Hongcheng Wang
Peter Jang
James A. Stori
e-mail: jastori@uiuc.edu
Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801
Telephone: (217) 244-7762; fax: (217) 333-1942

A Metric-Based Approach to Two-Dimensional (2D) Tool-Path Optimization for High-Speed Machining

Conventional tool-path generation strategies are readily available to generate geometrically feasible trajectories. Such approaches seldom take into consideration physical process concerns or dynamic system limitations. In the present work, an approach for improving a geometrically feasible tool-path trajectory based on quantifiable process metrics is developed. Two specific measures of toolpath quality are incorporated into the iterative improvement algorithm: instantaneous path curvature and instantaneous cutter engagement. These metrics are motivated by a desire to minimize acceleration requirements and maintain a stable steady-state cutting process during high-speed machining. The algorithm has been implemented for two-dimensional contiguous end-milling operations with flat end-mills, and case studies are presented to illustrate the approach. [DOI: 10.1115/1.1830492]

Keywords: High-Speed Machining, Tool-Path, Engagement, Curvature

Introduction

High speed machining (HSM) is a key enabling technology in an increasing number of industries. In the aerospace industry, structural components are increasingly being machined as monolithic structures from a single billet. The result is drastically reduced part counts, assembly costs, and even maintenance costs. The Boeing F/A 18 E/F tactical aircraft realized a 42% reduction in parts and a 25% weight savings over previous models, attributed not in small part to the design changes made practical by the application of high speed machining technology [1]. In the tooling industry, high speed machining technology continues to grow increasingly important for maintaining economic competitiveness. Successful applications of HSM to the production of tooling for forging, extrusion, sheet forming, die casting, and injection molding have been reported [2–5].

High-speed (HV) machine tools with 1000+ ipm traverse rates, and 50,000+ RPM high-power spindles, are readily available. However, the velocity capabilities of such machines are seldom reached in industrial practice. Often, conservative feedrates are employed in contouring due to concerns arising from process stability and acceleration limitations. To address these issues, significant attention has been focused on the important problems of dynamic modeling, parameter optimization, and feedrate scheduling. Often overlooked, however, is the potential for improvement in process efficiency through changes to the tool-path itself.

Tool-path planning has been traditionally approached from a purely geometric perspective. The vast majority of bulk material removal occurs in 2.5 D roughing operations. For such operations, tool-path trajectories are generated through conventional strategies such as contour-parallel or direction-parallel offsetting. When the dynamics and mechanics of the process and machine tool system are considered, conventional tool-path generation techniques are observed to be far from ideal. In the present work, an approach is developed to optimize a geometrically feasible toolpath based on quantifiable metrics driven by important technological concerns.

Technical Challenges in HSM. Numerous technological issues in high speed machining (HSM) pose particular challenges from the perspective of process planning, optimization, and control. Conventional tool-path strategies fail to adequately address concerns such as dynamic stability, acceleration demands, and constantly varying tool engagement that limit production rates, increase tool wear, and reduce part quality.

First, and possibly foremost, are limitations imposed by consideration of dynamic stability. Dynamic stability issues drive the majority of efforts to effectively implement high speed machining on the shop floor. Tlusty states: “It has long been recognized that the occurrence of chatter is one of the most significant limitations in increasing the metal removal rate in milling” [6]. The difference between successful stable cutting and catastrophic unstable cutting can be traced to subtle interactions between the dynamics of the tool, spindle, and workpiece. Instantaneous chip loads and cutter engagements. Numerous models have been developed for dynamic simulation of the machining process, and serve as increasingly accurate tools for selection of the spindle speed and cutting parameters for chatter-free machining [6–9]. Additionally, systems have been developed for avoidance or detection and control of chatter by varying the (constant) spindle speed or cutting with a varying spindle speed [10–12].

A second crucial issue in high speed machining is making effective use of the dynamic capabilities of the machine tool, actuators, and servo drives. Renton and Elbestawi [13] demonstrate that by making complete use of the performance envelope of each servo controlled axis, feed rates can be increased while simultaneously decreasing tracking path error. Richard Bertsche, of Bertsche Engineering, a manufacturer of high-speed machine tools, emphasizes the crucial role that acceleration capabilities play in the high speed machining process [14]. He notes that two machines with similar velocity yet different acceleration capabilities may differ by 50% in total processing time when cutting a moderately sized pocket. For many practical machining tasks, a high-speed machine tool may spend the majority of its time accelerating and decelerating [15]. During the inevitable periods of dwelling, tool temperatures climb sharply, drastically reducing tool life. Fatigue and wear of the machine tool itself is unnecessarily hastened by the extreme acceleration demands placed upon the actuators and structure. In such cases, a path that minimizes acceleration demands can have tremendous impact on cycle time.
Process Parameter Optimization and the Role of the Engagement Angle. Numerous researchers have approached the problem of optimization/selection of the process parameters for traditional and high-speed machining. Tsusty et al. [16] and Smith et al. [17] address issues of stability in endmilling and the importance of selecting the proper axial and radial depths of cut for chatter-free machining. Tsusty et al. [16] notes that although the radial engagement remains constant for straight path segments, chatter commonly is encountered during cornering as a result of the increased radial engagement (Fig. 1). Stori et al. [18,19] developed a simulation-based approach for parameter selection based on static model predictions of cutting forces, power, and tool deflection. A review of prior approaches to parameter optimization may be found in Stori et al. [18].

In 2.5 D milling, the angle of engagement between the tool and workpiece, $\theta$, is a convenient and unambiguous substitute for the radial depth of cut for arbitrary tool trajectories (Fig. 1). The angle of engagement, entry and exit angles, axial depth of cut, cutting (spindle) speed, and instantaneous feedrate (chip-load) uniquely determine the chip geometry [20]. In conjunction with the cutting tool, workpiece, and spindle/machine tool characteristics, it is these instantaneous process parameters that will determine cutting forces, deflections, and process stability.

The dependence of process stability on the spindle speed and axial depth of cut is well documented. The early work of Tobias [22], Merrit [23], and Koenisberger and Tsusty [24] led to the development of stability lobe diagrams that identify stable cutting regions as a function of the axial depth of cut and spindle speed for a given radial depth of cut. Numerous analytic and time domain procedures for process stability have been developed and validated in the literature [9,25,26]. While radial engagement has not been treated as a primary process variable in the majority of stability analyses, its importance may be clearly seen in the model predictions. A series of stability diagrams from [9] is shown in Fig. 2. As may be seen in the figure, the stability limits change dramatically as the radial engagement decreases from 100% (slotting) to 10%. Abrari et al. [27] identified stable cutting regions as a function of the width of cut and spindle speed for a given axial depth of cut in the ball endmilling process.

Given a conventional toolpath, it will not be possible to take advantage of a diagram such as that of Fig. 2 to achieve chatter-free cutting, as the instantaneous radial engagement is typically unknown, uncontrolled, and widely varying. As developments in dynamic stability analysis continue to highlight the important role of radial engagement, it is hoped that further efforts will be expended on toolpath generation and parameter selection strategies to mitigate the effects of varying engagement.

Feedrate Scheduling and Toolpath Modification. Despite the sensitivity of these instantaneous parameters to process stability, cutting forces, and process efficiency, there has been surprisingly little work done to modify tool-paths explicitly, such that favorable conditions of these instantaneous parameters may be achieved in the cutting of complex geometries. As the axial depth of cut and cutting speed are not readily varied within a continuous cutting operation with a flat end-mill, the engagement angle and instantaneous feedrate become critical process control variables. The feedrate may be scheduled independently of tool-path geometry, and a number of researchers have explored feedrate optimization/scheduling as a post processing step after a trajectory has been generated. Recently, Renton and Elbestawi [13], Farouki et al. [28], and Stori and Ferreira [29] have proposed feedrate scheduling approaches driven by axis velocity and acceleration limitations. Additionally, methods have been developed to schedule feedrates based on model predictions of constraints such as forces and tool deflections [30–32]. Kramer [21] detected regions of excessive engagement during zig-zag toolpath generation, and selected corresponding feedrates based on several engagement thresholds.

However, the potential for improvement in engagement variation and acceleration demands through changes to the tool-path geometry is significant. In only a very small number of instances has reference been made to explicit attempts to modify the toolpath geometry based on physical process concerns. Tsaì et al. [33] add additional circular arc segments to convex inside corners to keep cutter engagement below prescribed limits. They attribute

$$\theta_{\text{max}} = \pi - \arccos \left( \frac{2r^2 + 2R(s - r) - s^2}{2r(R - r)} \right)$$

Fig. 1 Varying engagement arising with contour-parallel offsets [21]

![Fig. 2 Stability lobes for 100%, 50%, 25%, and 10% radial immersion](image-url)
this technique to Iwabe et al. [34], Trusty et al. [16] and Smith et al. [17] avoid over-engagement in cornering by adopting a spiral-in approach.

Bierberman and Sandstrom [35] have developed an interesting approach for generating smooth low-curvature spiral paths for 2D planar pocketing. Through the solution of an elliptic PDE boundary value problem, a specified pocket boundary is morphed into smooth interior orbits. By spiraling between these PDE solution contours, a smooth low-curvature trajectory is generated. The approach has resulted in tool-paths with demonstrably lower tool-wear and reduced machining time for high-speed machining. Instantaneous cutter engagement is not constrained explicitly, although sharp corners are eliminated, and acceleration concerns greatly improved.

Metrics for Tool-Path Evaluation

One recurring challenge in process planning activities is the difficulty of concretely evaluating the quality and efficiency of competing alternative solutions. Planning decisions are often ad hoc, and approaches to computer-aided process planning (CAPP) typically rely on heuristics and rule structures for decision making. The notion of optimality is generally avoided, perhaps due to the extreme breadth of the solution space.

In the present work, an approach based on quantifiable metrics for tool-path evaluation is developed. While it may yet remain not possible to claim optimality of tool-path trajectories, the introduction of an approach based on quantifiable metrics will permit unbiased evaluation of algorithmic progress and comparison with conventional approaches.

The developed metrics for tool-path evaluation are based on the following assumptions:

• The ideal cutting operation is a steady-state process, and a target operating point for the operation can be identified a priori. This instantaneous cutting state target is characterized by an instantaneous engagement angle, feed, cutting speed, and depth of cut.

• It is desired to maintain the cutting process at, or near, this ideal operating point for as much of the operation as possible. The larger the deviation from the ideal operating condition, the more inferior the solution, and the longer the deviation from the ideal operating point, the more inferior the solution.

Given the above criteria, a path integral consisting of individual terms penalizing deviations from the ideal operating point for each of the instantaneous state variables represents a natural functional form for the metric. This approach has an analogy with the multiple objective optimization problem, where a set of constraints is replaced by a single objective function. The notion of optimality is generally avoided, perhaps due to the extreme breadth of the solution space.

In the present work, a set of metrics is developed to evaluate tool-paths. While it may yet remain not possible to claim optimality of tool-path trajectories, the introduction of an approach based on quantifiable metrics will permit unbiased evaluation of algorithmic progress and comparison with conventional approaches.

The developed metrics for tool-path evaluation are based on the following assumptions:

• The ideal cutting operation is a steady-state process, and a target operating point for the process can be identified a priori. This instantaneous cutting state target is characterized by an instantaneous engagement angle, feed, cutting speed, and depth of cut.

• It is desired to maintain the cutting process at, or near, this ideal operating point for as much of the operation as possible. The larger the deviation from the ideal operating condition, the more inferior the solution, and the longer the deviation from the ideal operating point, the more inferior the solution.

Given the above criteria, a path integral consisting of individual terms penalizing deviations from the ideal operating point for each of the instantaneous state variables represents a natural functional form for the metric. This approach has an analogy with the multiple objective optimization problem (see Ref. [36] for a review of such methods). In our case, however, the penalty is integrated along the entire path. If the path is represented as a sequence of continuous segments parameterized by a variable $u$, and $\vec{z}_i$ represents target values of each of the $i$ state variables (instantaneous engagement angle, feed, etc.), then the penalty function takes the form of Eq. (1), below. The scalars $w_i$ are the weights, and the exponent $p$ can vary from 1 to $\infty$. Typically, a value of $p = 1$ or 2 would be commonly used. Note that as $p$ increases, the solution approaches the weighted minimax, or Tchebycheff, method (minimize the maximum deviation of any single variable from the target).

$$U = \sum_i \sum_j \int_a^b (w_i |f_i(u) - \bar{z}_i|^p) du$$  \hspace{1cm} (1)

In the above expressions, $f_i(u)$ is a function that returns the instantaneous state at the point $u$, $\bar{z}_i$ is the target state value, $j$ is the segment number, and $i$ is the specific technical metric considered. Within the general framework described in Eq. (1), a variety of efficiency, quality, and process metrics can be supported.

Curvature. Path curvature is integrally tied to the acceleration and jerk required to track the required trajectory. In the high speed machining domain, it is desirable to minimize the actuator requirements and the time spent accelerating. In the extreme case of discontinuous cornering, the inherent dwell time leads to rapid unloading and loading of the machine axes and cutting tool, resulting in decreased part quality and excessive tool and machine wear. We assume that the path geometry is represented by a combination of individual continuous segments parametrized by a variable, $u$, $P(u) = (x(u), y(u), z(u))$. The path geometry must be augmented with a mapping into the time domain in order to achieve an intended feed rate profile. This mapping may be represented as an additional function $u(t)$, prescribing the parametric speed.

The acceleration vector is obtained through application of the chain rule, and differentiation:

$$a = \frac{d^2}{dt^2} P(u(t)) = \frac{d}{dt}(P(u(t))u(t)^2 + \frac{d}{du}(P(u(t))u(t))$$  \hspace{1cm} (2)

where the dot operator represents differentiation with respect to time, and the prime operator denotes differentiation with respect to the parameter $u$. In order to make the relationship between feedrate and geometric curvature explicit, the acceleration of a single axis ($x$) may be expressed in the following form [37]:

$$a = \frac{V^2}{\kappa} \kappa_t + \dot{V}t_x$$  \hspace{1cm} (3)

where $V = \sqrt{P(u)(u)}$ is the tangential velocity (feedrate), $\kappa_t$ is the $x$ component of the path normal scaled by the curvature, $\dot{V}$ is the feed-acceleration, and $t_x$ is the $x$ component of the unit tangent vector, $t = P(u)/\|P(u)\|$. The curvature, $\kappa$, can be expressed as [38]:

$$\kappa = \frac{\|P(u) \times \kappa P'(u)\|}{\|P'(u)\|^3}$$  \hspace{1cm} (4)

Note that the curvature is solely a function of the path geometry, $P(u)$, and can be represented analytically. In this case, the desire would be to minimize the curvature and hence acceleration demands, $f(\kappa) = |\kappa|$. Correspondingly, the target curvature would be 0, or $\kappa_t(u) = 0$.

Instantaneous Cutter Engagement. Varying cutter engagement is an inevitable consequence of parallel offsets, as has been observed by numerous researchers (Fig. 1). The instantaneous cutter engagement, $\theta$, is a function of the continuously changing in-process geometry of the workpiece. The engagement angle is therefore dependent on the history of the cutter path, and evaluation of the instantaneous cutter engagement must be carried out via geometric process simulation. Assuming that a target instantaneous cutter state can be identified at a given point in time for a particular tool-path trajectory, it is desired to minimize the deviation from this target, that is:

$$f_{\theta}(u) = |\theta - \bar{\theta}| \text{ and } \bar{z}_{\theta}(u) = |\theta|$$  \hspace{1cm} (5)

where $\theta$ is the target engagement angle and $\bar{\theta}$ is the engagement angle obtained via simulation. For continuous peripheral end-milling operations, the exit-angle will be $90^\circ$, and there is a one-to-one correspondence between the entry angle and the engagement angle. In such cases, the engagement angle will be sufficient to determine the direction and magnitude of the cutting force. For side cuts, typically encountered at the end of material removal, and for alternating situations of climb and conventional milling, additional knowledge of the cutting state is required (for example, both entry and exit angles).

In the present implementation, the penalty function incorporates deviations in the engagement angle. If it were desired to more accurately penalize cutting force variations, it would be necessary to augment the penalty function with a suitable mapping between entry–exit angles and a force vector.

In addition to targeting a particular value of the instantaneous engagement, it is also desirable to minimize the number of occur-
optimize the trajectory of the geometrically unconstrained tool-path as a starting point for the algorithm. The intent is to achieve for such operations through one of two conventional techniques: The contour-parallel and engagement angle metrics, respectively, according to application requirements, and the following cost function is obtained:

\[
U = \sum_{i} \sum_{j} \int_{0}^{1} w_{ij} f_{ij}(u) - \bar{z}_{ij}(u) |du
\]

where \( n \) is the number of continuous parametric segments comprising the tool-path. Penalties for discrete events such as the previously described disengagement and re-entry may be added outside of the path integral summation.

Multiple-Objective Planning Metric. Within the general framework of Eq. (1), several disparate metrics may be combined with appropriate weights to form a single utility, or cost function. For example, weights \( w_1 \) and \( w_2 \) may be assigned to the curvature, and engagement angle metrics, respectively, according to application requirements, and the following cost function is obtained:

\[
U = \sum_{i} \sum_{j} \int_{0}^{1} w_{ij} f_{ij}(u) - \bar{z}_{ij}(u) |du
\]

where \( n \) is the number of continuous parametric segments comprising the tool-path. Penalties for discrete events such as the previously described disengagement and re-entry may be added outside of the path integral summation.

Additional Planning Metrics. The above are not intended to constitute a comprehensive set of planning metrics for high speed machining. Rather, they serve to illustrate the types of concerns that may be considered with a metric-based approach. Below, we mention several other metrics that might find relevance for particular applications:

- rewarding increases in the distance from the process stability frontier, intended to provide robustness to the plan in the face of varying system dynamics.
- penalizing entry and exit angles most susceptible to tool damage and breakage.
- penalizing certain variations in uncut chip thickness, a technique useful for prolonging tool life.
- penalizing small chip thicknesses and low feed velocities that lead to excessive temperature increases.

Metric-Based Improvement Approach

The majority of volumetric material removal in milling is achieved through 2D roughing operations with a flat end-mill. For such operations, the periphery geometry is often rigidly constrained. Geometrically feasible toolpaths may be readily generated for such operations through one of two conventional techniques: The contour-parallel (spiral in–out) or direction-parallel (zig–zag) approach. We adopt the topology of the contour-parallel path as a starting point for the algorithm. The intent is to optimize the trajectory of the geometrically unconstrained (interior) tool-path segments so as to minimize deviations from the target instantaneous engagement, and reduce the total curvature.

The methodology developed for metric-based improvement of a geometrically feasible tool-path is outlined schematically in Fig. 3. A geometrically continuous interpolatory spline is used to represent the toolpath. Given a particular interpolation procedure (as described in more detail below), a set of interpolatory control points will uniquely determine the tool-path trajectory. The movement of an individual control point will result in a local change to the tool-path geometry in a neighborhood of that control point. Simulation is carried out at a much finer resolution of points along the accompanying spline trajectory. The algorithm proceeds by sequentially moving individual control points to reduce the metric function cost.

A single iteration of the algorithm consists of updating each of the control points exactly once. During each updating procedure, a local search is performed to determine the movement distance that minimizes the cost function. The two levels of resolution provided by the interpolation points and the simulation points are important; the much smaller set of interpolation points allows local control of the tool-path geometry, while the much larger set of simulation points provides the necessary resolution for the engagement simulation. The optimization algorithm is executed until reductions in the cost function fall below a specified termination tolerance for three consecutive iterations. When the cost function has stabilized for several consecutive iterations, further improvement is unlikely to occur through local updates to the geometry.

Below, we detail the tool-path representation, engagement simulation methodology, and the individual steps of the iterative improvement algorithm.

Tool-Path Representation. An interpolation procedure for a geometrically continuous sequence of cubic Bezier spline segments developed by Shirman and Sequin [39] has been used to represent the tool-path during optimization. An initial feasible path geometry is approximated to a controllable degree of accuracy by interpolation through a suitable number of interpolation points. The parametrized toolpath is comprised of individual Bezier spline segments between each of these interpolation points. They are several key features of this spline interpolation procedure that make it an attractive choice for our application:

- It has been designed to naturally avoid high curvature segments, even in instances when the angle between three consecutive interpolation points is highly acute.
- Several global properties of the spline can be adjusted easily by the user by manipulating three parameters.
- Special case rules guarantee perfectly straight segments in the case of collinear interpolation points, an important consideration for practical tool-path trajectories.
- The spline provides a guaranteed localized region of change with the adjustment of a single interpolation point. The movement of a single interpolation point can only affect the two spline segments to its left and the two segments to its right, since the derivative vector is affected only by the two neighboring interpolation points.
A brief description on the interpolation procedure of Shirman and Sequin follows; a more detailed exposition can be found in Ref. [39].

Figure 4 shows a cubic Bezier segment defined by the control points $c_0$, $c_1$, $c_2$, and $c_3$. The control points $c_0$ and $c_3$ are the interpolation points which define the start and end of the segment. The points $c_1$ and $c_2$ control the tangents at the endpoints as well as the shape of the curve. A cubic Bezier segment can be represented as follows:

$$ B(u) = (1-u)^3 c_0 + 3 u(1-u)^2 c_1 + 3 u^2 (1-u) c_2 + u^3 c_3 $$

(7)

where $c_0$, $c_1$, $c_2$, and $c_3$ are the control points and $u$ is a parameter ranging from 0 to 1. The toolpath is represented by a series of consecutive Bezier segments which pass through the given set of interpolation points $\{P_i\}$. As the start and end points of each Bezier segment must coincide with control points $c_0$ and $c_1$, the spline interpolation task requires the determination of suitable intermediate control points $c_1$ and $c_2$ for each segment. Alternatively, this task can be thought of as defining two control points $c_i$ and $c_{i+1}$ for each interpolation point $P_i$, where the control points lie at the location dictated by the derivative vectors $\vec{d}_{L_i}$ and $\vec{d}_{R_i}$ (see Fig. 5). Using this notation, the equation for a cubic Bezier segment between interpolation points $P_i$ and $P_{i+1}$ can be rewritten as follows:

$$ B(u) = (1-u)^3 P_i + 3 u(1-u)^2 c_{i+1} + 3 u^2 (1-u) c_i + u^3 P_{i+1} $$

(8)

Shirman and Sequin [39] define the normal to the derivative vector as follows:

$$ \vec{n} = \begin{cases} 
(a+b)\hat{a} + (a+b)\hat{b} + D[(b-a)\hat{a} - (b-a)\hat{b}] & \text{for } \angle P_{i-1} P_i P_{i+1} \leq 90^\circ \\
(a+b)\hat{n}_a + (a+b)\hat{n}_b + D[(a-b)\hat{n}_a - (a-b)\hat{n}_b] & \text{for } \angle P_{i-1} P_i P_{i+1} \geq 90^\circ 
\end{cases} $$

(9)

where $\vec{n}$ is the normal of the derivative vector, $a$ is the distance from interpolation point $P_i$ to $P_{i-1}$, $b$ is the distance from point $P_i$ to the $P_{i+1}$, $\hat{a}$ and $\hat{b}$ are orthogonal unit vectors with $\hat{a}$ in the direction from $P_{i-1}$ to $P_i$, and $\hat{b}$ and $\hat{n}$ are orthogonal unit vectors with $\hat{b}$ in the direction of $P_i$ to $P_{i+1}$, and $D$ is an adjustable shape parameter which ranges from $-1$ to $+1$. This parameter controls the direction of the normal; the default value of 0 gives a normal that bisects $\angle P_{i-1}P_iP_{i+1}$. Since both $\vec{d}_{L_i}$ and $\vec{d}_{R_i}$ share the same normal, first order differentiability is preserved at the transitions between segments. Shirman and Sequin define the magnitude of the derivative vector as follows:

$$ ||\vec{d}_{L_i}|| = B[C(c) + (1-C)a] $$

$$ ||\vec{d}_{R_i}|| = B[C(c) + (1-C)b] $$

(10)

where $c$ is the distance from $P_{i-1}$ to $P_i$, $B$ is an adjustable bulge parameter ranging from 0 to 1.5 (default is 1), and $C$ is an adjustable continuity parameter ranging from 0 to 1 (default is 1/2). The bulge parameter $B$ controls how close the curve follows the adjacent vertices; larger $B$ values lead to rounder curves. The continuity parameter $C$ changes the ratio of the two magnitudes; a $C$ value of 1 gives equal magnitudes, forming a $C^1$-continuous curve. The defaults for the adjustable parameters $D$, $B$, and $C$ were chosen by the authors to produce what they refer to as “visually pleasing” splines, where judgement was based upon how closely the spline follows the path of the interpolation points while minimizing sharp turns, particularly in the case where $\angle P_{i-1}P_iP_{i+1}$ is highly acute. These criteria, while subjective, share attributes desirable for machining toolpaths, since curvature minimization is a key metric that we wish to facilitate. The author’s default values for these parameters have proven satisfactory as starting points for the tool-path optimization procedure, although the spline characteristics can be changed as desired simply by adjusting these parameters.

In the case where three consecutive interpolation points are collinear, special rules are enforced in the determination of the normal of the derivative vector in order to ensure a linear spline segment. This extra precaution is taken to ensure that the spline

![Fig. 5 Derivative vectors and notation for the interpolatory spline of Ref. [39]](image-url)
created will accurately represent the shape implied by the interpolation points given by a user. The assumption is that if three consecutive interpolation points are collinear, a linear path segment is desired. Without the enforcement of special case rules, the cubic spline will instead contain some deviations from this desired form. The derivative vector is first rewritten as a linear combination of \( \hat{n}_a \) and \( \hat{n}_b \):

\[
\hat{n}_i = c_a \hat{n}_a + c_b \hat{n}_b
\]

where \( c_a \) and \( c_b \) are the lengths of the normal in the \( \hat{n}_a \) and \( \hat{n}_b \) directions, respectively. A correction factor can then be applied:

\[
\hat{n}_i = k_c' c_a \hat{n}_a + k_b' c_b \hat{n}_b
\]

where \( k_c' \) is zero when \( \angle \hat{n}_a \hat{n}_i \hat{n}_2 = 180^\circ \), and \( k_b' \) is zero when \( \angle \hat{n}_b \hat{n}_i \hat{n}_1 = 180^\circ \). The definition of the correction factors are:

\[
k_c' = \min((20(1 + \cos \angle \hat{n}_a \hat{n}_i \hat{n}_2)) \cdot 1)
\]

\[
k_b' = \min((20(1 + \cos \angle \hat{n}_b \hat{n}_i \hat{n}_1)) \cdot 1)
\]

The correction factors are defined such that the special rule is applied only if the cosine of the angle between the three points is less than \(-0.95\) (i.e., the three points are collinear within a certain tolerance). In order to maintain stability, the correction factors are made to sum up to 1:

\[
k_a = \frac{1}{2}(1 + k_c' - k_b')
\]

\[
k_b = \frac{1}{2}(1 + k_b' - k_c')
\]

It is important to note that choosing evenly spaced values of \( u \) in Eq. (8) will not result in evenly spaced points along the arclength of the spline. It is critical for the individual points on the spline to be evenly spaced for an accurate geometric simulation of the instantaneous engagement. The complete path, consisting of multiple Bezier segments which each consist of a finite number of simulation points, is created by iterating for values of \( u \) in order to achieve evenly spaced simulation points along the arclength within a threshold. To obtain an equally spaced set of interpolation points, we traverse through the spline using the divide-and-conquer method to obtain a simulation point, \( S_n \), that is a chord length distance of \( \Delta s \) from the previous simulation point.

**Engagement Simulation.** Geometric simulation is required to estimate the instantaneous cutter engagement. It is important to note that cutter engagement cannot, for arbitrary tool-path trajectories, be computed analytically. This is because the cutter engagement at a particular point in time is critically dependant on the in-process geometry, which is a function of the history of the tool-path. Figure 6 illustrates the pixel-based simulation procedure for cutter engagement. In the figure, the gray region represents material remaining to be machined. The interior (white) region has already been machined, and the present cutter engagement, \( \theta \), is to be estimated. As illustrated in the figure, both the local in-process geometry and tool are discretized. The workpiece is represented as an array of bits. Any unmachined area is represented bitwise by a 0, and the machined area is represented by a 1. The mask for the tool geometry is represented by a bitmap containing 1s within the circular region of the tool, and 0s elsewhere. It is necessary to generate this mask for the tool geometry only once. A mask of the outline of the tool is also created for engagement estimation. Figure 7 illustrates the engagement simulation procedure using these bitwise representations. The instantaneous engagement can be estimated by checking the status of the in-process geometry for the pixels that overlap the circumference of the cutting tool at its present location. Reading the state of the engagement may be efficiently implemented with a bitcount (number of nonzero bits) of the result of a logical “AND” operation between the workpiece (with the bits reversed) and the tool outline, as shown in the figure. The cutting process is then simulated by masking the tool bitmap onto the workpiece bitmap. This step requires translating the tool to the proper location, and replacing the corresponding elements of the workpiece with the result of a logical “OR” operation between the tool and workpiece. Both of these tasks require only very low-level memory management and computational tasks. For these particular computational operations, special-purpose graphics hardware is not necessary, nor helpful.
The simulation accuracy will depend on selection of two primary variables; the resolution of the simulation (measured by the tool radius in pixels), and the step size, $\Delta s$, between simulation points. A computational study was performed to determine the engagement simulation accuracy for varying resolution and step sizes (see Fig. 8). Experiments were performed for the case of slotting, where the engagement angle is known to be a constant 180°. Tests were performed for both linear paths in varying directions, as well as circular paths. The total error was taken to be the mean error plus the standard deviation expressed as a percentage of engagement. To achieve an expected error in engagement simulation of less than 5%, a tool radius and simulation step size should be chosen that fall within the white region of the figure for both the linear and circular arc trajectories. Figure 8 suggests that a simulation resolution of $r = 50$ pixels and a step size over radius $\Delta s/r$ of 0.2 is expected to result in engagement simulation errors of less than 5% for both linear and circular paths.

An important practical concern is the computational speed of the engagement simulation. Because the simulation only requires that two physical states be maintained (material/no material), both the memory and the computational capabilities of the microprocessor may be used extremely efficiently. Figure 9 illustrates the discretization of a tool bitmap using 8-bit compression. In our implementation, 32 bit compression has been used to fully utilize the capabilities of the 32 bit microprocessor architecture. With a 32 bit microprocessor, a row of 32 bits can be compactly stored as a single unsigned integer in ANSI C, and a bitwise logical operation can be used to simultaneously process these 32 bits in a single clock cycle for the microprocessor. This means that a discretized tool bitmap of 64 bit radius could be compactly stored in memory using a matrix of only $64 \times 2 = 128$ unsigned integers, and the engagement estimation (logical AND) and workpiece update (logical OR) would each require only this same number (128) of clock cycles.

Using bitwise compression, all bitwise comparison operations remain as previously described, but special care must be taken to ensure the appropriate bits are compared. In order for the tool to step with single bit resolution in a 32-bit compressed space, there must exist 32 individual tool bitmaps. Each of the tool bitmaps contain the same single tool image, offset by a single bit in the horizontal direction. In order to simulate a tool movement to a new position, the appropriate tool bitmap must be called upon.

In terms of computational efficiency, the time to run the simulation algorithm is affected by both the tool radius, $r$, (in pixels) and the number of simulation points $N$ (proportional to the length of the toolpath). Since the number of bitwise comparisons are directly proportional to the number of simulation points, the theoretical running time with respect to $N$ is $O(N)$. The number of bitwise comparisons for position updating depends on the tool area, while the engagement calculation depends on the number of
pixels along the circumference of the tool. Thus the theoretical running time with respect to \( r \) is \( O(r^2/32 + r) = O(r^2/32) \approx O(2r) \) for practical values of \( r \) (\(< 64\)). Experimental results verify that the time scales linearly with the number of simulation points and demonstrate the expected (slightly) nonlinear dependence on tool diameter (see Fig. 10). It is important to note that although the number of simulation points will vary from one given toolpath to another, the tool radius in bits is fixed for a given simulation accuracy.

**Local Simulation.** An important improvement to the efficiency of the algorithm can be made by performing only local engagement simulation. The engagement simulation must be executed every time a cost is to be recalculated when any single interpolation point is altered. Due to the iterative nature of the optimization process, this number is quite large in magnitude. However, due to the relatively small changes that can occur to the toolpath when a single interpolation point is altered, a local simulation technique may be employed for improving simulation speed. This technique requires identifying all critical path segments which require simulation. When moving the interpolation point \( P_i \), only the spline segments between \( P_{i-2} \) and \( P_{i+2} \) will be affected, due to the local modification property of the spline. These critical path segments must be simulated, along with any neighboring path segments whose engagement angles would be affected by the movement. Identifying these critical path segments requires a geometric search which identifies any sections near enough to be affected by the altered spline segment. This can be accomplished by performing a 2D Delaunay triangulation of the set of interpolation points to obtain all nearest neighbors to each point. The triangulation creates unique and non-intersecting edges from each point to all of its nearest neighbors, and is used to identify all neighboring points for each interpolation point. By accumulating the neighboring points for the interpolation points \( P_{i-2}, P_{i-1}, \ldots, P_{i+2} \), the critical path segments can be formed. Figure 11 illustrates the use of the Delaunay triangulation to find the neighboring points. For interpolation point \( P_i \), it is desired to obtain the nearest neighbor interpolation points to the primary critical spline segment; i.e., the neighbors to the points \( P_{i-2}, P_{i-1}, \ldots, P_{i+2} \). Using the Delaunay triangulation, which provides the nearest neighbors for any individual control point, the neighbors to these five points can be accumulated in order to obtain the neighboring critical spline segments.

Any of these critical segments which occur chronologically prior to the primary critical spline segment do not require updated engagement angles, since they cannot be affected by alterations made in the future portions of the path. However, these segments must be simulated in order to have the appropriate geometry when computing the engagement angles for the superseding segments. These superseding segments must then be simulated and their engagement angles must be updated.

Figure 12 shows the difference in performance between the global and local simulation methods. For the recommended tool radius of 50 bits, the results show approximately a 70% decrease in computational time for the local simulation over the global simulation, indicating significant gains in computational efficiency.

**Curvature Computation.** In contrast to the computationally expensive simulation required for instantaneous engagement estimation, the path curvature may be computed directly from the Bezier spline representation using Eq. (4). For the purpose of evaluating the metric cost, Eq. (4) is evaluated at each of the discrete simulation points, and the resulting summation approximates the path integral of Eq. (6). The absolute value of the signed curvature is used when computing the cost.

**Sensitivity Analysis and Updating of Control Points.** At the start of each iteration, a single interpolation point is chosen from the list of interpolation points that have not yet been updated. The point is chosen from the list at random to provide unbiased updating of the toolpath. The control point is moved a direction perpendicular to its tangent in the direction of descent relative to the cost function, as illustrated in Fig. 13. The distance moved is a scalar \( v \), and the location of the control point is updated as in Eq. (15). The final distance for the update, \( v^* \), is obtained through a one-dimensional search based on a quadratic approximation to the cost as a function of \( v \). Costs are first obtained for the movements of \( P_i \) by \( v_{\text{max}}, 0 \), and \(-v_{\text{max}}\), where \( v_{\text{max}} \) is taken to be the tool diameter. This movement is typically large enough such that the associated costs are much greater than that of the zero movement, making the three points a bracketing trio for the quadratic search. A parabola is fit through the bracketing trio, the ordinate for the quadratic search.

\[
P_i \leftarrow P_i + v \cdot \hat{n}_i
\]  

(15)

It is important to note that in order to find the global minimum, the optimization algorithm requires the function to be convex. The cost function, although often convex, is not guaranteed to be so. In particular, more than a single minima can exist within the initial
bracketing trio for complex path geometries. Nevertheless, the fact that each control point will undergo several local search updates (one per iteration) makes it unlikely that it will remain in a locally disadvantageous position.

**Additional Constraint to Maintain Geometric Feasibility.**

It is possible for a control point updating step based solely on the metrics to result in a geometrically infeasible toolpath. Specifically, two possibilities exist: a void may result in the resulting swept volume of the toolpath trajectory, or the outline of the processed shape may be altered from the original desired geometry. In order to guarantee the absence of these geometric violations, a check is implemented by comparing the simulated geometry of the original contour parallel toolpath to that of the optimized toolpath. By simply storing a bitwise representation of the desired final workpiece, $W_0$ created with the unmodified contour-parallel path, all that is required to avoid a geometric violation is a bitwise comparison between $W_0$ and current simulated geometry $W_i$. If the two bitmaps do not match exactly, a geometric violation has occurred due to the most recent movement of the interpolation point. In this case, the move is decreased and the new toolpath is again simulated and checked against the target workpiece. This process repeats until no geometric violation exists. If such a geometric check is implemented after every parabolic search, then the final workpiece geometry will be identical to the target geometry. In addition to avoiding geometric violations, it is also possible to use this overall strategy to avoid the formation of thin-wall sections in the workpiece. During the geometric check, a scaled-down diameter $D_s$ may be used to create an additional workpiece bitmap, where $D_s$ incorporates a clearance relative to the original tool diameter $D$. A toolpath which avoids a geometric violation using the reduced diameter $D_s$ corresponds to a toolpath which maintains a specific clearance using the actual tool diameter $D$. Assuming that no thin-walled sections exist in the original unmodified toolpath, the bitmap created with the reduced tool diameter will be identical to the bitmap created with the full tool diameter, save with slightly reduced external dimensions. The geometric check algorithm uses these reduced diameter tool bitmaps to check for violations only; the workpiece simulation itself is still performed using the original tool diameter. Using such an algorithm, the computational performance of the iterative improvement algorithm can be evaluated.
approach, the geometric check algorithm may be used to identify and avoid potential clearance issues such as thin-walled geometries as well as actual geometric violations.

Implementation of Iterative Improvement Algorithm. Algorithm 1 summarizes the overall iterative algorithm for improving a given feasible toolpath. The algorithm takes in a set of interpolation points from a given contour-parallel path and modifies the toolpath by iteratively making local modifications that reduce the overall cost function. Algorithms 2 and 3 are subroutines that are called from Algorithm 1. Algorithm 2 describes the steps in the local simulation procedure used to obtain the instantaneous engagement cost for the function cost and Algorithm 3 contains the geometric check procedure used to insure that any movement of the path does not create a geometric violation.

ALGORITHM 1. Iterative Improvement Algorithm Using Local Simulation

Given a set of interpolation points \( \{p_i\} \),

1. Preprocessing.
   - Generate interpolatory spline and set of accompanying simulation points \( \{s_j\} \).
   - Execute global simulation on spline toolpath to obtain cost at each \( s_j \).
   - Store bitwise representation of final (target) workpiece geometry as \( W_0 \).
   - Run global simulation, store cost as \( U_0 \).
   - Set \( n = 1 \) (iteration number).

2. Start of iterative improvement iteration.

2.1. Preprocessing of nearest neighbors to identify critical spline segments for each \( p_i \).
   - Generate Delaunay triangulation for set of control points \( \{p_i\} \).
   - Identify set of nearest neighbors \( \{n_i\} \), for each \( p_i \), from triangulation.
   - Construct the superset of critical nearest neighbors \( \{N_i\} \), for each \( p_i \), where \( \{N_i\} = \bigcup_{n=i-2}^{n+2} \{n_i\} \).
   - Group each \( \{N_i\} \) into path segments \( \{L_m\}_i \), where \( m \) is the number of path segments.

2.2. Local updating of an interpolation point \( p_i \).
   - Select a random \( p_i \) from interpolation points not yet updated.
   - Obtain bracketing trio for quadratic search.
   - Run local simulation (Algorithm 2) for \( p_i \) to obtain local cost \( U_{L,M} \).
   - Set \( p_i = p_i + v_{\text{max}} \hat{n}_i \).
   - Local reinterpolation; Execute local simulation to obtain cost \( U_R \) (Algorithm 2).
   - Set \( p_i = p_i - v_{\text{max}} \hat{n}_i \).
   - Local reinterpolation; Execute local simulation to obtain cost \( U_L \) (Algorithm 2).

Let \( v_M = 0, v_R = v_{\text{max}}, v_L = -v_{\text{max}} \).

Quadatic search to find optimal update distance, \( v^* \).

WHILE \( (v_R - v_L)/(2v_{\text{max}}) > 5\% \):
   - Fit parabola through \( (v_L, U_L), (v_M, U_M), (v_R, U_R) \), find minimizing value \( v^* \).
   - If \( (v^* < v_L) \) and \( (v^* > v_R) \), \( v^* = \min(v_L, v_M, v_R) \).

BREAK
   - Set \( p_i = p_i + v^* \hat{n}_i \).
   - Local reinterpolation; Execute local simulation to obtain cost \( U^* \) (Algorithm 2).
   - Update \( (v_M, U_M) \) to be the point with minimum \( U \) from \( (U_L, U_M, U_R, U^*) \).
   - Update \( (v_R, U_R) \) to be the point to the right of the new \( v_M \).
   - Update \( (v_L, U_L) \) to be the point to the left of the new \( v_M \).

END WHILE

Perform Geometric Check (Algorithm 3).

Mark \( p_i \) as updated.

If all \( p_i \) updated, proceed to Step 2.3, else return to Step 2.2.

2.3. Exit Condition
   - If \( (U_{i-1} - U_i)/U_i < 5\% \) for \( i = n, n - 1 \), and \( n - 2 \),
   - Done, otherwise, \( n = n + 1 \), return to step 2.

ALGORITHM 2. Local Simulation

Given interpolation point \( p_i \), corresponding interpolation segments \( \{L_m\}_i \), and penalty function \( U \).

Simulate all \( \{L_m\}_i \) that come before \( p_i \); do not update engagement angles.

1. All neighboring spline segments corresponding to interpolation points that occur prior to the moving control point are simulated to create the proper geometry for computing the engagement angles. The engagement angles are not updated, as they have not changed.

2. Simulate remaining \( \{L_m\}_i \) chronologically; update engagement angles.
   - All remaining critical neighboring spline segments are simulated. The engagement angles must be updated for these segments, as the tool path geometry has changed.

3. Compute penalty function \( U \) to obtain cost associated with \( p_i \).

ALGORITHM 3. Geometric Check

Given \( \{p_i\}, v^*, W_i, U_n \)

1. Preprocessing.
   - Store \( v_0 = v^* \).

2. Bitwise geometric check.
   - Reinterpolate spline using \( p_i \) modified by \( v^* \), run global simulation, store cost as \( U_i \).
   - Store bitwise representation of workpiece as \( W_i \).
   - If \( (\text{bitcount}((W_0)\text{OR}(W_i)) > 0) \), \( v^* = v^* - 0.2v_0 \).
   - Go back to Step 2.1.

   - If \( U_i > U_{n-1} \), \( v^* = 0 \).
   - Update \( p_i = p_i + v^* \hat{n}_i \).
   - Set \( U_n = U_i \).

Results

Figure 14 illustrates the progression of the iterative improvement algorithm on a representative test geometry. The state of the toolpath is shown at the end of each of the indicated iterations of the algorithm, where an iteration consists of a local updating of each of the control points. As may be observed in the figure, after the first three iterations, the visible changes to the toolpath geometry are relatively minor. This was typical in all of the examples that were tested, demonstrating the ability of the parabolic search-based algorithm to converge to a solution within several iterations.

To demonstrate the efficacy of the iterative improvement algorithm, four scenarios were evaluated for each of two pocket ge-
ometries. The two geometries tested, and the accompanying contour parallel toolpaths appear at the top of Figs. 15 and 16. The four scenarios for each geometry included a relatively high and a relatively low engagement weight for both the spiral-in and spiral-out trajectories. The parameters used in the resulting eight combinations are summarized in Table 1.

Four of the cases from Table 1 have been highlighted in Figs. 15 and 16. In Fig. 15, the results of the iterative improvement algorithm are compared for the case of a spiral-in (Case 1) versus a spiral-out (Case 3) trajectory on a nonconvex pocket geometry. Figure 16 highlights Cases 5 and 6 to demonstrate the effect of varying the respective weights for the curvature and engagement metrics. For all cases, a conventional contour-parallel toolpath was used as the starting point. The target engagement angle for Cases 1–4 was 70° while that for Cases 5–8 was 90°. These were chosen to correspond to intended radial depths-of-cut in the original contour-parallel tool-paths of \( s = 2r/3 \) and \( s = r \). Interpolation was carried out using the previously described Shirman-Sequin interpolation procedure [39] with parameters \( D = 0.0 \), \( B = 1.0 \), \( C = 0.5 \). The tool-path trajectories in the figures represent the centerline motion of the tool. To satisfy the geometric requirements, the peripheral (boundary) loop of the trajectory is not permitted to move. In other words, the optimized and initial tool-paths must machine identical volumes. The average time for the overall optimization process for each case was 11 min on a 750 MHz PC with a Pentium-P3 processor. With faster processors (3 GHz) readily available at moderate cost, and computational capabilities rapidly increasing, we believe that the computationally intensive iterative improvement approach will be practical for critical high-speed machining applications.

Plots of the instantaneous curvature and engagement before and after optimization are presented in Fig. 17 for each of the two case studies. Of immediate note is the reduction in the variation of the engagement angle after optimization. Although significant variations do remain present in the optimized trajectories, the number and magnitude of these fluctuations has been significantly reduced. As expected, the sharp peaks in curvature have been greatly reduced in the improved toolpath, and the toolpaths appear visibly smooth.

Figure 15, Cases 1 and 3, illustrates the difference between optimized spiral-in and spiral-out trajectories. From the figure, it is possible to consider explicitly the inherent trade-off between the two approaches. In the spiral-in approach, a large slotting pass is necessary around the periphery. This appears on the initial portion of the engagement plot. After the culmination of this slotting pass, the engagement variations are relatively small. In the spiral-out
trajectory, the initial slotting is nearly nonexistent, as the path initiates with a spiraling pattern. It is not possible to completely eliminate spikes in the engagement due to the geometric constraints; as a result, numerous cornering spikes appear in the final pass around the periphery.

In Fig. 16, the comparison between Cases 5 and 6 illustrates the effect of varying the respective weights of the two metrics in the cost function. The higher curvature weight scenario correspondingly lower engagement weight tends to smooth out some of the sharp turns and general waviness of the higher engagement weight toolpath, at the expense of a moderate increase in engagement variation. We have not pursued the development of a precise methodology for weight selection, as the evaluation of overall tool-path quality will typically be rather subjective. For this reason, the weights may be manually manipulated by an end-user to achieve a desired balance between the various objectives. The weights used for all test cases are given in Table 1. In order to avoid unnecessary disengagement and re-entry, a discrete re-entry weight was incorporated into all of the case studies whose magnitude was 5% of the total cost.

From the graphical illustrations in Figs. 15 and 16, it may be observed that the optimized trajectories are both visibly smoother, as well as geometrically feasible (the machined area is equivalent to that of the initial contour-parallel tool-path). The optimized tool-paths are also slightly shorter (approximately 4–6%) than the
Table 1  Metric weights, simulation parameters, and improvement in target engagement for the eight test cases

<table>
<thead>
<tr>
<th>Case</th>
<th>Geometry</th>
<th>Spiral Type</th>
<th>$w_{\text{engagement}}$</th>
<th>$w_{\text{curvature}}$</th>
<th>$R$ (bits)</th>
<th>$\theta_s$ (deg)</th>
<th>$\Delta s/R$</th>
<th>Engagement Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No Island</td>
<td>In</td>
<td>0.99</td>
<td>0.01</td>
<td>50</td>
<td>70</td>
<td>0.2</td>
<td>30.7</td>
</tr>
<tr>
<td>2</td>
<td>No Island</td>
<td>In</td>
<td>0.64</td>
<td>0.36</td>
<td>50</td>
<td>70</td>
<td>0.2</td>
<td>23.4</td>
</tr>
<tr>
<td>3</td>
<td>No Island</td>
<td>Out</td>
<td>0.85</td>
<td>0.15</td>
<td>50</td>
<td>70</td>
<td>0.2</td>
<td>11.3</td>
</tr>
<tr>
<td>4</td>
<td>No Island</td>
<td>Out</td>
<td>0.64</td>
<td>0.36</td>
<td>50</td>
<td>70</td>
<td>0.2</td>
<td>9.1</td>
</tr>
<tr>
<td>5</td>
<td>Pocket w/Island</td>
<td>In</td>
<td>0.97</td>
<td>0.03</td>
<td>50</td>
<td>90</td>
<td>0.2</td>
<td>11.7</td>
</tr>
<tr>
<td>6</td>
<td>Pocket w/Island</td>
<td>In</td>
<td>0.87</td>
<td>0.13</td>
<td>50</td>
<td>90</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>7</td>
<td>Pocket w/Island</td>
<td>Out</td>
<td>0.92</td>
<td>0.08</td>
<td>50</td>
<td>90</td>
<td>0.2</td>
<td>11.9</td>
</tr>
<tr>
<td>8</td>
<td>Pocket w/Island</td>
<td>Out</td>
<td>0.72</td>
<td>0.28</td>
<td>50</td>
<td>90</td>
<td>0.2</td>
<td>4.1</td>
</tr>
</tbody>
</table>
original tool-paths. Figure 18 contains histograms of the engagement angle distribution as a fraction of tool-path length (expressed as a percentage) both before and after optimization for the case studies of Figs. 15 and 16. The ideal distribution would be a single bar representing 100% of the toolpath being located at the target engagement (70° or 90°). The distribution for the original unmodified toolpaths shows a bimodal nature, with only 40–50% of the toolpath within an engagement range of ±10° of the target engagement. After application of the iterative improvement algorithm, the distribution has shifted towards the target values to varying degrees for the different scenarios. For example, Case 1 demonstrates the most significant improvement in the engagement angle distribution. In the contour-parallel toolpath for Case 1, less than 40% of the toolpath was near the target engagement value of 70° (±10°). After optimization, approximately 70% of the toolpath falls within this target range of 60–80°. We report this improvement in the fraction of toolpath within the target region in the right-most column of Table 1 for all of the eight cases—39.3% for Case 1. In the remaining seven cases, the percentage of the toolpath in the target engagement range has increased between 0.2 and 23.4%. For the high engagement weight cases, percentage of the toolpath in the target engagement range has increased between 11.3 and 30.7%.

Conclusions
Despite the tremendous daily economic impact of 2D CNC milling operations, little effort has been spent in pursuit of alternatives to the standard contour-parallel approach. Improved toolpath generation techniques offer the potential for gains in production efficiency with negligible capital expenditure. While it may not be possible to claim global optimality of the resulting toolpath trajectories, the case studies demonstrate that significant gains can be realized through the metric-based approach.

Metrics to reduce the path curvature, as well as variations in instantaneous cutter engagement have been used to drive the iterative improvement algorithm. As instantaneous cutter engagement is dependent on the history of the toolpath, an efficient pixel-based local simulation technique has been developed to explore changes to the engagement cost function with respect to local changes in the spline geometry. Variations in instantaneous cutter engagement are a significant concern from a process stability and efficiency perspective. It is apparent from the engagement plots that the magnitude and number of problematic instantaneous

---

*The histograms reflect only those regions of the toolpath that were free to be updated during optimization. For example, the initial slotting pass for the spiral-in cases would be excluded, as it would be identical before and after optimization.
engagement variations may be reduced. Further, smooth trajectories have been generated that satisfy the geometric constraints and remove precisely the same material as the contour-parallel trajectory.

Acknowledgments

The first author gratefully acknowledges the support of the University of Illinois Research Board. The second and third author gratefully acknowledge the support of the National Science Foundation through award DMI-99-84214. The authors would also like to thank Dr. Bieterman and Dr. Sandstrom of the Boeing Corporation for motivating the consideration of acceleration in the toolpath generation problem.

Nomenclature

- $a$: distance from interpolation point $P_{i-1}$ to $P_i$
- $\hat{a}$: unit direction vector from $P_{i-1}$ to $P_i$
- $b$: distance from $P_{i+1}$ to $P_i$
- $\hat{b}$: unit direction vector from $P_{i+1}$ to $P_i$
- $B$: adjustable bulge parameter for toolpath interpolation
- $B(u)$: point on Bezier segment evaluated at $u$
- $C$: adjustable continuity parameter for the toolpath representation
- $(c_0, c_1, c_2, c_3)$: Bezier segment control points
- $c_a$: component of the normal $\hat{n}_i$ in the $\hat{a}$ direction
- $c_b$: component of the normal $\hat{n}_i$ in the $\hat{b}$ direction
- $(c_{L_i}, c_{R_i})$: Bezier segment control points associated with $P_i$
- $D$: adjustable shape parameter for the toolpath representation
- $d$: tool diameter
- $(\hat{d}_{L_i}, \hat{d}_{R_i})$: derivative vectors of the toolpath at $P_{i-1}$ and $P_{i+1}$
- $d_s$: scaled-down tool diameter used in simulation for maintaining clearance
- $(k_a, k_b, k'_a, k'_b)$: correction factors applied for collinear segments in spline interpolation
- $L_m$: line segments formed from collection of points $N_i$
- $M_i$: set of neighboring interpolation points for each $P_i$
- $M'_i$: set of neighboring interpolation points for each $P'_i$
- $\hat{n}_i$: normal of toolpath curve at $P_i$
- $\hat{n}_a$: unit normal to $\hat{a}$
- $\hat{n}_b$: unit normal to $\hat{b}$
- $N_i$: set of interpolation points in critical line segments for each $P_i$
- $p$: exponent in penalty function
- $P_i$: interpolation point on the toolpath representation
- $r$: tool radius
- $\Delta s$: chord-length distance between two consecutive simulation points
- $S_n$: point on the toolpath representation used in engagement simulation
- $T_i$: total cost of total toolpath based on the weighted metrics
- $u$: parameter of Bezier spline
- $U$: total cost, value of penalty function

Fig. 18 Histogram of the engagement distribution for the cases of Figs. 16 and 17
\[ v = \text{distance in the } \hat{n}_i \text{ direction that the point } P_i \text{ is moved} \]
\[ w_j = \text{scalar weight on } j\text{th cost metric} \]
\[ W_0 = \text{bitwise representation of desired workpiece} \]
\[ W_j = \text{current representation of workpiece from engagement simulation} \]
\[ Z = \text{minimum diametrical clearance maintained between two tool passes} \]
\[ z_{ij} = \text{target value for each cost metric} \]
\[ \theta = \text{instantaneous cutter engagement} \]

References


