IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2018, accepted June 13, 2018, date of publication June 19, 2018, date of current version July 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2848938

A GPU-Accelerated Approach for Collision Detection and Tool Posture Modification in Multi-Axis Machining

JING WANG, MING LUO, (Member, IEEE), AND DINGHUA ZHANG
Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an 710072, China

Corresponding author: Ming Luo (luoming@nwpu.edu.cn)

This work was supported by the China Major National Science and Technology Project under Grant 2015ZX04001202.

ABSTRACT Collision detection and avoidance between solid bodies are one of the most important problems in path planning for robotics machining or multi-axis machining. While planning toolpath for multi-axis milling, accurate collision detection is usually time-consuming among complex solid bodies in the computer environment. Furthermore, how to avoid the collision automatically within limited space in the path planning stage still needs lots of experience. To this end, this paper presents a general collision detection and tool posture automatic adjustment approach for the multi-axis milling process. First, by analyzing the contact state of the tool-workpiece, the calculation model of the interference quantity is determined. A unified tool constraint mathematical model based on the interference quantity and the interference type is established. Second, three types of tool adjustment strategies are constructed, the sequential quadratic programming method is used to solve the model and the graphics processing unit-based high-performance computing technology is employed to accelerate the solution process. Finally, the developed method is validated for automatic collision and tool posture adjustment in the five-axis milling of a blisk. The presented method can be integrated into commercial CAD/CAM software for rapid tool collision detection and tool orientation modification.

INDEX TERMS CAM, collision avoidance, high performance computing, GPU, machining, manufacturing, tool posture modification.

I. INTRODUCTION

Fast collision detection and avoidance between solid bodies [1], [2] are common problems in the navigation [3], missile control [4] and multi-axis machining [5], [6] field. In manufacturing industry, while machining of complex parts, such as aero-engine blisks and compressor blades, collisions and interference may occur and lead to damage to the part due to their complex shape. Tool interference analysis and control is always the research interest in multi-axis machining. After decades of development, it has made great progress [7]–[10]. Tool collision can be classified into two types: local gouging [11] and global interference [12]. Local gouging is also known as the tool bottom interference, the main task is to analyze interference between the cutting tool and machined surface near the cutting contact point, it can be avoided by changing tool orientation or moving the tool along its orientation [5]. Global interference is also known as the collision interfere, the main task is to analyze potential collision between the cutting tool and the machined surface, neighboring surface as well as fixtures. The main developed methods include distance based method [13], convex hull method [14], mapping method [15] and bounding box based methods [16]. However, these developed methods are usually slow and time-consuming in computing and is difficult to be fully integrated into the commercial software.

With the development of computing technology, the graphics processing unit (GPU) computing based technology have demonstrated its power in the calculation with large amounts of data [17]. GPU uses massively parallel architecture, which uses lots data parallel logic processing unit, it is now commonly used in the path planning [18] and collision avoidance problems [19]. Since the computation cost in tool posture and collision detection is very high in the tool path planning and simulation stages for multi-axis machining,

it is also employed to deal with CAD/CAM problems. Bi *et al.* [20] developed a GPU based method to generate collision-free and orientation-smooth tool paths for five-axis NC finishing machining of complicated shapes, accessibility cones of cutter location points as well as tool orientation optimization for a ball-end cutter are accelerated with GPU. Hsieh and Chu [21] applied GPU technology to estimate the error in the 5-axis flank milling optimization process based on the particle swarm optimization, the error amount is simultaneously calculated by the parallel processing units of GPU. Morell-Giménez *et al.* [22] used GPUs to improve computation in tool path generation. Abecassis *et al.* [23] applied GPU technology to speed-up Z-buffer or N-buffer machining simulations, especially for collisions detection between the tool and the part, and their results show significant performance improvement on the computation time. Balabokhin and Tarbutton [24] applied GPUs to reduce the simulation time in parallel tool path generation for an arbitrary milling zone on a free-form surface and a generalized cutter. Lynn *et al.* [25] used GPUs to support the development of a voxelized CAM package that allows for rapid toolpath generation for complex parts.

The above research has shown the great potential of applying GPU computing technology in CAD/CAM, especially for path planning. However, little research has been done on tool posture modification when the collision occurs in path planning process. This paper presents a general tool posture modification method to avoid collision between the cutter and workpiece in the multi-axis machining process, and the calculation is accelerated with the GPU parallel computing technology. The rest of the paper are organized as follows: the contact condition between the tool and the workpiece is analyzed and modelled in Section II, cutting tool constraints model is established in Section III, detailed constraints are described in Section IV, solutions of the constraint model based on GPU computing is given in Section V. Validation of the proposed method is given in Section VI with discussions. Finally, conclusions are given in Section VII.

II. TOOL-WORKPIECE CONTACT ANALYSIS AND MODELING

In the machining process, the relative position relationship between the cutting tool and the workpiece is complex. Therefore, establishing the corresponding relation model for processing tool interference detection and tool posture modification is a must. Modeling and analysis of tool-workpiece contact condition will be discussed in this section.

A. THE UNIVERSAL CUTTING TOOL MODEL

The swept envelope of a cutting tool plays an important role in cutter location calculation, tool axis control, and interference detection. When a cutting tool is rotating, its envelope is a revolution body and can be expressed in the form of tool-axis $\boldsymbol{l}$ and its generatrix $\mathscr{f}$, as shown in Fig. 1. To simplify the calculation, a tool coordinate system $O_{t,x_t y_t z_t}$ is established,

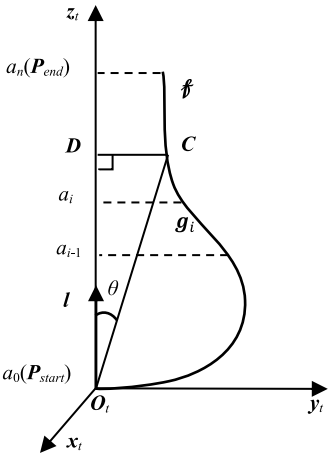

FIGURE 1. The universal cutting tool model.

the tool axis coincides with the axis $\boldsymbol{l}$ and $\boldsymbol{z}_t$ axis. The generatrix of the tool envelop can be expressed by a piecewise function below:

$$\mathscr{f} = \begin{cases} \boldsymbol{g}_i, & z \in [a_0, a_1] \\ \boldsymbol{g}_i, & z \in [a_{i-1}, a_1], \quad \boldsymbol{g}_i \\ \boldsymbol{g}_i, & z \in [a_{n-1}, a_n] \end{cases} = (y(t_i)\cos\varphi, y(t_i)\sin\varphi, z(t_i)) \tag{1}$$

Where $[a_{i-1}, a_i]$ represents the value range of function g_i along z_t axis, a_0 and a_n denote the two endpoints of the tool axis z_t, $h(t_i) = \{y(t_i), z(t_i)\}$ represents the generatrix function, and φ is the angle around the tool axis with range $-\pi \leq \varphi < \pi$.

B. TOOL-WORKPIECE CONTACT ANALYSIS

The contact conditions between the cutting tool and the workpiece can be classified into two types: non-interference and interference. When there is no interference, the tool locates outside of the workpiece, therefore no overcut will occur and there is no need to modify the tool posture. Conversely, when interference occurs, since the tool cuts into the workpiece, overcut will occur. It is necessary to modify the tool posture to avoid interference. Depending on the location of the interference between the tool and the workpiece, three are three types of interference: (1) Rear interference: it occurs at the bottom of the cutting tool. (2) Local interference: it occurs between the cutter flutes and the workpiece. (3) Global interference: there is interference between the cutter shank and the workpiece.

C. TRANSFORMATION BETWEEN THE WORKPIECE COORDINATE SYSTEM AND TOOL COORDINATE SYSTEM

In the machining process, the workpiece is usually complex and the profile of the workpiece is always changing. Since the cutter is almost the same, it is suitable to do the analysis of interference in the tool coordinate system. Define a workpiece coordinate $O_{w,x_w y_w z_w}$, let $\boldsymbol{p}$ be the cutter contact point,

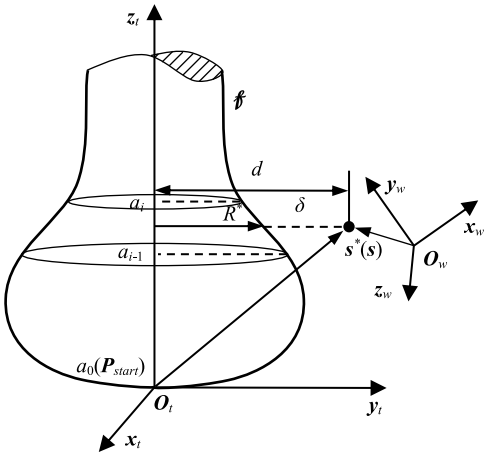

FIGURE 2. The cutting tool coordinate.

$\boldsymbol{l}$ be the corresponding tool orientation, $\boldsymbol{C}$ be the cutter location point, $\boldsymbol{n}$ be the face normal at $\boldsymbol{p}$. Let z_t axis of cutting tool coincide with $\boldsymbol{l}$, as shown in Fig.2. For a point $\boldsymbol{s}$ in the workpiece coordinate system, it is marked as $\boldsymbol{s}^*$ in the tool coordinate system. Then coordinate transformation relationship can be expressed as:

$$\boldsymbol{s^*} = (\boldsymbol{s} - \boldsymbol{O_t})\,\boldsymbol{M} \tag{2}$$

where $\boldsymbol{M}$ is the transformation matrix:

$$\boldsymbol{M} = \begin{bmatrix} M_{xx} & M_{yx} & M_{zx} \\ M_{xy} & M_{yy} & M_{zy} \\ M_{xz} & M_{yz} & M_{zz} \end{bmatrix} \tag{3}$$

After transforming points from the workpiece coordinate system to the tool coordinate system, it is easy to calculate the interference between the cutting tool and the workpiece.

D. INTERFERENCE DISTANCE CALCULATION

Once the interference occurs, the value should be calculated before automatic adjustment. For a universal tool described by the axes and its generatrix, the distance d from a point $\boldsymbol{s}^* = (x^*, y^*, z^*)$ to the cutter rotation axis can be expressed as:

$$d = \sqrt{(x^*)^2 + (y^*)^2} \tag{4}$$

By substituting $z(t_i) = z^*$ into the generatrix equation $h(t_i) = \{y(t_i), z(t_i)\}$, the corresponding tool radius R^* for point $\boldsymbol{s}^*$ can be obtained. Define the interference value on the tool elevated section where point $\boldsymbol{s}^*$ locates as δ, then the interference value can be derived as:

$$\delta = d - R^* \tag{5}$$

Let $\varepsilon > 0$ be the allowable interference tolerance, then if $\delta > -\varepsilon$ hold, the point $\boldsymbol{s}^*$ on the workpiece does not interfere with the tool; otherwise, interfere occurs. The interference type can be determined by judging projection position of $\boldsymbol{s}^*$ on the axis z_t.

Automatically Programmed Tools (APT) are the most commonly used tools in CNC machining, its definition is shown in Fig.3. It can be divided into three segments, $[z_0, z_2]$, $(z_2, z_3]$, $(z_3, z_4]$ and z_0, z_1, z_2, z_3, z_4 are expressed as:

$$\begin{cases} z_0 = 0 \\ z_1 = R\tan\alpha \\ z_2 = (R + r\sin\alpha)\tan\alpha \\ z_3 = R\tan\alpha + \dfrac{r}{\cos\alpha} - \sin\beta \\ z_3 = R\tan\alpha + \dfrac{r}{\cos\alpha} - \sin\beta + L \end{cases} \tag{6}$$

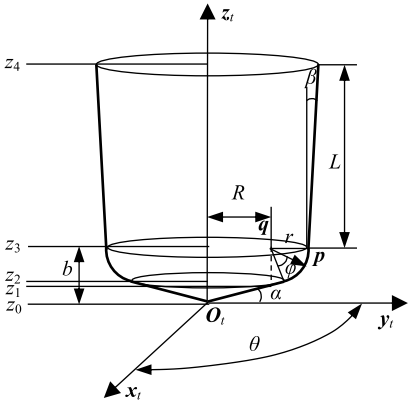

FIGURE 3. Geometry of an API tool.

For an APT tool, it can be considered as a special case of universal cutting tool defined in the previous section, the interference type can be defined as follows for a point $\boldsymbol{s}^* = (x^*, y^*, z^*)$:

(1) If $z^* \in [z_0, z_2]$, $R^* = z^*\tan\alpha$, the interference is $\delta = d - R^*$, the interference is the tool bottom rear interference.

(2) If $z^* \in [z_2, z_3]$, the interference is the curvature interference.

(3) If $z^* \in [z_3, z_4]$, the interference is the collision interference.

(4) If $z^* \notin [z_3, z_4]$, no interference will happen.

Although the above description can be applied for interference detection and calculation between the workpiece and tool for most cases, it cannot get accurate results for some special cases. As shown in Fig. 4, when the tool generatrix functions include $z = a$, $y \in [y_1, y_2]$, for any point $\boldsymbol{s}^*$ on the workpiece locating near the generatrix, i.e., $z^* \in [a - \varepsilon, a + \varepsilon]$, $y^* \in [y_1, y_2]$, then interference between the point and the tool exists and the interference type can be determined by the value of z^*. Conversely, if $z^* \notin [a - \varepsilon, a + \varepsilon]$ or $y^* \notin [y_1, y_2]$, then there is no interference between the point $\boldsymbol{s}^*$ and the cutting tool within that section of generatrix. However, interference may occur with other parts of the tool, hence detection with other generatrix sections is needed.

E. OVERVIEW OF THE AUTOMATIC COLLISION INTERFERENCE DETECTION AND MODIFICATION

Due to the shape complexity of the workpiece and the tool, it is difficult to apply the analytical method for interference

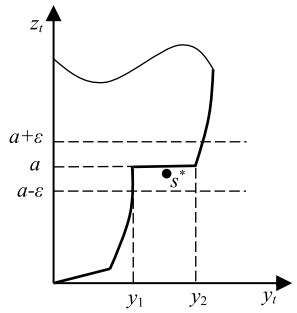

FIGURE 4. Interference detection for $z = a$.

calculation and adjustment. To get a common and feasible automatic approach, discrete points form the workpiece are used in this research and the flow chart for interference calculation and tool orientation modification is shown in Fig. 5. According to the above-described principles, the interference status and types between the tool and workpiece can be easily figured out. However, the calculation of interference type and a series of discrete points on workpiece and fixture is necessary with the above method. Therefore, the computation cost is very high. To overcome this problem, the GPU based parallel computing technology is applied for achieving rapid calculation of the interference value and it is discussed in the following sections.

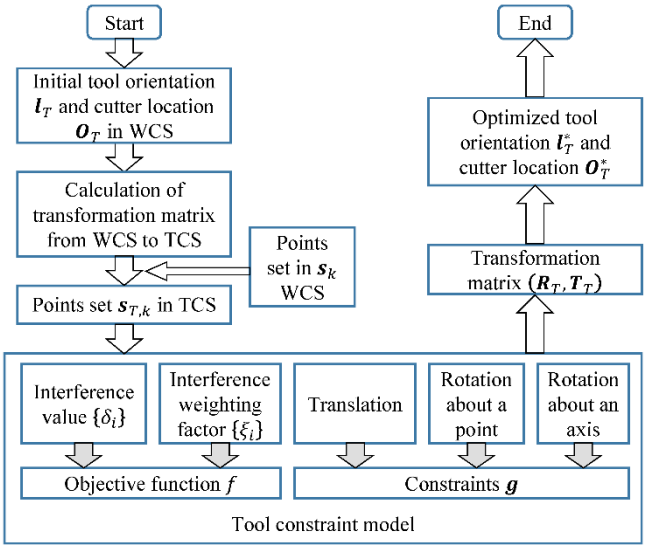

FIGURE 5. Flow chart for interference calculation.

III. CUTTING TOOL CONSTRAINTS MODEL

In the machining process, the interference type between different parts of the tool and the workpiece surface is different, therefore their influence on the global interference calculation is different. To solve this problem, an optimization model should be established to find a tool adjustment strategy. In this study, ξ_k, a weighting factor of interference is introduced here. To judge the interference between the cutter and the workpiece, transforming potential interference point set $\{s_k | k = 1, 2, \cdots, N\}$ from the workpiece coordinate system to the tool coordinate system and a new point set $\{s_{T,k} | k = 1, 2, \cdots, N\}$ is obtained, where N is the number of discrete points. The transformation process can be expressed by the following equations:

$$s_{T,k} = (s_k - O_T) M \tag{7}$$

where M is the transformation matrix from the workpiece coordinate to the tool coordinated system, O_T is the cutter location coordinate.

In the multi-axis machining process, the tool can do any rigid transformation including translation and rotation to avoid the collision. Let R_T represent rotation matrix about an arbitrary axis, T_T represents the translation matrix, s_k^* represent the point after the transformation. Then the transformation can be expressed as:

$$s_k^* = (s_k - O_1) M_1 \tag{8}$$

where

$$\begin{cases} M_1 = MR_T \\ O_1 = O_T - T_T M_1^{-1} \end{cases} \tag{9}$$

Let l_T be the initial tool orientation with interference, O_T be the corresponding cutter location. After the above transformation, the new tool orientation l_T^* and cutter location O_T^* are

$$\begin{cases} l_T^* = lM_1 M^{-1} \\ O_T^* = O_1 \end{cases} \tag{10}$$

Usually, the rotation R_T can be decomposed into rotation angles about three axes x_t, y_t, z_t as $\omega_x, \omega_y, \omega_z$ respectively; and the translation T_T can be decomposed into translation along three axes x_t, y_t, z_t as Δx, Δy and Δz, respectively. Therefore, the above six parameters $g = (\omega_x, \omega_y, \omega_z, \Delta x, \Delta y, \Delta z)^T$ can be used to constraint the tool posture for interference avoiding. Furthermore, according to the previous interference analysis in Section II, the weighted sum of squares function in terms of interference value is defined here as the optimization objective function for the tool posture refinement.

$$f(g) = \sum_{k=1}^{N} \xi_k \delta^2 (s_k) \tag{11}$$

In order to avoid interference in the machining process, the tool needs to be adjusted to the appropriate posture, so that the minimum $f(g)$ can be get. Then the following constraint model can be defined

$$\begin{cases} \min f(g) = \sum_{k=1}^{N} \xi_k \delta^2 (s_k) \\ s.t.\ g \in \Omega \end{cases} \tag{12}$$

where, $g \in \Omega$ is the tool posture constraints, Ω is a feasible set or feasible region, each set represents a feasible tool posture, ξ_k is the weighting factor of interference and it is related with

the interference positon between the cutter and the machined surface.

IV. ESTABLISHMENT OF CONSTRAINTS

To avoid interference between the tool and workpiece, there are several adjustment methods including translation and rotation of the tool. In this section, three types of tool adjustment method and their constraints will be discussed. In multi-axis milling process, there are totally six degrees of freedom, including three translation, Δx, Δy, Δz, and three rotation, ω_x, ω_y, ω_z. While adjusting the tool posture to avoid interference, the corresponding constraints in terms of translation or rotation within a certain range can be expressed as:

$$\begin{cases} \hbar_i(\boldsymbol{g}) = 0, & i = 1, 2, \cdots, p \\ c_j(\boldsymbol{g}) \le 0, & j = 1, 2, \cdots, q \end{cases} \tag{13}$$

where p and q are the corresponding constraints number.

A. ADJUSTMENT WITH TRANSLATION

In some cases, the tool can be adjusted by translation to avoid interference. That is, the tool can be moved along x_t, y_t and z_t for distance Δx, Δy and Δz, repectively. Then the constraints can be expressed as:

$$\begin{cases} \omega_x = 0 \\ \omega_y = 0 \\ \omega_z = 0 \\ dx_{min} \le \Delta x \le dx_{max} \\ dy_{min} \le \Delta y \le dy_{max} \\ dz_{min} \le \Delta z \le dz_{max} \end{cases} \tag{14}$$

Usually, adjustment of tool position with only translation is used in the milling process with fixed tool orientation, such as three-axis milling.

B. ROTATION ABOUT A SPECIFIC POINT

Suppose a point $\boldsymbol{s}$ in the tool coordinate system is rotated about point $\boldsymbol{O}_c$ and get a new point $\boldsymbol{s}_{tr}$:

$$\boldsymbol{s}_{tr} = \boldsymbol{s}\boldsymbol{R}_0 + \boldsymbol{T}_0 \tag{15}$$

where $\boldsymbol{R}_0$ is the rotation matrix and is express as:

$$\boldsymbol{R}_0 = \boldsymbol{R}_x(\omega_x)\boldsymbol{R}_y(\omega_y)\boldsymbol{R}_z(\omega_z) \tag{16}$$

and rotation matrices about x_t, y_t and z_t are

$$\boldsymbol{R}_x(\omega_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega_x & \sin\omega_x \\ 0 & -\sin\omega_x & \cos\omega_x \end{bmatrix} \tag{17}$$

$$\boldsymbol{R}_y(\omega_y) = \begin{bmatrix} \cos\omega_y & 0 & -\sin\omega_y \\ 0 & 1 & 0 \\ \sin\omega_y & 0 & \cos\omega_y \end{bmatrix} \tag{18}$$

$$\boldsymbol{R}_z(\omega_z) = \begin{bmatrix} \cos\omega_z & \sin\omega_z & 0 \\ -\sin\omega_z & \cos\omega_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The translation matrix $\boldsymbol{T}_0$ can be expressed as:

$$\boldsymbol{T}_0 = \boldsymbol{O}_c(\boldsymbol{I} - \boldsymbol{R}_0) \tag{20}$$

As shown in Fig.6, $\boldsymbol{s}_{new}$ is the projection of point $\boldsymbol{s}_{tr}$ on curve $\mathcal{C}$, which is the intersection of the plane Π and the cutter profile, the plane Π is perpendicular to the cutter axis. The point $\boldsymbol{s}_{new}$ is the new cutter contact point.

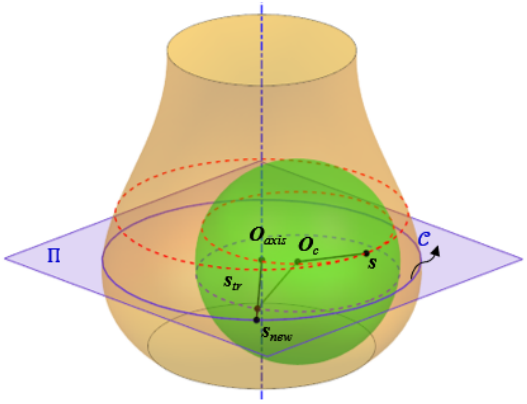

FIGURE 6. Tool rotation about a point.

Let $h(t) = \{y(t), z(t)\}$ be the generatrix function, and then the new cutter contact point is derived as:

$$\boldsymbol{s}_{new} = y(t)\frac{\boldsymbol{s}_{tr} - \boldsymbol{O}_{axis}}{\|\boldsymbol{s}_{tr} - \boldsymbol{O}_{axis}\|} + \boldsymbol{O}_{axis} \tag{21}$$

where $\boldsymbol{O}_{axis} = (0, 0, z(t))$, and $z(t)$ is the same value with $\boldsymbol{s}_{new}$ in z_t direction.

While using this tool adjustment strategy, both the translation and rotation matrices will be changed and the constraints are:

$$\begin{aligned} \hbar(\boldsymbol{g}) &: \begin{bmatrix} \Delta x & \Delta y & \Delta z \end{bmatrix}^T - \boldsymbol{T}_0 - \boldsymbol{s}_{new} + \boldsymbol{s}_{tr} = \boldsymbol{0} \\ c(\boldsymbol{g}) &: \begin{cases} \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T - \begin{bmatrix} \pi & \pi & \pi \end{bmatrix}^T \le \boldsymbol{0} \\ \begin{bmatrix} \pi & \pi & \pi \end{bmatrix}^T - \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T \le \boldsymbol{0} \end{cases} \end{aligned} \tag{22}$$

In practice, there are three types of tool posture adjustment: (a) The cutter rotates about $\boldsymbol{O}_{axis}$. This is the general case and strict constraint should be constructed based on above analysis. (b) The cutter rotates about the cutter contact point, and new cutter contact can be obtained with only rotation, thus $\boldsymbol{T}_0 = \boldsymbol{0}$. (c) The cutter rotates about the tool center. In this case, translation may be needed after rotation since the cutter may not contact with the workpiece. For ball end cutters, the third strategy is usually used to avoid the interference.

C. ROTATION ABOUT A SPECIFIC AXIS

Consider an axis in the space defined by point $\boldsymbol{A}(x_A, y_A, z_A)$ and point $\boldsymbol{B}(x_B, y_B, z_B)$, a point $\boldsymbol{s}$ rotates about the defined axis about angle γ and get a new point $\boldsymbol{s}_{new}$, then it is expressed as:

$$\boldsymbol{s}_{new} = \boldsymbol{s}\boldsymbol{R}_{AB}(\gamma) + \boldsymbol{T}_{AB} \tag{23}$$

where $\boldsymbol{R}_{AB}$ and $\boldsymbol{T}_{AB}$ are the rotation and translation matrix, respectively. They have the following expression:

$$\begin{cases} \boldsymbol{R}_{AB}(\gamma) = \boldsymbol{R}_x(\omega_x)R_y(\omega_y)\boldsymbol{R}_z(\omega_z)R_y(-\omega_y)\boldsymbol{R}_x(-\omega_x) \\ \boldsymbol{T}_{AB} = \boldsymbol{T}_A(I - \boldsymbol{R}_{AB}(\gamma)) \end{cases} \tag{24}$$

where

$$\boldsymbol{T}_A = \boldsymbol{A} - \boldsymbol{O}_t$$

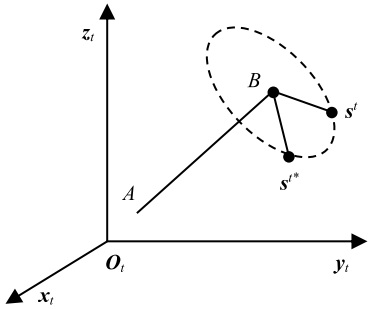

FIGURE 7. Tool rotation about an axis.

Therefore, the constraints in this case can be summarized as:

$$\begin{aligned} &\hbar(\boldsymbol{g}): \left[\Delta x \ \Delta y \ \Delta z\right]^T - \boldsymbol{T}_{AB} = \boldsymbol{0} \\ &c(\boldsymbol{g}): \begin{cases} \gamma - \pi \le 0 \\ \pi - \gamma \le 0 \end{cases} \end{aligned} \tag{25}$$

This kind of strategy is mainly used for the adjustment of the lead angle and tilt angle to refine the tool posture.

V. SOLUTION OF THE CONSTRAINT MODEL

The constraint model can be converted to the general form of nonlinear constrained optimization problems.

$$\begin{cases} \min \quad f(\boldsymbol{g}) \\ s.t \quad \hbar_i(\boldsymbol{g}) = 0, \quad i \in E = \{1, \ldots, l\} \\ \qquad c_j(\boldsymbol{g}) \ge 0, j \in I = \{1, \ldots, m\} \end{cases} \tag{26}$$

where $f(\boldsymbol{g})$ is a weighted sum of squared interference objective function, $\hbar(\boldsymbol{g})$ and $c(\boldsymbol{g})$ are the tool posture adjustment process constraints. An effective method to solve the above problems is the sequential quadratic programming (SQP method), the main idea of this approach is to convert the above problem into a series of sub-quadratic programming problems. Each subproblem can determine a descent direction, by reducing the measurement function to get the step length. Repeat these steps can obtain the final solution of the optimization problem.

A. SEQUENTIAL QUADRATIC PROGRAMMING

In order to convert the problem described in Eq. (26) into a quadratic programming subproblems, the nonlinear constraints are approximated by linear constraints, while the Hesse matrix of the new secondary objective function is a definite approximation of the original Hesse matrix of the Lagrange function, then we can get the following sub-quadratic programming problems.

$$\begin{cases} \min \nabla f(g_k)^T d + \dfrac{1}{2} d^T B_k d \\ s.t \quad \hbar_i(g_k) + \nabla \hbar_i(g_k)^T d = 0, \ i \in E = \{1, \ldots, l\} \\ \qquad c_j(g_k) + \nabla c_j(g_k)^T d \ge 0, \ j = \{1, \ldots, m\} \end{cases} \tag{27}$$

Where, B_k is the positive definite approximation of $\nabla_g^2 L(g_k, \lambda_k)$. By solving the above optimization problem, the solution d_k and the corresponding Lagrange multiplier λ_k can be obtained. Basic steps for solving the nonlinear constraint optimization problem with SQP are listed below:

(1) Set the initial value. Let $g_1 \in \boldsymbol{R}^n$, and $B_k \in \boldsymbol{R}^{n \times n}$ be a symmetric positive definite matrix, select control tolerance larger than zero. Calculate $\nabla f(g_1)$ and let $k = 1$.

(2) Solve the sub-problem. Solve the problem and get corresponding solution d_k and Lagrange multiplier λ_k.

(3) Conduct the linear search. Select a measurement function $W(g, u)$, determine the search step length ϱ_k. Let $g_{k+1} = g_k + s_k$, and $s_k = \varrho_k d_k$. If g_{k+1} satisfy the termination condition below, let $g^* = g_{k+1}$ and terminate the search. Otherwise, execute step 4.

$$\begin{cases} \displaystyle\sum_{j=1}^{l} \left|\hbar_j(g_{k+1})\right| + \sum_{j=1}^{m} \left|\min\left\{0, c_j(g_{k+1})\right\}\right| \le \varepsilon_1 \\ \left|f(g_{k+1}) - f(g_k)\right| \le \varepsilon_2 \\ \left\|\nabla_x L(g_{k+1}, \lambda_k)\right\| \le \varepsilon_3 \left\|\nabla f(g_{k+1})\right\| \\ \qquad or \ \left\|\nabla_x L(g_{k+1}, \lambda_k)\right\| \le \varepsilon_4 \end{cases} \tag{28}$$

Generally, $\varepsilon_1 = 0.01$, $\varepsilon_2 = 0.01$, $\varepsilon_3 = 10^{-5}$, $\varepsilon_4 = 10^{-3}$.

(4) Refine B_k and get B_{k+1}, let $k = k+1$ and go to step (2).

While calculating $f(g_k)$ in the above steps, repeated computation is needed for large amounts of data, resulting in a heavy computation burden for the CPU. To improve the computation efficiency, the repeated computation can be done on the GPU.

B. DETERMINATION OF SEARCH STEP ϱ_k

In the searching process, a measurement function $W(g, u)$ is established to let the iteration reach the feasible region. The function contains the objective function and constraint function and is defined as:

$$W(g, u) = f(g) + \sum_{i \in E} \mu_i \left|\hbar_i(g)\right| + \sum_{j \in I} \mu_j \max\left\{0, c_j(g)\right\} \tag{29}$$

where $\mu_{i(j)}$ is a penalty factor and can be determined by Lagrange multiplier, that is

$$\mu_{i(j)}^{(k)} = \begin{cases} \left|\lambda_{i(j)}^{(k)}\right|, & k = 1 \\ \max\left\{\left|\lambda_{i(j)}^{(k)}\right|, \dfrac{1}{2}\left(\mu_{i(j)}^{(k-1)} + \left|\lambda_{i(j)}^{(k)}\right|\right)\right\}, & k \ge 2 \end{cases} \tag{30}$$

In the linear search process, procedures for determine of search step are described below:

(1) Let $\phi(\varrho) = W(g_k + \varrho d_k, \mu_k)$, select $\varrho = 1$ for the first step.

(2) If $\phi(\varrho) \leq \phi(0) + 0.1\phi'(0)\varrho$ holds, terminate the iteration; otherwise, go to next step.

(3) Calculate $\bar{\varrho}$

$$\bar{\varrho} = \frac{\phi'(0)\varrho^2}{2\left(\phi'(0)\varrho^2 + \phi(0) - \phi(\varrho)\right)} \tag{31}$$

Let $\varrho = \max\{0.1\varrho, \min\{0.6\varrho, \bar{\varrho}\}\}$, go to step 2.

C. REFINEMENT OF MATRIX B_k

Using the quasi-Newton method to achieve a positive definite approximation of the Hesse matrix

$$L(g, \lambda) = f(g) - \sum_{i=1}^{l} \lambda_i h_i(g) - \sum_{j=1}^{m} \lambda_j c_j(g) \tag{32}$$

Let $\rho_k = g_{k+1} - g_k$, then the gradient difference of Lagrange function is

$$y_k = \nabla_g L(g_{k+1}, \lambda_k) - \nabla_g L(g_k, \lambda_k) \tag{33}$$

In order to get the definite matrix B_{k+1}, $y_k^T s_k > 0$ should be guaranteed. To fulfil the requirement, let

$$\chi_k = \vartheta y_k + (1 - \vartheta) B_k \rho_k, \vartheta \in [0, 1] \tag{34}$$

where

$$\vartheta = \begin{cases} 1, & y_k^T \rho_k \geq 0.2\rho_k^T B_k \rho_k \\ \dfrac{0.8\rho_k^T B_k \rho_k}{\rho_k^T B_k \rho_k - y_k^T \rho_k}, & y_k^T \rho_k < 0.2\rho_k^T B_k \rho_k \end{cases} \tag{35}$$

Therefore, $\chi_k = y_k$ can be get when $y_k^T \rho_k \geq 0.2\rho_k^T B_k \rho_k$. Otherwise, $y_k^T \rho_k = 0.2\rho_k^T B_k \rho_k$. Then the following can be guaranteed.

$$\rho_k^T \chi_k \geq 0.2\rho_k^T B_k \rho_k \geq 0 \tag{36}$$

Then the refined B_k becomes

$$B_{k+1} = B_k - \frac{B_k \rho_k \rho_k^T B_k}{\rho_k^T B_k \rho_k} + \frac{\chi_k \chi_k^T}{\chi_k^T \rho_k} \tag{37}$$

D. THE GPU-ACCELERATED APPROACH

While calculating the target value of the optimization function, it is necessary to calculate the interference quantity δ of all discrete points on the surface repeatedly. According to Section II, the coordinate point in the workpiece coordinate system needs to be transformed into the tool coordinate system during the calculation of the interference amount, thus a lot of matrix calculations are needed. The above operations need to be calculated one by one when executed in the CPU, which is time-consuming. Therefore, the GPU acceleration technology is introduced here to increase the speed of operation.

A GPU is a graphics processor and is also called a graphics processing unit. It is a dedicated graphics-rendering device that can be found on every personal computer, sharing the 2D and 3D graphics processing tasks of CPU. The GPU has significant advantages over the CPU in terms of processing power and memory bandwidth and does not need to pay a large price in terms of cost and power consumption, which makes it an important solution to solve the problem of a large amount of data calculation. Due to the high degree of parallelism of graphics rendering, the GPU can increase the processing capacity and memory bandwidth by adding parallel processing units and memory control units. GPU designers use more transistors as execution units instead of the CPU as complex control units and caches to increase the execution efficiency of a small number of execution units.

At present, GPU programming mainly adopts the heterogeneous mode of CPU+GPU. The CPU is responsible for the complex logic processing and transaction management and is not suitable for parallel computing. The GPU is responsible for the intensive large-scale data parallel computing. This kind of computing method that takes advantage of the GPU's powerful processing capacity and high bandwidth to compensate for the lack of CPU performance has significant advantages in exploiting the potential performance, cost, and cost-effectiveness of the computer. The parallel computing function running on the GPU is called the kernel. A kernel function is not a complete program, but a step in the entire program that can be executed in parallel. Figure 8 shows a complete CUDA program, which consists of a series of device-side kernel function parallel steps and host-side serial processing steps. These steps are performed in the corresponding order in the program.

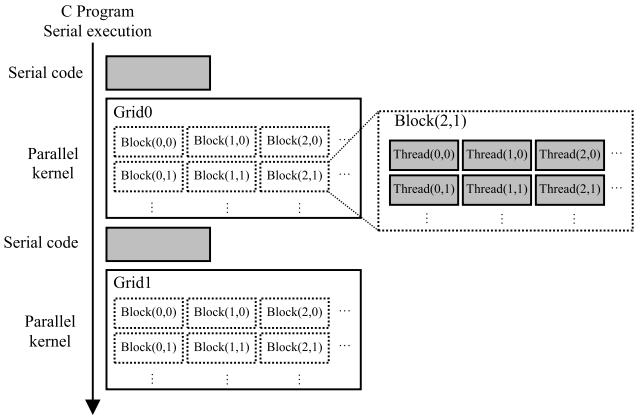

FIGURE 8. GPU based programming framework.

It can be seen from Fig. 8 that the kernel is organized in the form of a grid of threads. Each grid of threads is composed of a number of thread blocks, and each thread block is composed of several threads. There are two levels of parallelism in a kernel function, that is, the parallelism between blocks in the Grid and the threads in the Block. In practice, the kernel is implemented in units of blocks. The CUDA-introduced grid is only used to represent a set of blocks that can be executed in parallel. Blocks are unable to communicate in the parallel execution process, there is

no execution order, but threads can achieve communication through related operations. According to the practical problems and hardware conditions, opening up reasonable block and thread for parallel computing can effectively improve the computational efficiency.

The calculation principle of the objective function $f(\boldsymbol{g})$ is shown in Fig. 9. Create a float4 memory array A_0 with a length of N (N is the number of discrete points on the surface) and a float memory array A_1 of length M (M is the number of blocks opened in a Grid in the GPU). Store the coordinates of the discrete points on the surface of the WCS in A_0 and store the value of the objective function in the calculation process in A_1. At the beginning of the calculation, all the values in A_1 are set to 0. In the calculation process, a separate thread is used to calculate the interference quantity δ_i and the corresponding weight value λ_i between a discrete point s_i on the surface and the tool. Because the threads in the same block can communicate with each other, the application is applied to each block. The approximate operation acquires the weighted sum of squares of all the interferences in the block, and after all block calculations are completed, the result is passed back to A_1. The sum of all the values in A_1 is obtained by using the serial summation method in the CPU is the value of objective function $f(\boldsymbol{g})$ under $(\boldsymbol{M}_t, \boldsymbol{R}_t)$.

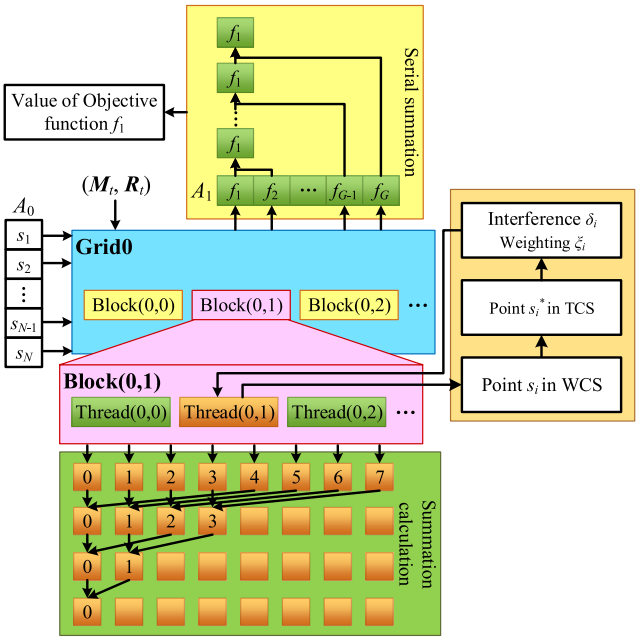

FIGURE 9. Calculation of objective value.

VI. EXAMPLES AND ANALYSIS

To validate the proposed method, the tool orientation adjustment for the five-axis machining of an aero-engine blisk is presented in this section. The cutting tool used here is a 10mm diameter flat end cutting tool. When collision occurs, cutter center point is $\boldsymbol{O}_t = (280.796, -7.177, 22.672)$ and the corresponding tool orientation is $\boldsymbol{l} = (0.976, -0.116, -0.186)$, as shown in Fig. 10. The developed method is used to eliminate collision between the cutting tool and the workpiece.

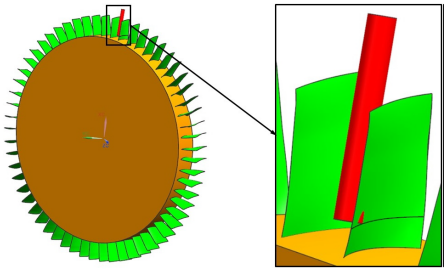

FIGURE 10. Tool collision in five-axis milling of blisk.

A. TRANSFORMATION WITH ONLY TRANSLATION

In this case, only translate transformation is performed. Let the adjustment range along x_t, y_t and z_t axis be $\Delta x \in [-2, 2]$, $\Delta y \in [-2, 2]$, $\Delta z \in [-2, 2]$, respectively. Since no rotation is performed, $\omega_x = \omega_y = \omega_z = 0$. The proposed GPU based method is performed and results are shown in Table 1 and Fig. 11, the optimization result is shown in Fig. 12. In Table 1, the "Points number" indicates the number of discrete points on the potentially effective area of the workpiece that may interfere with the tool. The more discrete points, the higher the calculation accuracy.

TABLE 1. Computation results with translation.

Points number	Computing time (ms)		Cutter center point
	CPU	GPU	
1199	393.5	84.55	(280.680,-7.078,23.340)
2111	742.4	92.31	(279.498, -6.886, 23.617)
3363	2199.7	124.59	(279.475, -6.517, 24.175)
4818	1813.4	109.57	(279.324, -6.527, 24.128)
6583	1853.6	152.29	(279.323, -6.709, 23.835)
8608	3399.6	185.42	(279.391, -6.789, 23.718)
10878	4365.2	145.25	(279.361, -6.765, 23.774)
13526	5297.6	152.98	(279.340, -6.757, 23.765)

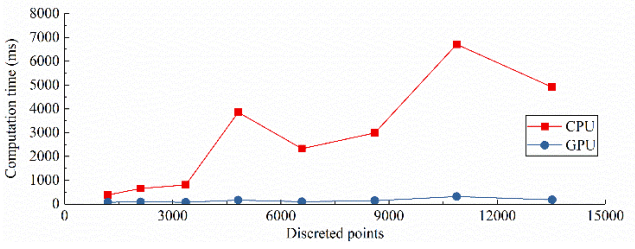

FIGURE 11. Comparison of computing time with CPU and GPU.

B. TRANSFORMATION WITH ONLY ROTATION

In this case, only rotation transformation is performed. Let the adjustment range about x_t, y_t and z_t axis be $\omega_x \in [-\pi, \pi]$,

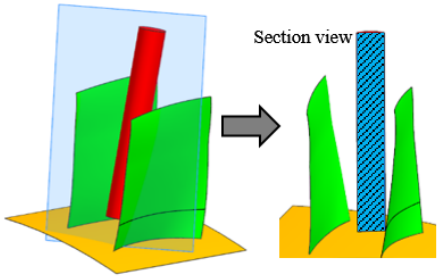

FIGURE 12. Optimization result with translation.

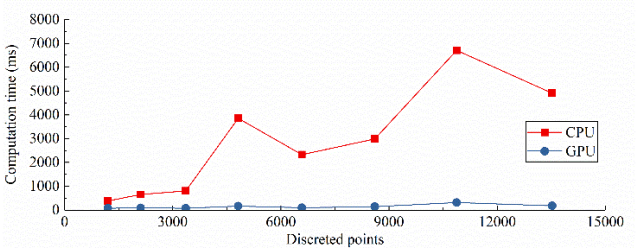

FIGURE 13. Comparison of computing time with CPU and GPU.

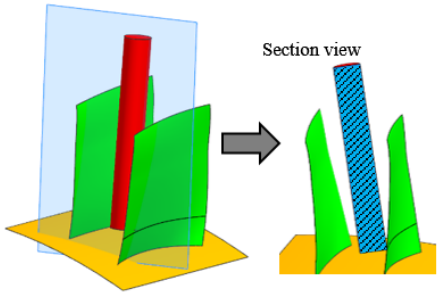

FIGURE 14. Optimization result with only rotation.

$\omega_y \in [-\pi, \pi]$, $\omega_z \in [-\pi, \pi]$, respectively. Since no translation is performed, $\Delta x = \Delta y = \Delta z = 0$. The proposed GPU based method is performed and results are shown in Table 2, Fig. 13 and Fig. 14.

TABLE 2. Computation results with rotation.

Points number	Computing time (ms)		Tool orientation
	CPU	GPU	
1199	381.4	71.67	(0.981,-0.030,-0.193)
2111	651.5	89.94	(0.980,-0.033,-0.197)
3363	803.6	72.61	(0.978,-0.039,-0.205)
4818	3849.9	167.49	(0.978,-0.039,-0.205)
6583	2327.7	94.69	(0.979,-0.036,-0.202)
8608	2985.9	146.03	(0.979,-0.035,-0.200)
10878	6706.7	312.52	(0.979,-0.035,-0.200)
13526	4909.2	181.25	(0.979,-0.036,-0.201)

C. TRANSFORMATION WITH BOTH TRANSLATION AND ROTATION

In this case, let the adjustment range along and about x_t, y_t and z_t axis be $\Delta x \in [-2, 2]$, $\Delta y \in [-2, 2]$, $\Delta z \in [-2, 2]$, $\omega_x \in [-\pi, \pi]$, $\omega_y \in [-\pi, \pi]$, $\omega_z \in [-\pi, \pi]$, respectively. The proposed GPU based method is performed and results are shown in Table 3 and Fig. 15, the optimization result is shown in Fig. 16.

TABLE 3. Computation results with combined transformation.

Points number	Computing time (ms)		Cutter center Point	Tool orientation
	CPU	GPU		
1199	1019.3	117.13	(280.590,-6.555,23.139)	(0.982,-0.022,-0.187)
2111	2011.2	244.92	(279.064,-6.359,23.457)	(0.982,-0.022,-0.187)
3363	3218.6	282.37	(279.200,-6.219,23.543)	(0.980,-0.022,-0.196)
4818	3371.9	340.77	(279.497,-6.247,23.483)	(0.980,-0.022,-0.196)
6583	6123.2	322.65	(279.598,-6.297,23.392)	(0.980,-0.022,-0.197)
8608	9585.9	522.47	(281.143,-6.505,23.021)	(0.979,-0.023,-0.202)
10878	10155.0	442.54	(279.049,-6.272,23.489)	(0.981,-0.021,-0.194)
13526	16632.9	403.37	(279.027,-6.275,23.546)	(0.982,-0.021,-0.189)

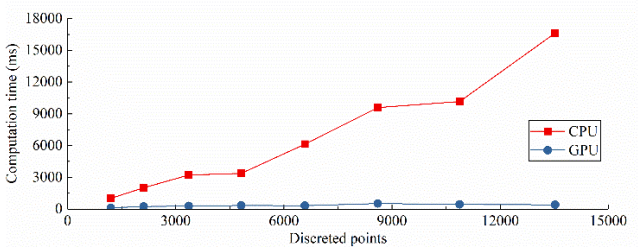

FIGURE 15. Comparison of computing time with CPU and GPU.

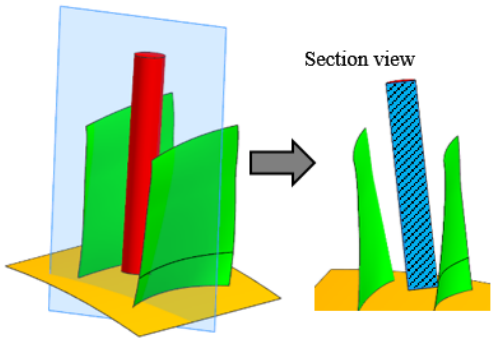

FIGURE 16. Optimization result with combined transformation.

For actual machining processes, there must be millions of points from the workpiece. However, only few positions on a machining trajectory may interfere, and the interference detection is actually very fast. The interference can be eliminated quickly with the developed method.

VII. CONCLUSION

This paper presents a tool orientation optimization model to achieve fast detection of interference and tool posture modification. The solution process of the optimization model is accelerated with the GPU technology. The main contributions of this paper are as follows:

(1) A general method for interference calculation for general cutters is developed, and the framework for fast collision detection and adjustment is established.

(2) A constraint model for the tool posture modification is proposed, and three tool general posture modification strategies are developed.

(3) The GPU based computing technology is applied for achieving rapid computing of the interference value in the tool posture modification process.

Beyond that, to get a smooth tool orientation change along the toolpath, future work can include the global smooth constraint. Furthermore, the presented method can also be used for the general collision detection and avoidance.

REFERENCES

[1] L. Glondu, S. C. Schvartzman, M. Marchal, G. Dumont, and M. A. Otaduy, "Fast collision detection for fracturing rigid bodies," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 1, pp. 30–41, Jan. 2014.

[2] S. D. Lynch, R. Kulpa, L. A. Meerhoff, J. Pettre, A. Cretual, and A.-H. Olivier, "Collision avoidance behavior between walkers: Global and local motion cues," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 7, pp. 2078–2088, Jul. 2018.

[3] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[4] Y. Wen, S. Wu, W. Liu, J. Deng, and X. Wu, "A collision forecast and coordination algorithm in configuration control of missile autonomous formation," *IEEE Access*, vol. 5, pp. 1188–1199, 2017.

[5] N. Wang and K. Tang, "Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath," *Comput.-Aided Des.*, vol. 39, no. 10, pp. 841–852, 2007.

[6] M. Luo, D. Yan, B. Wu, and D. Zhang, "Barrel cutter design and toolpath planning for high-efficiency machining of freeform surface," *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 9, pp. 2495–2503, 2016.

[7] H.-Z. Fan, S.-J. Wang, G. Xi, and Y.-L. Cao, "A novel tool-path generation method for five-axis flank machining of centrifugal impeller with arbitrary surface blades," *J. Eng. Manuf.*, vol. 231, no. 1, pp. 155–166, 2017.

[8] X. Liu, Y. Li, S. Ma, and C.-H. Lee, "A tool path generation method for freeform surface machining by introducing the tensor property of machining strip width," *Comput.-Aided Des.*, vol. 66, pp. 1–13, Sep. 2015.

[9] Y. Liang, Z. Lei, and W. Yuhan, "Optimizing tool size and tool path of five-axis flank milling with bounded constraints via normal mapping," *Adv. Mech. Eng.*, vol. 9, no. 10, pp. 1–10, 2017.

[10] M. Luo, H. Luo, D. Zhang, and K. Tang, "Improving tool life in multi-axis milling of Ni-based superalloy with ball-end cutter based on the active cutting edge shift strategy," *J. Mater. Process. Technol.*, vol. 252, pp. 105–115, Feb. 2018.

[11] A. Rao and R. Sarma, "On local gouging in five-axis sculptured surface machining using flat-end tools," *Comput.-Aided Des.*, vol. 32, no. 7, pp. 409–420, 2000.

[12] S. Ding, M. A. Mannan, and A. N. Poo, "Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces," *Comput.-Aided Des.*, vol. 36, no. 13, pp. 1281–1294, 2004.

[13] Y. L. Cai and G. Xi, "Global tool interference detection in five-axis machining of sculptured surfaces," *J. Eng. Manuf.*, vol. 216, no. 10, pp. 1345–1353, 2002.

[14] Y.-S. Lee and T.-C. Chang, "2-phase approach to global tool interference avoidance in 5-axis machining," *Comput.-Aided Des.*, vol. 27, no. 10, pp. 715–729, 1995.

[15] M. Balasubramaniam, S. E. Sarma, and K. Marciniak, "Collision-free finishing toolpaths from visibility data," *Comput.-Aided Des.*, vol. 35, no. 4, pp. 359–374, 2003.

[16] C. K. Chan and S. T. Tan, "Determination of the minimum bounding box of an arbitrary solid: An iterative approach," *Comput. Struct.*, vol. 79, no. 15, pp. 1433–1449, 2001.

[17] K. Punithakumar, P. Boulanger, and M. Noga, "A GPU-accelerated deformable image registration algorithm with applications to right ventricular segmentation," *IEEE Access*, vol. 5, pp. 20374–20382, 2017.

[18] A. Hidalgo-Paniagua, J. P. Bandera, M. Ruiz-de-Quintanilla, and A. Bandera, "Quad-RRT: A real-time GPU-based global path planner in large-scale real environments," *Expert Syst. Appl.*, vol. 99, pp. 141–154, Jun. 2018.

[19] K. Watanabe, J. I. Kaneko, and K. Horio, "Development of tool collision avoidance method adapted to uncut workpiece shape," *Int. J. Autom. Technol.*, vol. 11, no. 2, pp. 235–241, 2017.

[20] Q.-Z. Bi, Y.-H. Wang, and H. Ding, "A GPU-based algorithm for generating collision-free and orientation-smooth five-axis finishing tool paths of a ball-end cutter," *Int. J. Prod. Res.*, vol. 48, no. 4, pp. 1105–1124, 2010.

[21] H.-T. Hsieh and C.-H. Chu, "Particle swarm optimisation (PSO)-based tool path planning for 5-axis flank milling accelerated by graphics processing unit (GPU)," *Int. J. Comput. Integr. Manuf.*, vol. 24, no. 7, pp. 676–687, 2011.

[22] V. Morell-Giménez, A. Jimeno-Morenilla, and J. García-Rodríguez, "Efficient tool path computation using multi-core GPUs," *Comput. Ind.*, vol. 64, no. 1, pp. 50–56, 2013.

[23] F. Abecassis, S. Lavernhe, C. Tournier, and P.-A. Boucard, "Performance evaluation of CUDA programming for 5-axis machining multi-scale simulation," *Comput. Ind.*, vol. 71, pp. 1–9, Aug. 2015.

[24] A. Balabokhin and J. Tarbutton, "Iso-scallop tool path building algorithm 'based on tool performance metric' for generalized cutter and arbitrary milling zones in 3-axis CNC milling of free-form triangular meshed surfaces," *J. Manuf. Process.*, vol. 28, pp. 565–572, Aug. 2017.

[25] R. Lynn, D. Contis, M. Hossain, N. Huang, T. Tucker, and T. Kurfess, "Voxel model surface offsetting for computer-aided manufacturing using virtualized high-performance computing," *J. Manuf. Syst.*, vol. 43, pp. 296–304, Apr. 2017.

JING WANG was born in Xi'an in 1986. He received the B.S. degree in aircraft manufacturing engineering and the M.S. degree in aeronautical and astronautical manufacturing engineering from Northwestern Polytechnical University, Xi'an, China, in 2009 and 2014, respectively, where he is currently pursuing the Ph.D. degree. His main research is multi-axis machining.

MING LUO (M'15) received the B.S., M.S., and Ph.D. degrees in aeronautical and astronautical manufacturing engineering from Northwestern Polytechnical University, Xi'an, China, in 2005, 2008, and 2012, respectively. He was a Visiting Scholar with the University of Nottingham in 2016. From 2012 to 2017, he was an Assistant Research Fellow with the Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University. Since 2017, he has been an Associate Research Fellow with the Key Laboratory of Contemporary Design and Integrated Manufacturing Technology. He is the author of over 50 articles, and holds four patents. His research interest includes multi-axis machining, data-driven intelligent machining, and machining process monitoring and optimization. He is an Invited Reviewer for many international journals, e.g., the IEEE/ASME TRANSACTIONS ON MECHATRONICS, *International Journal of Machine Tools and Manufacture*, *Computer-Aided Design*, *Mechanical Systems and Signal Processing*, and *Journal of Materials Processing Technology*. He was a Guest Editor of the *International Journal of Manufacturing Research*.

DINGHUA ZHANG was born in 1958. He received the B.S., M.S., and Ph.D. degrees in advanced manufacturing engineering from Northwestern Polytechnical University, Xi'an, China, in 1981, 1984, and 1989, respectively.

He was with Cornell University, Ithaca, NY, USA, and Rochester University, Rochester, NY, USA, as a Visiting Scholar from 1996 to1999. He is currently the Director of the Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an, China. His research interests include smart NC machining, mold design and manufacturing, digital manufacturing system, and digital testing. He is a member of ASME, the Chinese Mechanical Engineering Society, and the Chinese Aerospace Society.

He was a recipient of following awards: the Development and Application of Precision Five-Axis NC Machining Technology for Turbine Blisk of Aero-Engine, Second Prize of National Scientific and Technological Progress, 2006; the Development and Application of a CAD/CAM System for Turbomachinery, Third Prize of National Scientific and Technological Progress, in 1992; the NPU—an Interactive Computer Graphics NC programming System, First Prize of Shaanxi Scientific and Technological Progress, in 1991; the Precision Casting Mold CAD/CAM system for Hollow Turbine-Blade of Aero-engine, First Prize of Scientific and Technological Progress, Aero-Industrial Ministry, China, in 1999; and the Accurate and Efficient Calibration Method for a Selenium Flat-panel Detector-based Volume Tomographic Angiography Imaging System, SPIE Medical Imaging'99 Honorable mention Award.