i \cap \partial\mathcal{F}_j$ are connected curves), **(II)** a series $\langle\gamma_{ij}\rangle$ of real numbers γ_{ij} $(i,j = 1, \ldots n)$ and **(III)** the functions, η, w and Φ, on $\mathcal{U} \subset \mathcal{P}$ that satisfy the following conditions:

(1) Grain Condition

For each $\mathcal{G}_i= \boldsymbol{r}(\mathcal{F}_i)$, there is a continuous **monotonic** function α_i *s.t.* $(\mathcal{G}_i, \eta_i, w_i, \Psi_i)$ is a grain, where $\eta_i \equiv \eta|_{\mathcal{F}_i}$ $w_i \equiv w|_{\mathcal{F}_i}$ and $\Psi_i \equiv \alpha_i \circ \Phi|_{\mathcal{F}_i}$. (By $f|_{\mathcal{F}_i}$, we denoted the restriction of a map f to $\mathcal{F}_i$).

(2) Continuity and Monotonicity of the Weaving Function Φ

The function Φ is **continuous** and **monotonic** in $\mathcal{V}$. (By monotonicity, we mean that any level line of Φ divides the continuum $\mathcal{V}$ into two simply connected regions.)

(3) **Compatibility of Paths**

If $i \neq j$, $\partial G_i \cap \partial G_j \neq \varnothing$ and $\partial G_i \cap \partial G_j \cap \partial \mathcal{V} = \varnothing$, then

for all $\boldsymbol{u}_o \in (\partial \mathcal{F}_i \cap \partial \mathcal{F}_j) - \partial(\partial \mathcal{F}_i \cap \partial \mathcal{F}_j)$,

$$\boldsymbol{a}_{ij} = \begin{cases} \pm \boldsymbol{R}\boldsymbol{b}_i & \text{if } (\boldsymbol{b}_i^T \boldsymbol{a}_i^+) \cdot (\boldsymbol{b}_j^T \boldsymbol{a}_j^+) > 0 \\ \boldsymbol{a}_k & \text{otherwise and if } \; w_k = \max\{w_i^+, w_j^+\} \end{cases}$$

and there exists a pair $(\gamma_{ij}, \gamma_{ji}) \in Q'' \equiv \{(1, 1), (1, 1/2), (1/2, 1)\}$ such that

$$(\gamma_{ij}/w_i^+) \cdot \boldsymbol{b}_i^T \boldsymbol{R} \boldsymbol{a}_i^+ + (\gamma_{ji}/w_j^+) \cdot \boldsymbol{b}_j^T \boldsymbol{R} \boldsymbol{a}_j^+ = 0 \text{ and}$$

$$\left(\frac{1}{w_{ij}}\right) = \begin{cases} \left(\frac{\gamma_{ij}}{w_i^+}\right) \cdot \left|\boldsymbol{b}_i^T \boldsymbol{R} \boldsymbol{a}_i^+\right| & \text{if } (\boldsymbol{b}_i^T \boldsymbol{a}_i^+) \cdot (\boldsymbol{b}_j^T \boldsymbol{a}_j^+) > 0 \\ 1/\max\{w_i^+, w_j^+\} & \text{otherwise} \end{cases}$$

Else

$$(\gamma_{ij}, \gamma_{ji}) = (0, 0)$$

where $\boldsymbol{b}_i \; (= -\boldsymbol{b}_j) \in (\underline{\dot{u}}, \underline{\dot{v}})\text{-space}$ such that $\boldsymbol{A}\boldsymbol{P}\boldsymbol{b}_i \in T_{r(\boldsymbol{u}_o)}\mathcal{S}$ is a **unit** tangent vector of $\partial G_i \cap \partial G_j$ with respect to the "natural" orientation of the boundary ∂G_i,

$$\boldsymbol{a}_i^+(\boldsymbol{u}_o) \equiv \lim_{(\boldsymbol{u} \in \mathcal{T}_i) \to \boldsymbol{u}_o} [\cos\eta(\boldsymbol{u}) \;\; \sin\eta(\boldsymbol{u})]^T, \; \boldsymbol{a}_{ij}(\boldsymbol{u}_o) = [\cos\eta(\boldsymbol{u}_o) \;\; \sin\eta(\boldsymbol{u}_o)]^T,$$

$$w_i^+(\boldsymbol{u}_o) \equiv \lim_{(\boldsymbol{u} \in \mathcal{F}_i) \to \boldsymbol{u}_o} w(\boldsymbol{u}) \text{ and } w_{ij}(\boldsymbol{u}_o) \equiv w(\boldsymbol{u}_o).$$

We refer to the function w as the *side step function*, the function η as the *direction field* and the function Φ as the *weaving* function of the path continuum $\mathcal{V}$ or $\mathcal{U}$. A pair $(\gamma_{ij}, \gamma_{ji}) \in Q''$ is referred to as a *matching pair.* Sometimes, we shall denote a path continuum $(\mathcal{V}, \{G_i\}, \langle\gamma_{ij}\rangle, \eta, w, \Phi)$ simply by $\mathcal{V}$ or $(\eta, w, \mathcal{V})$, of course, with the understanding that when we speak of "a continuum $\mathcal{V}$" we are also considering the underlying structure.

An immediate consequence of the grain condition (1) is that

a level set of the **weaving function** Φ is a streamline.

In that sense, the weaving function is a generalization of the stream function while we have more freedom in choosing its smoothness. By the function α_i in the definition, a stream function in a grain can be transformed to the weaving function (in a monotonic way). If we choose a smooth enough weaving function, then in a grain

$$\begin{Bmatrix} \cos\eta \\ \sin\eta \end{Bmatrix} = \frac{\boldsymbol{R}^T\boldsymbol{P}^T\nabla\Phi}{\sqrt{(\nabla\Phi)^T\boldsymbol{P}\boldsymbol{P}^T(\nabla\Phi)}} . \tag{19}$$

Unlike the stream function, the weaving function does not carry the information on the side step. Defining the stream function in a grain and the weaving function in a path continuum is inevitable to capture some global effects such as tool path loops—the direction angle and the side step carries only local information.

Finally, we gather path continua and construct a path structure.

(GEOMETRIC) PATH STRUCTURE:

A (Geometric) Path Structure is a 6-tuple $(\{\mathcal{V}^k\}, \{\mathcal{G}_i^k\}, \langle\gamma_{ij}^k\rangle, \eta, w, \Phi)$ $(i,j = 1,\ldots n(k),$ $k = 1,\ldots,K)$ of a collection $\{\mathcal{V}^k\}$ $(k = 1,\ldots,K)$ of a finitely many mutually-disjoint simply-connected open sets $\mathcal{V}^k = r(\mathcal{U}^k) \subset \mathcal{S}$ that properly partition the surface $\mathcal{S}$ (in that $\mathcal{S} = \bigcup_k cl(\mathcal{V}^k)$) together with **(I)** a collection of open sets $\mathcal{G}_i^k = r(\mathcal{F}_i^k) \subset \mathcal{S}$, **(II)** a series $\langle\gamma_{ij}^k\rangle$ of real numbers γ_{ij}^k and **(III)** the functions, η, w and Φ, on $\mathcal{P}$ *s.t.*

for each k, $(\mathcal{V}^k, \{\mathcal{G}_i^k, i = 1,\ldots n(k)\}, \langle\gamma_{ij}^k, i,j = 1,\ldots n(k)\rangle, \eta^k, w^k, \Phi^k)$ is a path continuum, where $\eta^k \equiv \eta|_{\mathcal{U}^k}$ $w^k \equiv w|_{\mathcal{U}^k}$ and $\Phi^k \equiv \Phi|_{\mathcal{U}^k}$.

We refer to the function w as the *side step function*, the function η as the *direction field* and the function Φ as the *weaving* function of the path structure. A pair $(\gamma_{ij}^k, \gamma_{ji}^k) \in Q''$ is referred to as a *matching pair*. Usually, we shall denote the path structure simply by $\{\mathcal{V}^k\}$, $(\eta, w, \{\mathcal{V}^k\})$ or (η, w), of course with some caution.

Note that the direction field and the side step function are defined in grains, path continua and path structures. Context will clarify the domains for them. Figure 14 shows the following relations of inclusion that summarize what we have defined in this subsection:

$\mathcal{S} \supset \mathcal{V}^k \supset \mathcal{G}_i^k$, in other words, *(A Surface)* $\supset$ *(A Path Continuum)* $\supset$ *(A Grain)*

and *(A Path Structure)* $\ni$ *(A Path Continuum)*.

Remark: Throughout all the definitions, we assumed that the orientation field, $\varphi = \varphi_o(u, v)$ and $\phi = \phi_o(u, v)$, is of differential class C^1 in each path continuum and the surface inverse kinematics is also of differential class C^1.

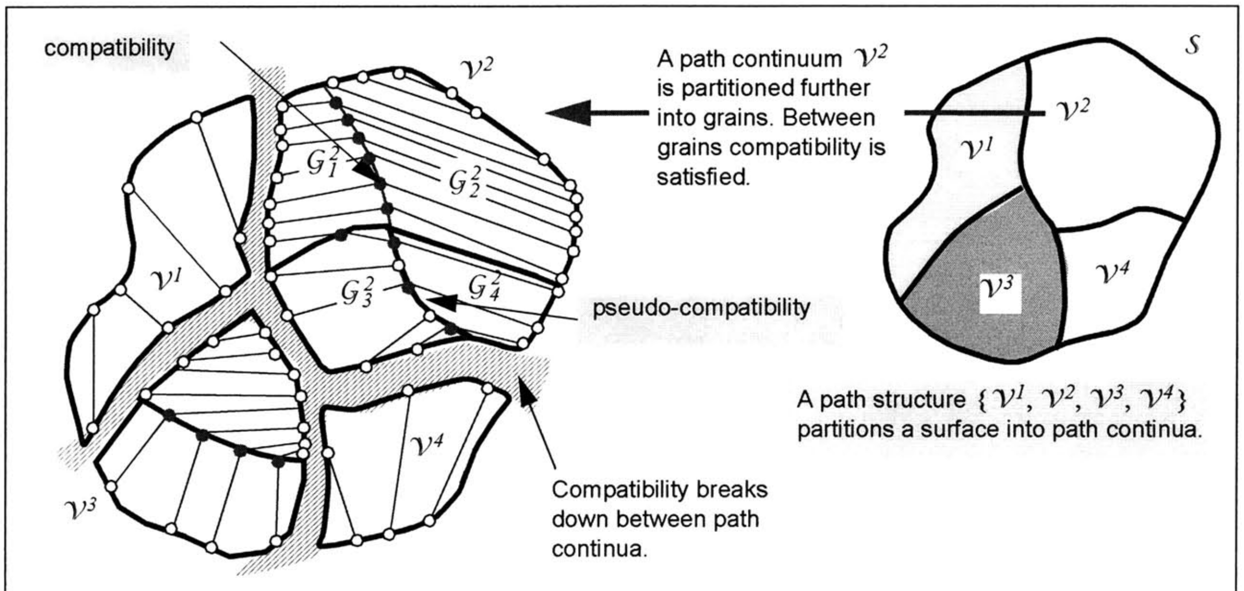

compatibility
$\mathcal{V}^2$
G_1^2
G_2^2
G_3^2
G_4^2
$\mathcal{V}^1$
$\mathcal{V}^3$
$\mathcal{V}^4$
A path continuum $\mathcal{V}^2$ is partitioned further into grains. Between grains compatibility is satisfied.
pseudo-compatibility
Compatibility breaks down between path continua.
S
A path structure $\{\mathcal{V}^1, \mathcal{V}^2, \mathcal{V}^3, \mathcal{V}^4\}$ partitions a surface into path continua.

Figure 14 Grains, Continua and a Path Structure

7. The Kinematic and the Dynamic Path Structure

The geometric path structure takes account of only the relation between the side step function $w(u, v)$ and the direction field $\eta(u, v)$. In this subsection, we include the speed $\vartheta(u, v)$ in our consideration and finish our preparing the formal statement of the problem.

(KIEMATIC) PATH STRUCTURE:

A (KINEMATIC) PATH STRUCTURE is simply a geometric path structure together with a function $\vartheta : \mathcal{P} \to \mathcal{R}$ that is continuous on each grain of the geometric path structure. The function ϑ is called the speed of the kinematic path structure. With a similar manner, we define a **KINEMATIC PATH CONTINUUM** and **KINEMATIC GRAINS.**

(DYNAMIC) PATH STRUCTURE:

A (DYNAMIC) PATH STRUCTURE is a kinematic path structure that satisfies the following conditions:

(1) the speed $\vartheta(u, v)$ is continuous in each path continuum of the kinematic path structure

(2) $\vartheta(\boldsymbol{u}_0) = 0$ if the direction field $\eta(\boldsymbol{u})$ is discontinuous at $\boldsymbol{u}_0 \in \mathcal{P}$.

With a similar manner, we define a **DYNAMIC PATH CONTINUUM** and **DYNAMIC GRAINS.**

In our most general setting, we seek a **dynamic path structure**. If we ignore the limits on the acceleration (*See* Subsection 2.5.2 Actuator Limits), we ask what is the optimal **kinematic path structure**. By imposing a stricter constraint, *e.g.* $\vartheta(u, v) = const$, we may seek the "shortest" path; in this case, we ask what **geometric path structure** is optimal. All the definitions are simply a refinement of the following loose statement: "the triad $(\vartheta(u, v), \eta(u, v), w(u, v))$ of piecewise continuous functions defines tool paths."

8. Summary

The compatibility equation is based on our hypothesis that the surface is covered by a series of regions in which tool paths are reasonably continuous or exhibit "enough regularity." The resulting Equation (15)

$$h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v) \cdot w$$

captures what we meant by regularity in a local (differential) sense. The "flux argument" in mathematical physics leads us to the same conclusion. This differential equation breaks down for non-smooth paths and we postulate another regularity along curves in the surface, which results in the jump conditions (including the pseudo-continuity). Finally, we defined a mathematical structure for stating our optimization problem formally. Our optimization problem is now to find the optimal geometric/kinematic/ dynamic path structure.

The compatibility is regarded as an equality constraint in our optimization problem, which declares the relation between the direction field and the side step function.

2.4 Cutting Time: The Cost Functional

Until now, we showed how to define tool paths in our framework and introduced an equality constraint of our optimization problem. Now, we define the cost functional in our optimization problem.

We make the argument that cutting time is today the major measure of machining performance and take the cutting time as the cost functional in our optimization problem. In applications such as die-mold machining, where time-to-market is the driving factor, other costs pale in comparison to the economic benefits of producing a finished component rapidly.

The time T_c that it takes to cut a surface can be divided into two parts, *effective cutting time* T_e and *non-effective cutting time* T_{ne}:

$$T_c = T_e + T_{ne}.$$

Effective cutting time is the period during which the cutting tool stays in contact with the surface of the part and removes material. The tool must touch the surface tangentially not to over-cut or under-cut the workpiece. While maintaining this tangential contact, the machine tool executes NC commands such as *G01*, *G02* and *G03*. During the rest of the time, the machine engages in activities not related directly to cutting. We refer to this period as *non-effective cutting time*. A typical non-effective activity is rapid motion between cuts in the *G00* mode.

1. Effective Cutting Time

Effective cutting time is modeled in the following way:

$$T_e = \sum_i \int_{\mathcal{C}_i} \frac{ds}{\vartheta_i(s)} = \sum_i \int_{\mathcal{C}_i} \frac{w_i(s)\, ds}{w_i(s)\, \vartheta_i(s)} \approx \iint_{\mathcal{P}} \frac{\overbrace{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot du dv}}{w(u, v) \cdot \vartheta(u, v)} \tag{20}$$

where the dummy variable s is the arc length of tool paths; $\mathcal{C}_i$ is the i^{th} tool path whose parameterization is $\boldsymbol{r}(u_i(s), v_i(s))$; $\mathcal{P}$ is the parameter space; $\vartheta_i(s) \equiv \vartheta(u_i(s), v_i(s))$ is the cutting speed along the i^{th} tool path; $w_i(s) \equiv w(u_i(s), v_i(s))$; the side step $w(u, v)$ and the speed $\vartheta(u, v)$ are the functions on $\mathcal{P}$ as explained in the preceding sections. Note that the sum for $w_i(s) \cdot ds$ corresponds to the surface area approximately and that surface area is defined as $\iint \|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot dudv$ in differential geometry.

Of course, the integration domain must include enough tool paths to make the approximation be valid. As in continuum mechanics, the domains we look at must be of much larger scale than the features.

2. Non-effective Cutting Time

The rest of cutting time is referred to as non-effective cutting time. Non-effective cutting time is implied by every break in the cut because the tool must be moved from the end of the previous cut to the beginning of the next cut. Typically, the tool is lifted, moved and lowered again for the new path at the end of each tool path. Therefore,

non-effective cutting time is proportional mainly to the **number of tool paths** in the domain.

This subsection is about how to count tool paths in a surface given a path structure. We consider a simple case first and proceed to the general case.

The Simplest Model

If there are **no tool path loops** and **no intermittent cuts** in the surface S, one of the simplest model for the non-effective cutting time is

$$T_{ne} = \frac{1}{2} \cdot \oint_{\partial S} \left| \left(\frac{\tau_o}{w}\right) \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} \right|$$

where τ_o is a constant of the proportionality and the integration is done along the boundary curve ∂S. A way to understand this equation is that $\oint |d\Psi|$ is the "number of tool paths" inside the surface, where Ψ is the stream function as defined in Equation (11).

Variations in each Movement and The Non-effective Penalty

In more detailed models, τ_o becomes a function in the following form:

$$\tau_o = \tau_o(\vartheta, \eta, w, u, v) > 0 .$$

This function τ_o is treated as a known function in our optimization problem. It characterizes the time it takes for a cutter to move from an end of a previous tool path to the next path. (The equation of motion, as presented in Subsection 2.2.6, can be used to determine this function.) By postulating the existence of the function τ_o, we implicitly assume the regularity in the non-effective activities. We refer to this function τ_o as the *non-effective penalty* (*NEP*) term (or function) of the problem.

We have lots of freedom in choosing the NEP function $\tau_o(\vartheta, \eta, w, u, v)$ unless the definition hurts reality. However, we specify the minimum requirement that the NEP function τ_o must meet.

First, we require that

(sub-linear growth) $\quad 0 \le \left(\frac{\partial \tau_o}{\partial w}\right) \le \left(\frac{\tau_o}{w}\right)$ in other words, $\frac{\partial}{\partial w}\left(\frac{\tau_o}{w}\right) \le 0$.

Remark: Of course, we can devise control schemes that break this "rule," quite artificially. However, they are typically inefficient. Considering the initialization cost such as the one related to the acceleration limit, this sub-linear growth condition is justified. Also note that we are considering very short distance around 1^{mm} and the initialization cost is quite high.

Readers will be pretty assured once they establish a physics-based model for the NEP term by themselves.

The sub-linear growth condition produces the following intuitive consequence:

(WIDEST SIDE STEP PRINCIPLE)

for a given direction field, cutting time is reduced if wider cuts are made, in other words, if the side step function w increases. Therefore, to minimize the cutting time, we maximize the side step (not violating the constraints of the problem).

Second, if we pursue the **dynamic path structure**, which is the most general case, we require the following condition:

(DYNAMIC CONDITION ON THE **NEP**) $$\left(\frac{\partial\tau_o}{\partial\vartheta}\right) \geq 0$$

related to the dependency on the "local speed" ϑ. This condition basically asserts that the faster an object moves, the longer time it takes to stop it.

Besides, if we pursue the **kinematic/geometric path structure** (as opposed to the dynamic path structure), we restrict the dependency further in the following way:

(KINEMATIC CONDITION ON THE **NEP**) $$\left(\frac{\partial\tau_o}{\partial\vartheta}\right) = 0 \quad .$$

Loosely speaking, this condition prevents the kinematic structure from behaving irrationally due to the infinite acceleration, which is not prohibited in the kinematic structure. In a geometric path structure, the condition is trivial because the speed is not defined there at the first place.

REMARK*: The argument on the necessity of the kinematic condition on the NEP term is convoluted. We only sketch the way to reach the condition. If $(\partial\tau_o/\partial\vartheta) \neq 0$ in the kinematic structure, it can be shown that the speed vanishes ($\vartheta = 0$) at the boundary of its path continuum for the optimal speed. However, the very reason why we define the function such that $(\partial\tau_o/\partial\vartheta) \neq 0$ is to capture the acceleration effect which is ignored in the kinematic path structure. If the speed vanishes, the inertia effect is lost. Therefore, we cannot account for the (local) inertia effect by defining τ_o such that $(\partial\tau_o/\partial\vartheta) \neq 0$. Instead, we do in an averaged sense ignoring the local effect. Suppose we have established a model, in a dynamic path structure, for the NEP term τ_o using a function τ_o^D, namely, $\tau_o = \tau_o^D(\vartheta, \eta, w, u, v)$. Then, in a kinematic path structure, we set the function in the following form:

$$\tau_o = \tau_o^D(\bar{V}, \eta, w, u, v)$$

where $\bar{V}$ is the speed averaged in a certain sense.

NON-EFFECTIVE CUTTING TIME FROM A PATH STRUCTURE

In reality, tool paths have loops (vortices) and intermittent cuts (sinks/sources). A path structure, as defined in Subsection 2.3.6 and 2.3.7, captures the intermittent cuts and loops and it partitions a surface into path continua. The boundary of each continuum is the place where the intermittent cuts are allowed to occur because we did not impose any continuity of tool paths across it. If pseudo-continuity occurs in a continuum, some grain boundaries are also the places where intermittent cuts are made. At those boundaries, the cutting tool moves across the streamlines as shown in Figure 13 and Figure 14.

In a general sense, we write:

$$T_{ne}[\vartheta, \eta, w, \{\mathcal{V}^k\}] = \sum_k T^o_{ne}[\vartheta^k, \eta^k, w^k, \mathcal{V}^k]$$

where $(\vartheta, \eta, w, \{\mathcal{V}^k\})$ is a path structure, η is the direction field, w is the side step function, ϑ is the cutting speed, $(\vartheta^k, \eta^k, w^k, \mathcal{V}^k)$ is the k^{th} continuum, and T^o_{ne} is the non-effective cutting time for each continuum. In our convention, brackets are used for functionals. The above statement is merely an assertion that

> we will generate tool paths in each continuum independently or separately (of course, after a path structure is given).

A simple model for the non-effective cutting time for the k^{th} continuum is

$$T^o_{ne}[\vartheta^k, \eta^k, w^k, \mathcal{V}^k] = \frac{1}{2} \cdot \left\{ \sum_{i \neq j}\sum \int_{\begin{pmatrix} \partial G^k_i \cap \partial G^k_j \\ -\partial \mathcal{V}^k \end{pmatrix}} \left(\frac{\tau_o \cdot (1 - \gamma^k_{ij})}{w^{k+}_i} \right) \cdot \left| (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} \right| + \oint_{\partial \mathcal{V}^k} \left(\frac{\tau_o}{w^{k+}} \right) \cdot \left| (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} \right| + \oint_{\partial \mathcal{V}^k} \frac{\|d\boldsymbol{r}\|}{V_o} \right\}$$

$$\sim \tau_o \cdot (\mathit{Number\ of\ Tool\ Paths}) + \frac{(\mathit{Length\ of\ the\ Contiuum\ Boundaries})}{2V_o} \tag{21}$$

where G^k_i is the ith grain of the k^{th} continuum, $(\gamma^k_{ij}, \gamma^k_{ji})$ is a matching pair and V_o is a given function on $\partial \mathcal{V}^k$ that characterizes the allowable speed along the continuum boundary. The first two terms count tool paths in a continuum and properly weight the number with the time τ_o per a non-effective movement. Especially, the first term deals with the pseudo-continuity. The last term has a different origin, which is largely proportional to the length of the continuum boundary because we make a cut along the continuum boundaries in the beginning of any finish machining. By this, a boundary effect is neglected (*see* **REMARK III** in Subsection 2.3.4). This term also prevents the path structure from being "granulated" or "layered," in other words, partitioning the surface into too small or narrow continua and violating our regularity (or "continuum") hypothesis.

Homogeneous Non-Effective Penalty Model

The NEP function τ_o captures a "local" variations that might exist in a system. The sub-linear growth and dynamic/kinematic conditions are the minimum requirements that must be imposed on it to ensure "reasonable" behavior of our resultant tool paths. Partly because there are typically "many" tool paths in a surface and partly for simplicity, we sometimes ignore the local effect taking a certain average. We refer to the path optimization problem in which the local variations are neglected as a *homogeneous NEP model*. That is, in a homogeneous NEP model, we dictate:

(Homogeneity Condition) $$\tau_o = Const\ .$$

It is easily verified that this model satisfies all the minimum requirements previously mentioned. In this model, non-effective cutting time (as defined in Equation 21) is proportional to the number of tool paths.

Procedural Estimation

The greedy approach will be introduced as an approximate solution in a subsequent chapter. The greedy approach is to generate tool paths while pursuing directions that minimize a local measure neglecting non-effective cutting time. For the greedy paths, we take a two-phase approach to evaluate the non-effective cutting time: we account for the non-effective cutting time in a procedural way after the generation of the effective movements of a cutting tool. We actually construct the non-effective movements and evaluate the non-effective cutting time directly instead of using the formula (Equation 21).

3. Summary

Cutting time is the cost functional in our optimization problem. While a machine tool is engaged in an effective activity, the cutting tool removes material within tolerance. The effective cutting time is evaluated through the integration of the reciprocal of area removal rate over designed surface. Non-effective movements occur at the boundary of path continua around which the cutting tool should be decelerated and accelerated again for the next path. The non-effective cutting time accounts for how many interruptions occur during the finishing period. It is largely proportional to the number of tool paths in the surface.

We minimize the functional $T_c = T_e + T_{ne}$ as above explained by finding an optimal kinematic/dynamic path structure (or, roughly speaking, the functions $\vartheta(u, v)$, $\eta(u, v)$ and $w(u, v)$) which are subject to the compatibility equations, and some inequality constraints which follow.

2.5 Inequality Constraints: Cusp Height Limits, Actuator Limits, Collision Avoidance, Cutting Force Limits, etc.

The problem as it now stands is unbounded. We now consider the constraints that make the machining problem a bounded optimization problem. They include an accuracy requirement and physical limitations imposed on the milling process.

1. The Cusp Height Limit: Geometric Tolerance or Accuracy

In reality, every engineering product must be designed with some room for dimensional errors, considering the limitations of a particular manufacturing process. Loose tolerance may hurt the functionality of the product while demanding high accuracy results in high manufacturing cost. We show how to accommodate this accuracy requirement in our framework.

Cusps, their Heights and Limits

It is a fundamental limitation in the free-form surface milling process that *cusps* (or *scallops*) are unavoidable as shown in Figure 15-(*a*). This is because curvatures of a surface differ from the curvature of a cutting tool. The *cusp height h* as shown in Figure 15-(*a*) determines how close the machined surface is to the designed surface. We ensure that the cusp heights are kept below a certain limit value. This maximum allowable cusp height is referred to as the *cusp-height-limit* and it is denoted by h_o. We regard this requirement, $h \le h_o$, as an inequality constraint. In this thesis, we do not ask what the appropriate value for the cusp-height-limit is; the cusp-height-limit h_o is fixed as an input in our optimization problem.

The Side-step-limit

The better conforms a cutting tool to the "local" shape of a surface at a point, the farther two adjacent tool positions can be apart (along the normal section perpendicular to a streamline at the point). The conformity is best observed in the normal plane of the streamline that passes the given point, which is shown in Figure 15-(*b*). When the tool positions are arranged at a particular interval along the normal section, the cusp height reaches the specified cusp-height-limit h_o. We refer to the length w_o of the interval as the *side-step-limit*. Instead of requiring $h \le h_o$, we now require that $w \le w_o$ so that the machined surface can be within a specified tolerance.†

Approximation of the Side-step-limit

Deciding the value of the side-step-limit at a point involves us in solving a highly non-linear equation. Simplifying the problem, the normal section is approximated by a parabola or a circle. Hence, it is only the **normal curvature** of the surface in the direction of $\boldsymbol{n} \times \boldsymbol{t}$ (or $\eta + 90^\circ$-direction) that determines the

† Our notion of side step w is close to the meaning of cross-feed [52], step-over [33], side-step [18], CC path interval [22 and 28] and so on. Our notion of side-step-limit w_o is close to the meaning of machined strip width [53], allowable side-step [18] and so on.

approximated normal section, (a parabola or a circle). Thanks to the parabolic (or circular) approximation, given a point on the surface, the side-step-limit is treated as a known function of the direction ***t*** or η of the streamline, *i.e.*

$$w_o = w_o(\eta, u, v). \tag{22}$$

REMARK: Clearly, the side-step-limit could be written as $w_o(\eta, u, v, h_o, R)$ because it does in fact depend on the cusp-height-limit and the tool radius. We choose not to write in that way because h_o and R can be seen as given constants as opposed to variables for a particular instance of our problem.

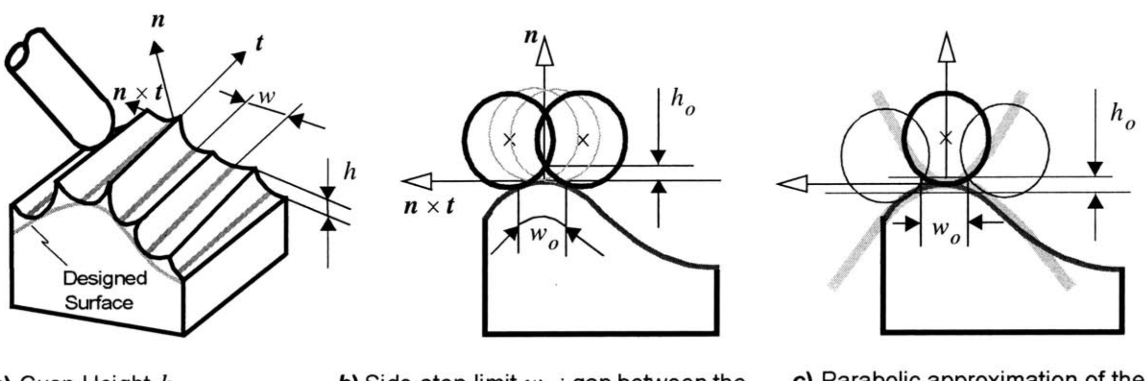
FORMULA FOR THE SIDE-STEP-LIMIT

In practice, additional simplification is made. For example, we paraphrase the following approximation for the side-step-limit that was originally introduced by Lin and Koren [22, *c.f.* 54, 18, 52 and 28]:

$$w_o(\eta, u, v) \approx \sqrt{\frac{8h_o}{\kappa_b - \kappa_s}} = \sqrt{\frac{8h_o}{(\kappa_b - \kappa_1)\cdot sin^2\{\eta - \eta_o(u, v)\} + (\kappa_b - \kappa_2)\cdot cos^2\{\eta - \eta_o(u, v)\}}} \tag{23}$$

where h_o is the given cusp height limit, $\kappa_b(\equiv 1/R)$ is the curvature of the cutting tool, κ_s is the *normal curvature* of the surface in the direction of $\boldsymbol{n}\times\boldsymbol{t}$ and κ_1/κ_2 is the maximum/minimum principal normal curvature at a given point. Note that $\kappa_s \equiv \kappa_{r(\boldsymbol{u})}(\boldsymbol{n}\times\boldsymbol{t}) = \kappa_1 sin^2\underset{\sim}{\eta} + \kappa_2 cos^2\underset{\sim}{\eta}$ according to *Euler's formula* [115], where $\underset{\sim}{\eta}$ $(= \eta - \eta_o(u, v))$ is the principal direction angle of the vector ***t*** as defined in Subsection 2.2.3. In this case, both the normal section and the ball are approximated by parabolas. Furthermore, the gap is measured as a Euclidean distance (instead of being measured along the normal section) and the two parabolas are centered at the same point as shown in Figure 15-(*c*). The formula diverges when the curvature κ_s approaches the curvature κ_b of the ball. We construct a more comprehensive formula for the side-step-limit and it is given in APPENDIX C.

$$h \le h_o \quad \Leftrightarrow \quad w(u, v) \le w_o(\eta, u, v)$$



a) Cusp Height *h* ***b)*** Side-step-limit w_o: gap between the two paths along a normal section. ***c)*** Parabolic approximation of the normal section and the ball.

Figure 15 Cusps and Coverage corresponding a cusp-height-limit

Summary

We treat the following condition as an inequality constraint of the optimization problem under consideration:

$$w(u, v) \leq w_o(\eta(u, v), u, v) \tag{24}$$

where $w(u, v)$ is the **unknown** side step function and $w_o(\eta, u, v)$ is the side-step-limit, which is a **known function**.

Remark I: Consider Figure 15-(*c*), which explains Lin and Koren's approximation. This shows that the side-step-limit can be thought of as a width of a cut by an individual tool path.

Remark II*: Recall the shifting procedure, as defined in Subsection 2.2.7. The "gap" between two neighboring paths was measured as the harmonic mean of the side step function along the orthogonal transverse curve $\boldsymbol{r}_{\perp}(\alpha)$. For the side-step-limit, the interval is measured along the normal section. The justification is that the orthogonal transverse curve and the normal section share their tangent direction—they agree with each other upto their first order approximation. Their second order approximations differ by the *geodesic curvature*† of the orthogonal transverse curve. (Considering this, we can conclude that the above condition tends to be conservative.) Of course, we can refine the formula for the side-step-limit including the derivatives of the direction field, *i.e.*

$$w_o = w_o(\eta, \eta_u, \eta_v, u, v).$$

It is possible to include even higher order derivatives to refine it. The first order approximation valid as long as the side step w is small enough with respect to the geodesic curvature $\kappa_{\perp}^{g}$ of the orthogonal transverse curve $\boldsymbol{r}_{\perp}(\alpha)$. Specifically, the condition is

$$w \cdot \kappa_{\perp}^{g} << 1 .$$

2. Actuator Limits: a Physical Limitation

Recent advances in spindle technology opened the door to the area of high-speed machining. Unfortunately, actuation technology has not kept pace with the spindle technology—often, machine tools are equipped with the actuators that exhibit relatively-poor performance in comparison with their powerful spindles. The result is that the motion actuation turns out to be the bottleneck in cutting performance of high speed machining. In this subsection, we introduce inequality constraints that capture the performance limits of actuators.

† Geodesic curvature of a surface curve is the curvature of its projection curve onto the tangent plane of the surface [*e.g. see* 116].

General Actuator Limits

Every motor has a torque-speed characteristic curve that determines its maximum torque in terms of its velocity. As shown in Figure 16-(*a*), the feasible combinations of torque τ_i and speed ω_i of the ith motor are in some empirical set, namely,

$$(\tau_i, \omega_i) \in \mathcal{A}_i \subset \mathcal{R}^2 .$$

It is general enough to consider the feasibility set $\mathcal{A}_i$ a compact set. We explain below how to express the torque and the speed of the motors in terms of the direction field $\eta(u, v)$ and the speed $\vartheta(u, v)$ of a path structure.

Motor (or Joint) Velocities and the Kinematics

Recall the *surface inverse kinematics* $\boldsymbol{f}$ (Equation 7) and the *restricted surface inverse kinematics* f^o. The restricted surface inverse kinematics maps surface parameters (u, v) to motor angles (or displacements) θ_i, *i.e.*

$$\theta_i = f_i(u, v, \varphi_o(u, v), \phi_o(u, v)) = f_i^o(u, v) \qquad (i = 1, \ldots, N)$$

where N is the number of actuators of the machine tool under consideration. The surface inverse kinematics affects the performance of a machine tool, in effect, warping the space of the motor displacements. The *Jacobian matrix* of the restricted surface inverse kinematics expresses this effect locally, creating a linear transformation. If the Jacobian matrix is represented as $\boldsymbol{C} = [c_{ij}]$, then by definition, $c_{i1} \equiv \partial f_i^o / \partial u$ and $c_{i2} \equiv \partial f_i^o / \partial v$. At every point on the surface, this matrix maps the parameter velocity $\dot{\boldsymbol{u}}$ to the motor velocity ω_i:

$$\omega_i = c_{i1} \cdot \dot{u} + c_{i2} \cdot \dot{v} \qquad \left(= \frac{d\theta_i}{dt}\right) . \tag{25}$$

Through a further transformation, ($\dot{\boldsymbol{u}} = \vartheta \cdot \boldsymbol{h}(\eta, u, v)$ as shown in Equation 6), the motor velocity at a point $(u, v) \in \mathcal{P}$ can be expressed in terms of the speed and the direction angle as follows:

$$\omega_i(u, v) \equiv \vartheta(u, v) \cdot \{c_{i1}(u, v) \cdot h_1(\eta(u, v), u, v) + c_{i2}(u, v) \cdot h_2(\eta(u, v), u, v)\} . \tag{26}$$

Motor (or Joint) Torques and the Equation of Motion

Through one more differentiation, the joint accelerations $d^2\theta_i / dt^2$ can be written in terms of the direction angle, the speed and their partial derivatives. If we insert these joint accelerations into the **equation of motion** (Equation 8) together with the joint velocities, we have the joint torques in the following form of equations:

$$\tau_i = \tau_i(\eta_u, \eta_v, \eta, \vartheta_u, \vartheta_v, \vartheta, u, v) . \tag{27}$$

Remark: Generally, motor torques need to overcome (1) the inertia force and (2) the cutting force to track a given path. We make the argument that the cutting force is quite low in our case. We are considering a finishing stage using a high speed machine tool. The objective of high speed machining is to move as fast as possible while taking light cuts. In addition, the cutting force is low in a finishing stage because a relatively-small amount of material is removed.

General Actuator Limits in terms of the Speed and the Direction

Thanks to Equation (26 and 27), the actuator limits, $(\tau_i, \omega_i) \in \mathcal{A}_i \subset \mathcal{R}^2$, can be expressed in terms of ϑ, η and their partial derivatives where the direction field η and the speed ϑ are smooth. Therefore, the general actuator limits can be accommodated in our framework. In this general setting, the unknown of the optimization problem is the **dynamic path structure**. Of course, solving the optimal machining path problem is quite complicated under these general actuator limits. On the other hand, the following speed limits result in a very convenient form. If we impose only the speed limits, we regard the **kinematic path structure** as the unknown of our machining problem.

Speed Limits

The speed limit in Figure 16-(*a*) is shown as the right-extreme of the torque-speed curve. We can say then that: $|\omega_i| \leq \omega_o^i$ where ω_o^i is the speed limit of the ith motor ($i = 1...N$). If we now map these actuator velocity constraints back on to the $(\dot{u}, \dot{v})$-space using the *Jacobian matrix* $\boldsymbol{C}$ (Equation 25, *c.f.* 26), we will see simple half-planes of feasibility:

$$|\omega_i| = |c_{i1}(u, v) \cdot \dot{u}(u, v) + c_{i2}(u, v) \cdot \dot{v}(u, v)| \leq \omega_o^i \quad . \tag{28}$$

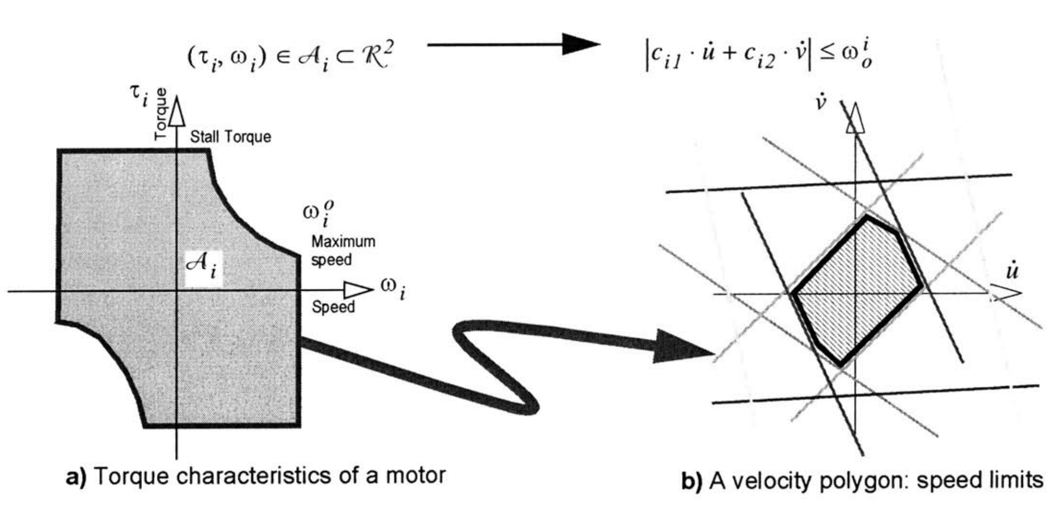

$(\tau_i, \omega_i) \in \mathcal{A}_i \subset \mathcal{R}^2$
$|c_{i1} \cdot \dot{u} + c_{i2} \cdot \dot{v}| \leq \omega_o^i$
τ_i
Torque
Stall Torque
ω_i^o
Maximum speed
$\mathcal{A}_i$
ω_i
Speed
$\dot{v}$
$\dot{u}$
a) Torque characteristics of a motor
b) A velocity polygon: speed limits

Figure 16 Actuator limits and a Velocity polygon

The intersection of the inequalities forms a **symmetric polygon** in $(\dot{u}, \dot{v})$-space as shown in Figure 16-(b). By additional **linear** transformations, ($\dot{u} = \boldsymbol{P}\dot{\underline{u}} = \underset{\sim}{\boldsymbol{P}}\dot{\underset{\sim}{u}}$ as given in Equation 4), the above equation also forms symmetric polygons in $(\dot{\underline{u}}, \dot{\underline{v}})$-space and $(\dot{\underset{\sim}{u}}, \dot{\underset{\sim}{v}})$-space. We refer to those symmetric polygons as *velocity polygons*.[†]

3. Other Constraints and Simplifications:

Collision Avoidance, Cutting Force, Tool Wear, Acceleration, Torque, Band Width, etc.

There are several other constraints that reflect the reality of machining. There are cutting force limits which arise from tool deflection limits. Like the actuator speed limits discussed in the previous subsection, the force limiting constraints also form a feasible area in the $(\dot{u}, \dot{v})$-space [59 *c.f.* 61]. In the context of high speed surface machining, we ignore them on the grounds that cutting forces are usually low. Other constraints include structural stiffness limits, system band-width, tool wear considerations, *etc.* We ignore these considerations in our initial analysis. We assume that the structure is rigid, and we assume that band-width limits can be captured to some extent with actuator limits. And in the outset, we assumed that the geometric feasibility is satisfied by defining the feasible orientation field.

There is a further simplification, which will be made in practice. The acceleration (or torque) limits result in a quite complicated form. We mention here some ways to ignore them, even though solving the problem with the acceleration limits is not entirely hopeless. We can respect the inertia effect indirectly through the consideration on the non-effective cutting time. In this case, the function τ_o should be "well" defined. In the greedy approach, tool paths are generated in two phases. We ignore acceleration limits in the first analysis, where we only consider the speed limits—we account for acceleration limits in the second phase by smoothing the tool path locally, removing small circular motions and reducing the speed. Another way is to set the speed limits ω_o^i conservatively depending on the designed surface. Some of those topics are interesting open questions. We do not cover them in this thesis.

4. Summary

We specify inequality constraints in the machining problem. They are expressed in terms of the participating functions and their partial derivatives about the parameters, u and v, in our framework. They capture the geometric accuracy requirement and the physical limitations imposed on the process. Among many, we will consider the **cusp-height-limit** and the **actuator speed limits** in detail.

[†] A similar concept is found in the context of robot manipulability [137, *c.f.* 138, 139 and 140].

2.6 Summary: Formulation of the Time Optimal Machining Path Problem

We have now derived the final form of the optimization problem. Specifically, we have formulated the objective function and the constraints in a form that is, at least in theory, amenable to numerical analysis. The key elements in our model, which make this formulation compact, are: (1) our recognition that families of tool paths can be captured as fields, which are defined over parameter space $\mathcal{P}$ as $\vartheta(u, v)$ and $\eta(u, v)$, (2) our decision to model the distance between tool paths as a function $w(u, v)$, (3) our ability to capture the actuator speed limits as geometric constraints on the tangent space, (4) the discovery of the compatibility condition in the context of tool path generation and (5) the use of existing results from path planning literature to express the cusp height limit as a function of tool radius and surface geometry parameters.

The Formulated Problem in its Simplest Form

With the quantities and the notions above introduced, we present the formulated optimization problem in its simplest form. Loosing some accuracy of the statement, we attempt to present the key elements in a "familiar" form, without mentioning the path structure and the jump conditions.

Find the (piece-wise continuous and piece-wise smooth) functions of the parameters, $(u, v) \in \mathcal{P} \subset \mathcal{R}^2$:

Unknowns: cutting speed $\vartheta(u, v)$, direction angle $\eta(u, v)$, side step $w(u, v)$

(and the regions $\mathcal{V}^k \subset S$ $(k = 1, \ldots, K)$ that partitions the designed surface)

which minimize the machining time (*cost functional*):

$$T_c = \iint_{\mathcal{P}} \frac{\| r_u \times r_v \| \, du dv}{\vartheta(u, v) \cdot w(u, v)} + T_{ne}[\vartheta, \eta, w, \{\mathcal{V}^k\}]$$

subject to:

Compatibility: $h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v) \cdot w$ (*a.e.*)

Cusp Height Limit: $w(u, v) \le w_o(\eta(u, v), u, v)$

Actuator Speed Limits: $\vartheta(u, v) \cdot \left| \sum_j c_{ij}(u, v) \cdot h_j(\eta(u, v), u, v) \right| \le \omega_o^i, \; (i = 1, \ldots, N, j = 1, 2)$

where w_o, h_1, h_2, h_3 and c_{ij} are usual functions which are known. ω_o^i are positive constants. $T_{ne}[\ldots]$ is a *functional* by which we mean that we can decide its value at least *procedurally* with the functions in the square brackets being specified. We denote partial derivatives by subscripts, *e.g.* $\eta_u = \partial\eta / \partial u$.

Problem Statement in terms of the Path Structure

Of course, the accurate statement is:

Find the **kinematic path structure** $(\{\mathcal{V}^k\}, \{\mathcal{G}_i^k\}, \langle\gamma_{ij}^k\rangle, \vartheta, \eta, w, \Phi)$

which minimize the machining time *(cost functional)*:

$$T_c = \iint_{\mathcal{P}} \frac{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \, dudv}{\vartheta(u, v) \cdot w(u, v)} + \sum_k T_{ne}^o[\vartheta^k, \eta^k, w^k, \mathcal{V}^k]$$

subject to:

Cusp Height Limit: $w(u, v) \le w_o(\eta(u, v), u, v)$

Actuator Speed Limits: $\vartheta(u, v) \cdot \left|\sum_j c_{ij}(u, v) \cdot h_j(\eta(u, v), u, v)\right| \le \omega_o^i, \ (i = 1, \ldots, N, j = 1, 2)$

where the inequality constraints are not enforced on the vertices of the grains in the kinematic path structure.

In the above formulation, the actuator torque limits were neglected. If we consider them, the problem is to find a **dynamic path structure**. If we further ignore the actuator speed limits and set the cutting speed $\vartheta(u, v)$ to be constant, the problem is reduced to finding the "shortest path" covering a surface. In this case, we ask what is the best geometric path structure. Also, if we plug $w(u, v) = w_o(\eta(u, v), u, v)$ into the compatibility equation, the compatibility turns out to be a partial differential equation about the direction angle $\eta(u, v)$. Solving this PDE is equivalent to generating iso-cusp-height tool paths.

Summary

Traditionally, tool paths have been considered a time-series of CC-points (together with the corresponding orientation vectors). This serial description is suited for commanding a machine tool, but it is too unstructured to deal with an optimization problem. Deviating from the conventional approach, we introduce a new framework in this thesis. We capture a family of tool paths with a (velocity) vector field. The streamlines or integral curves of the vector field are candidate tool paths.

The speed $\vartheta(u, v)$ and the direction field $\eta(u, v)$ can specify the vector field. The side step function $w(u, v)$ characterizes the "gap width" between two neighboring paths. The shifting procedure declares the actual gap width to be the harmonic mean of the side step function along the orthogonal transverse curve. We choose tool paths from the infinitely-many streamlines of the velocity vector field using the side step function.

The side step function must satisfy a certain compatibility equation with respect to a direction field, at least, locally. There are various ways to capture this compatibility equation mathematically. A way to see the compatibility is to accept the existence of a continuous function over a region on the surface, whose level sets we declare to be the streamlines of the velocity vector field. We refer to this function as the stream function. In the region where a stream function is defined, we do not allow any intermittent cuts. Assuming enough smoothness of the stream function, we derive a partial differential relation between the side step function and the direction field. When we weaken the smoothness of the stream function, we have jump conditions. The jump conditions are limits of the differential continuity equation as the underlying direction field approaches a non-smooth field. In a region on the surface, we allow tool paths to disappear with certain regularity. For example, along a (transverse) curve in the region, we can remove every other path. We refer to this phenomenon as the pseudo-continuity. On the region where the pseudo-continuity occurs, we define a continuous function, which we call the weaving function of the region. The weaving function is a generalization of the stream function in that the level sets of the weaving function are the streamlines of the direction field. Weaving functions and stream functions are important to capture global effects such as loops of tool paths.

Consequently, our basic viewpoint is that

> the triad $(\vartheta(u, v), \eta(u, v), w(u, v))$ of functions, which meet certain compatibility conditions, defines tool paths.

We refine this basic idea to state our problem formally. We define the following terms, which essentially partition a designed surface. We define grains as basic building blocks of tool paths, in which tool paths exhibit enough regularity. In a grain, the compatibility in its strict sense must hold. Compatible grains are collected and form what we refer to as a path continuum. In a path continuum, neither tool paths nor participating functions are smooth but a cutting tool can move continuously across grains. We consider a path continuum independent of other path continua in the sense of path generation. Finally, we define a collection of non-overlapping path continua as a geometric path structure. A geometric path structure and the speed (function) form a kinematic path structure. A dynamic path structure is a kinematic path structure with some restrictions being imposed on the speed and the direction field. Typically, the dynamic structure is smoother than the kinematic path structure. Our optimization problem is then reduced to finding an optimal geometric/kinematic/dynamic path structure. What type of path structure we seek depends on a particular set of constraints that we impose on the problem.

We regard cutting time as the cost in our optimization problem. Effective cutting time is the period during which the cutting tool makes a tangential contact with the designed surface and cuts the part within a specified dimensional tolerance. The rest of the time is non-effective cutting time. The effective cutting time is modeled mathematically by the integral of the reciprocal of the area removal rate over the surface. We evaluate the non-effective cutting time by literally counting the number of tool paths and assigning proper weights to them.

The cusp height limit is an inequality constraint of our problem. It is the dimensional tolerance of a part to be machined. We control the cusp height through adjusting the side step. The side-step-limit captures how wide an area is covered by an individual path and depends on the direction of the path. The limited capability of the actuators of a machine tool results in the actuator torque-velocity constraint. The general form of the actuator limits are quite complicated. The speed limits of actuators are easier to be stated. They form regional constraints in the tangent spaces of the designed surface. The Jacobian matrix of the surface inverse kinematics plays an important role in this model. Generally, we specify the inequality constraints in terms of the participating functions and their partial derivatives. Even though there are many other constraints that reflect the reality of machining, we mainly consider the cusp height limit and the actuator speed limits. In fact, solving the very simple form of the time-optimal machining problem is already challenging.

In summary, the time-optimal machining problem is to find a (kinematic/dynamic) path structure that minimizes the total cutting time subject to the cusp height limit and the actuator limits.

CONCLUDING REMARKS

A contribution of this thesis is a general formulation of the tool path optimization problem. We have theoretically considered geometric compatibility relations, machine tool kinematics, motor limits and geometric finish requirements. Kinematics, surface geometry, surface finish, cost and path intervals are seamlessly integrated into one formulation.

Our viewpoint is significantly different from others—a path is modeled as a streamline of a vector field. This field description allows us to deal with families of tool paths conveniently. The most basic requirement of surface machining is that the cutting tool should visit or cover all the points on the designed surface. This requirement is hard to be stated mathematically in the conventional framework while the field description expresses this requirement naturally.

3 General Consequences of the Formulation

—*A QUALITATIVE STUDY ON THE PROBLEM*—

We formulated the time-optimal machining problem in the preceding chapter. In this chapter, we provide immediate consequences that can be drawn from the formulated problem. This chapter reveals the general structure of the problem and prepares us to find the numerical solutions. Mainly, we study the compatibility equation and the velocity polygons in detail. The results are "trivial" but we mention them explicitly.

In Section 1, we show how to allow more freedom in constructing the tool paths from a known path structure by generalizing the orthogonal shifting procedure. The compatibility equation makes this generalization possible. In addition, we detail the procedure by introducing formulas for numerical analysis. In Section 2, we show how it is possible to reduce the compatibility equation, which is a partial differential relation, to a system of ODEs, presuming that the direction field is given. The characteristic theory of PDEs plays a key role in this conversion. Interesting enough, we stated the problem mainly using partial derivatives while ordinary line derivatives on a tool path are "more intuitive," but now, when we "solve" the problem, the partial derivatives are converted back to ordinary derivatives. From this point on, we develop our reasoning based on the assumption that the direction field of a continuum is known. Section 3 shows that the (optimal) side step in a continuum is fixed once the direction field is given. In Section 4, we see that the (optimal) speed is also known from the direction field. In Section 5, given a direction field, we show that the optimization in a continuum is decomposed into the optimization processes on individual streamlines. In Section 6, we show how to approach each decomposed problem to determine all the unknowns along a streamline. Therefore, one way to see the problem is to think of the direction field as the independent "variables" (or variations) of the optimization problem, which we discuss in Section 7. The remaining task is to decide the direction field and the continuum (boundaries) of a path structure. Finding the direction field inside a given continuum is relatively easy task; however, the most difficult part in our problem is to find the optimal continuum boundaries.

3.1 The Generalized Shifting Procedure

In this section, we explain how to select tool paths among the streamlines, given a direction field $\eta(u, v)$ and a compatible side step $w(u, v)$ in a continuum $\mathcal{U} \subset \mathcal{P}$ or $\mathcal{V} \subset \mathcal{S}$.

The Orthogonal Shifting Procedure

In Subsection 2.1.7, we defined the shifting procedure, essentially using Equation (10)

$$\int_0^{\bar{w}} \frac{d\alpha}{w_\perp(\alpha)} = 1$$

where the integration was made along the **orthogonal** transverse curve $\boldsymbol{u}_\perp(\alpha)$ parameterized by its arc length α. The next tool path was defined as the streamline that passes the point $\boldsymbol{u}_\perp(\bar{w}) \equiv \boldsymbol{u}_*$.

The Generalized Shifting Procedure

The compatibility equation allows us to use other curves, which are not necessarily orthogonal to the streamlines of the direction field.

The problem is to find a point $\boldsymbol{u}_f$ on the next tool path from a point $\boldsymbol{u}_o$ on a previous tool path. Let a curve on the parameter space $\mathcal{P}$ be denoted by $\boldsymbol{u}_z(\xi)$, whose parameter is ξ. We require that (1) the curve $\boldsymbol{u}_z(\xi)$ be centered at the point $\boldsymbol{u}_o$ and also that (2) it must not be parallel to the parameter velocity vector field $\dot{\boldsymbol{u}}$ that is specified by the given direction field $\eta(u, v)$.[†] Such a curve $\boldsymbol{u}_z(\xi)$ is called a *transverse curve*. Let the value of the parameter ξ at the end point $\boldsymbol{u}_f$ be denoted by ξ_f, *i.e.* $\boldsymbol{u}_z(\xi_f) = \boldsymbol{u}_f$. Then, we solve the following equation to find the end point parameter ξ_f of the transverse curve:

$$\int_{\mathcal{C}_z} \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} = 1 \text{ where } \mathcal{C}_z \equiv \{\boldsymbol{r}(\boldsymbol{u}_z(\xi)) : 0 \le \xi \le \xi_f\}. \tag{29}$$

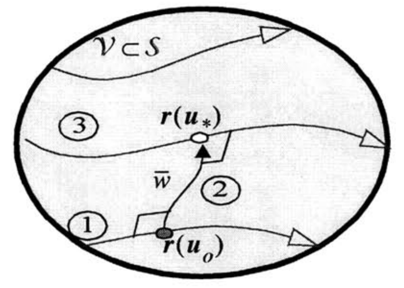
We pick the streamline that passes the point $\boldsymbol{u}_z(\xi_f)$ as the next path as shown in Figure 17-(*b*).

(The Equivalence)* The orthogonal shifting procedure (Equation 10) is equivalent to the above generalized shifting procedure (Equation 29). Recall that the stream function, say Ψ, has the same value at all the points on a streamline. In our case, it implies that $\Psi(\boldsymbol{u}_*) = \Psi(\boldsymbol{u}_f)$. Therefore,

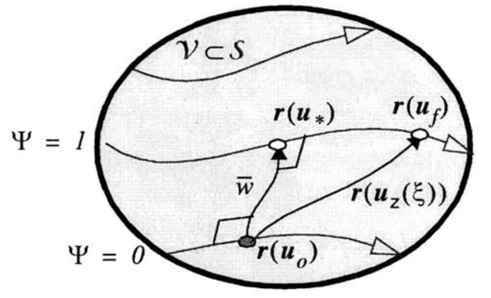
$$\left(\Psi(\boldsymbol{u}_*) - \Psi(\boldsymbol{u}_o) = \int_0^{\bar{w}} \frac{d\alpha}{w_\perp(\alpha)}\right) = \left(\Psi(\boldsymbol{u}_f) - \Psi(\boldsymbol{u}_o) = \int_{\mathcal{C}_z} \frac{1}{w} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}\right) = 1$$

which asserts the equivalence. This is shown in Figure 17.

[†] In equations, $\dot{\boldsymbol{u}}^T \cdot \boldsymbol{R} \cdot \frac{d\boldsymbol{u}_z}{d\xi} \neq 0$ and $\boldsymbol{u}_z(0) = \boldsymbol{u}_o$.



a) The method defined in the standard shifting procedure



b) Generalized Shifting Procedure

Figure 17 Generalized Shifting Procedure

Numerical Details*

In detail, we solve

$$\int_0^{\xi_f} \frac{d\xi}{w_z(\xi)} = 1 \qquad \text{where } \frac{1}{w_z(\xi)} \equiv \left(\frac{du_z}{d\xi}\right)^T \cdot \left.\frac{\boldsymbol{GPRa}}{w(\boldsymbol{u})}\right]_{\boldsymbol{u} = \boldsymbol{u}_z(\xi)}, \text{ and } \boldsymbol{a} \equiv \begin{bmatrix} \cos\eta(\boldsymbol{u}) \\ \sin\eta(\boldsymbol{u}) \end{bmatrix}.$$

This is equivalent to Equation (29) (by definition). Note that $\boldsymbol{G}$ is the 1st fundamental matrix, $\boldsymbol{P}$ is the modal matrix and $\boldsymbol{R}$ is the rotation by the right angle.

To avoid the iteration in solving the equation, the following 1st and 2nd order approximation can be used:

$$\xi_f = w_z(0), \qquad Err \equiv 1 - \int_0^{\xi_f} \frac{d\xi}{w_z(\xi)} \sim O\left(\frac{\Delta w_z}{w_z}\right)$$

$$\xi_f = \frac{1}{4}\left\{w_z(0) + 3 \cdot w_z\left(\frac{2}{3} \cdot w_z(0)\right)\right\} \qquad Err \sim O\left(\left(\frac{\Delta w_z}{w_z}\right)^2\right), \text{ where } \Delta w_z \equiv |w_z(\xi_f) - w_z(0)|.$$

Of course, the simplest choice for the transverse curve is a straight line in the parameter space.

Summary

Thanks to the compatibility equation, it is possible to find the tool paths given a path structure, not necessarily resorting to the orthogonal transverse curve. The necessary equations for the numerical analysis were provided.

3.2 (Differential) Compatibility as an ODE

In Section 2.3, we introduced a relation between the direction field $\eta(u, v)$ and the side step $w(u, v)$ in partial differential form. This differential relation is regarded as an equality constraint in our optimization problem. It is similar to conservation laws in continuum mechanics as shown in Figure 18-(*a*). If there are no intermittent cuts (or sinks/sources in fluid mechanics term) and the direction field is smooth, the compatibility results in a partial differential relation (Equation 15):

$$h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v) \cdot w$$

where h_1, h_2 and h_3 are known functions. And we have seen the jump conditions which we specify along the grain boundaries where the direction field $\eta(u, v)$ is not necessarily smooth.

The Characteristic Theory of 1st Order Quasi-Linear PDEs

In this section, we further develop the compatibility relation, which consists of the partial derivatives, to a system of ODEs, tentatively presuming the direction field $\eta(u, v)$ to be given. If we insert such a presumably known direction field into above Equation (15), we see a *1st order PDE* about a side step $w(u, v)$. Using the *characteristic theory of PDEs*, which is well-known [*e.g.*, *see* 119, 120 and 121], we have a system of ODEs:

$$\frac{du}{h_1} = \frac{dv}{h_2} = \frac{dw}{h_3 \cdot w} = ds \tag{30}$$

for a parameter s. The integration of first two terms ($du/ds = h_1$ and $dv/ds = h_2$, *c.f.* Equation 9) produces an integral curve $(u(s), v(s))$ in the parameter space $\mathcal{P}$. In the PDE theory, we refer to such integral curves as *(base) characteristic lines*. Base characteristic lines are shown in the bottom of Figure 18-(*b*). If you compare Equation (30) with Equation (9), it is not difficult to realize that

> the **base characteristic lines** are the **parameter streamlines** of the direction field and the above parameter s is the **arc length parameter** of a **surface streamline**,

which might have been conjectured.

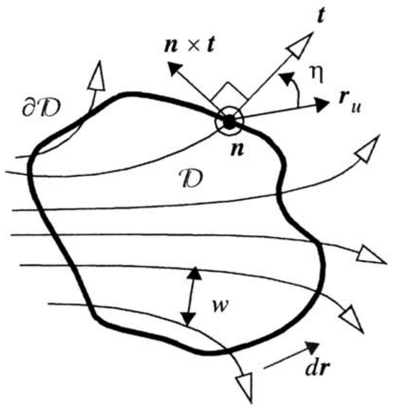
After getting a characteristic line $(u(s), v(s))$, we integrate the last equality:

$$w(s) \equiv w(u(s), v(s)) = w(0) \cdot exp\left(\int_0^s h_3(s)ds\right) \tag{31}$$

where $w(0)$ is the side step at a start point of the streamline.† Note that $w(0)$ is not yet decided.

† Of course, the term $h_3(s)$ is the abbreviation of the following rather long expression:

$$h_3(s) \equiv h_3(\eta(u(s), v(s)), \eta_u(u(s), v(s)), \eta_v(u(s), v(s)), u(s), v(s)) .$$



The number of toolpaths in = that out

⇔

$$\oint_{\partial\mathcal{D}} (1/w)\cdot(n\times t)^T dr = 0$$

⇔

$$\int_{\mathcal{D}} d[(1/w)\cdot(n\times t)^T dr] = 0$$

⇔

$$h_1(\eta, u, v)\cdot\frac{\partial w}{\partial u} + h_2(\eta, u, v)\cdot\frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v)\cdot w$$

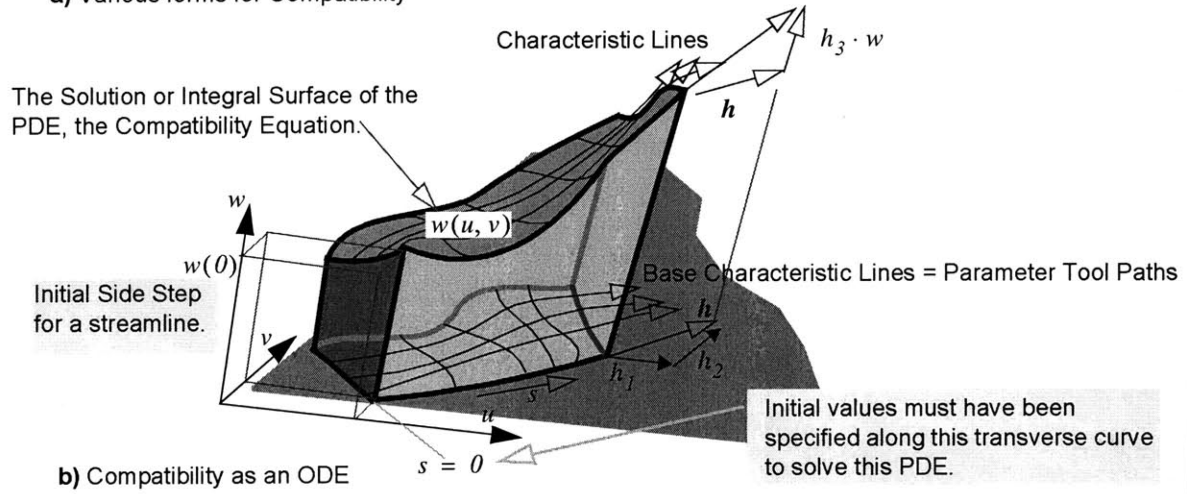
⇔

$$\frac{du}{h_1} = \frac{dv}{h_2} = \frac{dw}{h_3\cdot w} = ds$$

⇔

$$w(s) \equiv w(u(s), v(s)) = w(0)\cdot \exp(\textstyle\int_0^s h_3 ds)$$

a) Various forms for Compatibility


Characteristic Lines
$h_3 \cdot w$
The Solution or Integral Surface of the PDE, the Compatibility Equation.
h
w
$w(u, v)$
$w(0)$
Initial Side Step for a streamline.
Base Characteristic Lines = Parameter Tool Paths
v
h
h_2
h_1
s
u
Initial values must have been specified along this transverse curve to solve this PDE.
$s = 0$

b) Compatibility as an ODE

Figure 18 Characteristic Lines of the Compatibility

The integral curves $(u(s), v(s), w(s))$ in the abstract (u, v, w)-space are called *characteristic lines*, in the PDE theory. This integration is visualized in Figure 18-(b).[†]

> Therefore, once the direction field is known in a path continuum (together with the matching pairs), it is possible to extract all the information on the side step $w(u(s), v(s))$ along a streamline except for the "initial" side step $w(0)$. By the repetition of the same process for other streamlines, we construct the side step function $w(u, v)$ in the continuum once (1) the (initial) side steps along a transverse curve and (2) the direction fields (together with the matching pairs) are known.

Of course, we apply (continuity and coverage) jump conditions across grain boundaries to integrate the streamlines along the characteristic lines. It is inconvenient to mention this fact whenever the integration appears. Henceforth, we regard the integration in Equation (31) as the one that is made together with the jump conditions.

[†] The integration for the radius of streamtubes in fluid mechanics is analogous to this integration [*e.g.* 155].

3.3 The Widest Cut Principle

In this section, it will be shown how above Equation (31), combined with the cusp-height-limit inequality constraint (Equation 24, $w \le w_o$), decides the unknown initial side step $w(0)$ after we find a "good" direction field.

Generalized Side Step Integration

Once the direction field and the matching pairs are known in a continuum, it is possible to construct the following graph:

$$\left\{(s, H(s)) : \quad 0 \le s \le L\right\} \qquad \text{where } H(s) \equiv ln\left(\frac{w(s)}{w(0)}\right) = \int_o^s h_3(s) ds$$

and s is the arc length parameter of a streamline (or a characteristic line) $\boldsymbol{r}(u(s), v(s))$. Figure 19 shows a typical shape of the graph $\{(s, H(s))\}$ and indicates possible jump conditions across grain boundaries. We point out that the function $H(s)$ is *upper semi-continuous*.

The Side-Step-Limit

In addition, once the direction angle is known especially along the streamline $(u(s), v(s))$, it is possible to construct the following graph:

$$\left\{(s, H_o(s)) : \quad 0 \le s \le L\right\} \qquad \text{where } H_o(s) \equiv ln\left(w_o(s)\right) \equiv ln\left(w_o\left(\eta(u(s), v(s)), u(s), v(s)\right)\right)$$

and $w_o(\eta, u, v)$ is the side-step-limit as defined in Equation (22, Section 2.5.1). Because of the cusp height limit constraint, $w(s) \le w_o(s)$, the region above the function $H_o(s)$ is the infeasible region into which the graph

$$\left\{(s, W(s)) : W(s) \equiv ln\left(w(s)\right), \quad 0 \le s \le L\right\}$$

is not allowed to intrude itself. The infeasible region is shown in Figure 19.

The Widest Cut Principle

We write down the cutting time, as defined in Equation (20 and 21, Section 2.4), in abbreviated form:

$$T_c = \iint_{\mathcal{P}} \frac{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot dudv}{w(u, v) \cdot \vartheta(u, v)} + \frac{1}{2} \cdot \left\{ \int_{\text{along all the grain boundaries}} \frac{\tau_o}{w(u, v)} \cdot (1 - \gamma) \cdot \left|(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}\right| + \oint \frac{\|d\boldsymbol{r}\|}{V_o} \right\}. \tag{32}$$

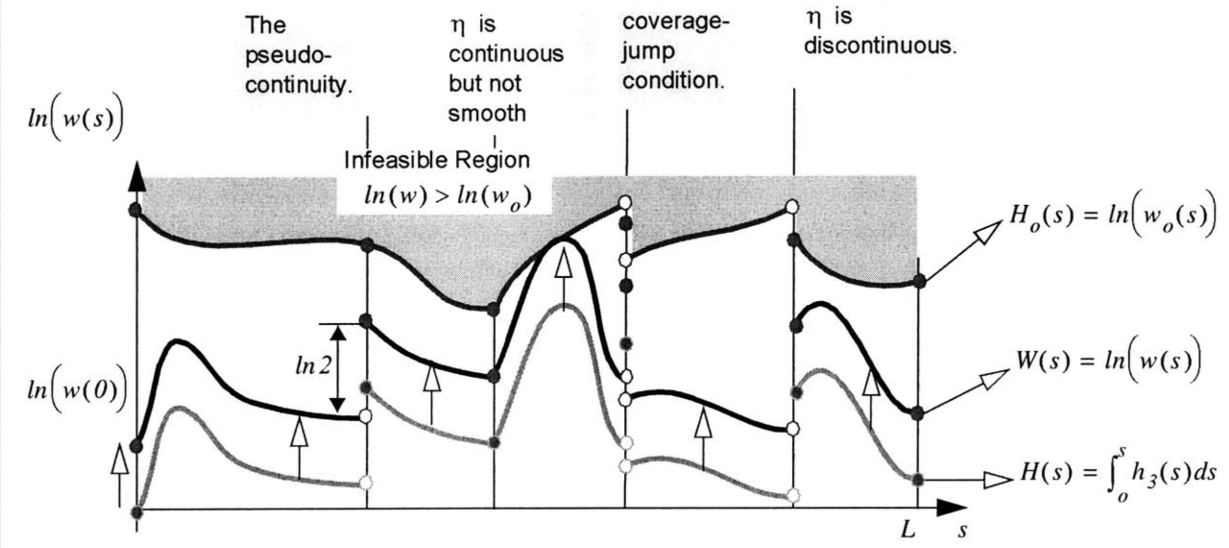


Figure 19 The Widest Cut Principle: move H(s) up as high as possible while remaining in the feasible region.

From the above equation, it is observed that the cutting time is reduced as the side step (function) $w(u, v)$ increases; (of course, this is because the NEP function τ_o satisfies the SUB-LINEAR GROWTH CONDITION, $\partial(\tau_o/w)/\partial w \le 0$, as defined in Section 2.4). In addition, if you read through the problem statement in Section 2.6, it is observed that the cusp height limit constraint, $w(s) \le w_o(s)$, is the unique constraint that limits the side step $w(u, v)$. Therefore, we reach the following "trivial" conclusion: to minimize the cutting time, we maximize the side step function subject to the cusp height limit constraint, for a specified direction field (and matching pairs) in a continuum.

If we apply this principle for the streamlines in a continuum, the initial side step $w(0)$ of each streamline is decided by the following equation:

$$w(0) = inf\left\{\frac{w_o(s)}{exp(\int_0^s h_3(s)ds)} : 0 \le s \le L\right\} . \tag{33}$$

This is shown in Figure 19. Now, we realize that given a direction field (and matching pairs in a continuum), the optimal side step is fixed by Equation (33 and 31), (resorting to the compatibility equation and the cusp height limit constraint).

SUMMARY FOR THE OPTIMAL SIDE STEP

In short, a direction field decides a side step function. This conclusion is even stronger than the one we stated in the previous section.

3.4 The Fastest Cut Principle

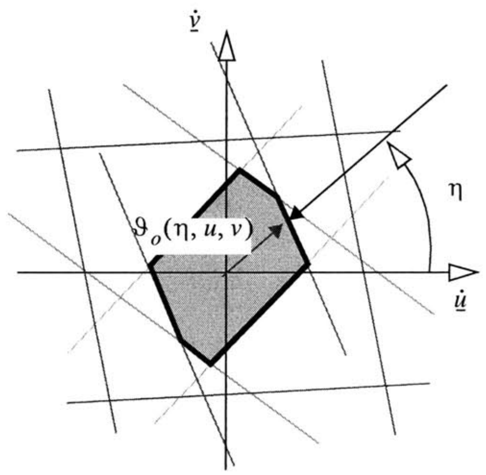
As before, we assume that the direction field is presumably known in this section. We now ask a similar question for the speed function $\vartheta(u, v)$ in a kinematic path continuum: what is the optimal speed of a continuum for a specified direction field?

THE OPTIMAL SPEED UNDER GENERAL ACTUATOR LIMITS*

Given a direction field, the streamlines $(u(s), v(s))$ can be extracted by Equation (30) as before. Assigning the optimal speed distribution $\vartheta(s) \equiv \vartheta(u(s), v(s))$ along a specified (robot) trajectory—in our case, the streamline—is a problem that has been well-known in the robotics community. We can solve the problem by invoking what we call the BDSA (*Bobrow-Dubowsky-Shin Algorithm*) [88 and 89]. Now,

we have the procedure to find the optimal speed function in a continuum for a given direction field,

at least, theoretically. In this thesis, we postpone discussing the general actuator limits on drawing this general conclusion, (not because it is hopeless to deal with them but) because we have a lot to do for even simpler cases. Instead, we will study the actuator speed limits in detail.



(a) A Velocity Polygon in a $(\dot{\underline{u}}, \dot{\underline{v}})$-Space and the Maximum Speed in the given direction.

$$\left\{(\vartheta\, cos\eta, \vartheta\, sin\eta) : \vartheta \le \vartheta_o(\eta, u, v),\ 0 \le \eta \le 2\pi\right\}$$

$$= \left\{(\dot{\underline{u}}, \dot{\underline{v}}) : \dot{\boldsymbol{u}} = \boldsymbol{P}\dot{\underline{\boldsymbol{u}}}\ and\ \left|c_{i1} \cdot \dot{u} + c_{i2} \cdot \dot{v}\right| \le \omega_o^i\right\}$$

$$T_c = \iint_{\mathcal{P}} \frac{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot dudv}{w(u, v) \cdot \vartheta(u, v)} + \frac{1}{2} \cdot \left\{ \int \left(\frac{\tau_o}{w}\right) \cdot (1 - \gamma) \cdot \left|(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}\right| + \oint \frac{\|d\boldsymbol{r}\|}{V_o} \right\}$$

(b) Cutting Time with respect to the Speed. Because of

(KINEMATIC CONDITION) $\dfrac{\partial \tau_o}{\partial \vartheta} = 0$, we have

$$\vartheta \nearrow \longrightarrow T_c \swarrow$$

Figure 20 Maximum Speeds and The Cutting Time

Velocity Polygons

Recall that the actuator speed limits are captured by the velocity polygons in $(\dot{u}, \dot{v})$, $(\dot{\underline{u}}, \dot{\underline{v}})$ or $(\dot{\underset{\sim}{u}}, \dot{\underset{\sim}{v}})$-space. A velocity polygon in a $(\dot{\underline{u}}, \dot{\underline{v}})$-space is shown in Figure 20-(*a*). Recall also that a pair (ϑ, η) is a polar coordinate in the $(\dot{\underline{u}}, \dot{\underline{v}})$-space. At a point on the surface, the boundary of the velocity polygon can be captured by a piecewise smooth function $\vartheta_o : \mathcal{R} \times \mathcal{P} \to \mathcal{R}$, $(\eta, u, v) \to \vartheta_o(\eta, u, v)$ as shown in Figure 20-(*a*), which evaluates the maximum speed for a given direction angle η.[†] Then, the actuator speed limits, $|c_{i1} \cdot \dot{u} + c_{i2} \cdot \dot{v}| \le \omega_o^i$ as defined in Equation (28), are stated in the following inequality:

$$\vartheta(u, v) \le \vartheta_o(\eta(u, v), u, v) \quad .$$

The Fastest Cut Principle

By ignoring the actuator torque limits, we imply that we pursue the kinematic path structure (instead of the general dynamic path structure). In Section 2.4.2, we dictated the kinematic condition to the NEP function $\tau_o(\vartheta, \eta, w, u, v)$ (for a kinematic path structure), namely $(\partial\tau_o / \partial\vartheta) = 0$. Consider the formula (Equation 20 and 21) for the cutting time as abbreviated in Figure 20-(*b*) and also notice that there are no other constraints but the speed limits that limit the speed. Now, it is evident that given a direction field $\eta(u, v)$, we must maximize the speed at every point to minimize the cutting time subject only to the speed limits, $\vartheta(u, v) \le \vartheta_o(\eta(u, v), u, v)$. Therefore, if we are (not subject to the actuator torque limits but) subject only to the speed limits, it is necessary to satisfy

$$\vartheta(u, v) = \vartheta_o(\eta(u, v), u, v) \quad .$$

Therefore, for the purpose of determining the optimal speed distribution for a specified direction field, it is not necessary to perform even a single integration (while the side step was determined globally through the integration as seen in the previous section). This is because the "global" condition for the speed is the acceleration limits but they are ignored in the kinematic path structure. However, we point out that our overall problem is not purely local even for the kinematic path structure because we need global thinking to determine the direction field itself, which was presumed to be known in this section.

Summary for the Optimal Speed

Like the optimal side step function, the optimal speed distribution is fixed for a specified direction field.

[†] In an equation, we define: $\vartheta_o(\eta, u, v) \equiv min\left(\left\{\omega_o^i / \left|\sum_j c_{ij}(u, v) \cdot h_j(\eta, u, v)\right|, i = 1, \ldots, N\right\}\right)$.

And the boundary of the velocity polygon is

$$\{(\vartheta cos\eta, \vartheta sin\eta) : \vartheta = \vartheta_o(\eta, u, v),\ 0 \le \eta \le 2\pi\} \subset (\dot{\underline{u}}, \dot{\underline{v}})\text{-}space .$$

3.5 The Individual Cut Principle

In this section, we also assume that a direction field $\eta(u, v)$ is given in a continuum and ask what the remaining unknowns in a continuum must be. The unknowns include (1) the side step (2) the speed (3) the grains and (4) the matching pairs of grain boundaries. This is an even broader question than what we asked in the previous sections. This section essentially confirms that the unknowns in a continuum must be optimal along an individual streamline in a continuum. Therefore, the optimal path problem in a 2-dimensional continuum is decomposed into the optimization along each 1-dimensional streamline in the domain, as soon as a direction field is given. More procedural or detailed form of this principle is given in the next section when we attempt to find the optimal matching pairs and grains. We begin by recalling the definition of matching pairs and defining few additional terms.

Definitions

A Matching Pair is a pair of numbers that characterizes how many paths are connected (or disconnected) across a grain boundary.[†] When we defined the path continuum,[‡] we restricted the feasible matching pairs $(\gamma_{ij}, \gamma_{ji})$ to the set

$$Q'' \equiv \left\{(1, 1), \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right)\right\}.$$

The case $(0, 0)$ is not included because we do not allow incompatibility inside a path continuum. The case $(1, 1)$ represents the "true continuity" and the cases, $(1, 1/2)$ and $(1/2, 1)$, are for pseudo-continuity.

A Pseudo-Compatible Grain Boundary is a section of a grain boundary along which pseudo-continuity occurs. **Pseudo-Compatible Points** of a streamline are the intersection of the streamline with pseudo-compatible grain boundaries. We point out that finding the unknowns, (3) the grains and (4) the matching pairs, is equivalent to finding pseudo-compatible points of every streamline (because the matching pairs are $(1, 1)$ in other locations).

The Reference Side Step: We tentatively assume that the matching pairs are all $(1, 1)$ in a given continuum. That is, we presume that all the points are true compatible. Under this presumed condition, it is possible to construct the (optimal) side step in the continuum by resorting to the widest cut principle as presented in Section 3.3. The (optimal) side step that is decided under the assumption of true compatibility is called a *reference side step*. We will refer to the reference side step at a point $(u, v) \in \mathcal{P}$ as $w_{ref}(u, v)$. In the strict sense, we should have denoted it by $w_{ref}[\eta](u, v)$ since it depends on the direction field.

The Cutting Time per a Streamline: Let $\{(u(s), v(s)) : 0 \le s \le L\}$ be a streamline in a continuum, where s is the arc length parameter. We denote the arc length parameters at the pseudo-compatible points along the streamline by $s_i \in [0, L]$ ($s_i < s_{i+1}$ and $i = 1, 2, \ldots N-1$). Of course, the positions $s_i \in [0, L]$ and even the number N is not known yet. We add 2 end points in the sequence by setting $s_0 = 0$ and $s_N = L$. The

[†] *See* Subsection 2.3.5.
[‡] *See* Subsection 2.3.6.

following term τ_c^{ref} is called the *cutting time of the streamline* or *the cutting time per a streamline*:

$$\tau_c^{ref} \equiv \int_0^L \left(\frac{w_{ref}(s)}{w(s)}\right) \cdot \frac{ds}{\vartheta(s)} + \frac{1}{2}\sum_i \left(\left[\tau_o \cdot (1-\gamma_{i,i+1}) \cdot \left(\frac{w_{ref}}{w}\right)\right]_{s \to s_i^-} + \left[\tau_o \cdot (1-\gamma_{i+1,i}) \cdot \left(\frac{w_{ref}}{w}\right)\right]_{s \to s_i^+}\right) \tag{34}$$

where $w_{ref}(s)$ ($\equiv w_{ref}(u(s), v(s))$) is the reference side step along the streamline, $\vartheta(s)$ is the speed, $w(s)$ is the side step, τ_o is the NEP term, $\gamma_{i,i+1}$ is a component of matching pairs and $s \to s_i^{\pm}$ represents the right (or left) limit process to the pseudo-compatible points s_i. Just for convenience, we set the matching pairs at the end points such that $(\gamma_{01}, \gamma_{10}) = (1, 0)$ and $(\gamma_{N,N+1}, \gamma_{N+1,N}) = (0, 1)$. In order to see the origin of this term, it is recommended to compare the cutting time τ_c^{ref} per a streamline with the cutting time T_c as defined in Equation (20, 21 and 32).

For example, suppose that there are no pseudo-compatible points. In this case, $w = w_{ref}$ and, as a result, the cutting time per a streamline (Equation 34) is reduced to

$$\tau_c^{ref} \equiv \int_0^L \frac{ds}{\vartheta(s)} + \frac{1}{2} \cdot \left(\tau_o\Big|_{s=0} + \tau_o\Big|_{s=L}\right).$$

As seen here, it is not difficult to notice that the term τ_c^{ref} is the cutting time that it takes for a cutter to move along a streamline from $s = 0$ to $s = L$. This cutting time is called the *reference cutting time* of a streamline. Note also that, in each interval $\{s_i < s < s_{i+1}\}$, it is proved that the ratio w_{ref}/w is constant.

THE INDIVIDUAL CUT PRINCIPLE

It is possible to prove:

(THE INDIVIDUAL CUT PRINCIPLE) to minimize the cutting time T_c, we must minimize the cutting time τ_c^{ref} along each streamline.

We defer the proof to the readers but we provide some hints to it. (**SKETCH OF THE PROOF**) At first, we show that $ds \wedge w d\Psi$ is an area-2-form, namely $\|\boldsymbol{r}_s \times \boldsymbol{r}_\Psi\| \cdot ds d\Psi$, where s is the arc length parameter of a streamline, w is the side step and Ψ is the stream function. Also note that the following triviality $d\Psi = (d\Psi/d\Phi) \cdot d\Phi$, where Φ is a (weaving) function. Then, we transform the integration for the cutting time T_c as defined in Equation (20 and 21) from the domain $\mathcal{P}$ or (u, v)-domain into the (s, Φ)-domain. Additionally, we show that there exist a weaving function, Φ, such that $w_{ref}/w = d\Psi/d\Phi$.

REMARK: The individual cut principle is physically appealing and even trivial in that the term τ_c^{ref} is literally the cutting time per a streamline. On the other hand, the result is somewhat "surprising" or at least non-trivial considering that we used inexact expression in equation (20) such as $\sum w_i(s)\, ds \to Area$ when we model the problem.

SUMMARY

To minimize the cutting time, it is enough to consider a series of independent optimization problems along streamlines.

3.6 Optimal Matching Pairs at Pseudo-Compatible Points

Until now in this chapter, we showed how to determine the optimal side step $w(u, v)$ and the optimal speed $\vartheta(u, v)$ in a given continuum presuming that (1) the direction field $\eta(u, v)$ and (2) the matching pairs $(\gamma_{ij}, \gamma_{ji})$ have been known. Besides, the previous section showed that there is a measure τ_c^{ref} on a streamline that must be minimized for the optimality of the global cutting time T_c. In this section, we ask the following question: given a direction field in a path continuum, what are the positions of grain boundaries where the matching pairs are either $(1, 1/2)$ or $(1/2, 1)$? Such positions were defined as the pseudo-compatible points. The question is equivalent to finding the optimal matching pairs.

First, we show that there is an upper bound, (only in a practical sense), for the desired number of pseudo-compatible points. Second, we show a "theoretical" approach that helps us to understand the structure of the problem. Finally, we present a procedure to solve this problem approximately. As in the previous sections, we assume that the direction field in a continuum is fixed throughout this section. The individual cut principle allows us to consider streamlines of the direction field, one by one at a time.

1. An Upper Bound of the Number of Pseudo-compatible Points

Observing the cutting time τ_c^{ref} (Equation 34) of a streamline, we recognize that its non-effective portion (which is the right part of the equation) increases from the reference cutting time at least by

$$\tau_{ne}^{inf} \equiv inf\left\{\frac{\tau_o}{4} \cdot \left(\frac{w_{ref}(s)}{w_o(s)}\right) : 0 \le s \le L\right\}$$

whenever a pseudo-compatible point is inserted. On the other hand, the effective portion (together with the non-effective portion at the two end points) of the cutting time cannot be improved, from the reference cutting time, more than

$$\Delta\tau_c^{max} \equiv \int_0^L \left(1 - \frac{w_{ref}(s)}{w_o(s)}\right) \cdot \frac{ds}{\vartheta(s)} + \frac{1}{2} \cdot \left(\tau_o \cdot \left(1 - \frac{w_{ref}(s)}{w_o(s)}\right)\Bigg|_{s=0} + \tau_o \cdot \left(1 - \frac{w_{ref}(s)}{w_o(s)}\right)\Bigg|_{s=L}\right).$$

If the number $(N - 1)$ of pseudo-compatibility points is greater than $\Delta\tau_c^{max}/\tau_{ne}^{inf}$, the total cutting time is definitely greater than the reference cutting time. Therefore,

$$N \le N_U \equiv \left(\frac{\Delta\tau_c^{max}}{\tau_{ne}^{inf}}\right) + 1 .$$

This limits the number of pseudo-compatible points along a streamline. Note that the upper bound is computable once the direction field is given.

> REMARK* If the reference side step converges zero at a point ($s = s_o$) on the streamline, the upper bound N_U diverges. In this case, we remove "small" interval including the point to evaluate the term τ_{ne}^{inf}. The neighborhood can be found by

$$|s - s_0| < inf\{w_0(\eta, u(s_0), v(s_0)) : 0 \le \eta < 2\pi\}/2 .$$

This does not hurt the practicality because the exact information in such intervals cannot be reflected in the realization of the tool paths by the shifting process.

Remark: Below, we construct an analogous problem to help readers understand the trade-offs between the effective portion and the non-effective portion of the cutting time along a streamline.

An Analogous Problem:

Find a piece-wise continuous function, $f: [0, 1] \to R$

which minimize: $J = J_1 + J_2 = \int_0^1 f(x) \cdot dx + kN$ $\longleftarrow$ $T_c = T_e + T_{ne}$

subject to: $f(x) \ge 1 - x^2, \forall x \in [0, 1]$ $\longleftarrow$ Cusp Height: $w \le w_o$

$\frac{df}{dx} = -1, a.e.$ $\longleftarrow$ Compatibility

where N is the number of positions of discontinuity and k is a positive constant.

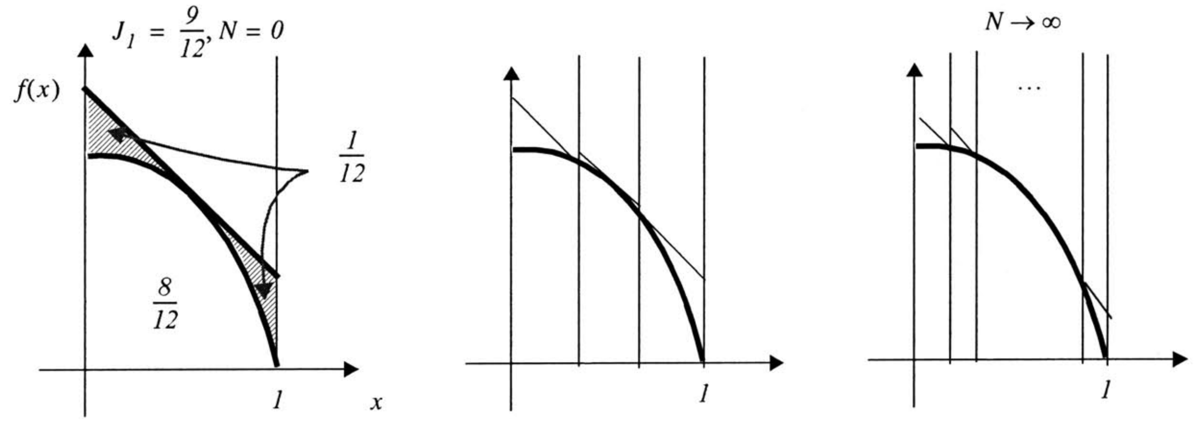


Figure 21 By introducing infinitely-many positions of discontinuity, we can make

$$J_1 \to \int_0^1 (1 - x^2) \cdot dx = 8/12 , J_2 = kN \to \infty .$$

The minimum of J occurs for a particular set of (finite) N^* positions of discontinuity. Finding the particular set of positions of discontinuity is **computation-intensive**.

However, it is not difficult to find an upper bound for the optimal number N^*. As shown in the first diagram, J_1 can be reduced by $1/12$ at most. Thus, an upper bound for N^* is $1/(12k)$, i.e.

$$N^* < 1/(12k) .$$

2. Optimal Matching Pairs for the General Case through the Individual Cut Principle

If we have the finite bound N_U, the unknown positions can be set by $N_U - 1$ variables s_i such that

$$0 < s_1 \leq s_2 \leq \ldots \leq s_{N_U - 1} \leq L .$$

Either equivalently, it is possible to change the unknown variables to new N_U variables such that

$$\alpha_1, \alpha_2, \ldots, \alpha_{N_U - 1}, \alpha_{N_U} \qquad \left(\sum \alpha_i = 1 \quad \text{and} \quad \alpha_i \geq 0 \right)$$

where we set $s_i - s_{i-1} = \alpha_i L$.

For a set of fixed α's, we have only 3 possible cases to be assigned to the matching pair at each corresponding position s, namely the elements of the feasibility set $Q'' \equiv \{(1, 1), (1, 1/2), (1/2, 1)\}$. Therefore, we have $3^{N_U - 1}$ cases. For each case of the combination of matching pairs, it is possible to evaluate the cutting time (Equation 34) of a streamline and to choose the best matching pairs (through enumeration). Let the cutting time of the streamline minimized for the fixed α_i's be denoted by $\tau_c^{ref}(\alpha_i)$. By minimizing this function $\tau_c^{ref}(\alpha_i)$, we obtain the pseudo-compatible points and their matching pairs. Of course, this becomes a computation-intensive approach as the upper bound increases.

Remark I: The continuity of the function $\tau_c^{ref}(\alpha_i)$ is an open question at this point.

Remark II: Another approach is to pre-partition the interval $[0, L]$ with finitely many equal disjoint sub-intervals. This approach has the advantage in that the integration can be pre-determined. The disadvantage is that the number of cases to be determined is quite high as we require more precision.

3. An Example of Optimal Matching Pairs for a Simple Case

We outline this subsection below. At first, we define the notion of the *overlap degree*. And then, we set the example what we will deal with. Finally, we present an algorithm, which, we claim, finds the optimal grain boundaries (together with the matching pairs) for the special example.

The Overlap Degree

We define the **Overlap Degree** $\mu(u, v)$ at a point $(u, v) \in \mathcal{P}$ such that

$$\mu \equiv log_2\left(\frac{w_o}{w}\right)$$

where w_o is the side-step-limit and w is the side step at a point on the surface. This quantity measures how much two neighboring tool paths overlap "locally"; in a sense, it is a measure of inefficiency. Taking "2" as the base of the logarithm is just for convenience, which will be evident soon.

The Example

We here assume that (1) the side-step-limit is constant (along a streamline), (2) the overlap degree on the streamline is piecewise linear (namely, $d\mu/ds = const > 0$), (3) the speed is constant and (4) the NEP term is also constant, (which we call the homogenous NEP model). The example is shown in Figure 22.

Few characteristics of the example are that the side step is piecewise exponential and that the side step is decreasing in each continuous section.

Optimal Pseudo-Compatible Points

We present the algorithm that finds the best set of pseudo-compatible points for the example that we have just set; the restrictions must be well recognized.

(The Simple Path Matching Procedure)

(1) Construct a streamline, $\{(u(s), v(s)) : 0 \le s \le L\}$.

(2) Assuming that all the matching pairs are $(1, 1)$, (which is the case for "true continuity"), evaluate the side step along the streamline resorting to the widest cut principle. We refer to this side step as $w_{ref}(s)$, which we defined as the **Reference Side Step** in the preceding section.

(3) Evaluate the overlap degree with the side step as evaluated in the above step. We refer to this overlap degree as $\mu_{ref}(s)$, which we call the **Reference Overlap Degree.**

(4) Mark the points where $\mu_{ref}(s) = n$, for every natural number n. (If there are infinitely many such points in some interval, we only mark the 2 boundary points of the interval). We refer to the marked points as **Cut-Off Points**.

(5) Set the matching pairs at each cut-off point to be $(1/2, 1)$ except for the end points.

Then, in the interval $\{s : n < \mu_{ref}(s) < n + 1\}$, we obtain the optimized side step $w(s)$ and the optimized overlap degree $\mu(s)$ as follows:

$$w(s) = 2^n \cdot w_{ref}(s) \qquad \text{and} \qquad \mu(s) = \mu_{ref}(s) - n .$$

We would appeal to readers' intuition by showing the procedure in Figure 22, rather than provide the proof. The figure also shows that the path matching procedure is analogous to removing sections of the "reference" tool paths.[†]

[†] There are found similar ideas in the computer graphics community [*See. binary thinning out technique* in 157, *c.f.* 156, 158, 27 and 159].

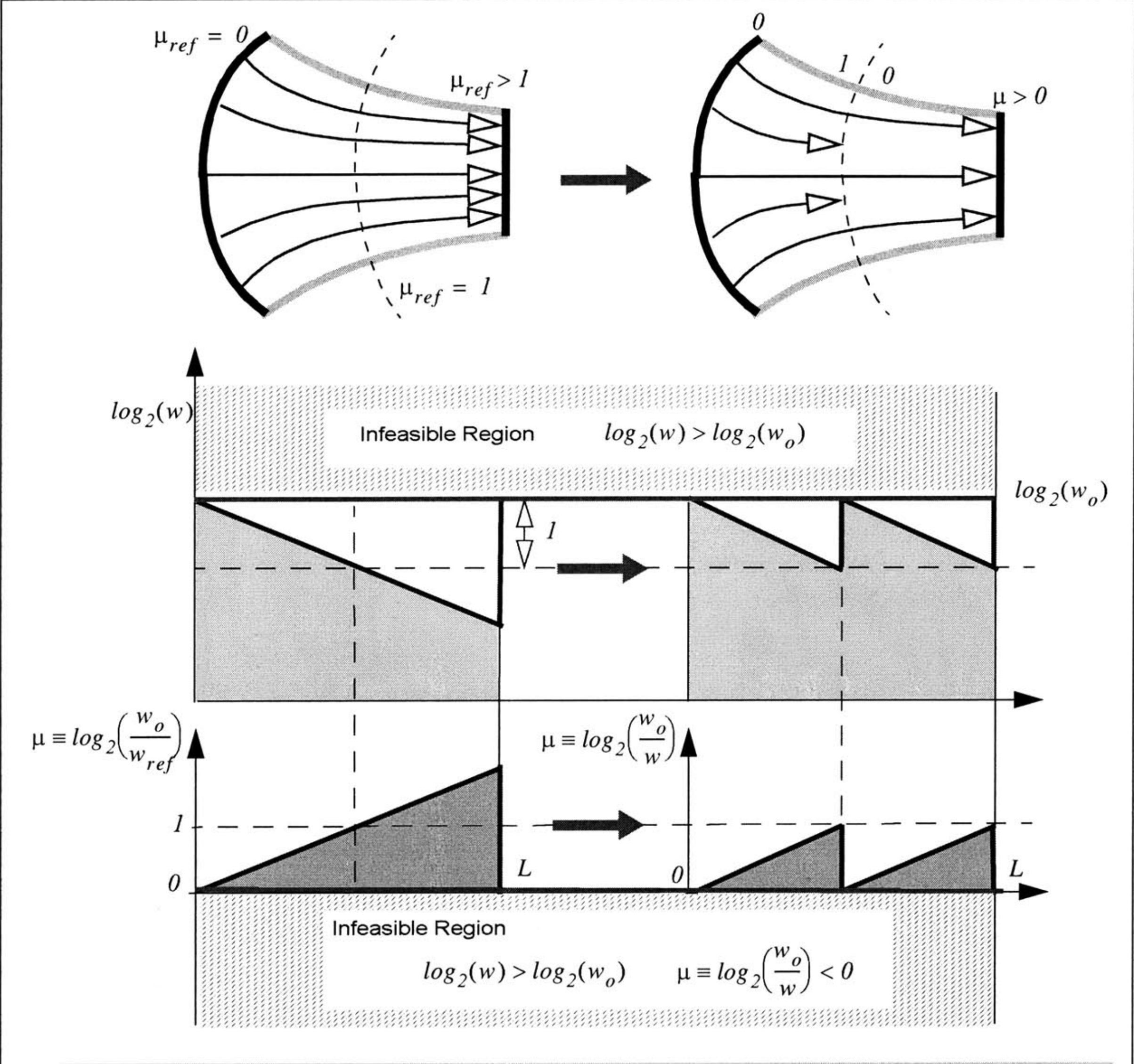


Figure 22 The Path matching For Exponentially Converging Field
The Key Point is that

the **effective cutting time** is reduced because the side step increase while the **non effective cutting time** does not change. Note that we are dealing with homogenous NEP models, namely $\tau_o = const$.

REMARK*: If we insert additional matching pairs, for example, $(1/2, 1/3)$ and $(1/3, 1/2)$, into the feasibility set Q'', the above algorithm does not work and the problem becomes harder. However, we point out that this difficulty is not the reason why we restricted the cases for matching pairs. The more fundamental reason is that we may develop very narrow grains with such extended feasibility set. If grain width becomes too narrow, the path matching cannot be regarded as a kind of regularity. In other words, the realization of path matching is impossible with narrow grains. Devising a penalty term for narrow grains is ill-suited for our framework, which was possible for narrow continua (*c.f.* Equation 21). To accommodate more feasible matching pairs, we need to operate a surgery on our framework.

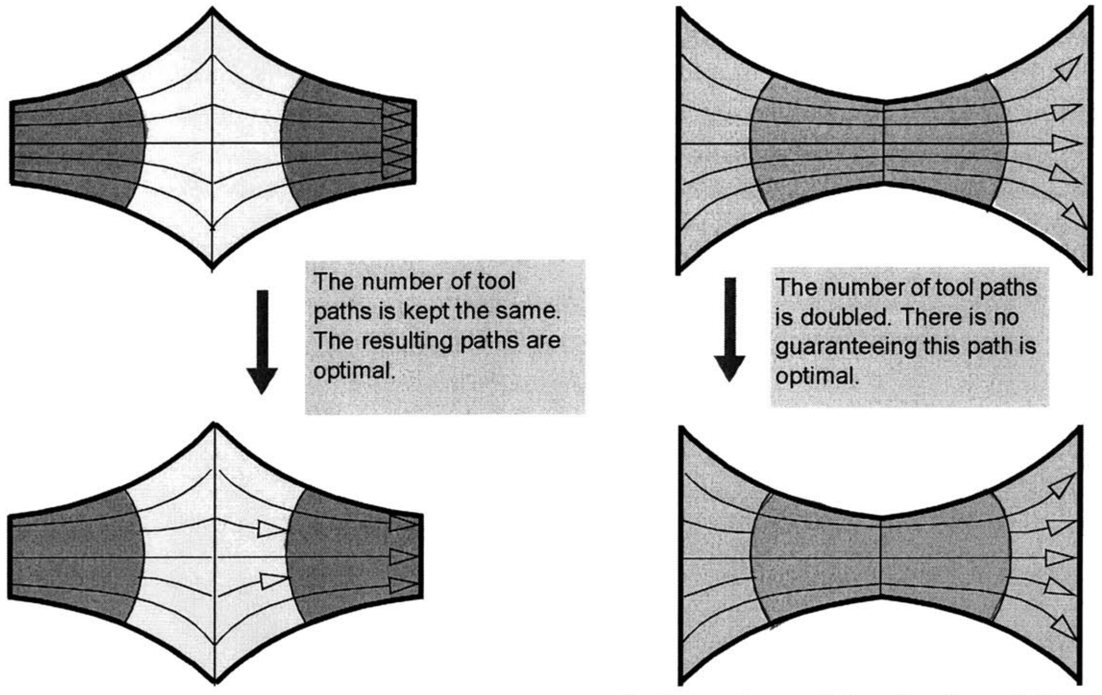

The number of tool paths is kept the same. The resulting paths are optimal.
The number of tool paths is doubled. There is no guaranteeing this path is optimal.
(a) Diverging and Converging: It Works!
(b) Converging and Diverging: It can Fail!

Figure 23 Slightly Extended Application of the Path Matching Procedure

A Slightly Extended Application

It is possible to devise an almost identical algorithm for a slightly general case. If a streamline is partitioned into two intervals such that the front interval is exponentially diverging and the other interval is exponentially converging, we have the optimal pseudo-compatible points by applying the same path matching procedure in each interval. The case is shown in Figure 23-(*a*). However, the applicability of the procedure breaks down for the inversed case as shown in Figure 23-(*b*) because the number of tool paths increases.

4. A Heuristic Approach for Controlling Matching Pairs in the General Case

We now consider the general case: (1) the NEP function τ_o is not homogenous, (2) the streamlines are not necessarily converging or diverging exponentially and (3) the speed is varying from place to place. It is recognized that the general case is even more difficult to solve than the Case (*b*) in Figure 23. The general "combinatorial approach," as given in Subsection 2, becomes complex as the amount of local variation increases. In this subsection, we present a heuristic approach that is "extrapolated" from the simple example presented in the previous subsection.

(THE GENERAL PATH MATCHING PROCEDURE)

(1) Construct a streamline, $\{(u(s), v(s)) : 0 \le s \le L\}$.

(2) Construct the reference side step $w_{ref}(s)$, the corresponding overlap degree $\mu_{ref}(s)$ and the cut-off points as before.

(3) The cut-off points naturally partition the interval $[0, L]$. We combine intervals as wide as possible such that each combined interval is either strictly converging or diverging. Let such N intervals be denoted by (s_i^1, s_i^2).

(4) **For** each strictly diverging or converging interval (s_i^1, s_i^2), $(i = 1, \ldots, N)$,

only if

$$\int_{s_i^1}^{s_i^2} \frac{ds}{\vartheta(s)} \ge \frac{1}{2} \cdot \left(\tau_o\Big|_{s = s_i^1} + \tau_o\Big|_{s = s_i^2}\right),$$

then we apply the previous simple path matching procedure.

(5) We adjust few boundary sections to ensure that each matching pair must be in the feasibility set Q''.

5. Summary for Matching Pairs

Once a direction field is given in a path continuum, the optimal matching pairs and the optimal grain boundaries are obtained, at least, in theory. This is because the individual cut principle reduces the optimization problem defined in a 2-dimensional path continuum to the one on a 1-dimensional streamline. However, the problem is computation-intensive as the variations of the local speed, the NEP function and the side step along the streamline increase. We also presented a heuristic approach to find an approximate solution.

3.7 Concluding Remarks: Structural Knowledge on the Problem and Search Methods

All the cumulated results, which we presented in this chapter, reveal the structure of the problem: once a direction field is given in a path continuum, all other unknowns are fixed in it. In this section, we discuss the implication of the results that were presented in this chapter, towards finding the numerical solutions through discretizing the problem. We make few suggestions on the search spaces that are suited for machining path optimization. In other words, we show how it is possible to proceed towards solving this rather difficult optimization problem. Of course, there are many alternatives for the discretization of our problem.

(1) The Direction Field together with Continuum Boundaries

We presuppose the continuum boundary and the direction field. Then, it is possible to evaluate the cutting time through the procedures given in this chapter. Of course, varying the continuum boundary and the direction field, we attempt to find the optimal paths. We need certain discretization of the direction field with a chosen basis functions. To give the variations on the continuum boundary, we need, at least, a heuristic criterion.

(2) The Direction Field, the Side Step and the Grain Boundaries

We tessellate the surface into "small" regions, each of which we regard as an (approximate) grain. On every edge of the tessellation, we assign variables that represent matching pairs. We assign variables in the grains to represent the direction field, too. (Of course, stream functions or weaving functions can serve this purpose). By including the case $(0, 0)$ for the matching pairs, we set the grain boundaries and the continuum boundaries in an equal footing. The compatibility equation is used to represent the relation between the side step and the direction field across the grain boundaries in a certain discretized form. The cutting time must be expressed with such discrete variables. And then, we use various techniques in mathematical programming to solve the problem. In this case, we use only the fastest cut principle to reduce the number of independent variables. Other principles are not invoked directly.

(3) Approaching From Continuous Vector Fields

We may even search for a continuous vector field ignoring the discontinuity of the optimal solution hoping that a continuous vector field could approach the solution within a certain resolution (in finitely many steps). In this case, finding the (best) continuum boundaries with a given vector field emerges as an auxiliary problem. (The algorithms for extracting the vector field topology are related closely to this problem, but not directly.) Even with this crude method, our framework is able to construct a diminishing sequence of cutting time rather practically, which is not straightforward in the conventional framework.

Summary

We use the structural knowledge on the problem for devising various numerical techniques that solve the machining path problem. We briefly sketched few approaches here. Each technique is characterized by how much such structural knowledge is used to set the (discretized) independent variables.

3.8 Summary

In this chapter, we presented the consequences that are drawn from the variational formulation of the machining path problem.

The realization of the actual tool paths from a given path structure is done resorting to the (generalized) shifting procedure. Across the direction field, we construct a transverse curve and find the intersection of the transverse curve with the next tool paths.

The characteristic theory for PDEs converts the compatibility equation to a system of ODEs. Given a direction field, the "relative side step" is determined along a streamline. In addition, the cusp height limit changes the side step into an "absolute scale." Therefore, the (optimal) side step in a continuum is fixed completely once a direction field is given. We call this the **widest cut principle**.

The actuator speed limit constraint fixes the (optimal) speed for a given direction angle. The velocity vector must be on the boundary of the velocity polygon in a tangent space of the surface. The relation is defined in a pointwise manner. We call this the **fastest cut principle**.

The optimization of the 2-dimensional continuum is decomposed into the optimization processes along individual streamlines. The cutting time per a streamline is the cost function of this 1-dimensional optimization problem. We call this the **individual cut principle**.

The three principles are "intuitively correct." Our variational formulation is converted back to the ODE form from PDEs, mathematically. The equivalence between them can be proved and it shows that the "continuum hypothesis", which we have made, is reasonable. Especially, the compatibility equation plays a critical role in keeping this kind of consistency.

Finding the pseudo-compatible points and their matching pairs in a given continuum is the most numerically-intensive subproblem. We solve the problem effectively for small number of pseudo-compatible points. Of course, we resort to the individual cut principle in solving this problem. Additionally, we presented a heuristic procedure, which we call the **path matching procedure**. The heuristics is that we remove tool paths in a region where tool paths clog too much.

Therefore, if (1) a continuum boundary and (2) a direction field are given, other unknown variables are determined there, either exactly or approximately. An intuitive approach is to regard them (or their discretized variables) as independent variables of our optimization problem. Giving variations on the continuum boundary is less straightforward. Besides the approach that uses the whole structural knowledge on the problem, it is possible to devise other methods that convert our variational problem to finite dimensional optimization problems. We briefly introduced a few of them.

4 Greedy Tool Paths

—A FIRST CUT FOR THE TIME OPTIMAL TOOL PATH—

The general problem formulated in Chapter 2 is the focus of long term research. It is challenging to solve analytically or numerically. The greedy approach, described below, attempts to find an approximate solution using certain local information. The idea is to pick, at an array of sample points on a surface, **directions** in which the tool path performs best from a material (or area) removal rate point of view while respecting the actuator speed limits, and to generate streamlines by integrating the direction field.

In Section 1, we define what the locally best directions are and show how to find them at each point on the surface. We make the analysis on the tangent space at a point on the surface, where velocity polygons and local normal curvatures of the surface reside. In Section 2, we generalize the notion of the stream function (or the weaving function) and define the generating function. We use it to fit a direction field to the sampled "best" directions. In Section 3, we define the notion of maximal basins. Continuum boundaries must be extracted after finding the desirable direction field to find the "start points" of streamlines. Finding the "best" locations for continuum boundaries is the least straightforward part in our problem. Instead of solving the problem exactly, we use maximal basins as the continua. After partitioning the surface into basins, it is straightforward to generate the tool paths. We briefly explain how it works in Section 4; we basically use the principles and procedures developed in the previous chapter. Finally, we show an example of greedy tool paths.

Greedy tool paths are not necessarily optimal because we neglected the compatibility (and acceleration limits) in finding the locally best directions. The study on how well the greedy tool paths work is the topic of the next chapter. At least, the greedy tool paths prune the general search and tend to outperform arbitrarily generated tool paths.

4.1 Generating a Greedy Direction Field

In this section, we define what the greedy directions are, and describe how we find them. We also show a piecewise continuous vector field that the greedy directions form.

1. Greedy Directions

A local measure of material removal rate is what we define as the *sweep rate*: the product of the speed and the side-step-limit. The sweep rate is denoted by Γ_o, namely $\Gamma_o \equiv \vartheta \cdot w_o$. The direction of maximum sweep rate is also the direction of greatest area coverage locally, and effectively, greatest material removal. (We will discuss the implications of locally maximizing this performance index in the next chapter.)

The localized problem, which we will solve for the best directions, is stated as follows:

MAXIMIZE the sweep-rate $\Gamma_o(\equiv \vartheta \cdot w_o)$, **SUBJECT TO** the speed limit constraints $|\omega_i| \le \omega_o^i$.

As discussed in Subsection 2.5.2, the speed limit constraints ($|\omega_i| \le \omega_o^i$) form a *velocity polygon* in the $(\dot{u}, \dot{v})$-space, (and at the same time, in $(\underline{\dot{u}}, \underline{\dot{v}})$ space being transformed by the modal matrix $\boldsymbol{P}$). As shown in Figure 24, we need to find the point on the velocity polygon with the highest sweep rate. Such points on the boundary of a velocity polygon are called *binding points* and the corresponding directions are called *greedy directions.* We will refer to the greedy directions as η^g. By symmetry, there is always at least one pair of *greedy directions* as shown in Figure 24. The ellipse or dumbbell shaped curves, shown in Figure 24 and Figure 25, are *iso-sweep-rate contour lines*, namely $\Gamma_o = const$ on them. In terms of the polar coordinates of the tangent space at $(u, v) \in \mathcal{P}$, the equation for an iso-sweep-rate contour line is $\vartheta = \Gamma_o / w_o(\eta, u, v)$. To find the binding points, we seek for a point on the velocity polygon, whose iso-sweep-rate contour line encloses the velocity polygon.

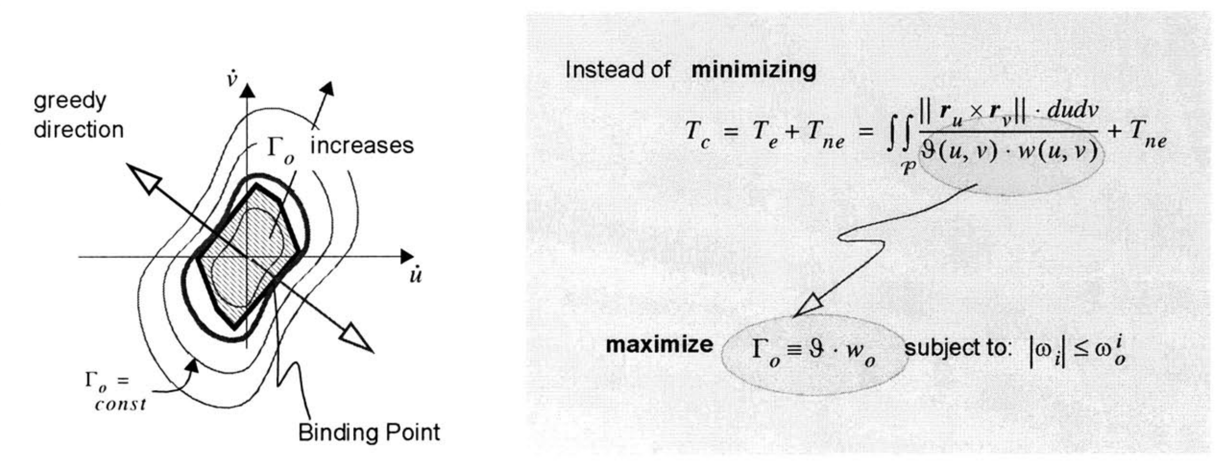
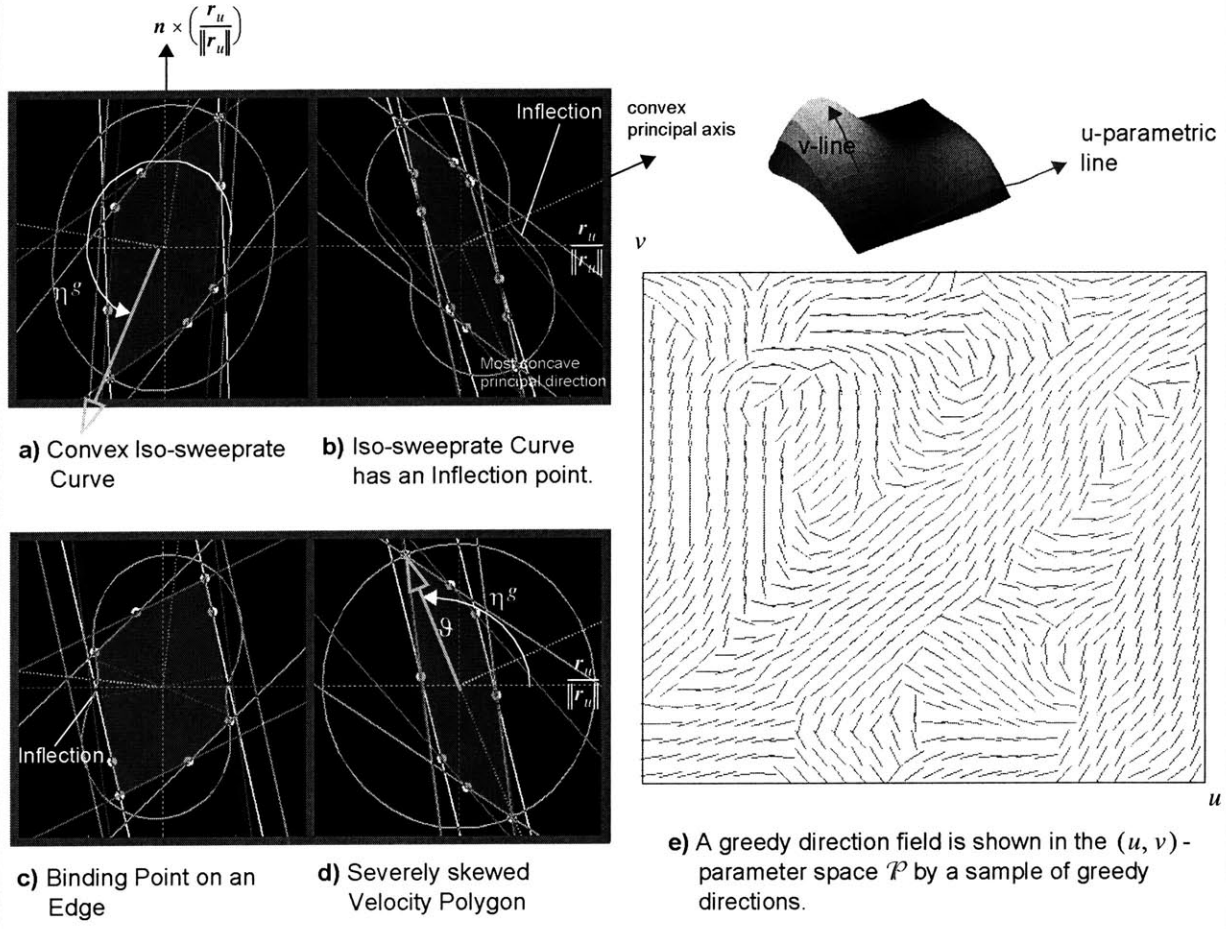

greedy direction
$\dot{v}$
Γ_o increases
$\dot{u}$
$\Gamma_o = const$
Binding Point
Instead of minimizing
$T_c = T_e + T_{ne} = \iint_{\mathcal{P}} \frac{\|r_u \times r_v\| \cdot dudv}{\vartheta(u,v) \cdot w(u,v)} + T_{ne}$
maximize $\Gamma_o \equiv \vartheta \cdot w_o$ subject to: $|\omega_i| \le \omega_o^i$

Figure 24 Localization of the Problem



$n \times \left(\frac{r_u}{\|r_u\|}\right)$
η^g
Inflection
$\frac{r_u}{\|r_u\|}$
Most concave principal direction
convex principal axis
v-line
u-parametric line
v
a) Convex Iso-sweeprate Curve
b) Iso-sweeprate Curve has an Inflection point.
Inflection
η^g
ϑ
$\frac{r_u}{\|r_u\|}$
u
c) Binding Point on an Edge
d) Severely skewed Velocity Polygon
e) A greedy direction field is shown in the (u, v)-parameter space $\mathcal{P}$ by a sample of greedy directions.

Figure 25 Velocity polygons, iso-sweep-rate contour lines and a greedy direction field

NOTE: In the drawings above, we transformed velocity polygons and the iso-sweep-rate contour lines from the $(\dot{u}, \dot{v})$-space into the **tangent plane** (or the $(\underline{\dot{u}}, \underline{\dot{v}})$-space) of a surface through the **linear transformation** (or the modal matrix) $\boldsymbol{P}$; in the transformed space, the norm (distance from the origin) corresponds to the speed. The directions of the principal curvatures are also shown; the iso-sweep-rate contour lines keep mirror symmetry about the principal axis. *Notice how anisotropic the performance envelope (velocity polygon) can be. The data shown is for a Hexapod machine Tool. All 5-axis machines have anisotropy.*

2. Finding Greedy Directions Efficiently

As shown in Figure 25, *iso-sweep-rate contour lines* tend to look flattened along the most convex direction. At "very hyperbolic" points [113], where *Gaussian curvatures* $\kappa_1\kappa_2$ are negative, the distortion can be severe and it is possible that iso-sweeprate contour lines have inflection points as shown in Figure 25-(*b*). If an iso-sweeprate contour line is convex as shown in Figure 25-(*a*) and (*d*), only vertices are candidate binding points. If it has inflection points, the binding point can be on an edge of the velocity polygon as shown in Figure 25-(*c*). Typically, inflection is rare and the binding at an edge is even more infrequent. We therefore first determine whether the iso-sweeprate contour lines are convex; only if they are non-convex, we search the concave range for candidate binding points on edges. This strategy yields an efficient way to determine the binding points or the greedy directions. More on this procedure is found in **APPENDIX C**.

Remark and Review:* By the fastest cut principle (as explained in Section 3.4), the speed ϑ, if it is optimal, must satisfy the following equation:

$$\vartheta = \vartheta_o(\eta, u, v)$$

where η is the direction angle and ϑ_o is the function that returns the maximum speed for a given direction angle as defined in Section 3.4. Also, recall that the side-step-limit w_o is defined as a function in the following form (Equation 22):

$$(\eta, u, v) \to w_o(\eta, u, v).$$

Therefore, maximizing the sweep rate Γ_o subject to the speed limit is equivalent to minimizing

$$\Gamma_{oo}(\eta, u, v) \equiv \vartheta_o(\eta, u, v) \cdot w_o(\eta, u, v)$$

with respect to the direction angle η for a given point $(u, v) \in \mathcal{P}$. We didn't explain the problem in that way because the geometric approach, as presented in this section, provides more insights.

3. The Greedy Direction Field

A sampling of greedy directions is shown in Figure 25-(*e*). The greedy directions η^g as determined above can form a **piece-wise continuous** vector field

$$[h_1(\eta^g(u, v), u, v), h_2(\eta^g(u, v), u, v)]^T$$

on the parameter space $\mathcal{P}$, where h_1 and h_2 were defined in Equation (6). The basic idea now is to construct its streamlines by numerical integration. We will choose from these streamlines to define tool paths, appropriately spaced to satisfy the cusp height limit constraint. In the following sections, we describe how this is achieved.

Remark: We point out that we have freedom in choosing one direction from the two equally preferable greedy directions, η^g and $\eta^g + \pi$.

4.2 Fitting a Continuous Vector Field to Sampled Directions: Generating Functions

In the previous section, we discussed how to generate a field of preferable cutting directions using the greedy local approximation. Streamlines of the field will yield tool paths.

Unfortunately, it is not straightforward to generate streamlines of a piecewise continuous field that is generated by the greedy directions. There are two reasons for this. First, finding a right set of initial (and final) positions for the integration is complicated. In other words, a direction field on a surface is not enough for specifying a **path structure** that also includes the continua, the side step, the speed and the weaving function in its definition. Especially, it is a difficult problem to find reasonable continuum boundaries for a given discontinuous direction field. Second, more fundamentally, there are two directions that are equally preferable; the tool can move either forward or backwards in the greedy direction. The question, now we ask, is how the greedy direction field η^g can be approximated by a convenient field.†

To circumvent these difficulties, we fit a continuous vector field to the greedy direction field using a least squares approximation. By squaring directional discrepancies, we suppress the ambiguity in the two equally preferable directions. We present the simplest way to achieve the fit among many possible alternatives. We further restrict the fitted field requiring the streamlines of the fitted field to be the contour lines of a continuous function on $\mathcal{P}$, which we call a *generating function*. This approach of fitting a field with level sets of a generating function is convenient because the algorithms for finding the level curves of a function are well developed in the field of computational geometry [*e.g.*, 150 and 151]. There is, however, a limitation on the classes of vector fields that can be generated by this method; for example, spiral paths are excluded from the candidate paths.

We define the generating function in the immediately following subsection. In Subsection 2, we describe linear combinations of *bi-cubic Bernstein polynomials*, which we will use as our generating function. In Subsection 3, we describe a weighted least-squares procedure for fitting.

1. The Generating Function in a Conservative Path Structure

We define the notion of generating functions as follows:

> A CONSERVATIVE (geometric/kinematic/dynamic) PATH STRUCTURE is a (geometric/kinematic/ dynamic) path structure whose weaving function is **smooth** and **continuous** on the surface $\mathcal{S}$, or equivalently, on the parameter space $\mathcal{P}$. The weaving function in the conservative path structure is called its GENERATING FUNCTION. We denote the generating function by $V(u, v)$.

By definition, generating functions keep all the properties of a weaving function. First of all, their level lines must be streamlines of the path structure, namely

† This problem is seemingly similar to vector field interpolation that has received some attention in the fluid mechanics community, where discrete vector data are often generated from experiments or simulations and fitted to continuous vector fields [146, 147, 148 and 149, *c.f.* 160 and 161].

$$\begin{Bmatrix} \cos\eta \\ \sin\eta \end{Bmatrix} = \frac{\boldsymbol{R}^T\boldsymbol{P}^T\nabla V}{\sqrt{\nabla V^T\boldsymbol{P}\boldsymbol{P}^T\nabla V}} \qquad \text{where} \quad \nabla V \equiv \begin{Bmatrix} V_u \\ V_v \end{Bmatrix} \tag{35}$$

$\boldsymbol{P}$ is the modal matrix and $\boldsymbol{R}$ is the rotation by the right angle (*c.f.* Equation 18, 19 and A-43).

A conservative path structure has limitation in that it cannot represent "point sinks" in a fluid mechanics term. Especially, spiral paths are not accommodated in a conservative structure. Just for its simplicity, we fit a generating function to the greedy directions and attempt to find a conservative path structure.

2. Generating Functions spanned by Bi-cubic Bernstein Polynomials

We reduce the degree of freedom of the generating function with finitely many *Bernstein basis functions*. For simplicity, we consider only *bi-cubic* cases. The method is, however, readily applicable to the cases of other orders and even to non-polynomial bases.

We presuppose the form of the generating function:

$$V(u, v) = \sum_i \sum_j V_{ij} \cdot B_{i,3}(u) \cdot B_{j,3}(v) \qquad (i, j = 0, 1, 2, 3, V_{ij} \in \mathcal{R})$$

where $B_{i,3}(u) \equiv {}_3C_i \cdot u^i \cdot (1-u)^{3-i}$. That is, a generating function is represented by the 16 parameters, $V_{ij} \in \mathcal{R}$.

It should be pointed out that neither translation nor dilation of a generating function that is built from polynomials changes the direction field. To prevent the generating function from floating up and down, we fix a parameter: $V_{00} = 0$. This is equivalent to fixing $V(0, 0) = 0$. To prevent arbitrary dilation, we fix a norm. Specifically, we add the following two constraints:

$$\sum\sum V_{ij}^2 = 1 \qquad \text{and} \qquad V_{00} = 0.$$

3. A Weighted Least Squares Method

In this section, we show how we can fit a continuous direction field to the greedy directions using the Bernstein polynomials as approximants. Any fitting or approximation procedure must be based on a measure of discrepancy. Least-squares methods are well known in the approximation of data. Here, we fit fields generated from a Bernstein basis, and use the angular deviation $\Delta\eta_k$ as the measure of error at the k^{th} sample point $\boldsymbol{u}_k$. We weight the deviations with a weighting α_k at each sample point. Below, we sketch briefly the rationale for the formulation of these two terms, and the aggregated objective function to be minimized.

Weighted Least Squares Error

For small values, we use the approximation $\Delta\eta_k \approx \sin\Delta\eta_k$. The weighted error can then be viewed as:

$$\frac{\sum_k \alpha_k (\Delta\eta_k)^2}{\sum_k \alpha_k} \approx \frac{\sum_k \alpha_k \sin^2\Delta\eta_k}{\sum_k \alpha_k} \equiv E .$$

Since $\sin^2\Delta\eta_k = 1 - \cos^2\Delta\eta_k$, the objective can be rewritten as that of maximizing $1 - E$, which is given by:

$$\lambda \equiv 1 - E = \frac{\sum_k \alpha_k \cos^2\Delta\eta_k}{\sum_k \alpha_k} . \qquad (36)$$

This is merely a manipulation to simplify the final form.

Weighting Factor α_k

The criteria for α_k are first, that it be positive, and second, that it lead to a convenient final form. Without derivation, we state the following expression for α_k:

$$\alpha_k = \nabla V^T \cdot \boldsymbol{P}\boldsymbol{P}^T \cdot \nabla V|_{\boldsymbol{u}_k}$$

where $\boldsymbol{P}$ is the modal matrix, ∇V ($\equiv [V_u \quad V_v]^T$) is the "gradient" of the generating function and $\boldsymbol{u}_k$ is the k^{th} sample point in the parameter space $\mathcal{P}$.

> **HINT:** This term is the square of the denominator of the unit velocity vector as shown in Equation (35). It allows the terms in our objective λ, as shown in Equation (36), to be summed into a convenient form.

Final Form of the Objective

It can then be shown that the objective function to be maximized is given as a *Rayleigh quotient*:

$$\lambda = \frac{\mathbf{v}^T \cdot \boldsymbol{I} \cdot \mathbf{v}}{\mathbf{v}^T \cdot \boldsymbol{II} \cdot \mathbf{v}}$$

where $\mathbf{v} = [V_{01}\ V_{02}\ \dots\ V_{33}]^T$. The matrices $\boldsymbol{I}$ and $\boldsymbol{II}$ are given by

$$\boldsymbol{I} \equiv \sum_k [(\nabla \boldsymbol{B})^T (\boldsymbol{b}\boldsymbol{b}^T)(\nabla \boldsymbol{B})]_{\boldsymbol{u} = \boldsymbol{u}_k}, \qquad \boldsymbol{II} \equiv \sum_k [(\nabla \boldsymbol{B})^T (\boldsymbol{P}\boldsymbol{P}^T)(\nabla \boldsymbol{B})]_{\boldsymbol{u} = \boldsymbol{u}_k},$$

$$\boldsymbol{b} \equiv \begin{bmatrix} h_1(\eta^g(u,v) + 90^o, u, v) \\ h_2(\eta^g(u,v) + 90^o, u, v) \end{bmatrix}, \quad \nabla \boldsymbol{B} \equiv [\nabla B_{01}\ \nabla B_{01}\ \dots\ \nabla B_{33}]\ , \quad \nabla B_{ij} \equiv \begin{bmatrix} B'_{i,3}(u) \cdot B_{j,3}(v) \\ B_{i,3}(u) \cdot B'_{j,3}(v) \end{bmatrix}$$

where we denoted the derivatives by primes.

MINIMIZATION OF THE ERROR

It is well known that the maximum of the Rayleigh quotient $\lambda(\mathbf{v})$ is the maximum *eigenvalue* of the following eigen-system:

$$\boldsymbol{I} \cdot \mathbf{v} = \lambda \cdot \boldsymbol{II} \cdot \mathbf{v}.$$

By finding the *eigenvector* corresponding to the maximum eigenvalue, we solve our problem. We can always normalize an eigenvector satisfying the constraint $\|\mathbf{v}\| = \sum\sum V_{ij}^2 = 1$. This is a standard procedure in optimization.

4. Summary

In this section, we have described the mechanism of fitting a vector field to the greedy directions with a least squares approximation. The use of Bernstein bases, the error measures and the manipulations shown reduce the problem to a form for which standard tools are applied, reducing it to a tractable problem.

Below, we show an example of this fitting method. A bi-quadratic and a bi-cubic generating function were used. Note that the level lines of the generating functions are the streamlines.

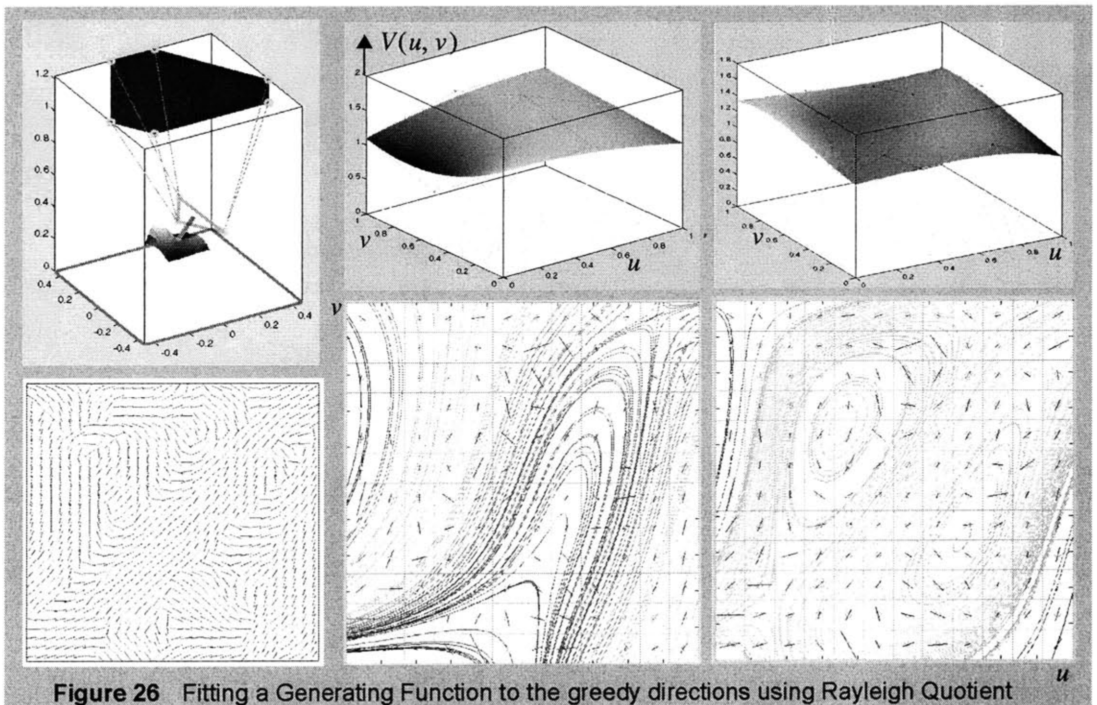


Figure 26 Fitting a Generating Function to the greedy directions using Rayleigh Quotient

4.3 Extracting Basins From A Generating Function: Optimal Continua

In this section, we assume that a direction field $\eta(u, v)$ has been generated as described in the previous section. Streamlines of the direction field are only the skeletal guidelines for the tool paths. They do not, however, completely define the tool paths (or the path structure) because the path continua and the side step $w(u, v)$ remain undecided. We need boundaries of the path continua when we place tool paths with the shifting procedure. Finding the "exact" optimal continuum boundary is a complicated problem. Instead, we will use maximal basins as the continua, which we define in this section. In a loose way, basins can be related to *basins of attraction* in dynamic systems. In fact our terminology draws inspiration from the field of dynamics.

We start by defining notions of basins, a partition into basins, a minimal partition into basins and maximal basins. All the definitions are made with the understanding that a generating function (and, as a result, a direction field) is known. And we briefly mention how we find the maximal basins for a given generating function. Finally, we explain how it is possible to extend the individual cut principle in this situation (of a conservative path structure) to find the optimal path continua. It is observed that we greatly simplified the problem by considering only conservative path structures even though the problem still remains "combinatorial."

1. Maximal Basins

Streamlines have start points and end points. Start points may occur at the boundary of the surface being considered, or at line (or point) sources inside the boundary. End points too can occur at the boundary or at line (or point) sinks. Given a direction field $\eta(u, v)$, we refer to **connected** curves formed by "start points" of its streamlines as inlets. Start points occur at the boundary of the region within which the tool path is being generated. Outlets are the curves where streamlines stop. Choosing *inlets* and *outlets* in a certain way, we partition a surface. Each region is characterized by a connected inlet and a connected outlet, and it is each such region that we refer to as a *basin*. We refer to a collection of basins, which partition a surface, as a *partition into basins*. When the basins in a partition are selected so that no two basins can be combined to create another basin, we call the partition *minimal* and the basins in it maximal. In a more formal way, we define *maximal basins* by the following series of definitions:

A BASIN $\mathcal{B}$ (subordinate to a direction field $\eta(u,v)$ or a generating function $V(u,v)$) is a simply connected open set on the parameter space $\mathcal{P}$ (or the designed surface $\mathcal{S}$) which satisfies the following two conditions:

(1) The subset of boundary $\partial\mathcal{B}$ of the basin $\mathcal{B}$ through which streamlines of the direction field (or the level lines of the generating function) flow into the basin $\mathcal{B}$ is **connected**. Such a subset of the boundary is called the INLET of the basin. Similarly, we define the OUTLET of the basin and it must be connected.

(2) The length of the streamlines is continuous along the inlet of the basin.

A PARTITION (INTO BASINS subordinate to a direction field) is a class $\{\mathcal{B}_i, i=1\ldots N\}$ of basins $\mathcal{B}_i$ (subordinate to the direction field) that partition the surface (in that $\cup\, cl(\mathcal{B}_i) = \mathcal{P}$ and

$\mathcal{B}_i \cap \mathcal{B}_j = \varnothing$, if $i \neq j$).

A REDUCTION OF A PARTITION $\{\mathcal{B}_i, i=1...N\}$ (INTO BASINS subordinate to a direction field) is another partition $\{\mathcal{B}_i', i=1...N'\}$ (into basins subordinate to the direction field) that can be generated from the partition $\{\mathcal{B}_i, i=1...N\}$ by one of the following two operations:

(1) Combine two basins in $\{\mathcal{B}_i, i=1...N\}$ and let the combined region be a basin in the new partition. Of course, the combined region must be a basin. By this operation, the new partition has one less element than the original partition.

(2) Divide a basin in $\{\mathcal{B}_i, i=1...N\}$ by a streamline into two regions, keep one of them as a basin and combine the other with another basin in such a way that the combined region is a basin in the new partition. By this operation, the number of basins does not change.

A MINIMAL PARTITION into basins is a partition whose reduction is impossible to be made.

Finally, we define maximal basins simply in the following way:

MAXIMAL BASINS are elements of a minimal partition.

2. The Critical Points and Turning Points of a Generating Function

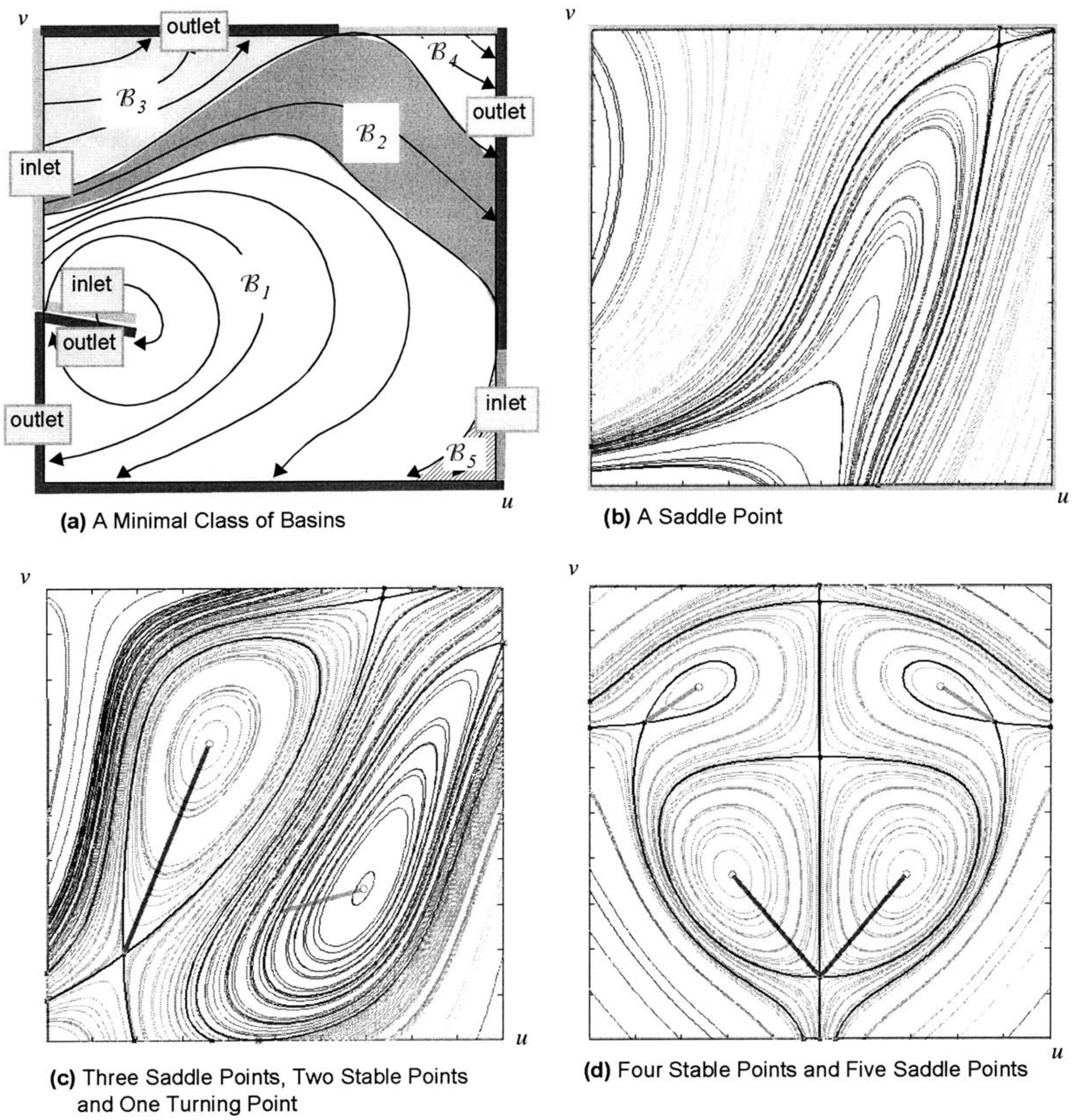
The basins shown in Figure 27-(*a*) are maximal. Inspecting the figure, we realize that the critical points and the turning points of the given generating function are the important cues for finding the minimal partition into basins. We do not go into detail here of how to determine the maximal basins; several algorithms related to this maximal basin problem, are known for contour drawing and for vector field visualization [*c.f.* 150, 152, 153 and 154]. In the same figure, few more examples for the maximal basins are shown. We point out that the minimal partition is not unique.

Within a (maximal) basin, we can integrate the given direction field and find streamlines that connect the inlet of the basin with its outlet. Instead of finding exact optimal continua, we will use the maximal basins as the continua of the path structure in the greedy approach. (A slightly more comprehensible approach is to combine some maximal basins and construct a continuum as large as possible.) The next section provides the procedure that we will take with the greedy approach using the basins while the immediately following subsection briefly describes how it is possible to find the optimal continuum "almost exactly."

3. The Individual Cut Heuristics for Maximal Basins*

We write down the cutting time again, as defined in Equation (20 and 21, Section 2.4), in abbreviated form:

$$T_c = \iint_{\mathcal{P}} \frac{\|r_u \times r_v\| \cdot dudv}{w(u,v) \cdot \vartheta(u,v)} + \frac{1}{2} \cdot \left\{ \int \frac{\tau_o}{w} \cdot (1-\gamma) \cdot |(n \times t)^T dr| + \oint \frac{\|dr\|}{V_o} \right\}.$$



(a) A Minimal Class of Basins

(b) A Saddle Point

(c) Three Saddle Points, Two Stable Points and One Turning Point

(d) Four Stable Points and Five Saddle Points

Figure 27 Maximal Basins and Minimal Partitions into Basins

As explained in Subsection 2.4.2, the main purpose of the last term for the non-effective cutting time is to prevent the path structure from developing too narrow continua. Once the direction field is given and the maximal basin boundary is checked to be much shorter than the total length of the tool paths, it is not unreasonable to neglect the last term marked in the above equation. *Under this ideal situation, the* **individual cut principle**, *as shown in Section 3.5, can be applied for each maximal basin (as opposed to the continuum) by extending the feasibility set for the matching pairs from Q'' to $Q' \equiv Q'' \cup \{(0, 0)\}$*. One more adjustment that we need is to take account of the "periodicity" of the "orbits," which does not increase the complexity of the problem significantly. This general search is mentioned here, just for completeness. In our first analysis, we regard a maximal basin as a continuum and, as a result, take Q'' for the feasibility set instead of the extended feasibility set.

4.4 Determining Side Step and Eliminating Redundant Paths

In this section, we also assume that a direction field $\eta(u, v)$ and its maximal basins have been generated as described in the previous section. The remaining task is to find another unknown function, the side step $w(u, v)$, and place tool paths in each maximal basin. In fact, we already developed all the necessary procedures in the previous chapter. We briefly review the key procedures.

1. Path Placement

The problem that we are now concerned with is how to determine a "seed point" (or a shifted point) from which the next tool paths can be constructed by forward and backward integration. In Section 3.1, we, in fact, showed the **generalized shifting procedure** using transverse curves to answer this question. The simplest way is to choose a straight line in the parameter space $\mathcal{P}$ for the transverse curve which is perpendicular to the previous tool path on the surface. The procedure is essentially similar to the one presented in the literature by other researchers [*c.f.* 18, 22 and 28]. Unlike other approaches, we can obtain higher order formulas because we took the "continuum" model.

2. The Reference Side Step

For a given direction field $\eta(u, v)$ in a basin, we tentatively assume that the matching pairs are all $(1, 1)$ in a basin and use the following equation in order to determine the reference side step:

$$w_{ref}(s) = inf\,\{w_o(s)/exp(\int_0^s h_3(s)ds)\ :\ 0 \le s \le L\} \cdot exp(\int_o^s h_3 ds)$$

where s is the arc length parameter of the streamline measured from the inlet curve and L is the arc length of the streamline (Equation 31 and 33). We refer to the tool paths that we place from the reference side step as *reference tool paths* and the corresponding cutting time as *reference cutting time*.

A DETAIL*

In performing the integration $\int_o^s h_3 ds$ in the above equation, it is convenient to evaluate the term h_3 directly from the generating function $V(u, v)$ instead of from the direction field $\eta(u, v)$. We use the following formula (Equation A-55):

$$h_3 = \frac{\nabla V^T \boldsymbol{W} \nabla V}{2 \cdot f^{3/2} \cdot g} \qquad \text{where} \quad \nabla V \equiv \begin{Bmatrix} V_u \\ V_v \end{Bmatrix}, \qquad G^{ad} = \begin{bmatrix} g_{22} & -g_{12} \\ -g_{21} & g_{11} \end{bmatrix},$$

$$\boldsymbol{W} \equiv G^{ad}\,\{2g\boldsymbol{HR} + (g_u \cdot V_v - g_v \cdot V_u) \cdot \boldsymbol{1}\} - g\{V_v G_u^{ad} - V_u G_v^{ad}\},\ f \equiv \nabla V^T G^{ad} \nabla V,\ g \equiv Det(\boldsymbol{G}),$$

$\boldsymbol{H}$ is the Hessian matrix of the generating function $V(u, v)$, $\boldsymbol{G}$ is the 1st fundamental form coefficient matrix and $\boldsymbol{R}$ is the rotation by the right angle. The step by step derivation is provided in APPENDIX **A.2**.

3. Path Elimination based on Path Matching

We improve the cutting time of the reference tool path following the **path matching procedure** as defined in Section 3.6. The procedure finds an approximate solution for the side step. Essentially, we eliminate some sections of the reference tool paths in a maximal basin based on a certain criterion. Loosely speaking, we achieve the path matching by selecting every n^{th} path and eliminating other paths in the region where $n - 1 < \mu_{ref} < n$ (for a natural number n). Note that μ_{ref} is **the reference overlap degree**, which we defined as $\mu_{ref} \equiv \log_2(w_o / w_{ref})$. The overlap degree can be thought of as a measure of inefficiency of tool paths.

4. Construction of Non-effective Movements

After generating the effective movements (or the streamlines), we construct the non-effective movements by connecting an end of a previous tool path with a next tool path. We do this in another greedy fashion. We pick an end of an effective movement and find the nearest end point of another tool path until all the tool paths are connected.

5. Summary of the Procedure

The following steps summarize the procedures developed in this chapter:

```
(GREEDY TOOL PATHS GENERATION)

    1. Generate an array of sampling points on the designed surface and find the greedy
       direction at each point.
    2. Fit a direction field $\eta(u, v)$ to the data generated in Step 1.
    3. Find a minimal partition $\mathcal{T}$ (= $\{\mathcal{B}_i, i=1...N\}$) of the field $\eta$.

For each maximal basin $\mathcal{B}_i \in \mathcal{T}$, do
    4. Generate the reference tool paths;
    5. Eliminate the sections of the tool paths considering path matching whenever
       possible,
until all the maximal basins in $\mathcal{T}$ are visited.

    6. Construct the non-effective movements; match the ends of a path with the
       beginning of the next path in an efficient way.
```

4.5 Examples of Greedy Tool Paths

Example 1

We have implemented the procedures described in this chapter and applied it to a *bi-quartic Bezier tensor product surface patch* to be machined with a hexapod 6-axis machine tool. Figure 28-(*0*) shows the machine tool and the designed surface. About $24cm \times 24cm$ area will be machined. The following properties were set to define the capability of the machine tool:

The speed limit of each link	$\omega_o^i = 6cm/s$
The tool length	$l = 15.14cm$
Ball radius	$R = 0.5\ cm$
Averaged acceleration limit	$a_{max} = 0.6g$.

Note that the acceleration limit is used only for evaluating non-effective cutting time. More detailed dimensional information about the machine tool and the designed surface is given in Appendix **B** and Appendix **D**. In this example, a rather large cusp height limit, $h_o = 1mm$, was set to make it possible to visualize the solution. All the examples are for the surface normal machining, *i.e.* $\mathbf{q} = \mathbf{n}$. Each step is shown in Figure 28, which is:

Figure 28

(1) Pick greedy directions at 12×12 sampled points in the parameter space $\mathcal{P}$.

(2) Fit a generating function to the data. (The generating function and its level lines are shown in the figure. The level lines are the streamlines. Bi-cubic basis functions were used for this fitting.)

(3) Find a minimal partition of the field.

(3-1) Find the critical and turning points of the generating function.

(3-2) From the critical and the turning points, extract the basin boundaries.

(3-3) The basin boundaries are shown on the designed surface.

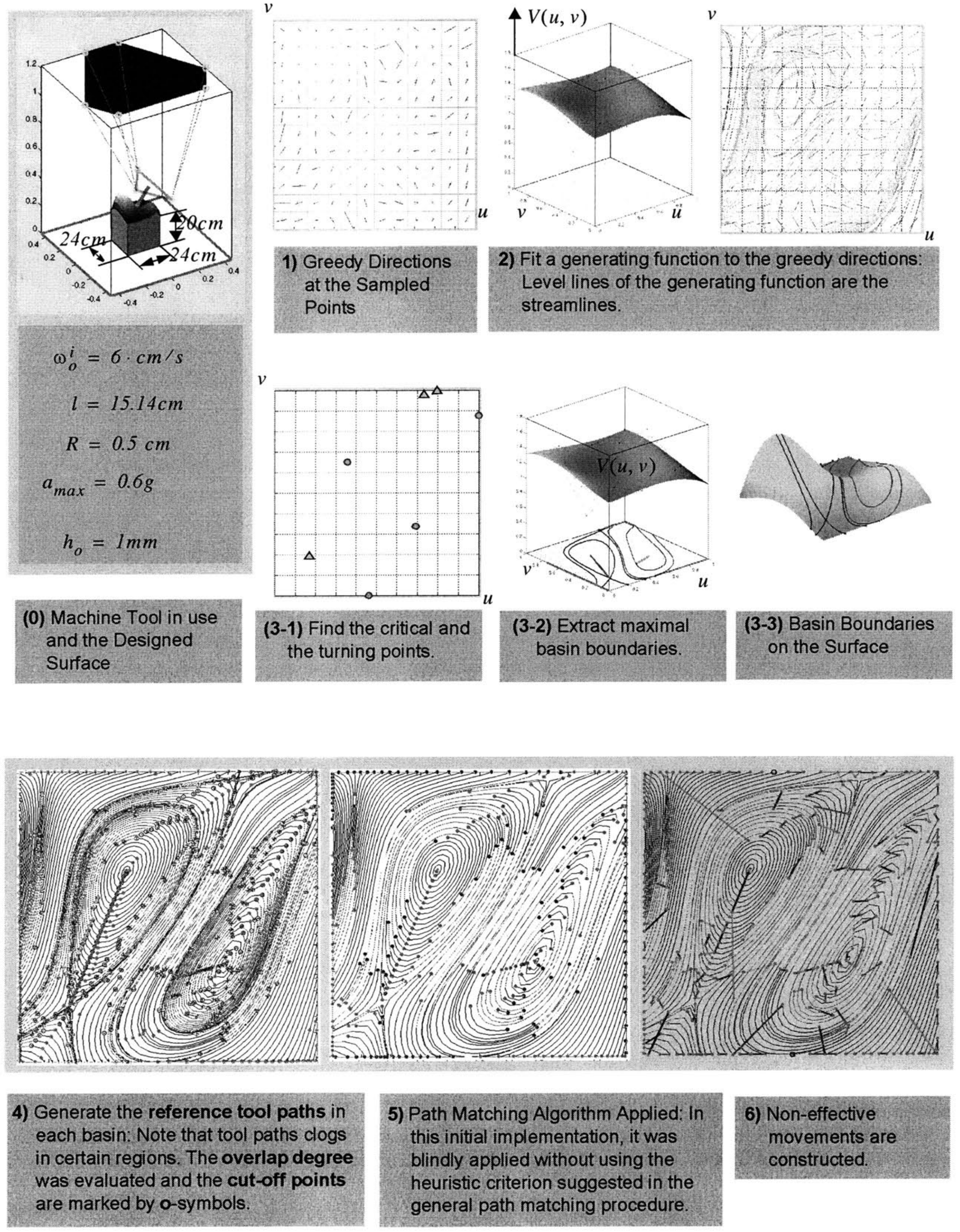
For each maximal basin, **do**

(4) Generate the reference tool paths in each basin;

(5) Eliminate the sections of the tool paths considering path matching whenever possible,

until all the maximal basins in the partition are visited.

(6) Construct the non-effective movements; match the ends of a path with the beginning of the next path in an efficient way.



$\omega_o^i = 6 \cdot cm/s$

$l = 15.14cm$

$R = 0.5\ cm$

$a_{max} = 0.6g$

$h_o = 1mm$

(0) Machine Tool in use and the Designed Surface

1) Greedy Directions at the Sampled Points

2) Fit a generating function to the greedy directions: Level lines of the generating function are the streamlines.

(3-1) Find the critical and the turning points.

(3-2) Extract maximal basin boundaries.

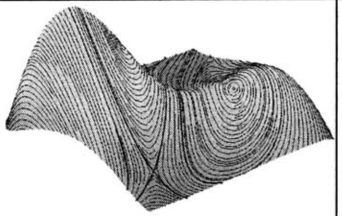
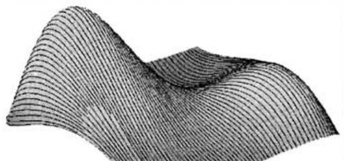
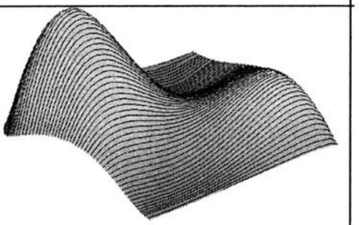
(3-3) Basin Boundaries on the Surface

4) Generate the **reference tool paths** in each basin: Note that tool paths clogs in certain regions. The **overlap degree** was evaluated and the **cut-off points** are marked by **o**-symbols.

5) Path Matching Algorithm Applied: In this initial implementation, it was blindly applied without using the heuristic criterion suggested in the general path matching procedure.

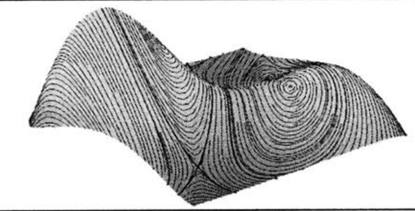
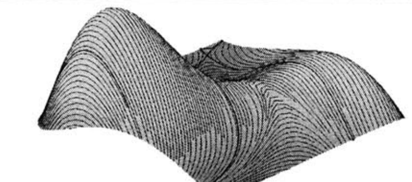
6) Non-effective movements are constructed.

Figure 28 Greedy Tool path Generation (To be continued in the next page)

(Con't) Greedy Tool path Generation	**7)** Greedy surface tool paths		**8)** Iso-u-paths		**9)** Iso-v-paths	
						
$h_o =$ $1mm$	$T_e = 534\ sec$	$T_{ne} = 57\ sec$	$T_e = 606\ sec$	$T_{ne} = 12\ sec$	$T_e = 674\ sec$	$T_{ne} = 13\ sec$
	$T_c = 590\ sec$		$T_c = 618\ sec$		$T_c = 686\ sec$	
	Advantage		5% less time		14% less time	
$h_o =$ $0.1mm$	$T_e = 1426\ sec$	$T_{ne} = 89\ sec$	$T_e = 1805\ sec$	$T_{ne} = 23\ sec$	$T_e = 2013\ sec$	$T_{ne} = 28\ sec$
	$T_c = 1514\ sec$		$T_c = 1827\ sec$		$T_c = 2041\ sec$	
	Advantage		17% less time		26% less time	

In Figure 28-(7), we present the surface tool paths which can be constructed by the surface parameterization. In (8), the *iso-u-paths* are shown; we can apply the same algorithm by setting the generating function such that $V(u, v) = u$. The *iso-v-paths* are generated by setting $V(u, v) = v$. Because path matching algorithm was applied to these two iso-parametric paths, the iso-parametric paths shown in the figure outperforms "usual" iso-parametric paths. We evaluated the cutting time for each tool path scheme. 5% and 14% of cutting time can be saved. In this example, the non-effective cutting time of the greedy path is higher than that of the iso-parametric paths. However, if we use a more realistic cusp-height-limit, the ratio T_{ne}/T_e is reduced significantly, and the greedy approach becomes more advantageous (roughly, $T_{ne}/T_e \sim (const) \cdot h_o^{1/4}$, $T_e \sim (const) \cdot h_o^{-1/2}$ and $T_{ne} \sim (const) \cdot h_o^{-1/4}$). Reducing the cusp height 10 times, we generated the tool paths. Now, 17% and 26% of cutting time is saved as shown in the table.

We provide another example in Figure 30 using the bi-quadratic generating function. Other parameters are set to be the same as the previous example. In Figure 29, we compare this with the previous bi-cubic case. The bi-cubic generating function performs slightly better for the small cusp-height-limit while the opposite happens for the large cusp-height-limit.

Figure 29 Bi-cubic vs. bi-quadratic Fitting				
$h_o = 1mm$	$T_e = 534\ sec$	$T_{ne} = 57\ sec$	$T_e = 518\ sec$	$T_{ne} = 38\ sec$
	$T_c = 590\ sec$		$T_c = 555\ sec$	
$h_o = 0.1mm$	$T_e = 1426\ sec$	$T_{ne} = 89\ sec$	$T_e = 1496\ sec$	$T_{ne} = 60\ sec$
	$T_c = 1514\ sec$		$T_c = 1555\ sec$	

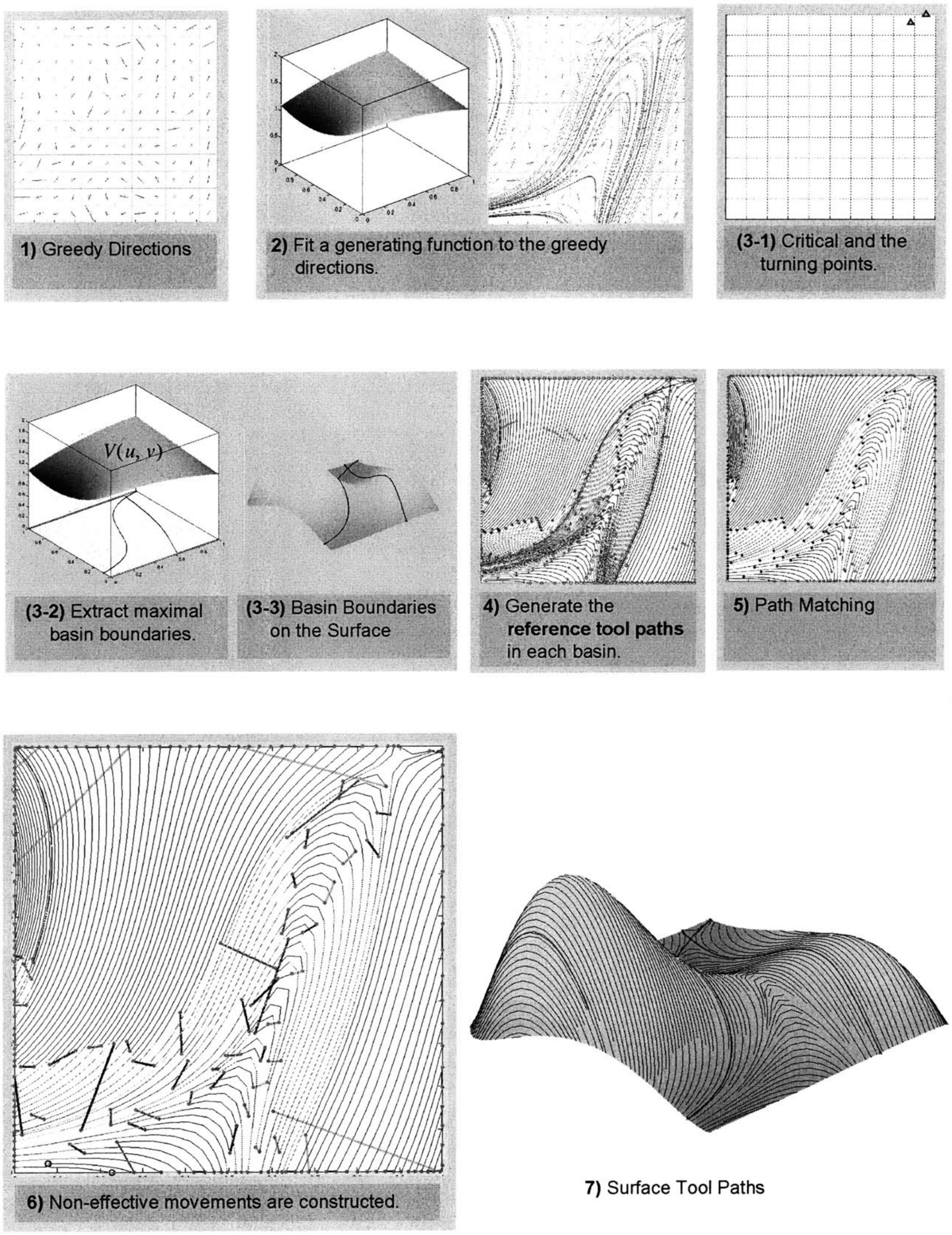
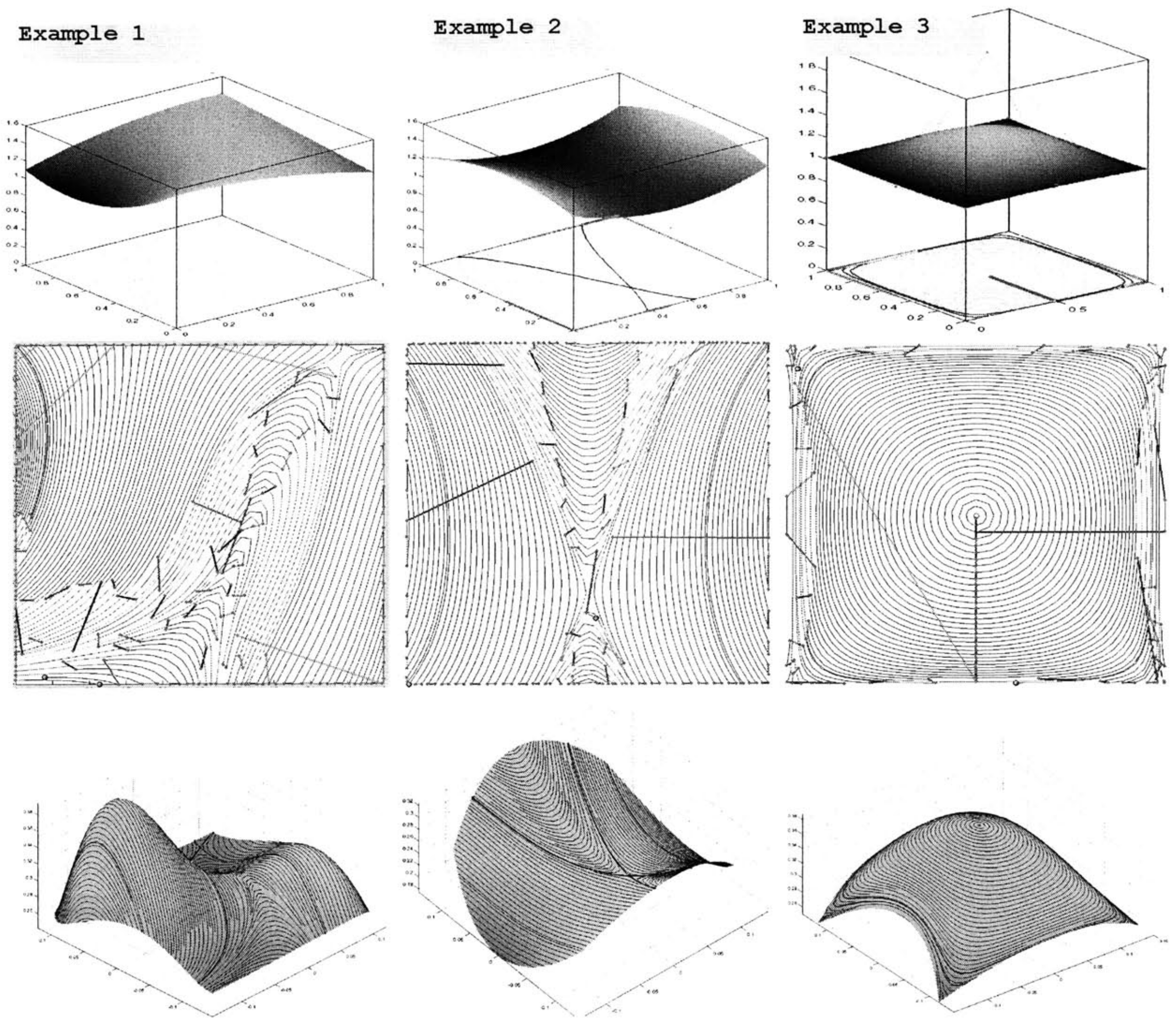

1) Greedy Directions
2) Fit a generating function to the greedy directions.
(3-1) Critical and the turning points.
V(u, v)
(3-2) Extract maximal basin boundaries.
(3-3) Basin Boundaries on the Surface
4) Generate the **reference tool paths** in each basin.
5) Path Matching
6) Non-effective movements are constructed.
7) Surface Tool Paths

Figure 30 Greedy Tool path Generation: a Bi-Quadratic Generating Function

MORE EXAMPLES

Two additional examples are shown. The control points of the Bezier surfaces are provided in APPENDIX **D**. 10-40% improvements in cutting performance is observed.



Result of the 3 examples	Example 1	Example 2	Example 3
1. Cusp Height Limit h_o (μm)	1000	1000	1000
2. Iso-u path ($T_c = T_e + T_{ne}$) (sec)	618(=606+12)	250(=239+11)	286(=277+9)
3. Iso-v path	686(=674+13)	373(=363+10)	413(=404+8)
4. Greedy path	555(=518+38)	239(=203+36)	248(=216+33)
5. Advantage over Iso-u path	**10%**	**4%**	13%
6. Advantage over Iso-v path	19%	36%	**40%**

NOTE: The non-effective cutting time was evaluated procedurally.

Summary: Greedy Tool Path Generation

In this chapter, we have seen how we generate greedy tool paths. Greedy tool paths maximize material removal rate of each individual path instead of minimizing the total cutting time. This approach is called greedy because it optimizes the local measure from the viewpoint of individual tool paths without considering neighboring tool paths.

We sample a proper number of points on a designed surface. At each sampled point, we find the greedy direction in which the material removal rate is maximized. To find a greedy direction, we necessarily find a point on a velocity polygon where the largest iso-sweeprate contour line passes. In this way, we take account of the kinematics and the "local" geometry of the machine tool in the greedy tool paths.

In theory, we could find the greedy directions everywhere and form a vector field. Once we formed a vector field, we would integrate it and generate tool paths using the procedures developed in the previous chapter. In practice, however, it is not straightforward to perform the integration along the greedy direction field, which is defined merely "procedurally" and discontinuous somewhere. The problem is more complicated because there are at least two equally preferable directions at every point on the surface. To circumvent such technical difficulties in integrating the greedy direction field, we fit a vector field, whose streamlines are level lines of a certain continuous function on the surface, to the sampled greedy directions. We minimize the weighted sum of squares of deviation angles. The continuous function that we used for the fitting is called a generating function, which is simply a special kind weaving function. By choosing weights appropriately in the least squares approximation, we reduce the problem to minimizing (or maximizing) a Rayleigh quotient. The optimization is now standard and easy to be solved.

There are, however, other unknowns to be determined. Among them, the continuum boundaries are crucial, which encode where a path starts (and ends). Since exact optimal continuum boundaries are hard to be found, we, instead, define maximal basins and use them as the path continua of our path structure. Basins are the regions subordinate to a direction field, which exhibit very "strong" regularity—a basin consists of only one inlet and only one outlet of streamlines of the direction field. If two basins in a partition cannot be combined to form another basin, we call the basins maximal and the partition minimal. The critical points and the turning points of the generating function is an important cue for finding maximal basins. Additionally, we have shown that after fining the maximal basins, finding the optimal continuum is reduced to an optimization problem along a (1-dimensional) streamline in the maximal basins in an approximate sense. We call this the individual cut heuristics, which must be distinguished from the individual cut principle. Even though the individual cut heuristics simplifies the problem considerably, the simplified problem still remains "combinatorial." In the greedy approach, we regard a maximal basin as a continuum, without searching after the optimal continuum.

After finding the direction field though the least squares approximation and taking the maximal basins as the continua, the remaining task is plain. The procedures that we need were provided in the previous chapter. In a sense, the greedy approach fixes, from the outset, the two most hard-to-be-found unknowns. Some examples of greedy tool paths were given in this section. As a rule, the chance is high that the greedy tool paths outperform arbitrarily generated tool paths. Greedy approach is invaluable when a machine tool approaches its singularities. Near singularities, behavior of the machine tool becomes extremely anisotropic as shown in Figure 25-(*d*), and trajectory planning becomes invaluable. The question now we raise is how well the greedy paths work, which we study in the next chapter.

5 Bound and Scale Analysis

Until now, we have formulated the machining problem, studied its qualitative nature and constructed some tool paths with a greedy strategy. The question we now ask is how well the greedy tool paths work in comparison with other "standard" tool paths. We develop the theory for greedy direction fields but it is readily applicable to any situation as far as a direction field is given. In a sense, we develop "indirect" performance indicators of direction fields. They are crude in nature but it allows us to judge economically whether it is worth while to generate tool paths along a particular direction field.

In Section 1, we show that there is a lower bound of (effective) cutting time and explain its physical meaning. In Section 2, we further assume that the direction field is linearly converging or diverging and derive formulas for cutting time in terms of easily obtainable parameters. We could have used other types of direction fields such as exponentially converging fields, which we defer to the readers. In addition, we show an example to use those formulas. Thanks to them, we can answer whether it is necessary to split a maximal basin and, as a result, to place intermittent cuts. In Section 3, we define the notion of availability (*A.V.*) and potential advantage (*P.A.*). The availability characterizes the overlap of tool paths while the potential advantage characterizes the amount of anisotropy in the kinematic performance of machine tools. They are the indirect measures that are evaluated by means of "blind" integrations. The greedy approach is to take advantage of the anisotropy keeping the amount of overlap as low as possible. We show how to achieve the trade-off between the two measures, (*A.V.*) and (*P.A.*). Finally, we present some examples.

5.1 The Strong Greedy Solution: A Lower Bound of Cutting Time

In this section, we introduce a lower bound of cutting time. The lower bound is evaluated from the greedy directions without iteration.

The Lower Bound of Cutting Time

We refer to the greedy direction angle at $(u, v) \in \mathcal{P}$ as $\eta^g(u, v)$. (The superscript g stands for "greedy.") In the previous chapter, we described how to obtain the greedy directions. Given the greedy directions, there are defined the corresponding speed ϑ^g and the side-step-limit w^g *s.t.*

$$w^g(u, v) \equiv w_o(\eta^g(u, v), u, v) \qquad \text{and} \qquad \vartheta^g(u, v) \equiv \vartheta_o(\eta^g(u, v), u, v)$$

where $w_o(\eta, u, v)$ is the side-step-limit and $\vartheta_o(\eta, u, v)$ is the maximum speed in the given direction as defined in Section 3.4. We refer to the triad $(\eta^g, w^g, \vartheta^g)$ as a *strong greedy solution*. Because of the following inequalities:

$$w(u, v) \le w_o(\eta^g(u, v), u, v) \equiv w^g(u, v) \qquad \text{and} \qquad \vartheta(u, v) \le \vartheta_o(\eta^g(u, v), u, v) \equiv \vartheta^g(u, v),$$

(which are the cusp-height-limit and the actuator speed limits), it is evident that the effective cutting time $T_e[\eta, w, \vartheta]$ of any feasible path structure satisfies the following inequality:

$$T_e[\eta, w, \vartheta] \equiv \iint_{\mathcal{P}} \frac{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot dudv}{\vartheta(u, v) \cdot w(u, v)} \ge \iint_{\mathcal{P}} \frac{\|\boldsymbol{r}_u \times \boldsymbol{r}_v\| \cdot dudv}{\vartheta^g(u, v) \cdot w^g(u, v)} \equiv T_L \qquad \forall\eta\,\forall w\,\forall\vartheta. \tag{37}$$

The quantity T_L, as defined in the equation above, is a lower bound of effective cutting time and, as a result, the lower bound of total cutting time $T_c \equiv T_e + T_{ne}$ $(T_{ne} > 0)$. Therefore, any attempt to finish machining a surface earlier than T_L would fail. Generally, a meaningful (or tight) lower bound is useful in various situations. For example, it can be used in a termination condition of an optimization process.

A Remark on the Strong Greedy Solution*

It is obvious that the strong greedy solution, $\eta(u, v) = \eta^g(u, v)$ and $w(u, v) = w^g(u, v)$, fails to satisfy the differential compatibility (Equation 15) **almost everywhere**. According to our model (Equation 21), the incompatibility on sets of measure zero (curves or points) results in finite non-effective cutting time. If the set of incompatibility is a set of full measure, the non-effective cutting time diverges to the infinity. Therefore, the strong greedy solution requires infinite non-effective cutting time. The physical situation is very similar to spotting, during which the majority of time is spent on switching positions. This is a natural consequence of neglecting the non-effective cutting time and the compatibility to get the greedy directions. While infinite non-effective cutting time is required for the strong greedy solution, a lower bound T_L of effective cutting time is furnished by it. Therefore, the strong greedy solution furnishes the minimum of effective cutting time. The greedy approach attempts, in a sense, to reduce the non-effective cutting time while keeping the effective cutting time near the lower bound T_L.

5.2 The Individual Cut Principle, the Individual Cut Heuristics and Additional Approximation

We outline this section. First, we recap the individual cut principle and the individual cut heuristics to make the context clear. Then, we present an approximate formula for the cutting time per a streamline. Finally, we show a simple example that uses the formula.

The Individual Cut Principle

In Section 3.5, we presented the individual cut principle, which implies that we must minimize the *cutting time per a streamline* to minimize the total cutting time. The cutting time per a streamline was defined as follows:

$$\tau_c^{ref} \equiv \int_0^L \left(\frac{w_{ref}(s)}{w(s)}\right) \cdot \frac{ds}{\vartheta(s)} + \frac{1}{2}\sum_i \left(\left[\tau_o \cdot (1 - \gamma_{i,i+1}) \cdot \left(\frac{w_{ref}}{w}\right)\right]_{s \to s_i^-} + \left[\tau_o \cdot (1 - \gamma_{i+1,i}) \cdot \left(\frac{w_{ref}}{w}\right)\right]_{s \to s_i^+} \right)$$

where the matching pairs $(\gamma_{i,i+1}, \gamma_{i+1,i})$ must be in the feasibility set

$$Q'' \equiv \left\{ \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right), (1, 1) \right\}$$

and other symbols are as defined in Equation (34). This principle is applicable only when a continuum and its direction field are provided. Resorting to this principle, it is possible to extract the (pseudo-compatible) grain boundaries **in a given continuum**, at least in theory.

The Individual Cut Heuristics

In Section 4.3.3, we provided the rationale for the individual cut heuristics to get the approximate continuum boundaries **in a maximal basin**. In this case, we also minimize the *cutting time per a streamline* but we search for the matching pairs the following extended feasibility set:

$$Q' \equiv Q'' \cup \left\{ (0, 0) \right\} = \left\{ (0, 0), \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right), (1, 1) \right\}.$$

The Cutting Time per a Streamline for Linear Fields

The individual cut principle and the individual cut heuristics reduce our problem to separate optimization problems each of which is defined on a 1-dimensional space (or a streamline). Even though they pretty simplify the problem, finding the optimal continuum boundaries still remains computation-intensive. Below, we simplify the problem further by considering a restricted case for the direction field in a (given) maximal basin—we invoked the individual cut heuristics by considering each basin and its streamlines one by one.

The restrictions are

(1) The field is linearly converging and the converging ratio is homogenous in a maximal basin, *i.e.* $\gamma \equiv d(w_o - w)/ds = const \geq 0$.

(2) There are no pseudo-compatible points.

(3) **The homogenous NEP model**: The NEP term τ_o is a constant.

(4) The side-step-limit and the speed are constants.

(5) The incompatible points are **equidistant** along a streamline.

With the restrictions above, we derive an approximate formula for cutting time per a streamline in Appendix E. Without derivation, we provide the following formula:

$$\tau_c(N) \equiv \tau_e(N) + \tau_{ne}(N) \approx \tau_1 \cdot [(2-\beta) + \{1-(1-\beta/N)^N\}/N] + \tau_o \cdot N \cdot \frac{\{1-(1-\beta/N)^N\}}{\beta}$$

$$= \frac{1}{\beta} \cdot \left\{1 - \left(1 - \frac{\beta}{N}\right)^N\right\}\left\{\tau_o N + \frac{\beta\tau_1}{N}\right\} + \tau_1 \cdot (2-\beta) \tag{38}$$

where $\tau_c(N)$ is the cutting time per a streamline for $(N-1)$ interruptions[†] (or the number of incompatible points),

$$\beta \equiv \left(\frac{L}{W}\right)\gamma, \qquad \tau_1 \equiv \frac{1}{2} \cdot \frac{L}{V}, \qquad \gamma \equiv \frac{d}{ds}(w_o - w) = const \geq 0, \qquad W \equiv \frac{1}{L} \cdot \int_0^L w_o \cdot ds, \qquad V \equiv \frac{1}{L} \cdot \int_0^L \vartheta_o \cdot ds,$$

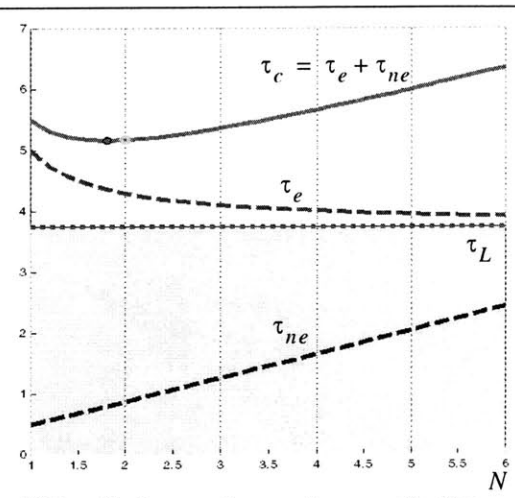
V is the average (maximum) speed along the streamline, L is the length of the streamline, W is the side-step-limit and τ_o is the non-effective penalty (NEP) term. We refer to the quantity β as the *compression ratio*.

We will apply the above "nominal result" for rather general (not necessarily linear) cases. In those cases, the estimation of the compression ratio β becomes expensive because the value of the parameter γ is not known in a point-wise manner prior to the integration of the side step. To circumvent this difficulty, we used the following estimation in an average sense, *i.e.*

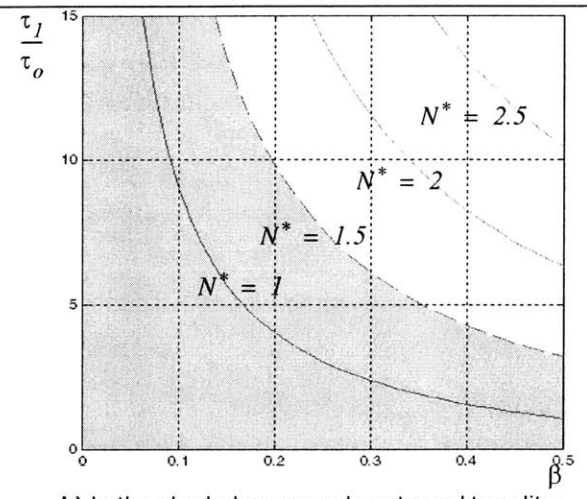
$$\beta \equiv \left(\frac{\gamma L}{W}\right) \approx \frac{L}{A} \cdot \int \left|\frac{1}{w_o}\frac{dw_o}{ds} - \frac{1}{w}\frac{dw}{ds}\right| \cdot dA \equiv \beta[\eta] \tag{39}$$

where A is the area of a maximal basin. We point out that the averaged compression ratio is evaluated, once a direction field $\eta(u, v)$ is specified in a maximal basin. Note also that $(dw/ds)/w$ $(= h_3)$ is known from the (differential) continuity.

[†] In other words, N is the number of continua.



a) The effective portion vs. the non-effective portion



b) In the shaded area we do not need to split a basin.

Figure 31 Effective Cutting Time v.s. Non-effective cutting Time

We plotted the equation in Figure 31-(*a*) for a specific value of β, τ_o and τ_1. The graph shows that as the number of incompatibility increases, the effective cutting time is reduced to a lower bound while the non-effective cutting time increases (*c.f.* the analogous problem in Figure 21, Section 3.6).

Remark: To be more specific, we check limit cases:

$$\text{as } N \to \infty, \qquad \tau_e \to \tau_L \equiv \tau_1(2-\beta) \quad \text{and} \quad \tau_{ne} \to (1-e^{-\beta})/\beta \cdot \tau_o N \to \infty$$

$$\text{as } \beta \to 0, \qquad \tau_e \to 2\tau_1 \quad \text{and} \quad \tau_{ne} \to \tau_o N.$$

The Desired Number of Continua

We differentiate Equation (38) and derive the desired number N^* of path continua in the given basin. The desired number N^* must satisfy the following condition:

$$\left.\frac{d\tau_c}{dN}\right|_{N=N^*} = 0.$$

The condition is arranged as follows:

$$\frac{\tau_1}{\tau_o} = \frac{N^{*2} \cdot [\{1-(1-\beta/N^*)^{N^*}\} - N^*(1-\beta/N^*)^{N^*}\{\ln(1-\beta/N^*) + \beta/(N^*-\beta)\}]}{\beta \cdot [\{1-(1-\beta/N^*)^{N^*}\} + N^*(1-\beta/N^*)^{N^*}\{\ln(1-\beta/N^*) + \beta/(N^*-\beta)\}]}. \tag{40}$$

We plot the condition on a non-dimensional chart in Figure 31-(*b*). The chart shows that the higher the compression ratio is, the more continua (or equivalently the more interruptions) we need. Besides, the desired number increases as the ratio (τ_1/τ_o) increases.

An Example

We now ask whether it is advantageous to split a maximal basin further after we applied the path matching procedure. This is one of the examples that show how we use the formulas that we have derived in this section.

We inserted a set of values for the compression ratio β, the half τ_1 of the average effective cutting time per a streamline and the NEP term τ_o as follows:

$$\beta = \frac{1}{2}, \qquad \tau_1 \equiv \frac{1}{2} \cdot \left(\frac{L}{V}\right) = \frac{(1m)}{2(0.2m/sec)} = 2.5\ sec \qquad \text{and} \qquad \tau_o = 0.5\ sec\,.$$

We point out that the maximum of the compression ratio is $1/2$ if the path matching procedure has been applied. In addition, other values were chosen rather conservatively so that the desired number N^* of path continua can be over-evaluated somewhat. Even in this conservative case, the formula (Equation 40) predicts the desired number of continua in a maximal basin to be 1.81 as shown in Figure 32. (Note that $N^* = 1$ implies no split, $N^* = 2$ is for the case when one split is made, and so on). Therefore, for most cases, it is expected that we cannot obtain a better path structure by splitting a maximal basin further into path continua after the path matching procedure. Of course, this is a conclusion that is drawn for typical cases with a crude argument. The answer depends on the specific situation where a particular problem stands. In general, if a machine tool exhibits poor performance in its maximum speed (in comparison with its performance in acceleration capability), it is better to split a maximal basin into path continua.

A Remark

We have derived the formulas (Equation 38 and 40) assuming the direction field is linearly converging. We might have derived similar formulas using other types of direction fields. For example, we could have used exponentially converging field, which we defer to the readers.

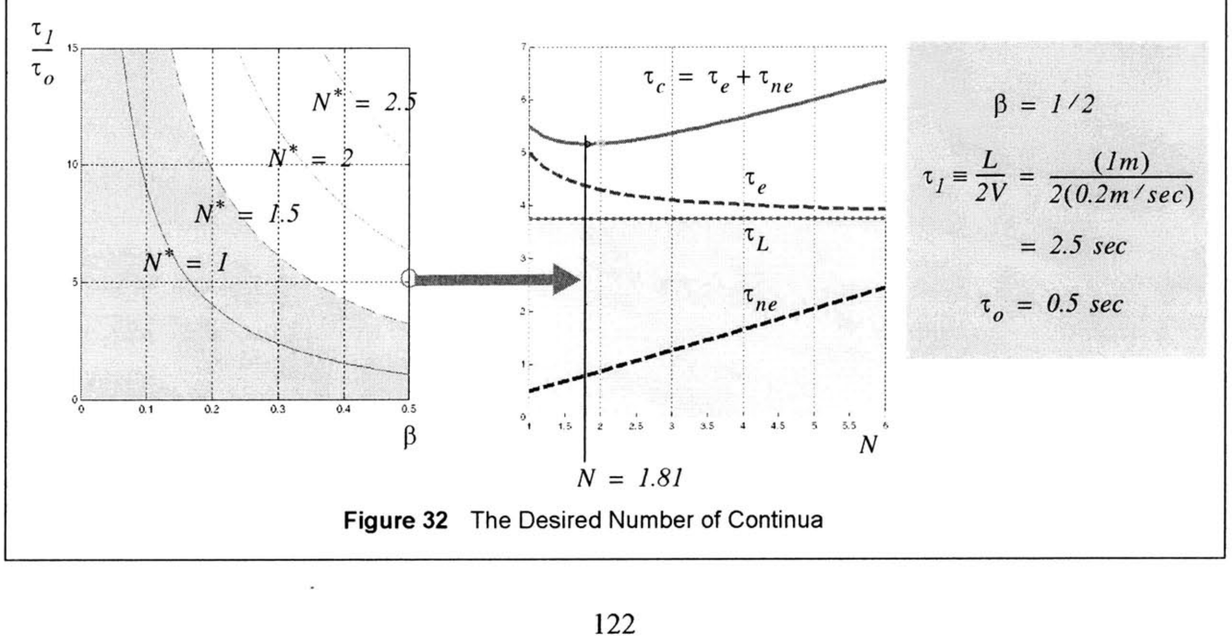


Figure 32 The Desired Number of Continua

5.3 Comparison with Iso-Cusp-Height Schemes

The greedy approach takes the local advantage at the expense of the overlap of neighboring tool paths. A path generation scheme in the opposite side is the iso-cusp-height scheme, which minimizes the overlap degree as explained in Section 2.1. It is interesting that the strong greedy path is an iso-cusp-height path, too. Therefore, at the expense of the non-effective cutting time, the strong greedy solution is approached by both the path generation schemes.[†] The deviation of the two types of tool paths from the strong greedy solution is our concern in this section. In a practical sense, we compare two schemes prior to actual path generation. Since reducing the effective cutting time is the objective of both the schemes, we compare their effective cutting time only.

A Lower Bound to Effective Cutting Time for a Particular Direction Field

As we defined the lower bound to effective cutting time for the greedy direction field, we define it for a particular direction field $\eta(u, v)$:

$$T_L[\eta] \equiv \iint_{\mathcal{P}} \frac{\| \boldsymbol{r}_u \times \boldsymbol{r}_v \| \cdot dudv}{\vartheta_o(\eta(u, v), u, v) \cdot w_o(\eta(u, v), u, v)}. \tag{41}$$

The effective cutting time of any tool paths that are aligned to the streamlines of the direction field $\eta(u, v)$ is greater than this lower bound, namely[‡]

$$T_L \le T_L[\eta] \le T_e^{ref}[\eta], \qquad \forall\eta$$

where $T_L \equiv T_L[\eta^g]$ as defined in Equation (37).

The Directional Loss

We define directional loss to measure how much a field η deviates from the greedy direction field η^g:

$$\left(D.L.\right)[\eta] \equiv \frac{T_L[\eta] - T_L}{T_L}. \tag{42}$$

This measures how efficient a fitting scheme for a given greedy direction field when it is applied for the fitted field. It is possible to evaluate the directional loss once a direction field is given or fitted.

† This limit process is equivalent to sending the continuum size to zero.

‡ $T_e^{ref}[\eta] \equiv \iint_{\mathcal{P}} \frac{\| \boldsymbol{r}_u \times \boldsymbol{r}_v \| \cdot dudv}{\vartheta_o(\eta(u, v), u, v) \cdot w_{ref}[\eta](u, v)}$

where $w_{ref}[\eta]$ is the reference side step for a given direction field η as defined in Section 3.5.

Availability

Because of inevitable overlaps, the effective cutting time of tool paths strays from the lower bound as expressed in the Equation (41). Availability is defined to measure this deviation from the lower bound. As shown in Figure 31-(*a*), the effective cutting time in a maximal basin cannot be improved by the splits more than $T_e^{ref}[\eta] - T_L[\eta]$, where $T_e^{ref}[\eta]$ is the effective cutting time of the reference tool path (*c.f.* Figure 21). Being motivated by this, we define *availability* $(A.V.)[\eta]$ of a (fitted) direction field $\eta(u, v)$ as follows:

$$\left(A.V.\right)[\eta] \equiv \frac{T_e^{ref}[\eta] - T_L[\eta]}{T_L[\eta]}.$$

The availability can be thought of as the loss due to **the overlap** that hinders the effective cutting time from achieving the lower bound $T_L[\eta]$.

Note that it is necessary to actually construct the (reference) tool paths in order to evaluate the effective cutting time $T_e^{ref}[\eta]$ of the reference tool paths. However, thanks to the linear theory (Equation 38), we estimate it by taking the following approximation:

$$\left(A.V.\right)[\eta] \equiv \frac{T_e^{ref}[\eta] - T_L[\eta]}{T_L[\eta]} \approx \frac{\tau_e(1) - \tau_L}{\tau_L} = \frac{\beta[\eta]}{2 - \beta[\eta]}$$

where β is the compression ratio as defined in Equation (39). By definition, the compression ratio β can be computed without performing streamline integration.

The Estimated Effective Cutting Time

Using the directional loss and the availability, we estimate the effective cutting time of the fitted field as follows:

$$T_e^{ref}[\eta] = \left\{\left(A.V.\right)[\eta] + 1\right\}\left\{\left(D.L.\right)[\eta] + 1\right\}T_L.$$

This is an economic way to evaluate the performance of a particular direction field avoiding the expensive task of path generation. This approximation can be used to compare two direction fields quickly. (Of course, this is a crude estimation.) If there are several direction fields, this measure can be used in a "branch and bound" search of the space of paths.

Remark: To evaluate the effective cutting time after the path matching procedure, we evaluate the compression ratio in the following way:

$$\beta[\eta] = \frac{1}{A} \cdot \int F \cdot dA \qquad \text{where} \qquad F \equiv MAX\left\{\frac{L}{w_o}\frac{dw_o}{ds} - \frac{L}{w}\frac{dw}{ds}, \frac{1}{2}\right\}.$$

In the linear theory, it is observed that the availability $(A.V.)$ is always estimated less than 33% because $\beta[\eta] \le 1/2$. In other words, it is always possible to machining an entire surface with the effective cutting time being less than $1.33 T_L[\eta]$.

The Potential Advantage

We can compute for a surface the worst directions η^w as shown in Figure 33, just as we determined the best directions. When kinematic concerns are ignored, a commercial CAM system may pick a bad direction for tool paths. The question is what the opportunity for improvement is. Clearly, for a flat, square patch, the opportunity is small. For other surfaces, the opportunity may be greater, and we define a metric called the *potential advantage* $(P.A.)$ to capture this opportunity:

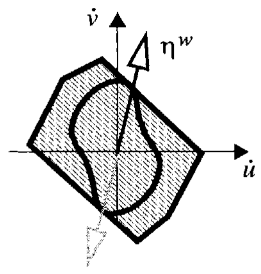


Figure 33 Finding the Worst Direction

$$(P.A.) \equiv \frac{T_L[\eta^w] - T_L}{T_L}$$

where $T_L[\eta]$ and T_L are defined in Equation (41 and 37). The more **anisotropic** the performance of the system, the greater the potential advantage is. If the potential advantage of a surface is low with respect to the availability, the benefits of the greedy tool paths are diminished. Under the circumstances, the most important criterion for the optimality is reducing overlap.

The Upper Bound for Iso-cusp-height Tool Paths

An extreme approach for the tool path problem is to ensure that there is no overlap at all on the surface. This approach is referred to as the iso-cusp-height scheme. By the worst field, the effective cutting time of any iso-cusp-height direction field η^c is bounded from above, namely

$$T_L \le \left\{ T_e^{ref}[\eta^c] = T_L[\eta^c] \right\} \le T_L[\eta^w] .$$

This is because the iso-cusp-height tool paths satisfy the following equation:

$$w^c(u, v) = w_o(\eta^c(u, v), u, v)$$

where $\eta^c(u, v)$ is the direction field and $w^c(u, v)$ is the side step of the iso-cusp-height tool paths; they are compatible (almost everywhere without overlap). Note that, for a general direction field η, its effective cutting time $T_e^{ref}[\eta]$ is only bounded from below.

The Risk Index

If $T_e^{ref}[\eta] > T_L[\eta^w]$, it is not worth while to proceed with the greedy approach. This is because any iso-cusp-height paths outperform the (greedy) paths in that case. In order to check whether the (greedy) tool paths can outperform the iso-cusp-height tool paths, we define the risk index $(R.I)$ in the following way:

$$\left(R.I.\right)[\eta] \equiv \left[\left\{\left(A.V.\right)[\eta] + 1\right\}\left\{\left(D.L.\right)[\eta] + 1\right\} - 1\right] / \left(P.A.\right).$$

The less this index, the more likely that the greedy approach succeeds. A definite (negative) conclusion is drawn: if $(R.I)[\eta] > 1$, then the effective cutting time of any iso-cusp-height tool paths is less than the effective cutting time of the tool paths that are generated from the direction field $\eta(u, v)$ and, as a result, it is better to generate the iso-cusp-height tool paths.

Summary

We summarize this section by providing following trivial inequalities:

1. $$T_L \equiv T_L[\eta^g] = \iint_{\mathcal{P}} \frac{\| r_u \times r_v \| \cdot dudv}{\vartheta_o(\eta^g(u, v), u, v) \cdot w_o(\eta^g(u, v), u, v)} = \iint_{\mathcal{P}} \frac{\| r_u \times r_v \| \cdot dudv}{\vartheta^g(u, v) \cdot w^g(u, v)}$$

$$\leq T_L[\eta] \equiv \iint_{\mathcal{P}} \frac{\| r_u \times r_v \| \cdot dudv}{\vartheta_o(\eta(u, v), u, v) \cdot w_o(\eta(u, v), u, v)}$$

$$\leq T_L[\eta^w] \equiv \iint_{\mathcal{P}} \frac{\| r_u \times r_v \| \cdot dudv}{\vartheta_o(\eta^w(u, v), u, v) \cdot w_o(\eta^w(u, v), u, v)} \qquad (\forall \eta).$$

2. The above inequality holds especially for the iso-cusp-height tool paths. Therefore,

$$\left\{T_L \equiv T_L[\eta^g]\right\} \leq \left\{T_L[\eta^c] = T_e^{ref}[\eta^c]\right\} \leq \left\{T_L[\eta^w]\right\} \qquad (\forall \eta^c).$$

3. For a general direction field, we only have the following inequality:

$$\left\{T_L \equiv T_L[\eta^g]\right\} \leq \left\{T_L[\eta]\right\} \leq \left\{T_e^{ref}[\eta]\right\} \qquad (\forall \eta).$$

5.4 Concluding with Examples

In this section, we present some examples for estimating the performance indices that has been defined in this chapter and judging the efficiency of the greedy tool paths over the iso-cusp-height tool paths. In addition, we show how well this estimation, which is based on the "linear" theory, works.

The Performance Indices

We take the same example as shown in the previous chapter. From the 8th to the 11th row in Figure 34, the lower bounds and the compression ratio are computed through area integration. We estimate the directional loss (*D.L.*), the availability (*A.V.*), the potential advantage (*P.A.*), and finally the risk index (*R.I.*) as shown from the 12th to 15th row. In the second and the third example, the risk index is low; while the first example exhibits high risk index around 1. The high risk index of the first example implies that it is better to choose iso-cusp-height approach for the surface. In this particular case, the high risk index mainly due to high directional loss (around 60%), which can be improved through improving the fitting method.

The uncertainty of the fifth row is due to the fact that the first and the last tool paths in a basin usually "cover" neighboring basins. Of course, the effective cutting time will dominate this uncertain portion of finishing time as the cusp-height-limit reduces. In the current implementation, the characteristic length *L* in Equation (39) was set very crudely as a constant over the designed surfaces as shown in Figure 34. In the 16th to 17th row, we computed the actual availability and the risk index. The estimation corresponds well to the actual values, taking account of the fact that the length *L* was set roughly. We can improve the result by using a respective characteristic length for each maximal basin.

The Effectiveness of the Estimation

We generated the greedy tool paths for various surfaces and estimated the performance indices. The difference between the estimated value and the actual value is shown in Figure 34. It is observed that the lower the compression ratio, the more accurate estimation tends to be made.

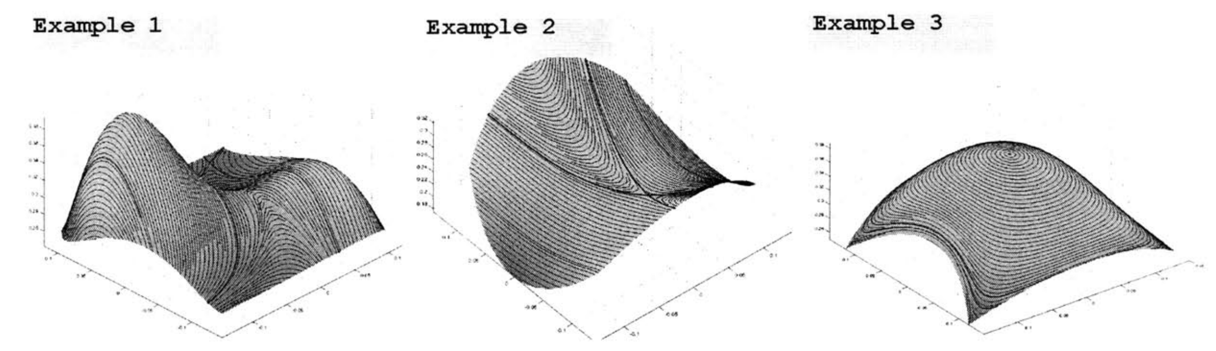
Concluding Remarks

In this chapter, we have defined various "indirect" performance indices of tool paths to compare the two intuitive approaches quickly. If we extend this idea further, instead of solving the exact problem, it is worth an attempt to minimize the following approximate cutting time:

$$\left\{\frac{2T_L[\eta]}{2-\beta[\eta]}\right\}_{S-\mathcal{D}} + |\mathcal{D}|$$

where the last term $|\mathcal{D}|$ is an appropriately-defined measure on the "discontinuity" set $\mathcal{D}$. The problem is

Figure 34 Evaluating the Performance Indices



		Example 1	Example 2	Example 3
1.	Cusp Height Limit h_o (μm)	1000	1000	1000
2.	Iso-u path ($T_c = T_e + T_{ne}$) (sec)	618(=606+12)	250(=239+11)	286(=277+9)
3.	Iso-v path	686(=674+13)	373(=363+10)	413(=404+8)
4.	Greedy path	555(=518+38)	239(=203+36)	248(=216+33)
5.	**Uncertainty	(-)9	(-)17	(-)19
6.	Advantage over Iso-u path	10%	**4**%	13%
7.	Advantage over Iso-v path	19%	36%	**40%**
8.	The Lower Bound T_L	229	134	142
9.	*The Worst Bound $T_L[\eta^w]$	510	300	353
10.	****Lower Bound $T_L[\eta^f]$	371	156	162
11.	***Compression Ratio $\beta[\eta^f]$	0.4388	0.3880	0.4238
12.	**Directional Loss** $(D.L.)[\eta^f]$	62.02%	16.42%	14.08%
13.	**Potential Advantage** $(P.A)$	122.9%	124.5%	148.6%
14.	By estimation, $(A.V.)[\eta^f] \approx$	28.11%	24.07%	26.89%
15.	By estimation, $(R.I.)$ *	87.51%	35.70%	30.12%
16.	**Actual $(A.V.)[\eta^f]$	35.20%-39.70%	24.63%-30.22%	21.43%-33.17%
17.	**Actual $(R.I.)$ *	96.86%-102.7%	36.22%-41.45%	25.93%-34.94%

*The worst lower bound was roughly estimated compared to other estimations in the current implementation; it is under-estimated because we did not find the exact worst direction. As a consequence, the potential advantage is under-estimated and the risk index is over-estimated.

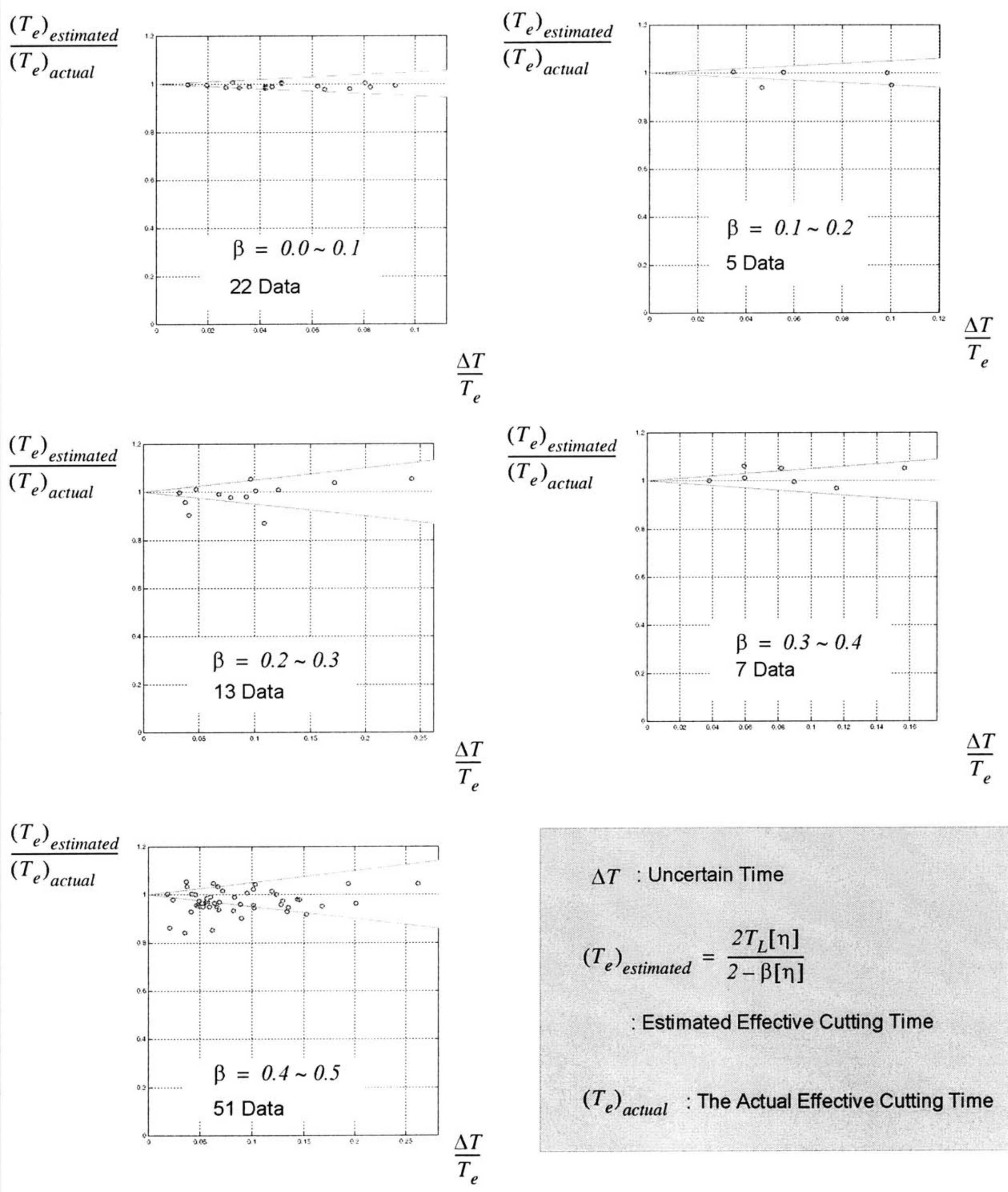
**This uncertainty comes from the fact that the last path in a basin cover a region somewhat redundantly.

***The characteristic length L was set as follows:

$$L \approx \sqrt{\frac{(Area)}{\pi / 4}}.$$

**** η^f stands for the fitted field.

now similar to the problem of image segmentation, which has been in attention in the computer vision community [*c.f. Mumford-Shah Functional, e.g. see* 162]. Of course, there are much room for refining the above functional from various perspective.



ΔT : Uncertain Time

$$(T_e)_{estimated} = \frac{2T_L[\eta]}{2-\beta[\eta]}$$

: Estimated Effective Cutting Time

$(T_e)_{actual}$: The Actual Effective Cutting Time

Figure 34 The Accuracy of the Estimation

6 Conclusions and Recommendations

This thesis (1) developed a new framework for CNC finish milling tool path generation for free-form surfaces, (2) studied the formulated problem theoretically, (3) implemented a path generation algorithm, which we called greedy, and (4) evaluated the performance of the generated paths. This thesis achieved the following two objectives: to lay down the theoretical basis for surface machining and to find a practical solution of the machining path optimization problem. The developed framework is fairly general and model the machining process effectively. We showed that the implemented greedy scheme can reduce the machining time to a considerable extent. It remains as future work to solve the optimization problem in a stricter sense.

We recommend a few improvements of the greedy tool path generation scheme, show how we may proceed to solve the general problem and place the problem in certain perspective. We wrap up this thesis with a summary.

Recommendations for the Greedy Scheme

It is recommended to improve the fitting method. Pre-segmentation of the region is expected to improve the fitting performance, and as a result, to reduce the directional loss. It is also recommended to weight the error measure with "local anisotropy" of the machine tool. In lieu of the Bernstein polynomials, other function spaces should also be studied. To the error measure, we may add penalty term for the severe overlap degree. We pre-assigned the orientation $(\varphi(u, v), \phi(u, v))$. We can make more realistic yet not too complicated assignment like $\varphi(u, v) = \varphi_o(\eta(u, v), u, v)$ and $\phi(u, v) = \phi_o(\eta(u, v), u, v)$ that depends on the direction angle η, too.

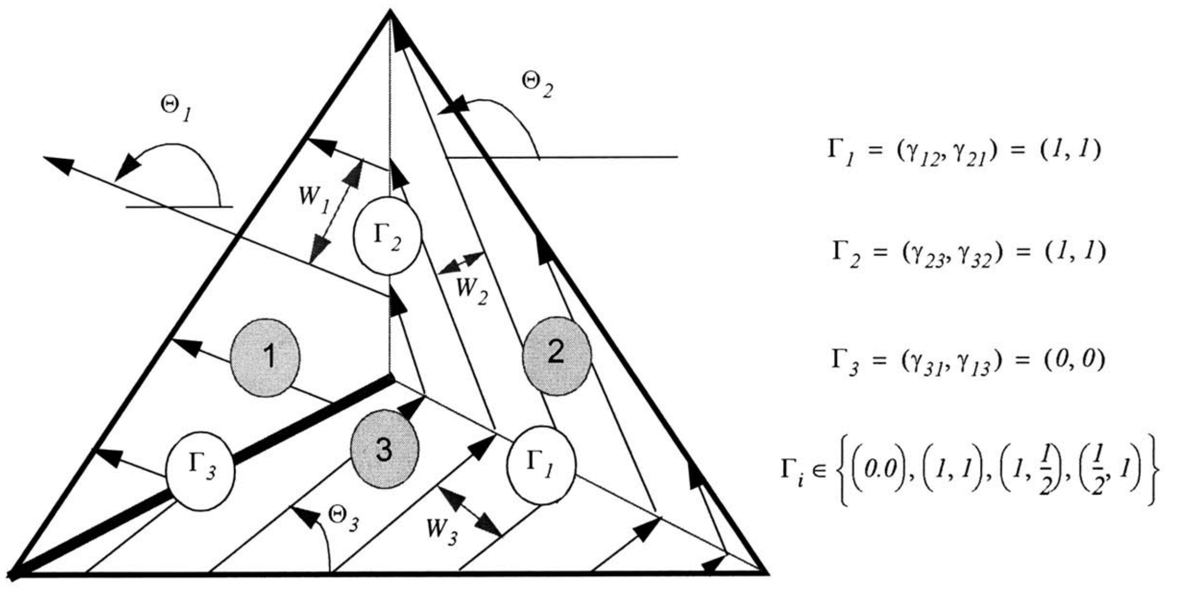
A Prelude to General Search

We take a simple case as an example and describe how we may approach the general problem. We show that it is possible to reduce our problem to a mixed-variable optimization problem.

We consider a triangular region on a plane as shown in Figure 35. We tessellate the region into three sub-triangles. We regard each sub-triangle as a grain of our path structure. We assume that the tool paths in each region are mutually parallel being spaced at an equal distance apart. Then, tool paths in the i^{th} triangle is identified by the two variables,

$$\Theta_i \quad \text{and} \quad W_i$$

where Θ_i is the direction angle and W_i is the side step of the i^{th} triangle. Just for showing the essence of this approach, we assume that the velocity polygon is given as an ellipse such that



Θ_1
Θ_2
Θ_3
W_1
W_2
W_3
Γ_1
Γ_2
Γ_3
1
2
3
$\Gamma_1 = (\gamma_{12}, \gamma_{21}) = (1, 1)$
$\Gamma_2 = (\gamma_{23}, \gamma_{32}) = (1, 1)$
$\Gamma_3 = (\gamma_{31}, \gamma_{13}) = (0, 0)$
$\Gamma_i \in \left\{(0.0), (1, 1), \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right)\right\}$

Figure 35 A Mixed-variable Formulation

$$\left(\frac{1}{V_i(\Theta_i)}\right)^2 = \left(\frac{1}{V_i^x}\right)^2 \cdot \cos^2(\Theta_i - \Theta_i^o) + \left(\frac{1}{V_i^y}\right)^2 \cdot \sin^2(\Theta_i - \Theta_i^o), \qquad (i = 1, 2, 3)$$

where Θ_i^o, V_i^x and V_i^y are certain constants that characterize the velocity "polygon." To the three edges of the tessellation, we assign the following variables for matching pairs,

$$(\gamma_{12}, \gamma_{21}),\ (\gamma_{23}, \gamma_{32}) \text{ and } (\gamma_{31}, \gamma_{13}) \in \left\{(0.0), (1, 1), \left(1, \frac{1}{2}\right), \left(\frac{1}{2}, 1\right)\right\} \equiv Q'$$

which encode whether the boundary is compatible or not.

Now, the effective cutting time is captured by the following formula:

$$T_e = \sum_{i=1}^{3} \sqrt{\left(\frac{A_i}{V_i^x}\right)^2 \cdot \left(\frac{\cos(\Theta_i - \Theta_i^o)}{W_i}\right)^2 + \left(\frac{A_i}{V_i^y}\right)^2 \cdot \left(\frac{\sin(\Theta_i - \Theta_i^o)}{W_i}\right)^2}$$

where A_i is the area of the i^{th} triangle. In a similar way, it is possible to express the non-effective cutting time and other constraints in terms of W_i, Θ_i and γ_{ij}. (We point out that the monotonicity condition on the weaving function is more easily captured when we use the stream function instead of the direction angle. However, we do not state problem with the stream functions not to distract the readers with such an abstract notion).

In this manner, we discretize the problem and form a mixed-variable nonlinear optimization problem [*e.g. see* 128]. In general, this approach results in a large scale problem, for which it is perhaps better to proceed for approximate solutions. It is recommended to apply various techniques of the mixed-variable optimization problems and to evaluate their fitness to our problem. It is not difficult to establish a similar discretization for general curved surfaces.

Placing the Problem in certain Perspective

Even though the following observations are, at best, superficial and speculative at the current stage of our research, we mention a few probable connections of our problem to other types of problems.

When we substitute the side step with the corresponding side-step-limit, the compatibility turns out to be a 1st order PDE about the direction field of the iso-cusp-height tool paths. In their article on iso-cusp-height tool paths [18], Suresh and Yang mentioned the difficulty that is associated with the sharp corners developed in their tool paths. It is expected that we can circumvent the technical difficulty by seeking for a *viscosity solution* [141 and 142] of the PDE using the *level set method* [*e.g., see* 122]. The level set method is naturally fits the problems of "hyperbolic or propagation" kind.

In spite of its remoteness from our problem, predicting the *micro-structure* of a material share certain similarity with the machining problem. We may regard the effective cutting time as the internal energy and the non-effective cutting time as the surface energy of the micro-structure [*e.g., see* 143]. Another analogous problem is *image segmentation* which has been in attention by the computer vision community.

The classical *Mumford-Shah functional* and the *Ambrosio-Tortorelli functional* are similar to our cost functional in that they concern with piecewise continuous functions and the discontinuity sets. The research on the segmentation problem is relatively-mature and there are abundant body of literature and ideas [*e.g.*, *see* 162 and 163]. Some techniques of this problem are based on the above-mentioned level set methods [123]. It is recommended to borrow ideas from the image segmentation problem to establish certain approximation of our cost functional.

It is not difficult to see the similarity of a tool path structure to a flow on a manifold or a phase space of a 2-dimensional dynamic system [*e.g.*, *see* 144 and 145]. Since Poincaré, *dynamic systems* has been an active research area. It is recommended to pursue the connections of our problem to that area, especially in the context of subdivision schemes. From a somewhat different angle, the problem of extracting *vector field topology* has been in attention in the area of visualization or fluid mechanics [152, 153 and 154]. Its inverse problems are *vector field interpolation* [146, 147, 148 and 149] and *vector field modeling* [164]. The results of those problems can be related to the vector field fitting and the path continuum extraction in our problem.

We further discuss the relation of our problem to *optimal control problems*, *dynamic programming* and the *TSP*. The dynamic programming approach attempts to solve all the optimal-control trajectories, namely by solving the *Hamilton-Jacobi-Bellman PDE* while the maximum principle-based reasoning attempts to find just one essential trajectory by solving "Hamilton's equation" or a 2-point boundary value problem. The optimal control flow in the phase space has a special "sink," which is usually set as the origin of the phase space. In the machining path optimization problem, there is no such a pre-specified sink on the surface. It is not expected to be able to construct the necessary conditions for optimality of "maximum principle type." On the other hand, naive enumeration of the TSP results in the complexity of "factorial order." It is well known that the dynamic programming reasoning reduces the number of necessary enumeration for the TSP to "merely exponential" one [125 and 129]. It is interesting that the above tessellation based mixed-variable formulation naturally exhibits exponentially-growing complexity as the number of grains in the tessellation increases. It is also recommended to relate the bound analysis of the TSP to the one of our problem.

Summary

In Chapter 1, we briefly explored the nature of the machining path problem in comparison with other path or scheduling problems. We observed that no existing path problem is suited for the machining process.

The free-form surface machining is a fundamental but time-consuming process in modern manufacturing. As a result, it is of considerable economic importance to reduce the machining time. In Chapter 2, we formulated the machining problem as a (non-classical) variational time-optimization problem. The conceived problem is not a conventional problem of time-optimal control because the domain of the performance measure is not a 1-dimensional but a 2-dimensional space. Instead of searching for a series of individual paths, we search for an optimal vector field to get a skeletal information on the time-optimal tool paths. Then, we place tool paths along the streamlines of the vector field. This viewpoint forms a "parallel" description method for tool paths as opposed to the sequential description method that conventional schemes take. The sequential description is suited for commanding a machine tool but it hardly handles the optimization problem.

A cutting tool must cover a surface properly during a machining process. The new "parallel" framework describes naturally this fundamental requirement of machining paths. At the center of this framework, the compatibility equation, which we have devised in this thesis, plays a crucial role in keeping the consistency of the formulated problem. It is an equality constraint in our optimization problem that relates the "gap width" of neighboring tool paths to the vector field.

In addition, this framework can capture various limitations of machining processes including the kinematic limits of machine tools and the geometric cusp height limits. They are the inequality constraints in our time-optimization problem. The anisotropy in the kinematic performance is considerable in any 5-axis machine tools and it is important to take into account the kinematics in path generation. The consideration on the kinematics becomes of more practical importance in the context of recent trend of developing parallel mechanisms and using high speed machine tools.

Tool paths can exhibit complicated patterns. This is because, in fact, tool paths are separate swathe, not a continuum. Tool paths can start and stop anywhere in the surface and can form "chaotic" patterns. However, such "chaotic" tool paths will result in machining inefficiency because it tends to increase the number of continuous tool paths or the amount of overlap; the cost for moving a tool from the end of a previous path to the beginning of the next path is quite high. We postulate that a surface is partitioned into regions in each of which tool paths are reasonably continuous. By this hypothesis, we exclude unnecessarily-complicated tool paths from our consideration but we still accommodate diverse path patterns. Path continua are such "reasonable" regions. Each path continuum is partitioned further into grains where paths are "strictly continuous." What is meant by the "reasonably" or "strictly" continuous paths are defined in terms of certain functions on a surface including a direction field, a side step function, a weaving function, stream functions and matching pairs. The participating functions must satisfy the compatibility equation. A partition of a surface into path continua together with the participating functions is defined as a path structure. Including the speed function in the list, we define kinematic and dynamic path structures. Now, the problem is reduced to finding a "good" path structure. This mathematical construction is necessary for stating the problem formally.

After formulating the problem that we have conceived, we deduced its consequences in Chapter 3. We derive the necessary conditions for the optimality of a tool path structure assuming that the direction field in a path continuum is given. We showed that the problem is reduced to a series of independent variational sub-problems defined on the 1-dimensional streamlines of the given direction field. The individual cut principle summarizes this condition. This principle state the following trivial reasoning: "if a path structure is optimal, each individual streamline or each tool path must be optimal." The widest cut principle and the fastest cut principle can be thought of as special cases of the individual cut principle. They state even more trivial reasoning: to finish machining a surface as early as possible, each individual cut must cover as wide a region as possible and the cutting tool must move as fast as possible. In spite of their triviality, we need some mathematical techniques to deduce the principles from the optimization problem that we formulated in Chapter 2. Specifically, we used the characteristics theory of PDEs and the transformation of the domain of area integration. While we state the problem mainly using partial derivatives, we convert the partial derivatives back to ordinary derivatives along streamlines of the presumably known vector field when we "solve" the problem. Even though the individual cut principle simplifies the problem significantly, there remain some unknowns that are hard to be found. They are the continua and the direction field of the optimal path structure. Especially, giving variations to the continuum boundaries is not straightforward.

In Chapter 4, we developed the greedy scheme to find a practical solution. The greedy scheme is to find the directions of maximum material removal rate and to place tool paths through the numerical integration of the direction field. To circumvent some technical difficulties, we fit a smooth (conservative) vector field to the sampled greedy directions. Instead of extracting the optimal continuum boundaries, we used the maximal basins as the continua of the greedy path structure. Once the greedy direction field and the maximal basins fix the two most inconvenient unknowns, which are the direction field and the continuum boundary, the remaining task is very plain. The greedy tool paths optimize a certain local measure and, therefore, they are not necessarily optimal. However, the chance is high that they perform better than arbitrarily generated tool paths. 10-40% improvement is observed over iso-parametric tool paths.

In Chapter 5, we developed an approximation theory. The theory establishes a criterion that judge economically whether it is probable for greedy tool paths to outperform iso-cusp-height tool paths. The more anisotropic the kinematics performance of a machine tool is, the more it is likely that the greedy tool paths succeed. We compare the anisotropy with the amount of overlap of the greedy tool paths. We define the potential advantage (P.A.) and the availability (A.V.) to measure the anisotropy and the amount of overlap, respectively. They are indirect measures for the performance of greedy tool paths. In fact, we can extend this reasoning further and compare any two direction fields economically. We concluded this thesis by briefly mentioning how it is possible to solve this problem in a stricter sense.

Appendix A Compatibility

A.1 Derivation

We fill in here the gaps between the compatibility (Equation 15)

$$h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v} = h_3(\eta, \eta_u, \eta_v, u, v) \cdot w \ (a.e.) \tag{A-1}$$

and Equation (14)

$$d\left[\frac{(n \times t)^T dr}{w}\right] = 0 \ (a.e.). \tag{A-2}$$

Recall that h_1, h_2 and h_3 are real-valued functions; $(u, v) \in \mathcal{P}$ is a point in the parameter space; η represents the direction angle; w represents the side step. Both η and w are functions on the parameter space $\mathcal{P}$.

Now, we start some "brain gymnastics."

Differential Forms

For the derivation of compatibility equation, differential forms are used. We summarize the necessary results especially for planes—2 dimensional Euclidean spaces. In our case, the 2-dimensional Euclidean space in analysis is the parameter space $\mathcal{P}$ and the coordinate functions are denoted by u and v instead of the usual choice, x and y. In this summary, we do not discern the definitions from what are deduced; for the rigorous formalization/axiomatization from various perspectives, readers are referred to some texts on this subject such as [103~108].

In the following statements, by the letters like f, f_1, f_2, g, g_1, g_2 and h, we denote usual functions on $\mathcal{P}$; by a and b, real numbers; by n and m, non-negative integers; by φ and ψ, differential forms.

(A-3) A *differential 0-form* is a function on $\mathcal{P}$.

(A-4) A *differential 1-form* is $f \cdot du + g \cdot dv$, for certain functions f and g on $\mathcal{P}$.

(A-5) A *differential 2-form* is $h \cdot dudv$, for a function h on $\mathcal{P}$.

(A-6) $h \cdot (f_1 \cdot du + g_1 \cdot dv) \pm (f_2 \cdot du + g_2 \cdot dv) = (h \cdot f_1 \pm f_2) \cdot du + (h \cdot g_1 \pm g_2) \cdot dv$

(A-7) $f \cdot dudv + g \cdot dudv = (f + g) \cdot dudv$

(A-8) $dudv = -dvdu$ †

(A-9) $dudu = dvdv = 0$

(A-10) $(f_1 \cdot du + g_1 \cdot dv)\,(f_2 \cdot du + g_2 \cdot dv) = (f_1 g_2 - f_2 g_1) \cdot dudv$

(A-11) $\varphi \cdot \psi = (-1)^{nm} \cdot \psi \cdot \varphi$, for an *n-form* φ and an *m-form* ψ

(A-12) $df = \frac{\partial f}{\partial u} du + \frac{\partial f}{\partial v} dv$ ‡

(A-13) $d(f \cdot du + g \cdot dv) = df \cdot du + dg \cdot dv$

(A-14) $d(a\varphi + b\psi) = a \cdot d\varphi + b \cdot d\psi$

(A-15) $d(\varphi\psi) = (d\varphi) \cdot \psi + (-1)^n \cdot \varphi \cdot d\psi$, for an *n-form* φ

(A-16) $d(d\varphi) = 0$ for a C^2 differential form φ

Matrix Representation and Some useful Relations

Adopting matrix representation can simplify some algebraic manipulation.

A mapping $\boldsymbol{r}_{*u}: T_u\mathcal{P} \to T_{r(u)}\mathcal{W}$, which is defined by

$$\boldsymbol{r}_{*u}(\dot{\boldsymbol{u}}) \equiv \frac{d}{dt}\boldsymbol{r}(\boldsymbol{u} + t \cdot \dot{\boldsymbol{u}})\bigg|_{t=0} \tag{A-17}$$

is called the *derivative mapping* of the surface map, $\boldsymbol{r}: \mathcal{P} \to \mathcal{W}$, at $\boldsymbol{u} = (u, v) \in \mathcal{P}$. It is not difficult to

† $dudv$ is sometimes denoted by $du \wedge dv$ and this product is called *wedge product* or *exterior product*. The reason why we do not specify the wedge product explicitly is that the context usually makes it clear which product is used in an expression.

‡ The operation denoted by the symbol d is called the *exterior derivative*.

show that this mapping is a *linear transformation* [109]. It is a well-known result of linear algebra that there is a corresponding matrix to a linear transformation with respect to a particular choice of bases. The corresponding matrix is called a *Jacobian matrix*. We choose the "natural" bases of (the tangent spaces of) the parameter space $\mathcal{P}$ and the workpiece space $\mathcal{W}$. The Jacobian matrix of the surface (map) $\boldsymbol{r}$ with respect to their natural bases will be denoted by $\boldsymbol{A}$, *i.e.*

$$r_{*_u}(\dot{u}) = A\dot{u} \text{ and } A \equiv [r_u \mid r_v] \tag{A-18}$$

by setting the correspondence, $r_{*_u} \leftrightarrow A$. As mentioned earlier, we adopt the convention that any vector is treated as a column vector for matrix representation.

The *Jacobian matrix* $\boldsymbol{A}$ is related to the *first fundamental form matrix* $\boldsymbol{G}$ by

$$G = A^T A. \tag{A-19}$$

The *modal matrix* $\boldsymbol{P}$ is defined in terms of the first fundamental form coefficients (Equation 4, Chapter 2) and satisfies the following relation

$$P^T G P = I. \tag{A-20}$$

Resorting to the elementary matrix theory, we get the following immediate result:

$$p^2 g = 1,\ (GP)^{-T} = P,\ GPP^T = I \text{ and } PP^T = G^{-1} = G^{ad}/g, \tag{A-21}$$

where $p \equiv Det(P)$, $g \equiv Det(G)$ and

$$G^{ad} = \begin{bmatrix} g_{22} & -g_{12} \\ -g_{21} & g_{11} \end{bmatrix}. \tag{A-22}$$

The modal matrix $\boldsymbol{P}$ and the Jacobian matrix $\boldsymbol{A}$ are essential in the matrix representation of tangent vectors of the surface. For example,

$$n \times t = APRa \tag{A-23}$$

$$dr = A \cdot du \tag{A-24}$$

where $a \equiv [\cos\eta \quad \sin\eta]^T$ and $\boldsymbol{R}$ is the 90 degree rotation matrix.

Abusing the notation somewhat, we adopt the matrix multiplication rule for exterior product, for example,

$$du\,du^T = \begin{Bmatrix} du \\ dv \end{Bmatrix} [du \ dv] = \begin{bmatrix} dudu & dudv \\ dvdu & dvdv \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot dudv = -R\,dudv = R^T dudv. \tag{A-25}$$

We also write:

$$d[f^T \cdot du] = df^T \cdot du = d[du^T \cdot f] = -du^T \cdot df \tag{A-26}$$

where $f \equiv [f \;\; g]^T$; this is a matrix representation of (A-13). The minus sign in the last term is from (A-11).

$90°$-rotation matrix $\boldsymbol{R}$ plays interesting roles in plane geometry as in (A-25). Among them, the following formula may interest you:

$$\boldsymbol{R}^T\boldsymbol{M}\boldsymbol{R} = Det(\boldsymbol{M}) \cdot \boldsymbol{M}^{-T}, \tag{A-27}$$

for any non-singular 2-by-2 matrix $\boldsymbol{M}$. Also,

$$da = d\begin{Bmatrix} cos\eta \\ sin\eta \end{Bmatrix} = \begin{Bmatrix} -sin\eta \\ cos\eta \end{Bmatrix} \cdot d\eta = \boldsymbol{Ra} \cdot d\eta = \boldsymbol{Ra} \cdot (\eta_u du + \eta_v dv). \tag{A-28}$$

The following is evident:

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = 1 \tag{A-29}$$

which merely states that $90 - 90 = 0$.

(A-2) Expanded:

We apply the rule (A-15) to (A-2):

$$d\left[\frac{1}{w}\right] \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} + \frac{1}{w} \cdot d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}] = 0. \tag{A-30}$$

Applying (A-12) further, we get

$$\frac{-1}{w^2}\left[\frac{\partial w}{\partial u} \;\; \frac{\partial w}{\partial v}\right] d\boldsymbol{u} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} + \frac{1}{w} \cdot d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}] = 0. \tag{A-31}$$

After all, we will show that $(LHT) = (RHT)$ is equivalent to (A-1), where

$$\begin{aligned} (LHT) &= \left[\frac{\partial w}{\partial u} \;\; \frac{\partial w}{\partial v}\right] d\boldsymbol{u} \cdot (\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} \\ (RHT) &= w \cdot d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}] \end{aligned}. \tag{A-32}$$

(LHT) Simplified:

We plug (A-23, A-24) into the 1-form $(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}$ in (A-32):

$$(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} = d\boldsymbol{r}^T(\boldsymbol{n} \times \boldsymbol{t}) = d\boldsymbol{u}^T \boldsymbol{A}^T \cdot \boldsymbol{APRa}. \qquad \text{(A-33)}$$

By (A-19), the 1-form is reduced to

$$(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r} = d\boldsymbol{u}^T \cdot \boldsymbol{GPRa}. \qquad \text{(A-34)}$$

Using this and (A-25), (LHT) is reduced to the following form:

$$(LHT) = \begin{bmatrix} \frac{\partial w}{\partial u} & \frac{\partial w}{\partial v} \end{bmatrix} \cdot d\boldsymbol{u} \cdot d\boldsymbol{u}^T \cdot \boldsymbol{GPRa} = dudv \cdot \begin{bmatrix} \frac{\partial w}{\partial u} & \frac{\partial w}{\partial v} \end{bmatrix} \cdot \boldsymbol{R}^T \boldsymbol{GPRa}. \qquad \text{(A-35)}$$

We apply (A-27 and A-21):

$$\boldsymbol{R}^T(\boldsymbol{GP})\boldsymbol{R} = gp\boldsymbol{P} = \sqrt{g} \cdot \boldsymbol{P}. \qquad \text{(A-36)}$$

Plugging this into (A-35), we get

$$(LHT) = \begin{bmatrix} \frac{\partial w}{\partial u} & \frac{\partial w}{\partial v} \end{bmatrix} \cdot \boldsymbol{Pa}\sqrt{g} \cdot dudv = \left(h_1(\eta, u, v) \cdot \frac{\partial w}{\partial u} + h_2(\eta, u, v) \cdot \frac{\partial w}{\partial v}\right) \cdot \sqrt{g} \cdot dudv, \qquad \text{(A-37)}$$

where we used the definition, $\boldsymbol{h} = [h_1 \;\; h_2]^T = \boldsymbol{Pa}$ (Equation 6).

(RHT) Simplified:

We apply (A-34 and A-26):

$$d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}] = -d\boldsymbol{u}^T \cdot d[\boldsymbol{GPRa}]. \qquad \text{(A-38)}$$

Recalling the general matrix multiplication rule and (A-15, A-28, A-29), we get

$$\begin{aligned} d[\boldsymbol{GPRa}] &= d[\boldsymbol{GP}] \cdot \boldsymbol{Ra} + \boldsymbol{GPR} \cdot d\boldsymbol{a} \qquad \text{(A-39)} \\ &= \left(du \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) + dv \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra} + (\eta_u du + \eta_v dv) \cdot \boldsymbol{GPRRa} \\ &= \left(du \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) + dv \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra} - (\eta_u du + \eta_v dv) \cdot \boldsymbol{GPa} \end{aligned}$$

We plug this into (A-38) and apply (A-8, A-9, A-36):

$$d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}] \tag{A-40}$$

$$= \left\{\left([0\ \ 1] \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) - [1\ \ 0] \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra} + ([0\ \ -1] \cdot \eta_u + [1\ \ 0] \cdot \eta_v) \cdot \boldsymbol{GPa}\right\} \cdot du dv$$

$$= \left\{[\eta_u\ \ \eta_v] \cdot \boldsymbol{R} \cdot \boldsymbol{GPa} + \left([0\ \ 1] \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) - [1\ \ 0] \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra}\right\} \cdot du dv$$

$$= \left\{\sqrt{g} \cdot [\eta_u\ \ \eta_v] \cdot \boldsymbol{PRa} + \left([0\ \ 1] \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) - [1\ \ 0] \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra}\right\} \cdot du dv$$

Collecting Terms:

Now plug (A-37, A-40) into (A-32) and set $(LHT)/\sqrt{g} = (RHT)/\sqrt{g}$:

$$h_1 \cdot \frac{\partial w}{\partial u} + h_2 \cdot \frac{\partial w}{\partial v} = \left\{[\eta_u\ \ \eta_v] \cdot \boldsymbol{PRa} + \left([0\ \ 1] \cdot \frac{\partial}{\partial u}(\boldsymbol{GP}) - [1\ \ 0] \cdot \frac{\partial}{\partial v}(\boldsymbol{GP})\right) \cdot \boldsymbol{Ra}/\sqrt{g}\right\} \cdot w. \tag{A-41}$$

We factored with $\sqrt{g}$ so that resulting terms could have some geometric meaning.

Just by setting

$$h_3 \equiv \left[[\eta_u\ \ \eta_v]\boldsymbol{P} + \left([0\ \ 1]\frac{\partial}{\partial u}(\boldsymbol{GP}) - [1\ \ 0]\frac{\partial}{\partial v}(\boldsymbol{GP})\right)/\sqrt{g}\right] \cdot \boldsymbol{Ra}, \tag{A-42}$$

(A-41) becomes (A-1). Now, we have shown that (A-1) is derived from (A-2).

A.2 Compatibility in terms of a Generating Function

The *generating function* V was defined as a smooth function on the parameter space $\mathcal{P}$, which satisfies the following relation to the direction angle η:

$$\boldsymbol{a}(\eta) \equiv \begin{Bmatrix} \cos\eta \\ \sin\eta \end{Bmatrix} = \frac{\boldsymbol{R}^T\boldsymbol{P}^T\nabla V}{\sqrt{\nabla V^T \boldsymbol{P}\boldsymbol{P}^T \nabla V}} \left(= \frac{\boldsymbol{P}^{-1}\boldsymbol{R}^T\nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right), \text{ where } \nabla V \equiv \begin{Bmatrix} V_u \\ V_v \end{Bmatrix}. \tag{A-43}$$

Here, we derive the compatibility somewhat independently instead of directly plugging (A-43) into (A-1); dealing with the *arctangent* function is somewhat inconvenient. Our objective is equivalent to expressing h_1, h_2 and h_3 in terms of the generating function V instead of the direction angle η.

Since the term $\nabla V^T \boldsymbol{G}^{ad} \nabla V$ shows up frequently, we abbreviate this term to f: $f \equiv \nabla V^T \boldsymbol{G}^{ad} \nabla V$.

(LHT): h_1 AND h_2

We do easier one first. Using (A-43), we have

$$\boldsymbol{h} = [h_1\ \ h_2]^T = \boldsymbol{Pa} = \boldsymbol{P} \cdot \frac{\boldsymbol{P}^{-1}\boldsymbol{R}^T \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}} = \frac{\boldsymbol{R}^T \nabla V}{\sqrt{f}}. \tag{A-44}$$

In other words,

$$h_1 = V_v / \sqrt{f} \text{ and } h_2 = -V_u / \sqrt{f}. \tag{A-45}$$

(RHT): h_3

Comparing (A-42) with (A-38 and A-40), we say that

$$h_3 \cdot dudv = \frac{d[(\boldsymbol{n} \times \boldsymbol{t})^T d\boldsymbol{r}]}{\sqrt{g}} = \frac{-d\boldsymbol{u}^T \cdot d[\boldsymbol{GPRa}]}{\sqrt{g}}. \tag{A-46}$$

Using (A-43, A-29 and A-21), we have

$$\boldsymbol{GPRa} = \boldsymbol{GPR} \cdot \frac{\boldsymbol{R}^T \boldsymbol{P}^T \nabla V}{\sqrt{\nabla V^T \boldsymbol{PP}^T \nabla V}} = \frac{\sqrt{g} \cdot \nabla V}{\sqrt{g \cdot \nabla V^T \boldsymbol{PP}^T \nabla V}} = \frac{\sqrt{g} \cdot \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}. \tag{A-47}$$

We apply the exterior derivative to the above equation in a component-wise manner, and apply (A-12):

$$d[\boldsymbol{GPRa}] = \frac{\partial}{\partial u}\left(\frac{\sqrt{g} \cdot \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right) \cdot du + \frac{\partial}{\partial v}\left(\frac{\sqrt{g} \cdot \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right) \cdot dv. \tag{A-48}$$

We perform the wedge product between this and $-d\boldsymbol{u}^T$:

$$-d\boldsymbol{u}^T \cdot d[\boldsymbol{GPRa}] = \left\{[0\ \ 1] \cdot \frac{\partial}{\partial u}\left(\frac{\sqrt{g} \cdot \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right) - [1\ \ 0] \cdot \frac{\partial}{\partial v}\left(\frac{\sqrt{g} \cdot \nabla V}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right)\right\} \cdot dudv \tag{A-49}$$

$$= \left\{\frac{\partial}{\partial u}\left(\frac{\sqrt{g} \cdot V_v}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right) - \frac{\partial}{\partial v}\left(\frac{\sqrt{g} \cdot V_u}{\sqrt{\nabla V^T \boldsymbol{G}^{ad} \nabla V}}\right)\right\} \cdot dudv = \{I_1 - I_2\} \cdot dudv,$$

where

$$I_1 \equiv \frac{\partial}{\partial u}\left(\frac{\sqrt{g}\cdot V_v}{\sqrt{\nabla V^T G^{ad} \nabla V}}\right) \tag{A-50}$$

$$= \left\{\left(\frac{g_u}{2\sqrt{g}}\cdot V_v + \sqrt{g}\cdot V_{uv}\right)\cdot\sqrt{\nabla V^T G^{ad}\nabla V} - \sqrt{g}\cdot V_v\cdot\frac{2\cdot(\nabla V)_u^T\cdot G^{ad}\nabla V + \nabla V^T G_u^{ad}\nabla V}{2\sqrt{\nabla V^T G^{ad}\nabla V}}\right\}/\{\nabla V^T G^{ad}\nabla V\}$$

$$= \{(g_u\cdot V_v + 2g\cdot V_{uv})\cdot\{\nabla V^T G^{ad}\nabla V\} - g\cdot V_v\cdot\{2\cdot(\nabla V)_u^T\cdot G^{ad}\nabla V + \nabla V^T G_u^{ad}\nabla V\}\}/\{2\cdot f^{3/2}\cdot\sqrt{g}\}$$

and similarly,

$$I_2 \equiv \frac{\partial}{\partial v}\left(\frac{\sqrt{g}\cdot V_u}{\sqrt{\nabla V^T G^{ad} \nabla V}}\right) \tag{A-51}$$

$$= \{(g_v\cdot V_u + 2g\cdot V_{uv})\cdot\{\nabla V^T G^{ad}\nabla V\} - g\cdot V_u\cdot\{2\cdot(\nabla V)_v^T\cdot G^{ad}\nabla V + \nabla V^T G_v^{ad}\nabla V\}\}/\{2\cdot f^{3/2}\cdot\sqrt{g}\}$$

Now, if we compare (A-49) with (A-46), we notice that

$$h_3 = \{I_1 - I_2\}/\sqrt{g} \tag{A-52}$$

$$= \frac{(g_u\cdot V_v - g_v\cdot V_u)\cdot\nabla V^T G^{ad}\nabla V - 2\cdot g\cdot\{V_v(\nabla V)_u^T - V_u(\nabla V)_v^T\}\cdot G^{ad}\nabla V - g\cdot\nabla V^T\{V_v G_u^{ad} - V_u G_v^{ad}\}\nabla V}{2\cdot f^{3/2}\cdot g}$$

Recalling that G^{ad} is a symmetric matrix, we do some tricks for the 2nd term:

$$-\{V_v(\nabla V)_u^T - V_u(\nabla V)_v^T\}\cdot G^{ad}\nabla V = \nabla V^T G^{ad}\cdot\{-V_v(\nabla V)_u + V_u(\nabla V)_v\} \tag{A-53}$$

$$= \nabla V^T G^{ad}\cdot\left[\begin{Bmatrix} V_u \\ V_v \end{Bmatrix}_u \begin{Bmatrix} V_u \\ V_v \end{Bmatrix}_v\right]\begin{Bmatrix} -V_v \\ V_u \end{Bmatrix} = \nabla V^T\cdot G^{ad}\cdot \boldsymbol{HR}\cdot\nabla V$$

where we denoted *Hessian* matrix by $\boldsymbol{H}$:

$$\boldsymbol{H} = \left[\begin{Bmatrix} V_u \\ V_v \end{Bmatrix}_u \begin{Bmatrix} V_u \\ V_v \end{Bmatrix}_v\right] = \begin{bmatrix} V_{uu} & V_{uv} \\ V_{uv} & V_{vv} \end{bmatrix}. \tag{A-54}$$

Then we are lead to the following equation:

$$h_3 = \frac{\nabla V^T \boldsymbol{W}\nabla V}{2\cdot f^{3/2}\cdot g} \tag{A-55}$$

where

$$W \equiv G^{ad}\ \{2gHR + (g_u \cdot V_v - g_v \cdot V_u) \cdot 1\}\ - g\{V_v G_u^{ad} - V_u G_v^{ad}\}\ .$$

In summary,

$$2 \cdot f^{3/2} \cdot g \cdot \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \end{Bmatrix} = \begin{Bmatrix} (2fg)V_v \\ -(2fg)V_u \\ \nabla V^T W \nabla V \end{Bmatrix} . \qquad \text{(A-56)}$$

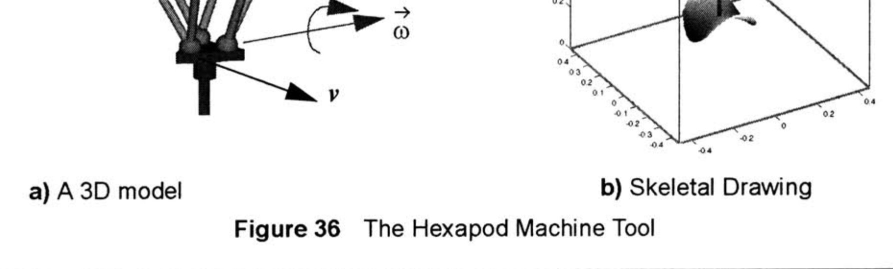
Now, the compatibility is expressed in the following form:

$$\frac{1}{\sqrt{f}} \cdot \left\{ V_v \cdot \frac{\partial w}{\partial u} - V_u \cdot \frac{\partial w}{\partial v} \right\} = \frac{1}{\sqrt{f}} \cdot \left\{ \frac{\nabla V^T \cdot \quad W \quad \cdot \nabla V}{\nabla V^T \cdot 2g \cdot G^{ad} \cdot \nabla V} \right\} \cdot w\ . \qquad \text{(A-57)}$$

APPENDIX B SURFACE INVERSE KINEMATICS

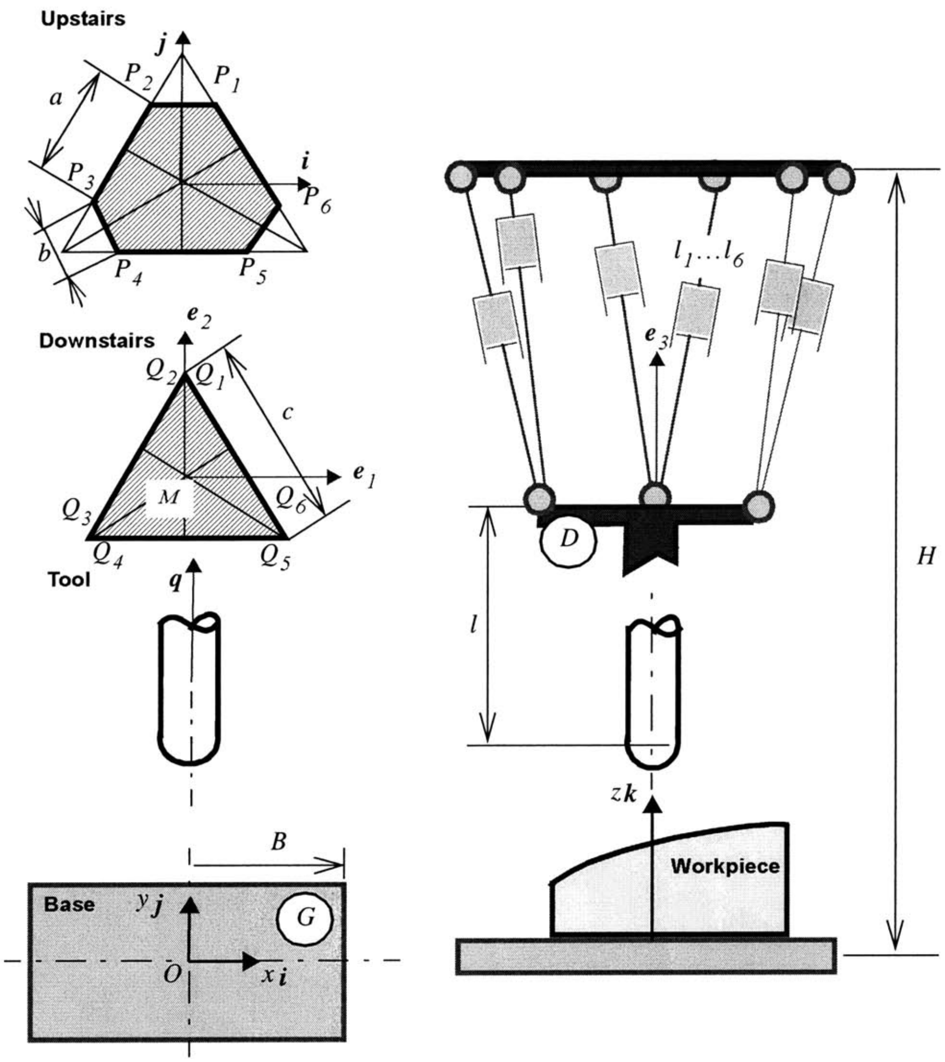
B.1 The Dimensions of the Hexapod Machine Tool

In Figure 36, a hexapod machine tool is shown. Its controller dictates the required motion to each link. As each link moves up and down with respect to the ceiling, the distance l_i between two spherical joints along the i^{th} link varies. In the end, we control the motion of the rigid body that hangs from the ceiling. We call the "triangular" rigid body the *downstairs* of the hexapod machine tool. The hexagonal ceiling is referred to as its *upstairs*. In our previous terminology, the downstairs is the spindle base (*c.f.* Figure 9). At the center of the downstairs, we attach an imaginary point M, which we regard as the cutter location point. We also attach an orthonormal basis $\{e_1, e_2, e_3\}$ on the downstairs as shown in Figure 37. The CL-point M together with the basis $\{e_1, e_2, e_3\}$ forms a reference frame, which we refer to as D. In the bottom



a) A 3D model **b)** Skeletal Drawing

Figure 36 The Hexapod Machine Tool



• Points Named

P_i: *Joint Positions* at *Upstairs*. ($i = 1 \ldots 6$)

Q_i: Corresponding *Joint Positions* on *Downstairs*.

M: The Cutter Location Point

O: Center of the *Base*. Origin of the *Ground Reference Frame*. Origin of *Workpiece Space*. In this special case, the 3 points are the same.

$G = Oxyz$: *Ground Reference Frame* with the corresponding *Basis* $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$. Inertial.

D: The reference frame fixed at the downstairs whose basis is $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ and whose origin is M. Non-inertial.

Figure 37 The Neutral Position of Hexapod

of the machine tool, we attach an orthonormal basis $\{i, j, k\}$. We refer to the bottom as the *base* of the machine tool. We attach a point O at the center of the base. The point O together with the basis $\{i, j, k\}$ forms a reference frame, which we refer to as G. The same frame can be taken for the workpiece space $\mathcal{W}$, either. (In this case, the change of frame between the workpiece space and the inertial reference frame is the trivial unity.) It is well known that the inverse kinematics of a hexapod (a parallel mechanism) is readily obtained (while getting its forward kinematics is quite challenging).

In Figure 37, we show the geometric dimensions that we need to specify, which are a, b, c, H, B and l. And, we specified their values:

$$a = 26'',\ b = 10'',\ c = 12'',\ H = 4',\ B = 18'' \text{ and } l = 6''.$$

They are the inputs in our problem. Here, we show how to get the coordinates of the vertices of the upstairs and the downstairs from the specified geometric input data.

THE VERTICES OF THE UPSTAIRS HEXAGON:

We refer to the vertices of the Hexagon as P_i $(i = 1\ldots6)$ as shown in the figure below. Let a^i denote its position (vector) in the workpiece space $\mathcal{W}$, whose basis is $\{i, j, k\}$. Then, the point is determined from the given geometric inputs in the following way:

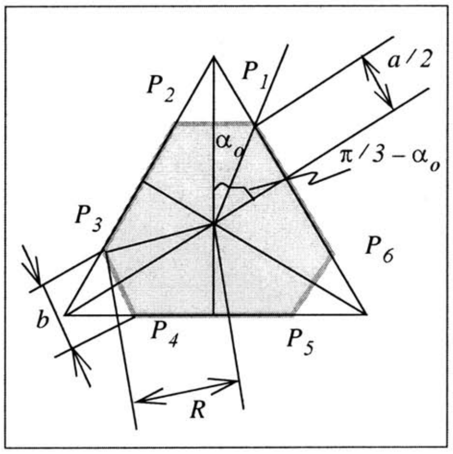
$$a^i \equiv \overrightarrow{OP_i} = [R \cdot \cos\alpha_i \quad R \cdot \sin\alpha_i \quad H]^T$$

where

$$\alpha_o = \arctan\left(\frac{\sqrt{3}}{2 \cdot (a/b) + 1}\right), \qquad (0 \le \alpha_o \le \pi/2)$$

$$R = \frac{b}{2} \cdot \sqrt{\frac{\{2 \cdot (a/b) + 1\}^2 + 3}{3}}$$

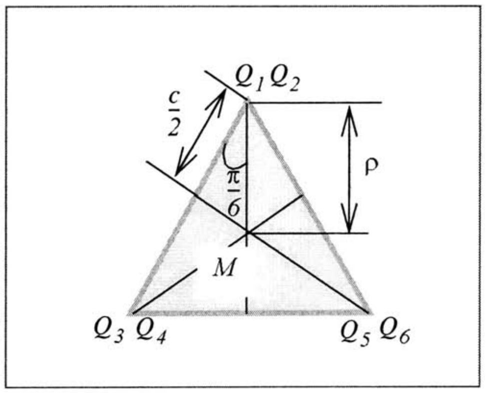
$$\langle\alpha_i\rangle \equiv \begin{cases} \alpha_1 = & \pi/2 - \alpha_o \\ \alpha_2 = & \pi/2 + \alpha_o \\ \alpha_{n+2} = & \alpha_n + 2\pi/3 \end{cases}, (i = 1\ldots6 \text{ and } n = 1\ldots4).$$



THE VERTICES OF THE DOWNSTAIRS TRIANGLE:

Just for convenience, we assign two points at each vertex of the downstairs triangle as shown in the figure below. The points are denoted by Q_i $(i = 1...6)$. By $\boldsymbol{b}^i$, we denote the coordinates of the vertices (observed) in the downstairs frame D. Now, we have:

$$\boldsymbol{b}^i = [\rho \cdot cos\alpha_i \quad \rho \cdot sin\alpha_i \quad 0]^T$$



where

$$\rho = \frac{c}{\sqrt{3}}$$

$$\langle \beta_i \rangle \equiv \begin{cases} \beta_1 = & \pi/2 \\ \beta_2 = & \pi/2 \\ \beta_{n+2} = & \beta_n + 2\pi/3 \end{cases} , i = 1...6 , n = 1...4 .$$

B.2 The Kinematic Control Law

The hexapod machine tool is a redundant machine tool for 5-axis machining because it has 6 independent axes. A simple way to resolve the redundancy is to impose a constraint between the generalized coordinates of the system. In the given particular hexapod, its controller imposes the following constraint:

$$\boldsymbol{e}_1^T \boldsymbol{j} = 0$$

which implies that $\overline{Q_4 Q_5} \, // \, xz\text{-}plane$, in the geometric sense.

B.3 The Kinematics

1. The Rigid Body Motion of the Downstairs

The above specific kinematic control law implies that the basis vector $\boldsymbol{e}_1$ of the downstairs is expressed in the following form:

$$\boldsymbol{e}_1 = cos\alpha \cdot \boldsymbol{i} - sin\alpha \cdot \boldsymbol{k} \qquad \text{or} \qquad \boldsymbol{e}_1 = [cos\alpha \quad 0 \quad -sin\alpha]^T \qquad \exists(\alpha \in \mathcal{R}).$$

The orientation $\boldsymbol{q}$ is a body-fixed direction and, in fact, it coincides with $\boldsymbol{e}_3$, *i.e.*

$$\boldsymbol{e}_3 = \boldsymbol{q} .$$

Since $\boldsymbol{e}_3$ should be perpendicular to $\boldsymbol{e}_1$, we have:

$$e_3^T e_1 = q^T (\cos\alpha \cdot i - \sin\alpha \cdot k) = q_x \cos\alpha - q_z \sin\alpha = 0.$$

Therefore,

$$\cos\alpha = \frac{q_z}{\sqrt{q_x^2 + q_z^2}} \quad \text{and} \quad \sin\alpha = \frac{q_x}{\sqrt{q_x^2 + q_z^2}}.$$

In summary,

$$e_1 = \frac{[\ q_z \quad 0 \quad -q_x\ ]^T}{\sqrt{q_x^2 + q_z^2}}$$

$$e_2 = e_3 \times e_1 = \frac{[-q_x q_y \quad q_x^2 + q_z^2 \quad -q_z q_y]^T}{\sqrt{q_x^2 + q_z^2}}$$

$$e_3 = [\ q_x \quad q_y \quad q_z\ ]^T.$$

Let Q be the rigid body rotation of the downstairs with respect to the workpiece space. By definition,

$$Q \equiv [\ e_1 \quad e_2 \quad e_3\ ] = \frac{1}{q_o} \begin{bmatrix} q_z & -q_x q_y & q_o q_x \\ 0 & q_o^2 & q_o q_y \\ -q_x & -q_z q_y & q_o q_z \end{bmatrix} \qquad \text{where} \quad q_o \equiv \sqrt{q_x^2 + q_z^2}.$$

Rotation Applied:

The vertices of the downstairs triangle in the workpiece space is

$$\overrightarrow{MQ_i} = Q\, b^i$$

when the CL-point M is at the origin of the workpiece space.

Translation Applied:

We further apply the translation:

$$\overrightarrow{OQ_i} = \overrightarrow{MQ_i} + \boldsymbol{r}^M = \boldsymbol{Q}\,\boldsymbol{b}^i + \boldsymbol{r}^M \quad .$$

2. Joint Displacement

Using the above equation, we have

$$\overrightarrow{P_iQ_i} = \overrightarrow{OQ_i} - \overrightarrow{OP_i} = \boldsymbol{Q}\,\boldsymbol{b}^i + \boldsymbol{r}^M - \boldsymbol{a}^i .$$

Note that $\boldsymbol{b}^i$ and $\boldsymbol{a}^i$ are constants. Now, we derive the distance between the two spherical joints along the i^{th} link in terms of $(\boldsymbol{r}^M, \boldsymbol{Q})$:

$$\begin{aligned} l_i^2 &= \overrightarrow{P_iQ_i}^T\overrightarrow{P_iQ_i} = (\boldsymbol{Q}\,\boldsymbol{b}^i + \boldsymbol{r}^M - \boldsymbol{a}^i)^T(\boldsymbol{Q}\,\boldsymbol{b}^i + \boldsymbol{r}^M - \boldsymbol{a}^i) \quad . \\ &= (\boldsymbol{b}^i)^T\boldsymbol{b}^i + (\boldsymbol{a}^i)^T\boldsymbol{a}^i + (\boldsymbol{r}^M)^T\boldsymbol{r}^M + 2\cdot\left((\boldsymbol{r}^M)^T\boldsymbol{Q}\boldsymbol{b}^i - (\boldsymbol{a}^i)^T\boldsymbol{Q}\boldsymbol{b}^i - (\boldsymbol{a}^i)^T\boldsymbol{r}^M\right) \end{aligned}$$

Note that we used $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{1}$. The equation above can be arranged in the following form:

$$l_i^2 = (\boldsymbol{b}^i)^T\boldsymbol{b}^i + (\boldsymbol{a}^i)^T\boldsymbol{a}^i + (\boldsymbol{r}^M)^T\boldsymbol{r}^M + 2\cdot\left((\boldsymbol{r}^M - \boldsymbol{a}^i)^T\boldsymbol{Q}\boldsymbol{b}^i - (\boldsymbol{a}^i)^T\boldsymbol{r}^M\right) \quad . \tag{B-1}$$

The displacement is also expressed by

$$l_i^2 = (\boldsymbol{b}^i)^T\boldsymbol{b}^i + (\boldsymbol{a}^i)^T\boldsymbol{a}^i + (\boldsymbol{r}^M)^T\boldsymbol{r}^M + 2\cdot\left((\boldsymbol{r}^M)^T\boldsymbol{c}^i - (\boldsymbol{a}^i)^T\boldsymbol{Q}\boldsymbol{b}^i\right) \quad . \tag{B-2}$$

where we set

$$\boldsymbol{c}^i \equiv \boldsymbol{Q}\boldsymbol{b}^i - \boldsymbol{a}^i \quad .$$

Both forms are useful.

3. Orientation Parametrization and the Surface Inverse Kinematics

Remarks on the Parametrization of the Orientation:

The orientation vector can be parametrized by two parameters because it is a unit vector in a 3-dimensional space. Generally, we cannot avoid the representation singularities of a unit sphere (using only 2 parameters). However, we point out that the cutting tool is not allowed to penetrate the designed solid and the orientation we need is not the entire unit sphere. The set of orientation under our interest is only a "north" hemisphere whose equator is on the tangent plane of the designed surface, which is topologically equivalent to a bounded 2-dimensional domain. As a result, we avoid the representation singularities easily if we confine the singularities within the "south" hemisphere. For example, we can adopt the two angles of a spherical coordinate system, whose singularity is in the "equator." It is also possible to use a stereographic projection whose singularity is on the "south" pole. In practice, it is more convenient to define the orientation with respect to the tangent plane of the surface. To accommodate this situation, we allow the parametrization to depend on the locations of the surface. Therefore, the orientation is parametrized in the following form:

$$\boldsymbol{q} = \boldsymbol{q}_o(\varphi, \phi, u, v)$$

where $\boldsymbol{q}_o$ is a parametrization (map), (φ, ϕ) are two parameters for the orientation and (u, v) is a point on the parameter space $\mathcal{P}$. By preassigning the vector field with $\varphi = \varphi_o(u, v)$ and $\phi = \phi_o(u, v)$, we have the orientation in the following form:

$$\boldsymbol{q}(u, v) = \boldsymbol{q}_o(\varphi_o(u, v), \phi_o(u, v), u, v)$$

forming a unit vector field on the surface.

The Requirement of Tangency:

The cutter location point M is defined by the requirement of tangency:

$$\boldsymbol{r}^M = \boldsymbol{r} + R \cdot \boldsymbol{n} + l \cdot \boldsymbol{q} \tag{B-3}$$

The Surface Inverse Kinematics:

Once the orientation vector is parametrized by two angles, φ and ϕ, we substitute $\boldsymbol{q}$ in Equation (B-1 or B-2) with the parameterization $\boldsymbol{q}_o(\varphi, \phi, u, v)$ and substitute the cutter location point $\boldsymbol{r}^M$ with $\boldsymbol{r}(u, v) + R \cdot \boldsymbol{n}(u, v) + l \cdot \boldsymbol{q}_o(\varphi, \phi, u, v)$. The surface inverse kinematics is readily obtained. Substitution with the assignment $\boldsymbol{q}(u, v)$ will yield the restricted surface inverse kinematics:

$$f^o : \mathcal{P} \rightarrow \mathcal{R}^6, \qquad (u, v) \rightarrow (l_1, l_2, l_3, l_4, l_5, l_6).$$

B.4 The Jacobian Matrix of the Surface Inverse Kinematics

Using Equation (B-1, B-2 and B-3), we derive the following formula for the Jacobian matrix $\boldsymbol{C}$ of the surface inverse kinematics map f^o. (Note that $\boldsymbol{C} \equiv [f_u^o\, f_v^o]$ and $\omega = \boldsymbol{C}\dot{\boldsymbol{u}}$).

$$\boldsymbol{C} = \boldsymbol{J}^M \cdot (\boldsymbol{A} + R \cdot \boldsymbol{N} + l \cdot \boldsymbol{S}) + \boldsymbol{J}^Q \cdot \boldsymbol{S}$$

where $\quad \boldsymbol{J}^M = [J_{ij}^M], \quad \boldsymbol{J}^Q = [J_{ij}^Q], \quad \boldsymbol{N} \equiv [\boldsymbol{n}_u\ \boldsymbol{n}_v], \quad \boldsymbol{S} \equiv [\boldsymbol{q}_u\ \boldsymbol{q}_v], \quad A \equiv [\boldsymbol{r}_u\ \boldsymbol{r}_v],$

$$J_{ij}^M = \frac{r_j^M + c_j^i}{l_i}, \qquad J_{ij}^Q = \left\{\boldsymbol{r}^M - \boldsymbol{a}^i\right\}^T \frac{\partial \boldsymbol{Q}^j \boldsymbol{b}^i}{l_i}, \qquad \boldsymbol{c}^i \equiv \boldsymbol{Q}\boldsymbol{b}^i - \boldsymbol{a}^i, \qquad \boldsymbol{r}^M = \boldsymbol{r} + R \cdot \boldsymbol{n} + l \cdot \boldsymbol{q},$$

$$l_i^2 = (\boldsymbol{b}^i)^T \boldsymbol{b}^i + (\boldsymbol{a}^i)^T \boldsymbol{a}^i + (\boldsymbol{r}^M)^T \boldsymbol{r}^M + 2 \cdot ((\boldsymbol{r}^M)^T \boldsymbol{c}^i - (\boldsymbol{a}^i)^T \boldsymbol{Q} \boldsymbol{b}^i), \quad q_o \equiv \sqrt{\boldsymbol{q}_x^2 + \boldsymbol{q}_z^2}$$

$$\boldsymbol{Q} \equiv \frac{1}{q_o} \begin{bmatrix} \boldsymbol{q}_z & -\boldsymbol{q}_x \boldsymbol{q}_y & q_o \boldsymbol{q}_x \\ 0 & q_o^2 & q_o \boldsymbol{q}_y \\ -\boldsymbol{q}_x & -\boldsymbol{q}_z \boldsymbol{q}_y & q_o \boldsymbol{q}_z \end{bmatrix},$$

$$\partial \boldsymbol{Q}^1 = \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{q}_x} = \frac{1}{q_o^3} \begin{bmatrix} -\boldsymbol{q}_x \boldsymbol{q}_z & -\boldsymbol{q}_z^2 \boldsymbol{q}_y & q_o^3 \\ 0 & \boldsymbol{q}_x q_o^2 & 0 \\ -\boldsymbol{q}_z^2 & \boldsymbol{q}_x \boldsymbol{q}_y \boldsymbol{q}_z & 0 \end{bmatrix}, \quad \partial \boldsymbol{Q}^2 = \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{q}_y} = \frac{1}{q_o} \begin{bmatrix} 0 & -\boldsymbol{q}_x & 0 \\ 0 & 0 & q_o \\ 0 & -\boldsymbol{q}_z & 0 \end{bmatrix}, \quad \partial \boldsymbol{Q}^3 = \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{q}_z} = \frac{1}{q_o^3} \begin{bmatrix} \boldsymbol{q}_x^2 & \boldsymbol{q}_x \boldsymbol{q}_y \boldsymbol{q}_z & 0 \\ 0 & \boldsymbol{q}_z q_o^2 & 0 \\ \boldsymbol{q}_x \boldsymbol{q}_z & -\boldsymbol{q}_x^2 \boldsymbol{q}_y & q_o^3 \end{bmatrix}.$$

B.5 The Weingarten Formula and Surface Normal Machining

The following is the *Weingarten formula* [*e.g. see* 117]:

$$\boldsymbol{N} = -\boldsymbol{A}\boldsymbol{G}^{-1}\boldsymbol{D}$$

where $\boldsymbol{N} \equiv [\boldsymbol{n}_u\ \boldsymbol{n}_v]$, $A \equiv [\boldsymbol{r}_u\ \boldsymbol{r}_v]$, $\boldsymbol{G}$ is the first fundamental form matrix and $\boldsymbol{D}$ is the 2nd fundamental form matrix. This formula simplifies the evaluation of the Jacobian matrix of the surface inverse kinematics. Especially, for the surface normal machining,

$$\boldsymbol{C} = \boldsymbol{J}^M \boldsymbol{A} - \{(R + l)\boldsymbol{J}^M + \boldsymbol{J}^Q\}\boldsymbol{A}\boldsymbol{G}^{-1}\boldsymbol{D}$$

because $\boldsymbol{q} = \boldsymbol{n}$ and $\boldsymbol{S} = \boldsymbol{N} = -\boldsymbol{A}\boldsymbol{G}^{-1}\boldsymbol{D}$.

APPENDIX C THE SIDE-STEP-LIMIT

C.1 The Side-step-limit

In this appendix, we refine the formula (Equation 23) of the side-step-limit $w_o(\eta, u, v)$. Given a principal direction angle $\eta \in [0, \pi/2]$, principal curvatures (κ_1 and κ_2), a tool curvature κ_b and a cusp-height-limit h_o, we use the following formula to evaluate side-step-limit:

$$w_o = \begin{cases} \{w_1/\gamma + \gamma\}\beta & \text{if } h_o \le h_L(\eta) \\ \{w_1/\gamma_L + \gamma_L\}\beta_L & \text{otherwise} \end{cases} \tag{C-1}$$

where

1. $\gamma \equiv \sqrt{1 + (\kappa_s\beta)^2}$, $\gamma_L \equiv \sqrt{1 + (\kappa_s\beta_L)^2}$,

2. $\kappa_s(\eta) \equiv \kappa_1 \sin^2\eta + \kappa_2 \cos^2\eta$, (*Euler's formula*)

: κ_1 and κ_2 are the most concave and convex principal curvature, respectively.

3. $w_1 = \begin{cases} 1 & \text{if } [s \equiv \kappa_s\beta/\sqrt{1 + (\kappa_s\beta)^2}] = 0 \\ [\ln(1+s)^{1/s} - \ln(1-s)^{1/s}]/2 & \text{otherwise} \end{cases}$,

4. β_L is decided by solving the following polynomial equation:

$$F(\beta_L) \equiv (\kappa_b\kappa_s)^2\beta_L^4 - 2\kappa_b\kappa_s^2\beta_L^3 + \kappa_b^2\beta_L^2 - 2\kappa_b\beta_L + 1 = 0$$

in the interval of $(1/\kappa_b) \le \beta_L < (2/\kappa_b)$ *if* $\kappa_s \ge 0$

in the interval of $0 \le \beta_L < (1/\kappa_b)$ *if* $\kappa_s < 0$,

using any numerical methods. Note that the **uniqueness** of the solution in the given interval is proved. For *Newton-Rhapson* method, use the following derivative:

$$F'(\beta_L) = 2\kappa_b\{\gamma_L^2(2\kappa_b\beta_L - 3) + 1\}.$$

5. $h_L \equiv \left[\kappa_b\gamma_L - \frac{3}{2}\kappa_s\right]\beta_L^2$, and

6. β is decided by solving the following nonlinear equation in the interval $0 \le \beta \le \beta_L$ (only if $h_o \le h_L$):

$$h_o = \left[\frac{1}{2}\kappa_s + \frac{\{\kappa_b\gamma - 2\kappa_s\}}{1 + \sqrt{\gamma^2 - (\kappa_b\gamma - \kappa_s)^2\beta^2}}\right]\beta^2 \qquad \text{(C-2)}$$

with given h_o. Use the information that the right side term is **monotonic** with respect to β.

For other three quadrants of the $(\underset{\sim}{u}, \underset{\sim}{v})$-space, the side-step-limit is evaluated by the symmetry. For most cases, $w_o \approx 2\beta$ is accurate enough. If the direction angle η is given instead of the principal direction angle, we use the following relation:

$$\eta = \underset{\sim}{\eta} + \eta_o$$

where η_o is defined by the following system of equations

$$\cos\eta_o = \boldsymbol{i}_U^T\boldsymbol{r}_u/\|\boldsymbol{r}_u\| \text{ and } \sin\eta_o = \boldsymbol{n}^T(\boldsymbol{r}_u \times \boldsymbol{i}_U)/\|\boldsymbol{r}_u\|.$$

Note that $\boldsymbol{i}_U$ is the "most convex" principal direction. Instead of solving Equation (C-2), we could solve the following 5th order polynomial equation:

$$\begin{aligned}
&[\beta^2]^5\ \kappa_b^2 \cdot \kappa_s^6\kappa_b^2 + \\
&[\beta^2]^4\ \kappa_b^2 \cdot \kappa_s^4\kappa_b^2[9 - 8h_o\kappa_s] + \\
&[\beta^2]^3\ \kappa_b^2 \cdot 8\kappa_s^2\kappa_b^2[\,3 - 5(h_o\kappa_s) + 3(h_o\kappa_s)^2] - \\
&[\beta^2]^2\ \kappa_b^2 \cdot 8\,[\kappa_b^2\{-2 + 4(h_o\kappa_s) - 7(h_o\kappa_s)^2 + 4(h_o\kappa_s)^3\} + 18\kappa_s^2] + \\
&[\beta^2]\ \kappa_b^2 \cdot 16[\kappa_b^2\{\,2 - 2(h_o\kappa_s) + (h_o\kappa_s)^2\}h_o + 12\kappa_s]h_o + \\
&[1\ ]\ \kappa_b^2 \cdot 16[\kappa_b^2h_o^2 - 4]h_o^2 \\
&= 0
\end{aligned}$$

Derivation

The normal section in a normal plane of a streamline at $\boldsymbol{u} \in \mathcal{P}$ is the intersection of the normal plane of a streamline with the designed surface as shown in Figure 38-(*a*). We refer to the arclength parametrization of the normal section as $\boldsymbol{r}^{NS}(\beta)$. κ_b is the curvature of the cutting tool.

The normal section is approximated at $\boldsymbol{u} \in \mathcal{P}$ by

$$\boldsymbol{r}^{NS}(\beta) = \boldsymbol{r}(\boldsymbol{u}) + \left(\left.\frac{d\boldsymbol{r}^{NS}}{d\beta}\right|_{\beta = 0}\right)\beta + \frac{1}{2}\left(\left.\frac{d^2\boldsymbol{r}^{NS}}{d\beta^2}\right|_{\beta = 0}\right)\beta^2 + \dots .$$

This is the parabolic approximation of the normal section. We center the normal section at $\boldsymbol{r}(\boldsymbol{u})$ and define:

$$\Delta\boldsymbol{r}^{NS}(\beta) \equiv \boldsymbol{r}^{NS}(\beta) - \boldsymbol{r}(\boldsymbol{u}) = (\beta)(\boldsymbol{n} \times \boldsymbol{t}) + \left(\frac{1}{2}\kappa_s\beta^2\right)\boldsymbol{n} + \dots$$

where $\boldsymbol{t}$ is the unit tangent vector of a streamline passing through the point and $\boldsymbol{n}$ is the unit surface normal vector at the surface point. κ_s is the normal curvature in the direction of $\boldsymbol{n} \times \boldsymbol{t}$. It is well known in differential geometry that the normal curvature κ_s corresponds to the curvature of the normal section, namely, $\kappa_s = \left\|(\boldsymbol{r}^{NS})''|_{\boldsymbol{u}}\right\|$.

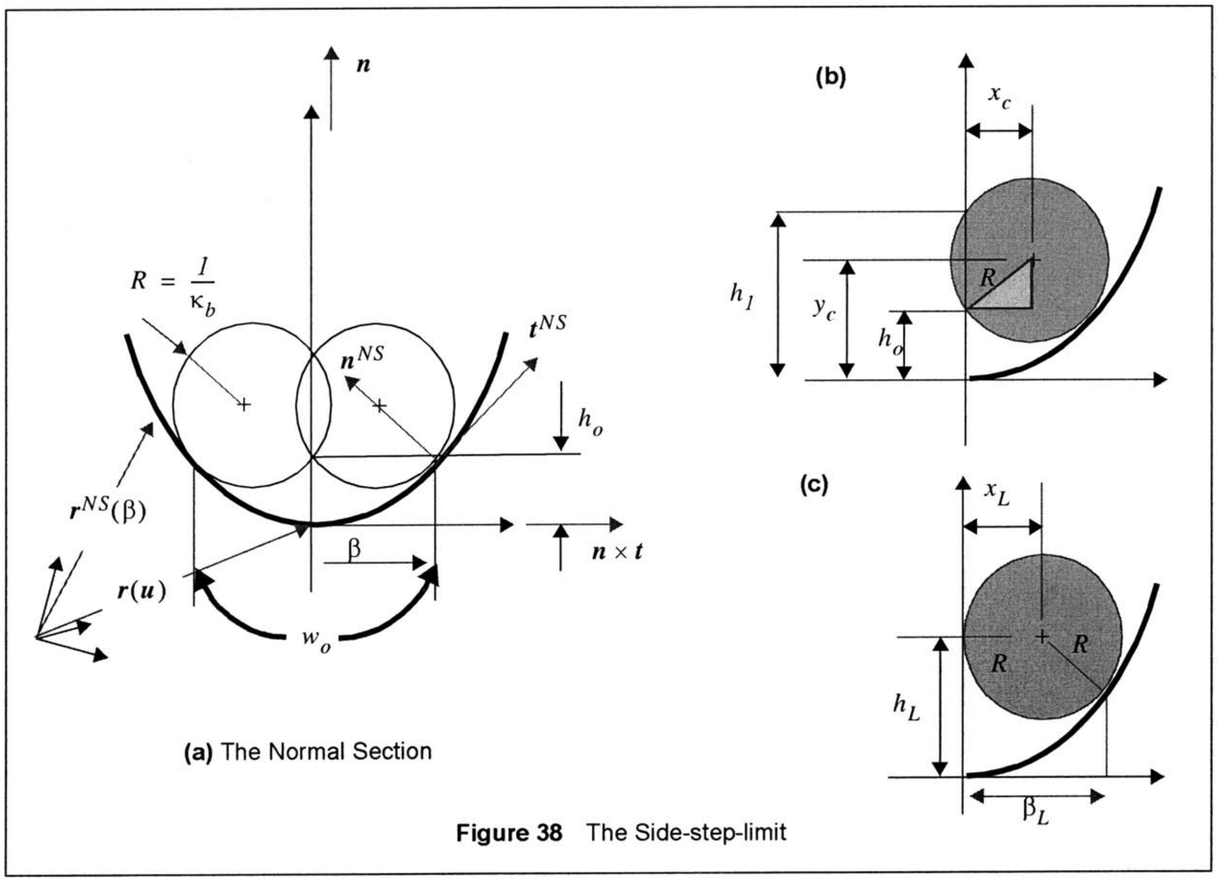


Figure 38 The Side-step-limit

We set a local coordinate system for the expression of the normal section. For brevity, we consider the following alternative expressions the same:

$$\Delta \boldsymbol{r}^{NS}(\beta) = \beta(\boldsymbol{n} \times \boldsymbol{t}) + \left(\frac{1}{2}\kappa_s\beta^2\right)\boldsymbol{n} = x(\boldsymbol{n} \times \boldsymbol{t}) + y\boldsymbol{n} = (x, y) = \left(\beta, \frac{1}{2}\kappa_s\beta^2\right).$$

We define $\boldsymbol{t}^{NS}$ and $\boldsymbol{n}^{NS}$ as the unit vectors of the normal section as shown in Figure 38-(*a*), namely

$$\boldsymbol{t}^{NS} = \frac{(1, \kappa_s\beta)}{\sqrt{1 + (\kappa_s\beta)^2}} \qquad \text{and} \qquad \boldsymbol{n}^{NS} = \frac{(-\kappa_s\beta, 1)}{\sqrt{1 + (\kappa_s\beta)^2}}.$$

Then, the center $\boldsymbol{r}_c$ of the ball is

$$\boldsymbol{r}_c \equiv (x_c, y_c) = (x, y) + \left\{\frac{1}{\kappa_b}\right\} \cdot \boldsymbol{n}^{NS} = \left(\beta, \frac{1}{2}\kappa_s\beta^2\right) + \left\{\frac{1}{\kappa_b}\right\}\frac{(-\kappa_s\beta, 1)}{\sqrt{1 + (\kappa_s\beta)^2}}. \tag{C-3}$$

Applying *Pythagoras' theorem* to the triangle in Figure 38-(*b*),

$$x_c^2 + (y_c - h_o)^2 = R^2 \qquad \text{or}$$

$$\left\{\kappa_b - \frac{\kappa_s}{\sqrt{1 + (\kappa_s\beta)^2}}\right\}^2 \beta^2 + \left\{\frac{1}{\sqrt{1 + (\kappa_s\beta)^2}} + \frac{1}{2}\kappa_b\kappa_s\beta^2 - h_o\kappa_b\right\}^2 = 1.$$

Solving the above equation for h_o results in

$$h_o = y_c - \sqrt{R^2 - x_c^2}. \tag{C-4}$$

The other solution $h_o = y_c + \sqrt{R^2 - x_c^2}$ is discarded since it predicts h_1 in Figure 38-(*b*). Figure 38-(*c*) shows that h_o is bounded by a certain number h_L and the corresponding β is denoted by β_L.

By inserting Equation (C-3) into Equation (C-4), we have

$$h_o = \left(\frac{1}{2}\kappa_s\beta^2 + \frac{1}{\kappa_b\gamma(\beta)}\right) - \left\{\frac{1}{\kappa_b}\right\}\sqrt{1 - \left(\kappa_b - \frac{\kappa_s}{\gamma(\beta)}\right)^2\beta^2}$$

where we set

$$\gamma \equiv \sqrt{1 + (\kappa_s\beta)^2}.$$

Directly using this formula causes computational instability near $\kappa_b = 0$. To avoid such "artificial singularities," we rationalize the equation and get the following equation:

$$h_o = \left[\frac{1}{2}\kappa_s + \frac{\{\kappa_b\gamma - 2\kappa_s\}}{1 + \sqrt{1 - \{\kappa_b^2(1 + (\kappa_s\beta)^2) - 2\kappa_s\kappa_b\gamma\}\beta^2}}\right]\beta^2 \equiv h(\beta). \qquad \text{(C-5)}$$

By expanding the above equation, we have the following *5th* order polynomial equation for β^2:

$$\begin{aligned}
&[\beta^2]^5\ \kappa_b^2 \cdot\ \kappa_s^6\kappa_b^2 + \\
&[\beta^2]^4\ \kappa_b^2 \cdot\ \kappa_s^4\kappa_b^2[9 - 8h_o\kappa_s] + \\
&[\beta^2]^3\ \kappa_b^2 \cdot 8\kappa_s^2\kappa_b^2[\ 3 - 5(h_o\kappa_s) + 3(h_o\kappa_s)^2] - \\
&[\beta^2]^2\ \kappa_b^2 \cdot 8\ [\kappa_b^2\{-2 + 4(h_o\kappa_s) - 7(h_o\kappa_s)^2 + 4(h_o\kappa_s)^3\} + 18\kappa_s^2] + \\
&[\beta^2]\ \ \kappa_b^2 \cdot 16[\kappa_b^2\{\ 2 - 2(h_o\kappa_s) + (h_o\kappa_s)^2\}h_o + 12\kappa_s]h_o + \\
&[1\ \ ]\ \ \kappa_b^2 \cdot 16[\kappa_b^2h_o^2 - 4]h_o^2 \\
&\ = 0
\end{aligned}$$

Monotonicity of $h(\beta)$

Since the *5th* order polynomial can have a complicated root structure, we want to solve β directly from Equation (C-5) for a given cusp-height-limit h_o. To do so, we check the monotonicity of the function $h(\beta)$:

$$\frac{dx_c}{d\beta} = \frac{\kappa_b\gamma^3 - \kappa_s}{\kappa_b\gamma^3} \geq 0, \text{ since } (\gamma \geq 1) \textit{ and } (\kappa_b \geq \kappa_s).$$

$$\frac{dy_c}{d\beta} = (\kappa_s\beta)\left\{\frac{\kappa_b\gamma^3 - \kappa_s}{\kappa_b\gamma^3}\right\} = (\kappa_s\beta)\left(\frac{dx_c}{d\beta}\right)$$

$$\left[\frac{dh}{d\beta}\right] = \left[\frac{(dx_c/d\beta)}{(y_c - h)}\right]I\beta \qquad \text{where } I \equiv \left[1 - \kappa_s h + \frac{1}{2}\kappa_s^2\beta^2\right]$$

(*1*) If $\kappa_s \leq 0$, then $I \equiv [1 - \kappa_s h + \kappa_s^2\beta^2/2] \geq 0$.

(*2*) If $\kappa_s > 0$, then starting from

$h \leq y_c$ (inspecting Figure 38-*b*)

$$-\kappa_s h \geq -\kappa_s y_c$$

$$[1 - \kappa_s h + \kappa_s^2\beta^2/2] \geq [1 - \kappa_s y_c + \kappa_s^2\beta^2/2] \equiv II$$

$$II = 1 + \kappa_s^2\beta^2/2 - \kappa_s y_c = \left\{1 + \frac{1}{2}\kappa_s^2\beta^2\right\} - \frac{\kappa_s}{\kappa_b}\left\{\frac{1}{\sqrt{1 + (\kappa_s\beta)^2}} + \frac{1}{2}\kappa_b\kappa_s\beta^2\right\} \quad = 1 - \frac{\kappa_s}{\kappa_b\gamma} = \frac{\kappa_b\gamma - \kappa_s}{\kappa_b\gamma} \geq 0$$

$$\therefore\ I \geq II \geq 0$$

Gathering both cases, we conclude that

$$\left[\frac{dh}{d\beta}\right] = \left[\frac{(dx_c/d\beta)}{(y_c - h)}\right] I\beta \geq 0 \qquad (\beta \geq 0)$$

which implies that $h(\beta)$ is monotonic.

Cutting off the Side-step-limit for Large Cusp-height-limits

The critical value h_L of $h(\beta)$ occurs at

$$x_c = \frac{1}{\kappa_b}$$

as shown in Figure 38-(*c*). The corresponding β is denoted by β_L, which is the solution of

$$x_c(\beta = \beta_L) \equiv \frac{\beta_L}{\kappa_b}\left(\kappa_b - \frac{\kappa_s}{\sqrt{1 + (\kappa_s\beta_L)^2}}\right) = \frac{1}{\kappa_b} \equiv R\,.$$

Expanding the above equation, we establish the following equation about β_L:

$$F(\beta_L) \equiv (\kappa_b\kappa_s)^2\beta_L^4 - 2\kappa_b\kappa_s^2\beta_L^3 + \kappa_b^2\beta_L^2 - 2\kappa_b\beta_L + 1 = 0\,. \qquad \text{(C-6)}$$

We also observe that $h(\beta)$ in Equation (C-5) starts to have an imaginary value from $\beta = \beta_L$. (Of course, this is not a proof, which we omit here). If we insert $\beta = \beta_L$ into the equation, the square root part will be zero. Thus, the limit value of cusp height is

$$h_L = \left[\frac{1}{2}\kappa_s + \frac{\{\kappa_b\gamma_L - 2\kappa_s\}}{1 + \sqrt{0}}\right]\beta_L^2 = \left[\kappa_b\gamma_L - \frac{3}{2}\kappa_s\right]\beta_L^2$$

where we set

$$\gamma_L \equiv \sqrt{1 + (\kappa_s\beta_L)^2}\,.$$

Elementary calculus is sufficient to prove that

If $\kappa_s \geq 0$, Equation (C-6)—$F(\beta_L) = 0$—has the unique solution in the interval:

$$\frac{1}{\kappa_b} \leq \beta_L < \frac{2}{\kappa_b}\,;$$

if $\kappa_s < 0$, the equation has the unique solution in the interval:

$$0 \le \beta_L \le \frac{1}{\kappa_b}.$$

The Side-step-limit:

Approximately,

$$w_o \approx 2\beta$$

where β is the solution of the Equation (C-5). Of course, this is valid only if $h_o \le h_L$. To be more exact,

$$w_o = 2 \cdot \int_o^{\beta} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx = 2 \cdot \int_o^{\beta} \sqrt{1 + (\kappa_s x)^2} dx = \frac{\beta}{\gamma}\left\{\frac{ln(1+s)^{1/s} - ln(1-s)^{1/s}}{2} + \gamma^2\right\}$$

where we set

$$s \equiv \frac{\kappa_s \beta}{\sqrt{1 + (\kappa_s \beta)^2}}.$$

Note that

$$\lim_{s \to 0} [ln(1+s)^{1/s}] = ln(e) = 1 \text{ and } \lim_{s \to 0} [ln(1-s)^{1/s}] = ln\left(\frac{1}{e}\right) = -1.$$

All the considerations made here lead us to the formula given in the outset of this section.

C.2 Iso-sweep-rate Contour Lines

Minimum Requirement of The Side-step-limit Formula

We can devise other approximate formulas for the side-step-limit. But, we require that the formulas should satisfy the following inequality:

$$\frac{dw_o}{d\underset{\sim}{\eta}} \ge 0 \text{ where } \underset{\sim}{\eta} \in [0, \pi/2]. \tag{C-7}$$

This is because the widest cut is made in the most convex direction.

Iso-sweep-rate Contour Lines in the Tangent Space

As mentioned earlier, $(\vartheta, \underset{\sim}{\eta})$ is the cylindrical coordinates of the $(\dot{\underset{\sim}{u}}, \dot{\underset{\sim}{v}})$-space, namely $\dot{\underset{\sim}{u}} = \vartheta cos\underset{\sim}{\eta}$ and $\dot{\underset{\sim}{v}} = \vartheta sin\underset{\sim}{\eta}$. In terms of the cylindrical coordinate system, iso-sweeprate contour lines can be expressed by the following equation:

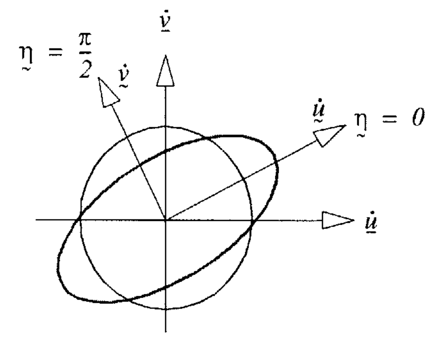
$$\vartheta(\underset{\sim}{\eta}) = \Gamma_o / w_o(\underset{\sim}{\eta}),$$

where Γ_o is the sweep rate. Then, resorting to Equation (C-7), we have

$$\vartheta(\underset{\sim}{\eta}=0) \geq \vartheta(\underset{\sim}{\eta}=\pi/2).$$

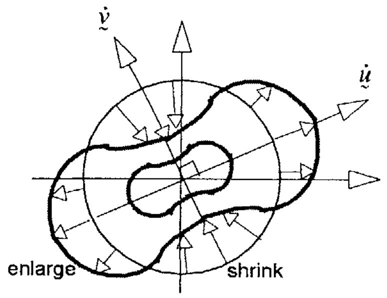
Geometrically, it means that the iso-sweep-rate contour lines are enlarged to the $\underset{\sim}{\dot{u}}$-axis and shrinks to the $\underset{\sim}{\dot{v}}$-axis. This is shown in Figure 39-(*a*). The contour can even have inflection points in the $(\underset{\sim}{\dot{u}}, \underset{\sim}{\dot{v}})$-space if the variation of the side-step-limit is large as shown in Figure 39-(*b*).

Conventional tool path generation schemes do not take account of the kinematic aspect of machining, which is equivalent to the assumption that the velocity limit is not a polygon but a circle in $(\underset{\sim}{\dot{u}}, \underset{\sim}{\dot{v}})$-space, namely $\vartheta \leq \vartheta_o = const$ represents the velocity polygon. For such an isotropic case, the most convex direction ($\underset{\sim}{\dot{v}}$-axis) is the direction of maximum sweep rate as shown in Figure 39-(*c*).

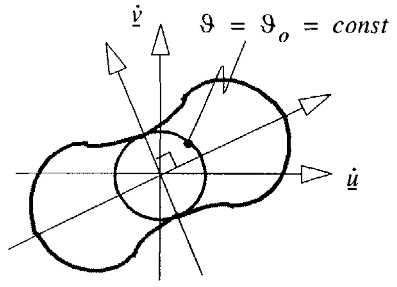


$\dot{v}$
$\underset{\sim}{\eta} = \frac{\pi}{2}$
$\underset{\sim}{\dot{v}}$
$\underset{\sim}{\dot{u}}$
$\underset{\sim}{\eta} = 0$
$\dot{u}$

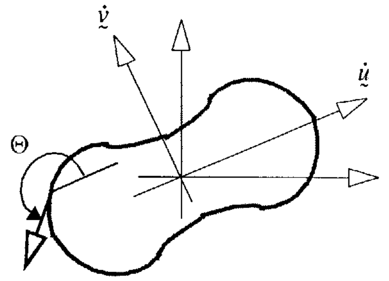
a) A Convex Iso-sweep Rate Contour Line



b) A Non-convex Iso-sweep Rate Contour Line



c) Constant Velocity Limit



c) The Direction Angle of a Iso-sweep-rate Contour Line

Figure 39 Iso-Sweep Rate Contour Lines

Tangent Angles and Inflection Points of Iso-sweeprate Contour Lines

The convexity of iso-sweep-rate contour lines is an important piece of information to find the greedy direction. To check the convexity, we define the tangent angle of the contour lines:

$$\Theta(\underset{\sim}{\eta}) \equiv atan_2\left(\frac{d\underset{\sim}{v}}{d\underset{\sim}{\eta}}, \frac{d\underset{\sim}{u}}{d\underset{\sim}{\eta}}\right) , \qquad \left(\frac{\pi}{2} \leq \Theta \leq \frac{5\pi}{2}\right)$$

which is the angle of the tangent vector $d\underset{\sim}{u}/d\underset{\sim}{\eta}$ of an iso-sweep-rate contour line from the $\underset{\sim}{u}$-axis. This is shown in Figure 39-(*d*). An iso-sweeprate contour line $(\underset{\sim}{u}(\underset{\sim}{\eta}), \underset{\sim}{v}(\underset{\sim}{\eta}))$ in $(\underset{\sim}{u}, \underset{\sim}{v})$-space for a given sweep rate Γ_o is represented by

$$\underset{\sim}{u} = \vartheta cos\underset{\sim}{\eta} = \left\{\frac{\Gamma_o}{w_o(\underset{\sim}{\eta})}\right\} \cdot cos\underset{\sim}{\eta} \qquad \text{and} \qquad \underset{\sim}{v} = \vartheta sin\underset{\sim}{\eta} = \left\{\frac{\Gamma_o}{w_o(\underset{\sim}{\eta})}\right\} \cdot sin\underset{\sim}{\eta} .$$

Then, we have

$$\Theta(\underset{\sim}{\eta}) = atan_2[w_o \cdot cos\underset{\sim}{\eta} - w_o' \cdot sin\underset{\sim}{\eta}, -(w_o \cdot sin\underset{\sim}{\eta} + w_o' \cdot cos\underset{\sim}{\eta})] , \ (\pi/2 < \Theta < 5\pi/2)$$

where $w'_o = 2(\kappa_1 - \kappa_2) cos 2\underset{\sim}{\eta} \cdot (dw_o/d\kappa_s)$

$$\frac{dw_o}{d\kappa_s} = \begin{cases} \beta\left(\dfrac{\gamma(2-4h_o\kappa_s + 2h_o\kappa_b\gamma^3 - \beta^2\kappa_b\kappa_s\gamma^3)}{(1 - 2h_o\kappa_s + \gamma^2)(\kappa_b\gamma^3 - \kappa_s)} + \beta w_3(\kappa_s\beta)\right) & if \ h_o \leq h_L \\ \beta_L^2\left(\dfrac{2\kappa_s\beta_L\gamma_L(\kappa_b\beta_L - 2)}{(\kappa_b\beta_L)^2(2\kappa_b\beta_L - 3) + \kappa_b\beta_L - 1} + w_3(\kappa_s\beta)\right) & otherwise \end{cases} , \ w_3(x) \equiv \begin{cases} \dfrac{x - arc\,sinh(x)}{x^2} & if \ |x| \neq 0 \\ 0 & otherwise \end{cases} .$$

Note that all the symbols are as defined in Equation (C-1). This formula is used to find the edge binding points and the inflection angles. For the edge binding points, we used the bisection search method without differentiation. For the inflection angles, we took the golden ratio search method, maximizing $\Theta(\underset{\sim}{\eta})$. Depending on the numerical methods we choose, the following derivative can be invoked:

$$\frac{d\Theta}{d\underset{\sim}{\eta}} = \frac{\left(\frac{d\underset{\sim}{u}}{d\underset{\sim}{\eta}}\right) \cdot \frac{d}{d\underset{\sim}{\eta}}\left(\frac{d\underset{\sim}{v}}{d\underset{\sim}{\eta}}\right) - \left(\frac{d\underset{\sim}{v}}{d\underset{\sim}{\eta}}\right) \cdot \frac{d}{d\underset{\sim}{\eta}}\left(\frac{d\underset{\sim}{u}}{d\underset{\sim}{\eta}}\right)}{\left(\frac{d\underset{\sim}{u}}{d\underset{\sim}{\eta}}\right)^2 + \left(\frac{d\underset{\sim}{v}}{d\underset{\sim}{\eta}}\right)^2} .$$

Note that

$$\frac{d\Theta}{d\underset{\sim}{\eta}} > 0$$

at the points where an iso-sweep-rate contour line is convex.

C.3 Edge Binding Criteria

We maximize the function $\Theta(\underset{\sim}{\eta})$ and eventually find the inflection point in the first quadrant. Let the direction angle of the inflection point be η_{inf} and corresponding slope be $\Theta_{inf} \equiv \Theta(\eta_{inf})$. For other quadrants, the inflection points are derived by symmetry.

Let an *ith* edge of the velocity polygon be represented by 2 vertices (being oriented anti-clock wise):

$$\dot{\underset{\sim}{u}}_i = [\dot{\underset{\sim}{u}}_i \;\; \dot{\underset{\sim}{v}}_i]^T \quad \text{and} \quad \dot{\underset{\sim}{u}}_{i+1} = [\dot{\underset{\sim}{u}}_{i+1} \;\; \dot{\underset{\sim}{v}}_{i+1}]^T .$$

We refer to the principal direction angle of the vertices as $\underset{\sim}{\eta}_i$, namely $\underset{\sim}{\eta}_i \equiv atan_2(\dot{\underset{\sim}{v}}_i, \dot{\underset{\sim}{u}}_i)$. Let

$$\Delta\dot{\underset{\sim}{u}}_i \equiv \dot{\underset{\sim}{u}}_{i+1} - \dot{\underset{\sim}{u}}_i .$$

The tangent angle of each segment is represented by

$$\Theta_i \equiv atan_2(\Delta\dot{\underset{\sim}{v}}_i, \Delta\dot{\underset{\sim}{u}}_i), \qquad (\pi/2 < \Theta_i < 5\pi/2).$$

If an edge has a binding point, the following criterion is satisfied:

$$2\pi - \Theta_{inf} < \Theta_i < \Theta_{inf} \qquad or \qquad 3\pi - \Theta_{inf} < \Theta_i < \pi + \Theta_{inf} .$$

In addition,

$$[\eta_i, \eta_{i+1}] \cap U \neq \varnothing$$

where

$$U = [\eta_{inf}, \pi/2] \qquad if\ \Delta\dot{\underset{\sim}{u}}_i < 0\ and\ \Delta\dot{\underset{\sim}{v}}_i < 0$$

$$U = [\pi/2, \pi - \eta_{inf}] \qquad if \quad <,>$$

$$U = [\pi + \eta_{inf}, 3\pi/2] \qquad if \quad >,>$$

$$U = [3\pi/2, 2\pi - \eta_{inf}] \qquad if \quad >,<.$$

Of course, if $\eta_i > \eta_{i+1}$, then the interval $[\eta_i, \eta_{i+1}]$ is interpreted considering the periodicity, namely $[\eta_i, \eta_{i+1}] \equiv [\eta_i, 2\pi) \cup [0, \eta_{i+1}]$ if $\eta_i > \eta_{i+1}$. If the edge passes the above criteria, we solve the following equation

$$\Theta(\eta) - \Theta_i = 0 \qquad \text{(C-8)}$$

in the corresponding interval U.

After finding the solution η_* of Equation (C-8), we check whether the point is on the segment but not on its extension. The ray in the direction of η_* is represented by

$$\begin{Bmatrix} \dot{\underset{\sim}{u}} \\ \dot{\underset{\sim}{v}} \end{Bmatrix} = \vartheta \begin{Bmatrix} cos\eta_* \\ sin\eta_* \end{Bmatrix}, \qquad \exists(\vartheta > 0).$$

The equation of the edge is expressed by

$$\begin{Bmatrix} \dot{\underset{\sim}{u}} \\ \dot{\underset{\sim}{v}} \end{Bmatrix} = t \begin{Bmatrix} \Delta\dot{\underset{\sim}{u}}_i \\ \Delta\dot{\underset{\sim}{v}}_i \end{Bmatrix} + \begin{Bmatrix} \dot{\underset{\sim}{u}}_i \\ \dot{\underset{\sim}{v}}_i \end{Bmatrix} \qquad \exists t \in (0, 1).$$

Equating both, we solve the following equation for parameters ϑ and t:

$$\begin{bmatrix} cos\eta_* & -\Delta\dot{\underset{\sim}{u}}_i \\ sin\eta_* & -\Delta\dot{\underset{\sim}{v}}_i \end{bmatrix} \begin{Bmatrix} \vartheta \\ t \end{Bmatrix} = \begin{Bmatrix} x_i \\ y_i \end{Bmatrix}.$$

In order that the point can be a candidate of the binding point, the following inequalities must be satisfied:

$$\vartheta > 0 \qquad \text{and} \qquad 0 < t < 1.$$

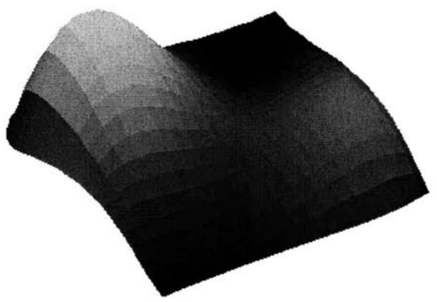
If the point passes all the above criteria, we include the point $\vartheta[cos\eta_* \; sin\eta_*]^T \in (\dot{\underset{\sim}{u}}, \dot{\underset{\sim}{v}})\text{-}space$ in the list of candidate (edge) binding points.

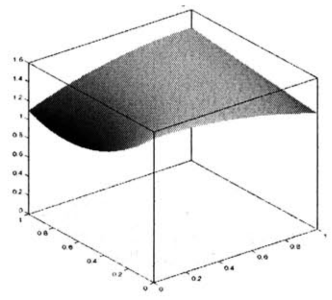
APPENDIX D EXAMPLES OF SURFACES

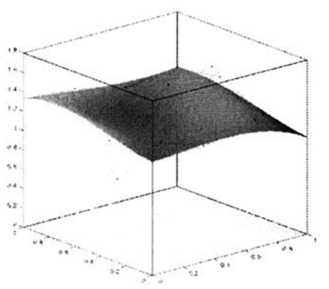
Here, we provide the control points R_{ij} for the surfaces that were taken as examples in Chapter 4. We also provide the control points V_{ij} for the generating function $V(u, v)$ that is fitted to the greedy directions.

D.1 Surface Example 1:

```
R00=[   -0.1200    -0.1200     0.2500](m)
R01=[   -0.1200    -0.0600     0.2500](m)
R02=[   -0.1200     0.0000     0.3611](m)
R03=[   -0.1800     0.0600     0.2500](m)
R04=[   -0.1200     0.1200     0.2500](m)
R10=[   -0.0600    -0.1200     0.2500](m)
R11=[   -0.0600    -0.0600     0.2500](m)
R12=[   -0.0600     0.0000     0.2500](m)
R13=[   -0.0600     0.0600     1.0833](m)
R14=[   -0.0600     0.1200     0.3056](m)
R20=[    0.0000    -0.1200     0.2500](m)
R21=[    0.0000    -0.0600     0.1944](m)
R22=[    0.0000     0.0000     0.1944](m)
R23=[    0.0000     0.0600     0.0278](m)
R24=[    0.0000     0.1200     0.2500](m)
R30=[    0.0600    -0.1800     0.3056](m)
R31=[    0.0600    -0.0600     0.5278](m)
R32=[    0.0600     0.0000     0.3056](m)
R33=[    0.0600     0.0600     0.1944](m)
R34=[    0.0600     0.1200     0.2500](m)
R40=[    0.1200    -0.1200     0.2500](m)
R41=[    0.1200    -0.0600     0.3056](m)
R42=[    0.1200     0.0000     0.3056](m)
R43=[    0.1200     0.0600     0.2500](m)
R44=[    0.1200     0.1200     0.2500](m)
```



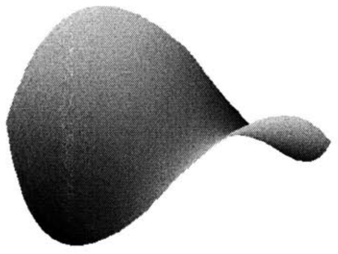


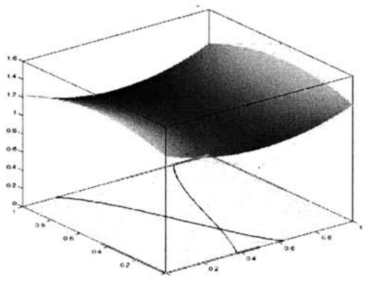


```
---------------------------------------------------------------------------
    Bi-quadratic [Vij] =
---------------------------------------------------------------------------
   0.00010000000000  -0.75318950554665  -0.43351053657144
  -0.00873161266664   0.28345116377007  -0.10811633357740
  -0.30453695395901  -0.21252059798077  -0.12147624308731

---------------------------------------------------------------------------
   Bi-cubic        [Vij] =
---------------------------------------------------------------------------
   0.00010000000000   0.04787356388696   0.35433969464262   0.15356826121058
   0.01738605276363  -0.10564499097978  -0.65479627927392   0.10882303739045
   0.14714797217375   0.53228756990644   0.18364673085155  -0.01008529754797
  -0.17350569539288  -0.10408369864959   0.11766457389464   0.05395849964030
===========================================================================
```

D.2 Surface Example 2:

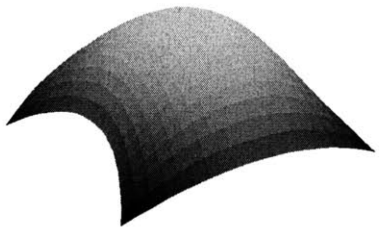
```
-------------------------------------------------------------
R00=[   -9.0000     -9.0000     20.0000]*(0.012500)(m)
R01=[  -12.0000     -3.0000     12.0000]*(0.012500)(m)
R02=[  -12.0000      3.0000     12.0000]*(0.012500)(m)
R03=[   -9.0000      9.0000     20.0000]*(0.012500)(m)
R10=[   -3.0000    -12.0000     28.0000]*(0.012500)(m)
R11=[   -3.0000     -3.0000     20.0000]*(0.012500)(m)
R12=[   -3.0000      3.0000     20.0000]*(0.012500)(m)
R13=[   -3.0000     12.0000     28.0000]*(0.012500)(m)
R20=[    3.0000    -12.0000     28.0000]*(0.012500)(m)
R21=[    3.0000     -3.0000     20.0000]*(0.012500)(m)
R22=[    3.0000      3.0000     20.0000]*(0.012500)(m)
R23=[    3.0000     12.0000     28.0000]*(0.012500)(m)
R30=[    9.0000     -9.0000     20.0000]*(0.012500)(m)
R31=[   12.0000     -3.0000     12.0000]*(0.012500)(m)
R32=[   12.0000      3.0000     12.0000]*(0.012500)(m)
R33=[    9.0000      9.0000     20.0000]*(0.012500)(m)
-------------------------------------------------------------
    [Vij] =
-------------------------------------------------------------
                   0   0.30058703471589  -0.05681922365427
  -0.41158334537802  -0.62397478156384  -0.50407602853921
   0.00000000000000   0.30058703471590  -0.05681922365427

=============================================================
```

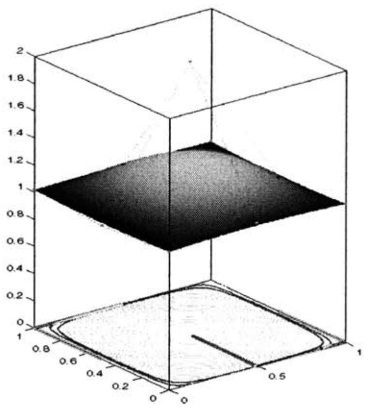




D.3 Surface Example 3:

```
----------------------------------------------------------------
R00=[  -12.0000      -9.4737     20.0000]*(0.012500)(m)
R01=[   -8.0000      -3.1579     25.3333]*(0.012500)(m)
R02=[   -8.0000       3.1579     25.3333]*(0.012500)(m)
R03=[  -12.0000       9.4737     20.0000]*(0.012500)(m)
R10=[   -4.0000     -12.6316     25.3333]*(0.012500)(m)
R11=[   -4.0000      -3.1579     36.0000]*(0.012500)(m)
R12=[   -4.0000       3.1579     36.0000]*(0.012500)(m)
R13=[   -4.0000      12.6316     25.3333]*(0.012500)(m)
R20=[    4.0000     -12.6316     25.3333]*(0.012500)(m)
R21=[    4.0000      -3.1579     36.0000]*(0.012500)(m)
R22=[    4.0000       3.1579     36.0000]*(0.012500)(m)
R23=[    4.0000      12.6316     25.3333]*(0.012500)(m)
R30=[   12.0000      -9.4737     20.0000]*(0.012500)(m)
R31=[    8.0000      -3.1579     25.3333]*(0.012500)(m)
R32=[    8.0000       3.1579     25.3333]*(0.012500)(m)
R33=[   12.0000       9.4737     20.0000]*(0.012500)(m)
----------------------------------------------------------------
    [Vij] =
----------------------------------------------------------------
   0.00010000000000  -0.03398149913045   0.00066044448484
   0.03571439157901   0.99808789770223   0.01531906506090
  -0.00000000000000  -0.03398149913045   0.00066044448484

================================================================
```





Appendix E

Cutting Time

PER A STREAMLINE IN A LINEARLY CONVERGING FIELD

In this appendix, we derive the formula (Equation 38, Section 5.2) for the cutting time per a streamline:

Equation (38)

$$\tau_c(N) \equiv \tau_e(N) + \tau_{ne}(N) \approx \tau_1 \cdot [(2-\beta) + \{1-(1-\beta/N)^N\}/N] + \tau_o \cdot N \cdot \frac{\{1-(1-\beta/N)^N\}}{\beta}$$

$$= \frac{1}{\beta} \cdot \left\{1-\left(1-\frac{\beta}{N}\right)^N\right\}\left\{\tau_o N + \frac{\beta\tau_1}{N}\right\} + \tau_1 \cdot (2-\beta)$$

where $\tau_c(N)$ is the cutting time per a streamline for $(N-1)$ interruptions (or the number of incompatible points),

$$\beta \equiv \left(\frac{L}{W}\right)\gamma, \qquad \tau_1 \equiv \frac{1}{2} \cdot \frac{L}{V}, \qquad \gamma \equiv \frac{d}{ds}(w_o - w) = const \geq 0,$$

V is the speed along the streamline, L is the length of the streamline, W is the side-step-limit and τ_o is the non-effective penalty (NEP) term.

E.1 Non-effective Cutting Time

Because we consider only the converging field, the overlap degree will vanish on the inlet of each new continuum to minimize the cutting time. The number n_i of tool paths in the i^{th} continuum is approximated by

$$n_i \approx \frac{Y_i}{w_o^i}$$

where Y_i is the length of the inlet of the i^{th} continuum and w_o^i is the (averaged) side-step-limit along the inlet as shown in Figure 40. We assumed that the direction field is linearly converging, *i.e.*

$$\gamma \equiv \frac{d}{ds}(w_o - w) = const.$$

The side step $w^i|_{outlet}$ on the i^{th} outlet is approximated by

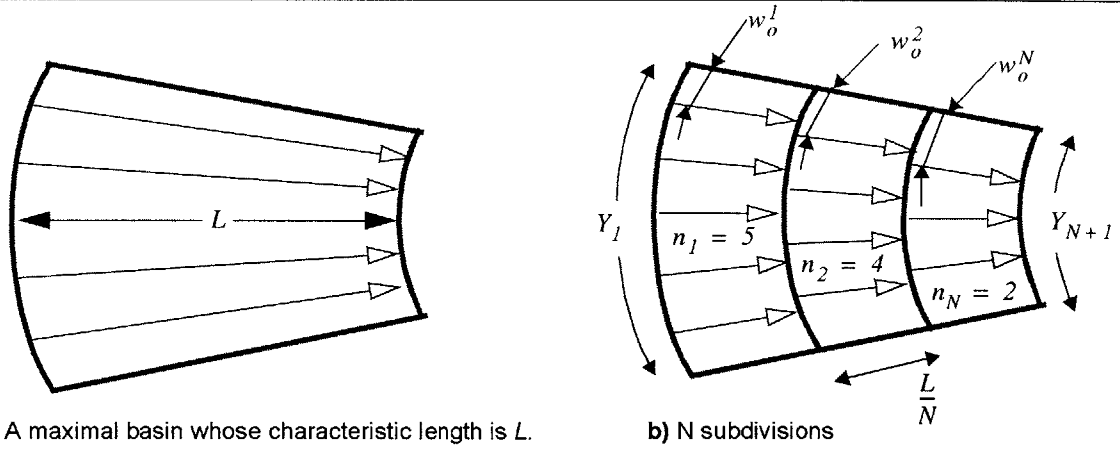
$$w^i|_{outlet} \approx w_o^{i+1} - \gamma \cdot \left(\frac{L}{N}\right) \qquad \text{(E-1)}$$

where N is the number of path continua and L is the averaged length of streamlines in the maximal basin under consideration. Note that

$$w_o^{i+1} = w_o^i\big|_{outlet}$$

because an outlet of a continuum is the inlet of its next continuum. With the same token, the length Y_{i+1} of the $(i+1)^{th}$ inlet is identical to the length of the i^{th} outlet.

The number of tool paths in the ith continuum must be conserved, *i.e.*



a) A maximal basin whose characteristic length is *L*. **b)** N subdivisions

Figure 40 Subdividing a Maximal Basin into Path Continua (*c.f.* Figure 22)

$$\left[n_i = \frac{Y_i}{w_o^i}\right] = \left[\frac{Y_{i+1}}{w^i|_{outlet}} = \left(\frac{Y_{i+1}}{w_o^{i+1}}\right)\cdot\left(\frac{w_o^{i+1}}{w^i|_{outlet}}\right) = n_{i+1}\cdot\left(\frac{w_o^{i+1}}{w^i|_{outlet}}\right)\right]. \tag{E-2}$$

From Equation (E-1 and E-2), we have

$$\left(\frac{n_{i+1}}{n_i}\right) \approx 1 - \left(\frac{\gamma}{w_o^{i+1}}\right)\cdot\left(\frac{L}{N}\right).$$

We consider the side-step-limit a constant, namely $w_o^i = W = const$. Consequently, the number of tool paths in each region is a geometric series:

$$n_i = n_1 \cdot \left(1 - \frac{\beta}{N}\right)^{i-1} \qquad \text{where } \beta \equiv \left(\frac{L}{W}\right)\gamma.$$

We now derive the non-effective cutting time T_{ne} of the maximal basin as follows:

$$T_{ne} \approx \tau_o \cdot (Number\ of Toolpaths) = \tau_o \cdot \left(\sum n_i\right) = \tau_o \cdot n_1 \cdot \left(\frac{N}{\beta}\right)\left(1 - \left(1 - \frac{\beta}{N}\right)^N\right).$$

Note that the non-effective cutting time per a streamline is T_{ne}/n_1. Therefore,

$$\tau_{ne}(N) = (\tau_o N)\cdot\left(\frac{1}{\beta}\right)\cdot\left(1 - \left(1 - \frac{\beta}{N}\right)^N\right) \tag{E-3}$$

E.2 Effective Cutting Time

We divide the effective cutting time T_e of the maximal basin into two terms as follows:

$$T_e = \int \frac{dA}{\vartheta \cdot w} = \int \frac{dA}{\vartheta \cdot w_o} + \int \frac{w_o - w}{\vartheta \cdot w_o} \cdot \frac{dA}{w}.$$

The first term is a constant for a given direction field. In this appendix, we refer to the first term as T_L. We reverse the approximation in Equation (20) for the second term, *i.e.*

$$T_e = T_L + \sum_i \int_{C_i} \frac{w_o - w}{\vartheta \cdot w_o} \cdot ds.$$

In the outset, we assumed that $w_o(s) - w(s) \approx \gamma \cdot s$, $w_o \approx W$ and $\vartheta \approx V$. Then,

$$T_e = T_L + \left(\sum_{i=1}^{N} n_i\right) \cdot \frac{\gamma}{2VW}\left(\frac{L}{N}\right)^2 = T_L + n_1 \cdot \left(\frac{N}{\beta}\right) \cdot \left(1 - \left(1 - \frac{\beta}{N}\right)^N\right) \cdot \left(\frac{\beta L}{2V}\frac{1}{N^2}\right).$$

With a similar manner, we are lead to

$$\tau_L \equiv \frac{T_L}{n_1} \approx \tau_1(2 - \beta) \qquad \text{where} \qquad \tau_1 \equiv \frac{1}{2} \cdot \frac{L}{V}.$$

Collecting both terms, we have derived the effective cutting time per a streamline in a maximal basin:

$$\tau_e(N) \equiv \frac{T_e}{n_1} = \tau_L + (\tau_e - \tau_L) = \tau_1 \cdot (2 - \beta) + \left(\frac{1}{\beta}\right) \cdot \left(1 - \left(1 - \frac{\beta}{N}\right)^N\right) \cdot \left(\frac{\beta \tau_1}{N}\right) \quad . \tag{E-4}$$

E.3 Total Cutting Time

Collecting terms in Equation (E-3 and E-4), we now have derived the cutting time per a streamline in a maximal basin:

$$\tau_c(N) \equiv \tau_e(N) + \tau_{ne}(N) = \frac{1}{\beta} \cdot \left\{1 - \left(1 - \frac{\beta}{N}\right)^N\right\}\left\{\tau_o N + \frac{\beta \tau_1}{N}\right\} + \tau_1 \cdot (2 - \beta)$$

as shown in Equation (38).

REFERENCES

SURFACES/CAD/CAM/CAPP

[1] Piegel, L. and Tiller, W., *Implicit and Parametric Forms*, The NURBS book, Springer, pp. 1-5, 1995.

[2] Hsiung, C. C., Chapter 3, Local theory of surfaces, Section 1, Parametrization, *A first course in differential geometry*, International Press, pp. 151-167, 1997.

[3] Sarma, R. and Dutta, D., An integrated system for NC machining of multi-patch surfaces, *Computer-Aided Design*, Vol. 29, No. 11, pp. 741-749, 1997.

[4] Kagan, P. and Fischer, A., Integrated mechanically based CAE system using B-Spline finite elements, *Computer-Aided Design*, Vol 32, pp. 539-552, 2000.

[5] Guillet, S. and Léon, J.C., Parametrically deformed free-form surfaces as part of a variational model, *Computer-aided Design*, Vol. 30, pp. 621-630, 1998.

[6] Ng, W.M.M. and Tan, S.T., Incremental tessellation of trimmed parametric surfaces, *Computer-Aided Design*, Vol. 32, pp. 279-294, 2000.

[7] Shimada, K. and Gossard, D. C., Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis, *Computer Aided Geometric Design*, Vol. 15, No. 3, pp. 199-222, 1998.

[8] Gu, P. and Norrie, D.H., *Intelligent Manufacturing Planning*, Chapman & Hall, 1995.

[9] Chou, J. J., *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces*, Ph. D. Thesis, Dept. of Computer Science, University of Utah, pp. 36, pp. 8, 1989.

[10] Mortensen, F. L., *Constant scallop height tool paths for sculptured surfaces*, M. S. Thesis, Brigham Young University, 1988.

MACHINE TOOLS

[11] Boer, C., Molinari-Tosatti, L. and Smith K. (Eds.), *Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements*, Springer, 1999.

[12] Wavering, A. J., Parallel Kinematic Machine Research at NIST: Past, Present, and Future, Boer, C., Molinari-Tosatti, L. and Smith K. (Eds.), *Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements*, Springer, pp. 26, 1999.

[13] Chou, J. J., *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces*, Ph. D. Thesis, Dept. of Computer Science, University of Utah, pp. 7, 1989.

[14] Kim, B. H. and Chu, C. N., Effect of cutter mark on surface roughness and scallop height in sculptured surface machining, *Computer-Aided Design*, Vol. 26, No. 3, pp. 179-188, March 1994.

Path Schemes

[15] Marciniak, K., *Geometric modeling for numerically controlled machining*, Oxford University Press, 1991.

[16] Choi, B. K. and Jerard, B. J., *Sculptured Surface Machining: Theory and Application*, Kluwer Academic Publishers, 1998.

[17] Huang, Y. and Oliver, J.H., Non-constant parameter NC tool path generation of sculptured surfaces, *Computers in Engineering*, Vol. 1, ASME, 1992.

[18] Suresh, K. and Yang, D. C. H., Constant scallop-height machining of free-form surfaces, *ASME Journal of Engineering for Industry*, Vol. 116, pp.253-259, May 1994.

[19] Choi, B. K., Park, J. W. and Jun, C. S., Cutter-location optimization in 5-axis surface machining, *Computer-Aided Design*, Vol. 25, No. 6, pp. 377-385, 1993.

[20] Dragomatz, D. and Mann, S., A classified bibliography of literature on NC milling path generation, *Computer-Aided Design*, Vol. 29, No. 3, pp. 239-247, 1997.

[21] Marshall, S. and Griffiths, J. G., A survey of cutter construction techniques for milling machines, *International Journal of Production Research*, Vol. 32, No. 12, pp. 2861-2877, 1994.

[22] Lin, R.S. and Koren, Y., Efficient tool-path planning free-form surfaces, *ASME Journal of Engineering for Industry*, Vol. 118, pp.20-28, Feb 1996.

[23] Suh, Y. S. and Lee, K., NC milling tool path generation for arbitrary pockets defined by sculptured surfaces, *Computer-Aided Design*, Vol. 22, No. 5, 1990.

[24] Han, Z. and Yang, D. C. H., Isophote based machining for feature intensive surfaces, *Proceedings of the ASME*, MED-Vol. 8, pp.483-495, 1998.

[25] Stori, J. A. and Wright, P. K., A constant engagement offset for 2-1/2D tool path generation, *Proceedings of the ASME*, MED-Vol. 8, pp.475-481, 1998.

[26] Kim, B. H. and Choi, B. K., Guide surface based tool path generation in 3-axis milling; an extension of the guide plane method, *Computer-Aided Design*, Vol. 32, pp. 191-199, 2000.

[27] Elber, G. and Cohen, G., Toolpath generation for free-form surface models, *Computer-Aided Design*, Vol. 26, No. 6, pp. 490-496, 1994.

[28] Lo, C-C., Efficient cutter-path planning for five-axis surface machining with a flat-end cutter, *Computer-Aided Design*, Vol. 31, pp. 557-566, 1999.

[29] Sarma, R. and Dutta, D., The geometry and generation of NC tool paths, *Transactions of the ASME, Journal of mechanical design*, Vol. 119, pp. 253-258, June 1997.

[30] Elber, G., Freeform surface region optimization for 3-axis and 5-axis milling, *Computer-Aided Design*, Vol. 27, No. 6, pp. 465-470, 1995.

[31] Lo, C. C., Two-stage cutter-path scheduling for ball-end milling of concave and wall-bounded surfaces, *Computer-Aided Design*, Vol. 32, pp. 597-603, 2000.

[32] Mullins, S. H., Jensen, C. G. and Anderson, D. C., Scallop elimination based on precise 5-axis tool placement, orientation, and stepover calculations, *Advances in Design Automation ASME* 65-2, pp.535-544, 1993.

[33] Warketin A., Ismail, F. and Bedi, S., Five-axis milling of spherical surfaces, *Int. J. Mach.Tools Manufact*, Vol. 36, No. 2, pp. 229-243, 1996.

[34] Warketin A., Ismail, F. and Bedi, S., Intersection approach to multi-point machining of sculptured surfaces, *Computer Aided Geometric Design* 15, pp. 567-584, 1998.

[35] Warketin A., Ismail, F. and Bedi, S., Multi-point tool positioning strategy for 5-axis machining of sculptured surfaces, *Computer Aided Geometric Design* 17, pp. 83-100, 2000.

[36] Jensen, C. G. and Anderson, D. C., Accurate tool placement and orientation for finished surface machining, *Journal of design and manufacture* 3, pp. 251-261, 1993.

[37] Rao, N., Bedi, S. and Buchal, R., Implementation of principal-axis method for machining of complex surfaces, *International Journal of Advanced Manufacturing Technology* 11, pp. 249-257, 1996.

[38] Kruth, J. and Klewais, P., Optimization and dynamic adaptation of the cutter inclination during five-axis mill-

ing of sculptured surfaces, *Annals of CIRP* 43, pp.443-448, 1994.

[39] Choi, B. K., Chung, Y. C., Park, J. W. and Kim, D. H., Unified CAM-system architecture for die and mold manufacturing, *Computer-Aided Design*, Vol. 26, No. 3, pp. 235-243, 1994.

[40] Lee, Y-S and Chang, T-C, CASCAM—An automated system for sculptured surface cavity machining, *Computers in Industry*, Vol. 16, pp. 321-342, 1991.

[41] Kuragano, T., FRESDAM system for design of aesthetically pleasing free-form objects and generation of collision-free tool paths, *Computer-Aided Design*, Vol. 24, No. 11, pp. 573-581, 1994.

[42] Held, M. and Andor, L., Pocket machining based on contour-parallel tool paths generated by means of proximity maps, *Computer-Aided Design*, Vol. 26, No. 3, pp. 198, 1994.

GOUGING/INTERFERENCE/ACCESSIBILITY/SIDE-STEP-LIMIT

[43] Choi, B. K. and Jun, C. S., Ball-end cutter interference avoidance in NC machining of sculptured surfaces, *Computer-Aided Design*, Vol. 21, No. 6, pp. 371-378, 1989.

[44] Lee, Y. S., Admissible tool orientation control of gouging avoidance for 5-axis complex surface machining, *Computer-Aided Design*, Vol. 29, No. 7, pp. 507-521, 1997.

[45] Lee, Y. S., Chang, T. C., 2-Phase approach to global tool interference avoidance in 5-axis machining, *Computer-Aided Design*, Vol. 27, No. 10, pp. 715-729, 1995.

[46] Zhu, C., Avoiding interference in manufacturing a free-formed surface with a cylindrical end milling cutter, *Computers in Industry*, Vol. 14, pp. 367-371, 1990.

[47] Elber, G., Accessibility in 5-axis milling environment, *Computer-Aided Design*, 26(11):796-802, 1994.

[48] Tang, K., Cheng, C. C. and Dayan, Y., Offsetting surface boundaries and 3-axis gouge-free surface machining, *Computer-Aided Design*, Vol. 27, No. 12, pp. 915-927, 1995.

[49] Hwang, J. S., Interference-free tool-path generation in the NC machining of parametric compound surfaces, *Computer-Aided Design*, Vol. 24, No. 12, pp. 667-676, 1992.

[50] Morishige, K., Takeuchi, Y. and Kase, K., Tool path generation using C-space for 5-axis control machining, Transactions of the ASME, *Journal of Manufacturing Science and Engineering*, Vol. 121, February, 1999.

[51] Balasubramaniam, M., Laxmiprasad, P., Sarma, S. and Shaikh, Z., Generating 5-axis NC roughing paths directly from a tessellated representation, *Computer-Aided Design*, Vol. 32, pp. 261-277, 2000.

[52] Vickers, G. W. and Quan, K. W., Ball-Mills Versus End-Mills for Curved Surface Machining, *ASME Journal of Engineering for Industry*, Vol. 111, pp. 22-26, Feb 1989.

[53] Marciniak, K., Influence of surface shape on admissible tool positions in 5-axis face milling, *Computer-Aided Design*, Vol. 19, No. 5, pp. 233-236, June 1987.

[54] Filip, D., Magedson, R. and Markot, R., Surface algorithms using bounds on derivatives, *Computer-Aided Geometric Design*, Vol. 3, pp. 295-311, 1986.

FEED RATE ADJUSTMENT

[55] Yazar, Z., Koch, K. F., Merrick, T. and Altan, T., Feed rate optimization based on the cutting force calculations in three-axis milling of dies and molds with sculptured surfaces, *Int. J. Mach. Tools Manufact*, 34(3), 365, 1994.

[56] Chu, C. N., Kim, S. Y., Lee, J. M. and Kim, B. H., Feed-rate optimization of ball end milling considering local shape features, *Annals of CIRP*, Vol. 46/1/1997.

[57] Park, S., Jun, Y. T., Lee C. W. and Yang, M. Y., Determining the cutting conditions for sculptured surface machining, *Int J. Adv Manuf Technol*, 8:61-70, 1993.

[58] Weinert, K., Enselmann, A. and Friedhoff, J., Milling simulation for process optimization in the field of die and mold manufacturing, Annals of the CIRP, Vol. 46, pp. 325-328, 1997

[59] Lim, E. and Menq, C., Integrated planning for precision machining of complex surfaces. Part I: Cutting-path and feedrate optimization, *Int. J. Mach. Tools Manufact*, 37(1), pp. 61-75, 1996.

[60] Lim, E., Feng, H. Y., Lin, Z. H., and Menq, C., The prediction of dimensional error for sculptured surface productions using the ball-end milling process, Part I: Chip geometry analysis and cutting force prediction, *Int. J.*

Mach. Tools Manufact, 35(8), pp. 1149-1169, 1995.

[61] Chou, J. J. and Yang, D. C. H., Command generation for three axis CNC machining, *Journal of Engineering for Industry*, Vol. 113, pp. 305-310, 1991.

OTHER PATH PROBLEMS

[62] Antonio, J., Optimal trajectory planning for spray coating, *IEEE International Conference on Robotics and Automation*, pp. 2570 -2577 vol.3, 1994.

[63] Ramabhadran, R. and Antonio, J. K., Fast solution techniques for a class of optimal trajectory planning problems with applications to automated spray coating, *IEEE Transactions on Robotics and Automation*, Volume. 134, Page(s): 519 -530, Aug. 1997.

[64] Ramabhadran, R. and Antonio, J. K., Fast solutions for a class of optimal trajectory planning problems with applications to automated spray coating, *Proceedings of the 34th IEEE Conference on Decision and Control*, Volume. 2, pp. 1612 -1617, 1995.

[65] Ramabhadran, R. and Antonio, J. K., Planning spatial paths for automated spray coating applications, *Proceedings.of IEEE International Conference on Robotics and Automation*, Volume. 2, pp. 1255 -1260, 1996.

[66] Chou, J. J., *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces*, Ph. D. Thesis, Dept. of Computer Science, University of Utah, pp. 104-108, 1989.

[67] Bisschop, J. J., Stegman, H., and Striekworld, M. E. A., The sequence of point operations on a CNC-machining center using a microcomputer, *International Journal of Production Research*, Vol. 26, No. 8, pp. 1375-1383, 1988.

[68] Arkin, E. M. and Hassin, R., Approximation algorithms for the geometric covering salesman problem, *Discrete Appl. Math.*, Vol. 55, pp. 197-218, 1994.

[69] Arkin, E. M., Fekete, S. P. and Mitchell J. S. B., Approximation algorithms for lawn mowing and milling, *Compositional Geometry: Theory and Application*, Vol. 17, pp. 25-50, 2000.

[70] Ntafos, N., Watchman routes under limited visibility, *Computational Geometry: Theory and Application*, Vol. 1, No. 3, pp. 149-170, 1992.

[71] Current, J. T. and Schilling, D. A., The covering salesman problem, *Transportation Sci.*, Vol. 23, pp. 208-213, pp. 1989.

[72] Huang, Y. Y., Cao. Z. and Hall. E. L., Region filling operations for mobile robot using computer graphics, *Proc. of 1986 IEEE Int'l Conference on Robotics and Automation*, pp. 1607-1614, 1986.

[73] Prabhu, P. V., Gramopadhye, A. K., and Wang, H. B., A general mathematical model for optimizing NC tool path for face milling of flat convex polygonal surfaces, Int. J. Prod. Res., Vol. 28, pp. 101-130, 1990.

[74] Deshmukh, A. and Wang, H. B., Tool path planning for NC milling of convex polygonal faces: minimization of non-cutting area, *Int. J. Manuf Technol*, **8**:17-24, 1993.

[75] Sarma, S. E., The crossing function and its application to zig-zag tool paths, *Computer-Aided Design*, Vol. 31, pp. 881-890, 1999.

[76] Park, S. C. and Choi, B. K., Tool-path planning for direction-parallel area milling, *Computer-Aided Design*, Vol. 32, pp. 17-25, 2000.

[77] Dong, H. L., and Vickers, G. W., Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis, *Computer-Aided Design*, Vol. 26, No. 11, Nov 1994.

[78] Marshall, S. and Griffiths, J. G., A new cutter-path topology for milling machines, *Computer-Aided Design*, Vol. 26, No. 3, March, 1994.

[79] Tam, H., Towards the uniform coverage of surfaces by scanning curves, *Computer-Aided Design*, Vol. 31, pp. 585-596, 1999.

TIME-OPTIMAL CONTROL

[80] Boltianskii, V., Martini, H., Soltan, V., Chapter 1, Nonclassical Variational Calculus in Geometric Methods and Optimization Problems, *Combinatorial Optimization*, Vol. 4, Kluwer Academic Publishers, 1999.

[81] Pontriagin, L. S., *The mathematical theory of optimal processes*, Interscience Publishers, 1962.

[82] Boltianskii, V., *Mathematical methods of optimal control*, Rinehart and Winston, 1971.

[83] Yang, H. S. and Slotine, J.-J. E., Fast algorithms for near-minimum-time control of robot manipulators, *The International Journal of Robotics Research*, Vol. 13, No. 6, pp. 521-532, Dec 1994.

[84] Kahn, M. E. and Roth, B., The near-minimum-time control of open-loop articulated kinematic chains, *Journal of Dynamic Systems*, Measurement, and Control, 93(3):164-172, 1971.

[85] Shiller, Z., Time optimal motion of robotic manipulators, Sc. D. Thesis, Dept. of Mechanical Engineering, MIT, 1987

[86] Meier, E-B. and Bryson, A. E. Jr., Efficient algorithm for time-optimal control of a two-link manipulator, *J. Guidance Control Dynam.* 13(5):859-866, 1990.

[87] Rajan, V. T., Minimum-time trajectory planning, *Proc. of IEEE Conference on Robotics and Automation*, pp. 759-764, 1985.

[88] Bobrow, J. E., Dubowsky, S. and Gibson, J. S., Time-optimal control of robotic manipulators along specified paths, *The International Journal of Robotics Research*, Vol. 4, No. 3, Fall 1985.

[89] Shin, K. G. and McKay, N. D., Minimum-time control of robotic manipulators with geometric path constraints, *IEEE Transactions on Automatic Control*, Vol. AC-30, No. 6, June 1985.

[90] Shin, K. G. and McKay, N. D., Robot path planning using dynamic programming, *Proc. 23rd Conference on Decision and Control*, Dec 1984.

[91] Huang, H. and McClamroch, N. H., Time-optimal Control for a robotic contour following problem, *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, April 1988.

[92] Bellman, R. and Kalaba, R. E., *Dynamic programming and modern control theory*, Academic Press, 1965.

[93] Bellman, R. and Dreyfus, S. E., *Applied dynamic programming*, Princeton University Press, pp. 200-203, 1962.

[94] Strang, G., *Introduction to applied mathematics*, Wellesley-Cambridge Press, pp. 589, 1986.

[95] Boltianskii, V., *Mathematical methods of optimal control*, pp. 13-18, pp.174-184, Rinehart and Winston, 1971.

[96] Boltianskii, V., *Optimal control of discrete systems*, pp. 43-50, John Wiley, 1978.

[97] Agrawal, S. K. and Fabien B. C., *Optimization of dynamic systems*, pp. 76-79, Kluwer Academic Publishers, 1999.

[98] Singh, S. K. and Leu, M. C., Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints, *The International Journal of Robotics Research*, Vol. 10, No. 2, April 1991.

[99] Agrawal, S. K. and Fabien B. C., Chapter 5, Dynamic optimization: Direct Solution, *Optimization of dynamic systems*, pp. 93-111, Kluwer Academic Publishers, 1999.

[100] Betts, J., A direct approach to solving optimal control problems, *Computing in Science & Engineering*, May-June, 1999.

[101] Bryson, A. E., Optimal Control—1950 to 1985, *IEEE Control Systems*, pp.26-33, 1996.

[102] Sussmann, H.J.; Willems, J.C., 300 years of optimal control: from the brachystochrone to the maximum principle, IEEE Control Systems Magazine, pp. 32-44, 1997.

Differential Geometry and Calculus

[103] Weintraub, S. H., *Differential forms: a complement of vector calculus*, Academic Press, 1997.

[104] Darling, R., *Differential forms and connections*, Cambridge University Press, 1994.

[105] Buck, R. C., *Advanced Calculus*, pp. 366-436, 1965.

[106] Flanders, H., *Differential forms with applications to the physical science*, Dover Pub., 1989.

[107] Theodore, F., *The geometry of physics: an introduction*, Cambridge University Press, 1997.

[108] Hsiung, C. C., Differential Forms, *A first course in differential geometry*, International Press, pp. 64-75, 1997.

[109] Hsiung, C. C., Differential map and Jacobian matrix, *A first course in differential geometry*, International Press, pp. 40-42, 1997.

[110] Hsiung, C. C., Geodesics, *A first course in differential geometry*, International Press, pp. 229-239, 1997.

[111] Hsiung, C. C., A Curve on a Surface, *A first course in differential geometry*, International Press, pp. 171-172, 1997.

[112] Hsiung, C. C., Principal Normal, Binormal, Osculating Planes, Normal Planes, Rectifying Planes, Surface

Normal, Normal Curvatures and Normal Sections, *A first course in differential geometry*, International Press, pp. 91, pp. 174, pp. 177, pp. 185, 1997.

[113] Hsiung, C. C., Gaussian Curvatures and Hyperbolic Points, *A first course in differential geometry*, International Press, pp. 189, 1997.

[114] Hsiung, C. C., Poincaré Theorem or Lemma, *A first course in differential geometry*, International Press, pp. 70, 1997.

[115] Hsiung, C. C., Euler's Formula, *A first course in differential geometry*, International Press, pp. 184, 1997.

[116] Hsiung, C. C., Geodesic Curvature, *A first course in differential geometry*, International Press, pp. 229, 1997.

[117] Hsiung, C. C., Weingarten Formula, *A first course in differential geometry*, International Press, pp. 207, 1997.

[118] Tricot, C., *Curves and Fractal Dimension*, Springer Verlag, 1995.

[119] Hildebrand, F. B., *Advanced Calculus for Applications*, 2nd ed., p.387, Englewood Cliffs, 1976.

[120] Arnold, V. I., Chapter 2, First-Order Partial Differential Equations, *Geometrical Methods in the Theory of Ordinary Differential Equations*, 2nd ed., p. 59-88, Springer-Verlag, 1988.

[121] Melikyan, A., Chapter 1, Method of Characteristics in Smooth Problems, *Generalized Characteristics of First Order PDEs,: Applications in Optimal Control & Differential Games*, pp. 7-54, Birkhãuser Boston, 1998.

[122] Sethian J. A. and Adalsteinsson, An overview of level set methods for etching, decomposition and lithography development, *IEEE Transactions of semiconductor manufacturing*, Vol 10., No. 1, Feb 1997.

[123] Osher, S. and Sethian, J., Fronts propagating with curvature dependent speed: algorithms based on the Hamilton-Jacobi Formulation, *J. Comp. Physics*, Vol. 79, 1988.

Combinatorial Optimization

[124] Papadimitriou, C. H., *Combinatorial optimization: algorithms and complexity*, Dover Publications, 1998.

[125] Papadimitriou, C. H., *Combinatorial optimization: algorithms and complexity*, pp. 450, Dover Publications, 1998.

[126] Kwan, M., Graphic programming using odd or even points, *Chinese Mathematics*, 1962.

[127] Gordan, M. and Minoux, M., *Graphs and Algorithms*, John Wiley & Sons, New York, 1984.

[128] Cao, Y.J., Jiang, L. and Wu, Q. H., An evolutionary programming approach to mixed-variable problems, *Applied mathematical modeling*, Vol. 24, pp. 931-942, 2000.

[129] Cooper, L. and Cooper, M. W., *Introduction to Dynamic Programming*, Pergamum Press, pp.142-145, 1981.

Mechanics and Kinematics

[130] Lanczos, C., *The variational principles of mechanics*, Dover Publications, pp.51-53, 1986.

[131] Truesdell, C. and Noll, W., *The nonlinear field theories of mechanics*, in Flugge, S. (*ed.*), Handbuch der Physik, Vol. III/3, Springer Verlag, Berlin, pp. 41, 1960.

[132] Synge, J. L., Classical Dynamics, pp.14, *Principles of Classical Mechanics and Field Theory*, in Flugge, S. (ed.), *Handbuch der Physik*, Vol. III/1, Springer Verlag, Berlin, 1960.

[133] Murray, R. M., Li, Z. and Sastry, S., *A mathematical introduction to robotic manipulation*, Boca Raton: CRC Press, 1994.

[134] Crandall, S. H., *Dynamics of mechanical and electromechanical systems*, Krieger Pub. Co., 1982.

[135] Crandall, S. H., *Dynamics of mechanical and electromechanical systems*, Krieger Pub. Co., pp. 115-117, 1982.

[136] Truesdell, C. and Rajagopal, K. R., An introduction to the mechanics of Fluids, Birkhãuser, pp. 7., 2000.

[137] Huynh, P. and Arai, T., Maximum velocity analysis of Parallel manipulators, *Proc. IEEE International Conference on Robotics and Automation*, pp.3268-3273, April 1997.

[138] Yoshikawa, T., Manipulability of robotic mechanisms, *Int, J. Robot. Res.*, Vol. 4, No. 2, Summer 1985.

[139] Yoshikawa, T., Dynamic manipulability of robot manipulators, *Int, J. Robot. Res.*, Vol. 2, No. 1, pp. 113-124, 1985.

[140] Doty, K. L., Melchiorri, C., Schwartz, E. M. and Bonivento, C., Robot Manipulability, *IEEE Transactions on*

Robotics and Automation, Vol. 11, No. 3, June 1995.

[141] Crandall, M. G., Ishii, H. and Lions, P-L., User's Guide to Viscosity Solutions of Second Order Partial Differential Equations, *Bull. AMS*, 27/1, pp. 1-62, 1992.

[142] Crandall, M. G. and Lions, P-L., Viscosity Solutions of Hamilton Jacobi Equations, *Trans. AMS*, 277, pp. 1-43, 1983.

[143] Muller, S., Variational models for micro-structure and phase transitions, *Calculus of Variations and Geometric Evolution Problems* (Ed. Hildebrandt, S. and Struwe, M.), LNM 1713, Lectures given at the 2nd Session of the CIME, pp. 178, Cetraro Italy, June 1996.

[144] Nikolaev, I. and Zhuzhoma, E., *Flows on 2-dimensional Manifolds: An Overview*, LNM 1705, Springer, 1999.

[145] Andronov, A. A., Leontovich, E.A., Gordon, L.L. and Maier, A.G., *Theory of Bifurcation of Dynamic Systems on A Plane*, Keter Press, 1971.

Computational Geometry and Vector Field Interpolation

[146] Zhong, J., Weng, J.and Huang, T. S., Vector field interpolation in fluid flow, *Digital Signal Processing*, pp. 323-329, 1991.

[147] Shaefer, J. T. and Doswell, C. A. III, On the interpolation of a vector field, *Monthly Weather Review*, Vol. 107, pp. 458-476, April 1979.

[148] Scueuermann, G., Tricoche, X. and Hagen, H., C1-interpolation for vector field topology visualization, *Proceedings of Visualization 99*, pp. 271-279, 533, 1999.

[149] Telea, A. and van Wijk J. J., Simplified Representation of Vector Fields, *Proceedings of the conference on Visualization 99: Celebrating ten years*, pp. 35-42, 1999.

[150] Preusser, A., Computing area filling contours for surface defined by piecewise polynomials, *Computer Aided Geometric Design*, **3**:267-279, 1986.

[151] Alfonzetti, S., An N-dimensional Algorithm to draw contour lines over triangular elements, *IEEE Transactions on Magnetics*, Vol. 32, No. 3, pp. 1473-1476, May 1996.

[152] Helman, J. and Hesselink, L., Representation and display of vector field topology in fluid flow data sets, *IEEE Computer*, pp. 27-36, 1989.

[153] Helman, J. and Hesselink, L., Visualizing Vector Field Topology in Fluid Flows, *IEEE Computer Graphics and Visualization*, pp. 36-45, 1991.

[154] Scheuermann, G., Krüger H., Menzel, M. and Rockwood, A. P., Visualizing nonlinear vector field topology, *IEEE Transaction on Visualization and Computer Graphics*, Vol. 4, No. 2, pp. 109-116, April-June 1998.

[155] Ueng, S., Sikorski C. and Ma, K., Efficient streamlines, streamribbon and streamtube constructions on unstructured grids, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, No. 2, pp. 100-108, June 1998.

[156] Turk, G. and Banks, D., Image-guided streamline placement, *Proceedings of Computer Graphics SIGGRAPH 96*, pp. 453-460, 1996.

[157] Saito, T. and Takahashi, T., *3. Curved Hatching* in Comprehensible rendering of 3-D shapes, *Computer Graphics*, Vol. 24, No. 4, Aug., pp. 197-206, 1990.

[158] Jobard, B. and lefer, W., Creating evenly-spaced streamlines of arbitrary density, *Proceedings of Eurographic workshop on Visualization in Scientific Computing 97*, pp. 43-55, 1997.

[159] Elber, G., Line art rendering via a coverage of isoparametric curves, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 3, pp. 231 -239, Sept. 1995.

[160] De Leeuw, W. and Van Liere, R., Collapsing flow topology using area metrics, *Proceedings of Visualization 99*, pp. 349-354, pp. 542, 1999.

[161] Heckel, B.; Weber, G. and Hamann, B.; Joy, K.I., Construction of vector field hierarchies, *Proceedings of Visualization 99*, pp. 19-25, pp. 505, 1999.

[162] Shah, J., A common framework for curve evolution, segmentation and anisotropic diffusion. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 136 -142, 1996.

[163] Braides, A., *Approximation of Free-Discontinuity Problems*, LNM 1694, Springer, 1998.

[164] Trichohe, X, Scheuermann, G. and Hagen, H., High order singularities in piecewise linear vector fields, *Pro-*

ceedings of 9th Conference on the Mathematics of Surfaces, Cambridge, September 2000.

3131-16