Int J Adv Manuf Technol (2016) 85:1887–1900
DOI 10.1007/s00170-016-8356-3

ORIGINAL ARTICLE

Accessible regions of tool orientations in multi-axis milling of blisks with a ball-end mill

Yongshou Liang[1] · Dinghua Zhang[1] · Junxue Ren[1] · Zezhong C. Chen[2] · Yingying Xu[1]

Received: 7 September 2015 / Accepted: 13 January 2016 / Published online: 29 January 2016

Abstract Noninterference tool orientations are hard to be determined in multi-axis milling of blisks, because the integrated structure of blisks introduces more geometrical constraints. To address this problem, an original approach without interference detection is proposed to solve accessible regions of tool orientations in milling of blisks. Based on the visibility of checking surfaces, only critical points on surface profiles are searched and processed to construct the accessible regions. In this approach, the start point of each profile is sought on boundaries of the main surface with a constant step length. From this point, a new critical point is first searched along a specified direction and then adjusted iteratively until it locates on a profile. When all critical points on profiles are searched like this, the searched points on each checking surface are reordered to form one or more than one closed curve to present the accessible region related to this checking surface. These curves are then mapped onto a unit sphere and divided into nonintersecting segments. After that, a concise approach is proposed to combine accessible regions of all checking surfaces into simply connected regions. This algorithm is finally verified with two kinds of blisks and compared with a referenced method. The results show that the proposed method can efficiently solve accessible regions in multi-axis milling of blisks with an expected accuracy.

✉ Yongshou Liang
liangys363@gmail.com

[1] School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an 710072, China

[2] Department of Mechanical and Industrial Engineering, Concordia University, Montreal H3G 1M8, Canada

Keywords Accessible region · Accessibility · Visibility · Blisk machining · Ball-end mill · Tool orientation · Interference detection

1 Introduction

Blisks are key parts used in new jet engines to improve engine's thrust-to-weight ratio and promote the performance of aircrafts. Instead of attaching separate blades to a machined rotor disk, a blisk is designed as a one-piece unit by joining them together. This integrated structure reduces the engine's weight, part count, aerodynamic losses, and possible crack initiation and propagation. However, it also introduces more geometrical constraints to the parts, which increase the complexity of tool orientation planning in milling process.

The key issue for planning tool orientations is the avoidance of local and global interference between cutters and parts. The local interference usually refers to local-gouging and rear-gouging, while global interference concerns global-collision between objects involved in machining. Principal axis method [1], rolling ball method [2], arc-intersect method [3], swept envelope method [4], multi-point machining method [5], rotary contact method [6], and hyper-osculating circles method [7] were used to avoid local-gouging and rear-gouging. These techniques dealt with the local interference to obtain gouging-free tool orientations or locations with the concept of curvature matching. However, global interference elimination is the main consideration in machining geometrically complex parts rather than gouging avoidance, the latter of which is a common technique as same as machining a single surface. Effective approaches were proposed to optimize noninterference tool orientations in machining impellers and blisks

by considering their complex geometries [8–10]. These researches focused on tool orientation optimization, while specific techniques to detect or avoid the interference were not introduced.

Global interference detection and avoidance are still a current technique challenge in computer numerical control (CNC) machining. Since the interference between a cutter and sculptured surfaces is difficult to detect, an intuitional approach is converting involved surfaces into points, lines, or subdivided meshes. Lin et al. [11] expressed the machined surface as point cloud and detect collisions between the cutter and these points. Chu et al. [12] discretized the machined surface into a set of sample points to check the interference. Chen et al. [13] expressed a blade surface as a triangular mesh body and solved the accessible regions to optimize and smooth tool orientations. These discretization or subdivision methods check all discretized points or subdivided meshes. The quality and accuracy of interference detection are largely dependent upon the quantity of sampled or subdivided objects. When increasing the quantity to improve quality and accuracy of interference detection, the computation cost will increase accordingly. An improved approach is presenting involved surfaces as a hierarchical structure. Only the interested parts involved in interference detection are further subdivided iteratively. Ding et al. [14] presented the cutter and cutter holder as an oriented bounding box (OBB), while approximated the machined surface as an octree. Then, the interference was detected hierarchically between tool OBBs and octants of the surface octree. Tang et al. [15] approximated the machined part as an octree of bounding sphere to recursively conduct collision detection in five-axis machining. The computation cost with these hierarchical approaches could be reduced because the further process is avoided when no interference is detected in an octant. However, these approaches are still time-consuming to detect the interference repeatedly. Hardware approach is another perspective to detect the collision by taking advantage of efficient and robust performance of the graphics processing unit (GPU) in graphics hardware. Wang et al. [16] proposed a two-phase strategy to detect collisions between the cutter and triangulated obstacles. The first phase used a fair-resolution collision map to detect collisions at most cutter location (CL) points, while the second phase conducted vector-based calculation to check collisions precisely for some ambiguous cases. Bi et al. [17] computed accessible regions in milling of an impeller with the assistance of occlusion query functionality of the graphics hardware. The hardware approach relies on expensive hardware and supporting software system [11]. Therefore, equipment costs should be taken into consideration.

According to the processing sequence, there are two kinds of approaches to planning noninterference tool orientations. The one firstly generates tool orientations according to some strategies and then adjusts them with interference detection, while the other one directly calculates collision-free spaces and then optimizes tool orientations within these spaces [18]. Some researchers planned noninterference tool orientations based on the first idea. Chuang and Young [19] calculated the distance between tool axis and blade surface, and then adjusted the tool location to avoid the interference in milling of centrifugal impellers. Chu et al. [12] checked the distance between the tool axis and the machined surface. After that, the cutter was rotated away from the interfered points to avoid the interference. Wu et al. [20] presented a rotation method to eliminate the interference in milling of impellers with a non-orthogonal four-axis machine tool. By adjusting the tool orientation with a calculated rotation angle, an interfered point was corrected to locate outside of the cutter's envelope. The interference detection and correction in the abovementioned researches are mainly conducted to guarantee the validation of existing tool orientations. Collision-free spaces of tool orientations are not obtained with these adjustment methods, without which further tool orientation optimization is not convenient to be conducted. Moreover, these adjustment methods usually execute interference detection and correction iteratively for several times, which require large computation time and resource.

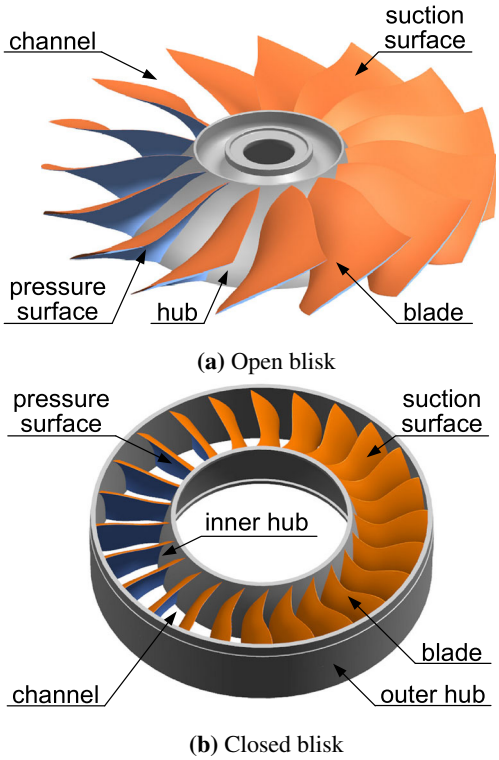
To avoid iterative adjustments of tool orientations, some researchers adopt the second idea to calculate collision-free spaces directly based on the visibility or accessibility of machined surfaces. Balasubramaniam et al. [21] used graphics hardware to calculate visibility cone of a triangulated surface. Lee and Chang [22] proposed a two-phase method to avoid global interference in five-axis machining. The first phase quickly obtained conservative feasible tool orientations based on the convex hull property of the control mesh, while the second phase checked detailed orientations with sample points in the localized area. Hsueh et al. [23] generated collision-free cones of tool orientations with a two-stage method. At each stage, collision detection was conducted between the cutter flank and dispersion points of machined surface. Wang and Tang [24] derived a feasible map from a visibility map to figure noninterference tool orientations. The global interference was checked by calculating the distance between the CL point and the project of an obstacle on the cutter bottom plane. Li and Zhang [25] searched the cutter accessibility in interference-prone regions by checking the distances between tool axis and sample points of the machined surface. Morishige et al. [26] introduced a two-dimensional configuration space (C-space) to calculate collision-free

tool orientations in milling of polyhedrons and sculptured surfaces. In this approach, the inclination angle was adjusted from the collided point to its nearest point on the boundary of the collision area and the free area in the C-space. Lu et al. [27] constructed the C-space considering local interference, global interference, and cusp height on the machined surface. Jun et al. [28] adopted an edge detection technique to search the boundaries of C-space feasible region. This procedure was first conducted in coarse-grid meshes and then in fine-grid meshes, which was also a layer-by-layer refining process. These researches successfully calculate noninterference spaces of tool orientations with adjustment or search method. However, the interference detection in these methods cannot be avoided, which is time-consuming when traversing all sample points or meshes [29]. Furthermore, the contradiction of accuracy and efficiency still exists in the discretization or subdivision process.

This work presents a novel search method to calculate accessible regions of tool orientations without interference detection in five-axis milling of blisks. By replacing a ball-end mill and checking surfaces with a line and their offset surfaces, respectively, the concept of visibility is used to calculate critical points that divide the surfaces into visible and invisible parts. The accuracy of this method can be specified by a user, which has little effect on the computation time. The strategy of this approach is briefly introduced in Section 2. The algorithms to search all critical points in two stages are presented in Section 3. When all critical points are obtained, data processing and accessible region combination are conducted in Section 4. In section 5, the proposed approach is verified and evaluated by applying it in two blisks and comparing with a referenced method; and the work is concluded in Section 6.

2 Strategy to solve accessible regions

As shown in Fig. 1, a channel of a blisk is formed by two adjacent blades and hubs. To machine a point located inside of this channel, tool orientations must be planned appropriately to avoid the potential collisions from the blades and hubs. An accessible region is a set of tool orientations along which the cutter will not collide or interfere with the machined part. When calculating the accessible region in milling of blisks with a ball-end mill, involved geometrical models are replaced by their offsets in this work. First, the ball-end mill is replaced by its central line, while original blade/hub surfaces are replaced by their offset surfaces. Second, vertexes and boundaries of these original surfaces are replaced by their offsets: spheres and tubes that are centered at them, respectively. Here, the offset distance, radius of spheres, and radius of tubes are equal to



(a) Open blisk

(b) Closed blisk

Fig. 1 Typical structure of blisks

tool radius plus machining allowance. It is obvious that the offset surfaces related to an original blade/hub surface can be classified into main and assistant offset surfaces. The offset of the original blade/hub surface is the main offset surface, while the spheres and tubs generated from the vertexes and boundaries are assistant offset surfaces. For convenience, when mentioning "surface," "checking surface," "main surface," or "assistant surface" in the following part, they usually represent their offsets, unless otherwise specified.

An accessible region of a CL point is constrained by its surrounding surfaces. As shown in Fig. 2, imagine that there is a virtual unit sphere centered at the CL point P_{CL}. When putting a light source at this point, the region on the unit sphere that the rays can reach to is not obstructed by these surfaces. It is called accessible region, while the obstructed region is called inaccessible region. In this figure, only one surrounding surface is shown to illustrate the concept of the accessible region. Considering the geometrical structure of blisks, the accessible regions for milling a blisk is one or more than one simply connected space. These spaces can be presented by their boundaries. Geometrically, the boundaries of each space on the unit sphere are constituted with central projections of these checking surfaces' profiles. Therefore, when calculating accessible regions in milling of a blisk, this work resorts to solving these profiles.

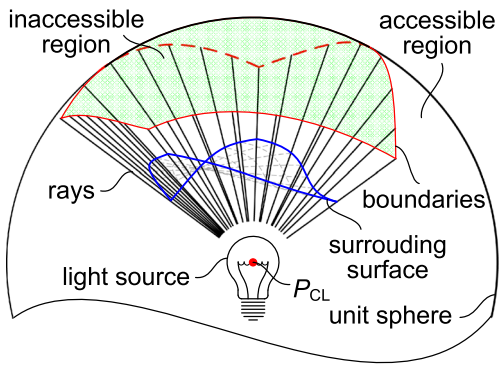


Fig. 2 Illustration of the accessible region

Profiles of the main and assistant surfaces for one original blade/hub surface are solved in a search process. When viewing from the CL point, each profile of the main surface will contact one profile of an assistant surface at a point. This point is located on a boundary of the main surface. It can be treated as the start point to search these profiles on the main and assistant surfaces. Therefore, the search process is conducted in two stages. The first stage searches the start points of all profiles on boundaries of the main surface, while the second stage searches critical points on checking surfaces' profiles from these start points. In the first stage, the search process starts at one vertex of the main surface, and then checks sample points along all surface boundaries with a constant step length using the Taylor expansion of boundaries. If the normal direction of the main surface at a sample point is perpendicular to the view direction, this point is solved as a start point. When all start points are solved like this, the search process turns into the second stage. In this stage, the dot product of the view direction and surface's normal direction, abbreviated as "dot product" in the following part, is used as an evaluation function to define critical points and determine search trajectory. To search the next critical point from a start point or a solved critical point on a checking surface, an initial point is first searched perpendicular to the gradient direction of the dot product with a constant step length. If the dot product at this initial point does not satisfy the condition of critical points, the algorithm searches along the gradient direction iteratively until a new critical point is found. The search process will go ahead like this and end when it reaches to one boundary of the checking surface. The searched critical points in this process present a profile on a checking surface. When profiles of all checking surfaces are searched and stored in segments, the critical points on these profiles are used to solve accessible regions.

Before solving accessible regions, the stored data is first processed. Since each critical point determines a tool orientation in which the CL point can just be accessible, they are expressed in a two-dimensional space defined by two orientation parameters. When viewing from the CL point, the profiles of the main and assistant surfaces for an original blade/hub surface should form one or more than one closed curve in this two-dimensional space. However, the segments of profiles have been stored according to the sequence when they were searched, which are not consistent with the sequence as they will appear on the closed curve. These sequences are first rearranged to make them stored correctly. Then the stored sequence of critical points in each segment is further checked. If the sequence of critical points within a segment is not consistent with the sequence of all segments, the critical points in this segment are stored reversely. Then each closed curve is assigned with a positive direction to distinguish the accessible and inaccessible regions. After that, intersection points of closed curves are calculated to divide them into nonintersecting segments.

The segments solved from an original blade/hub surface can present the accessible regions for this surface. When concerning all blade/hub surfaces, the final accessible regions are solved by combining all related accessible regions together. Starting from an intersection point of segments, it looks like a man walking along the positive direction of point sequence. When encountering a new intersection point of segments, the rightmost segment is selected as the new walking direction according to a proposed determining criterion. One or more than one closed route can be formed when the man passes through all intersection points of segments. The critical points appearing in each walking route construct a new closed curve to present a simply connected region in the two-dimensional space. These regions are the accessible regions to plan tool orientations in multi-axis milling of blisks.

3 Algorithm to search critical points

3.1 Search start points of profiles

In this part, the algorithm to solve start points of profiles is introduced in detail. Since the profiles divide a checking surface into visible and invisible parts, the visibility of a point is used to judge whether this point is a critical point on profiles. So the start points can be solved by finding the points on boundaries around which the visibility is changed.

The visibility of a point on a surface is determined by dot product of surface normal and view direction. As shown in Fig. 3, suppose a point on the ith checking surface is $S_i(u, v)$ in xyz coordinate system. Then the unit vector $\mathbf{N}_i(u, v)$ of

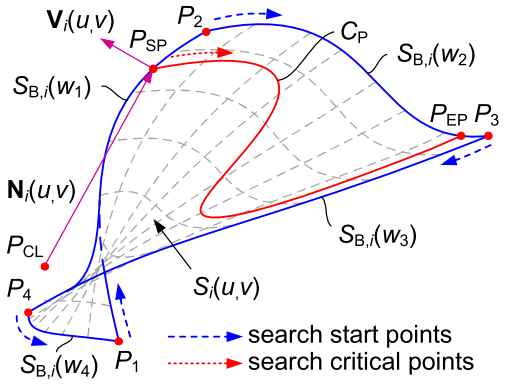


Fig. 3 Process of solving start points and critical on a main surface

surface normal at this point is calculated with Eq. 1. Here, the outward direction is used in the algorithm.

$$\mathbf{N}_i(u,v) = \frac{\frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial S_i(u,v)}{\partial v}}{\left|\frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial S_i(u,v)}{\partial v}\right|}$$

or

$$\mathbf{N}_i(u,v) = \frac{\frac{\partial S_i(u,v)}{\partial v} \times \frac{\partial S_i(u,v)}{\partial u}}{\left|\frac{\partial S_i(u,v)}{\partial v} \times \frac{\partial S_i(u,v)}{\partial u}\right|} \quad (1)$$

When viewing the surface from a CL point P_{CL}, dot product Q_i of view direction $\mathbf{V}_i(u,v)$ and surface normal $\mathbf{N}_i(u,v)$ can be calculated as shown in Eq. 2.

$$Q_i = \mathbf{N}_i(u,v) \cdot \mathbf{V}_i(u,v) = \mathbf{N}_i(u,v) \cdot \frac{S_i(u,v) - P_{CL}}{|S_i(u,v) - P_{CL}|} \quad (2)$$

In view of the calculation error, a small positive quantity ε is defined to classify the calculated results of dot product Q_i. When $Q_i < -\varepsilon$, point $S_i(u,v)$ is visible from point P_{CL}; when $Q_i > \varepsilon$, point $S_i(u,v)$ is invisible. When $|Q_i| \leq \varepsilon$, point $S_i(u,v)$ is considered to be on a profile of checking surface when viewing from point P_{CL}.

Suppose four boundaries of the ith checking surface are $S_i(u,0)$, $S_i(1,v)$, $S_i(u,1)$, and $S_i(0,v)$. For convenience, use $S_{B,i}(w_j)$ to represent the jth boundary. Here, w_j is the curve parameter ($w_j \in [w_{S,j}, w_{E,j}]$, $j = 1,2,3,4$), while $w_{S,j}$ and $w_{E,j}$ are the curve parameters at the start and end points of the jth boundary, respectively. Start points of profiles are first searched along boundaries $S_{B,i}(w_j)$ of a checking surface one by one. After that, critical points on profiles are then searched from these start points using the algorithm that will be introduced later. One of the start points is shown as point P_{SP} in Fig. 3, while the searched profile is shown as curve C_P.

The algorithm to solve start points is shown in Fig. 4. It starts at the start point of the first boundary by assigning curve parameter $w_{S,j}$ to $w_{C,j}$. Here, $w_{C,j}$ is the curve parameter at the current point. The visibility of this point is

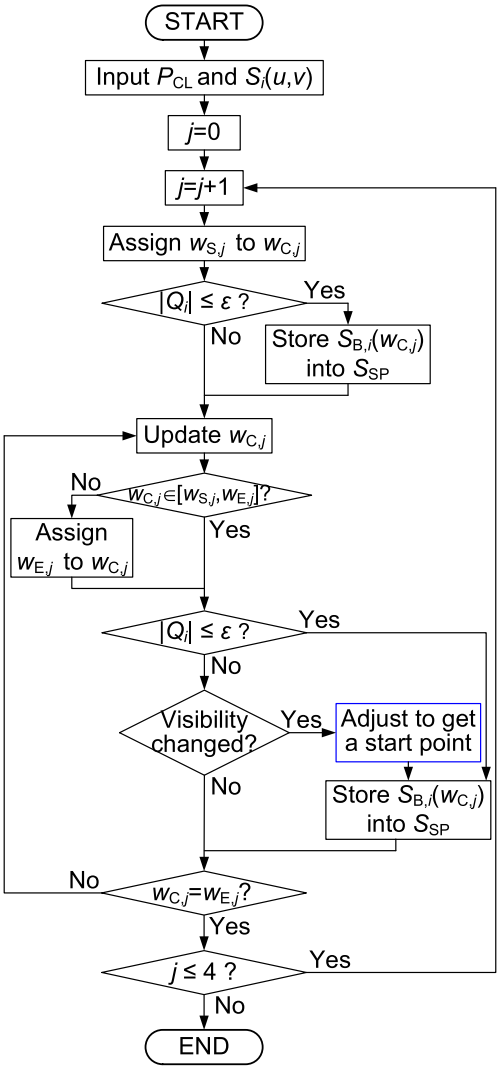


Fig. 4 Flow chart of the algorithm to solve start points

then checked using Eq. 2. If this point satisfies the condition of a profile, it is a start point and stored in a set S_{SP}. Otherwise, this point may be visible or invisible from the CL point. Since the current point $S_{B,i}(w_{C,j})$ is the start point on this boundary, regardless of its visibility, the search process will go ahead for checking the next sample point on the jth boundary.

The next sample point is determined by a constant step length L_S. When L_S is very small, the next sample point can be calculated with the Taylor expansion of boundary $S_{B,i}(w_j)$. Expand the boundary $S_{B,i}(w_j)$ with a first-order Taylor series at point $S_{B,i}(w_{C,j})$ as

$$S_{B,i}(w_j) = S_{B,i}(w_{C,j}) + S'_{B,i}(w_{C,j})(w_j - w_{C,j}) + O(w_j - w_{C,j}) \quad (3)$$

Here, $S'_{B,i}(w_{C,j})$ is the derivative of $S_{B,i}(w_j)$ at the current parameter $w_{C,j}$, while $O(w_j - w_{C,j})$ is a remainder. To

guarantee a constant distance to the current point, the next sample point should satisfy Eq. 4.

$$\left|S_{\mathrm{B},i}(w_j) - S_{\mathrm{B},i}(w_{\mathrm{C},j})\right| = L_{\mathrm{S}} \tag{4}$$

Then the curve parameter for the next sample point $w_{\mathrm{N},j}$ can be approximately calculated with Eq. 5 to update $w_{\mathrm{C},j}$. This simplification is reasonable because a tiny step error has little influence on the calculated tool orientations. Here, the sign of the second part in Eq. 5 is determined by whether $w_{\mathrm{E},j}$ is greater than $w_{\mathrm{S},j}$ or not.

$$w_{\mathrm{N},j} = w_{\mathrm{C},j} \pm \frac{L_{\mathrm{S}}}{\left|S'_{\mathrm{B},i}(w_{\mathrm{C},j})\right|} \tag{5}$$

When using parameter $w_{\mathrm{N},j}$ to update $w_{\mathrm{C},j}$, the algorithm turns into a loop to check all sample points on this boundary, as shown in Fig. 4. In this loop, the new $w_{\mathrm{C},j}$ needs to be checked whether it is within the defined interval $[w_{\mathrm{S},j}, w_{\mathrm{E},j}]$. If not, the end point of the boundary is treated as the new current point by assigning curve parameter $w_{\mathrm{E},j}$ to $w_{\mathrm{C},j}$; otherwise, the current point keeps unchanged. Then the visibility of the current point $S_{\mathrm{B},i}(w_{\mathrm{C},j})$ is checked by Eq. 2 as before. A new start point on the boundary may be searched in two cases, as shown in Fig. 4. The one is that a new start point is searched in visibility checking period. When dot product Q_i of the sample point just satisfies the condition $|Q_i| \le \varepsilon$, this point is a new start point. The other one is that dot product Q_i does not satisfy the condition $|Q_i| \le \varepsilon$, but the sign of Q_i inverses. It means that the visibility of the adjacent two points is changed. It may be changed from "visible" to "invisible" or from "invisible" to "visible." Anyway, there should be a point between them that satisfies the condition of a critical point, which should also be a start point. To seek a start point that is located in the range where the visibility of adjacent points inverses, the gradient of dot product Q_i at the last point is used to adjust $w_{\mathrm{C},j}$ adaptively in this local range. The dot product Q_i of a point on the checking surface can be denoted as $Q_i(u, v)$ in Eq. 6.

$$Q_i(u,v) = \left(\frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial S_i(u,v)}{\partial v}\right) \cdot (S_i(u,v) - P_{\mathrm{CL}}) \tag{6}$$

Expand the dot product $Q_i(u, v)$, and calculate its gradient $\nabla Q_i(u, v)$.

$$\begin{aligned}\nabla Q_{u,i}(u,v) = &\left(\frac{\partial^2 S_i(u,v)}{\partial u^2} \times \frac{\partial S_i(u,v)}{\partial v}\right.\\ &\left. + \frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial^2 S_i(u,v)}{\partial v \partial u}\right)\\ &\cdot (S_i(u,v) - P_{\mathrm{CL}})\end{aligned} \tag{7}$$

$$\begin{aligned}\nabla Q_{v,i}(u,v) = &\left(\frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial^2 S_i(u,v)}{\partial v^2}\right.\\ &\left. + \frac{\partial^2 S_i(u,v)}{\partial u \partial v} \times \frac{\partial S_i(u,v)}{\partial v}\right)\\ &\cdot (S_i(u,v) - P_{\mathrm{CL}})\end{aligned} \tag{8}$$

Here, $\nabla Q_{u,i}(u, v)$ and $\nabla Q_{v,i}(u, v)$ represent the components of gradient $\nabla Q_i(u, v)$ in u and v directions, respectively. For boundary $S_{\mathrm{B},i}(w_j)$ with only one curve parameter, use $Q_i(w_j)$ to represent $Q_i(u, v)$, while use $\nabla Q_i(w_j)$ to represent $\nabla Q_{u,i}(u, v)$ or $\nabla Q_{v,i}(u, v)$ uniformly. It is obvious that the dot product at the desired point is equal to zero. Expand $Q_i(w_j)$ with a first-order Taylor series at parameter $w_{\mathrm{C},j}$ as

$$Q_i(w_j) = Q_i(w_{\mathrm{C},j}) + \nabla Q_i(w_{\mathrm{C},j})(w_j - w_{\mathrm{C},j}) + O(w_j - w_{\mathrm{C},j}) \tag{9}$$

Substitute $Q_i(w_j) = 0$ into Eq. 9, the curve parameter $w_{\mathrm{A},j}$ of the next point in adjustments can be initially assigned as

$$w_{\mathrm{A},j} = w_{\mathrm{C},j} - \frac{Q_i(w_{\mathrm{C},j})}{\nabla Q_i(w_{\mathrm{C},j})} \tag{10}$$

Then the dot product at this point is calculated to check the visibility by comparing it with the given quantity ε. If it is still located outside of the given limitations, the curve parameter $w_{\mathrm{C},j}$ is updated iteratively with Eq. 10 until a new start point is found or the iteration number exceeds a defined number. This process is shown in Fig. 5 with an example. In this figure, the adjustments start at point $S_{\mathrm{B},i}(w_{\mathrm{C},j})$, and end at point $P_{\mathrm{S},5}$ with five steps. The fifth point $P_{\mathrm{S},5}$ is already a start point of a profile.

After that, the algorithm goes ahead to search along the current boundary until the whole boundary has been completely searched. Here, the case that the boundary is closed in the current direction is also considered and processed. To present the flow chart clearly, this part is not shown in Fig. 4. When four boundaries are completely searched, the start points of all profiles on this surface are obtained. In the

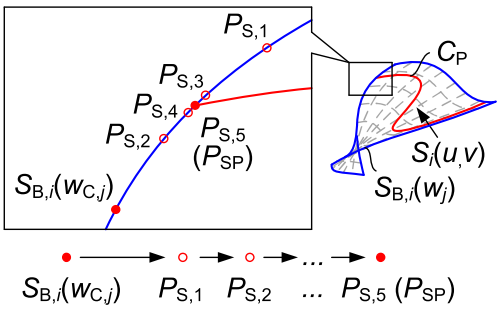


Fig. 5 Adjustments to get a start point

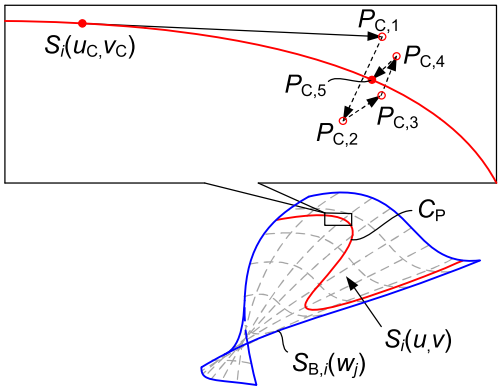


Fig. 6 Process of solving a new critical point

same way, check other surfaces and store the searched start points in set S_{SP}.

3.2 Search critical points

When a start point is solved and stored in set S_{SP}, critical points on the profile starting from this start point will be searched with a constant step length one by one in this part. The search process includes two parts. It firstly moves a large step to approach the desired point, and then adjusts with adaptive steps iteratively to reach the new critical point. As shown in Fig. 6, the first part gets point $P_{C,1}$ from the current point $S_i(u_C, v_C,)$, while the second part adjusts point $P_{C,1}$ to point $P_{C,5}$. Here, step lengths of adjustments shown in this figure are locally amplified to show the search process clearly.

The algorithm to solve a new critical point is shown in Fig. 7. At first, the search process is expected to moves

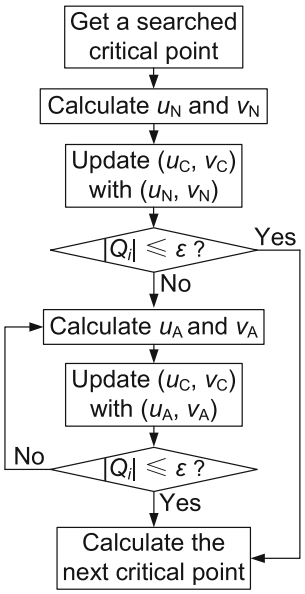


Fig. 7 Flow chart of the algorithm to search a critical point

along the profile with a large step. Along this profile, the dot product at each point keeps unchanged. In other words, the first search direction should be perpendicular to the gradient direction of the dot product. Suppose the variations of parameters in u and v directions are $\Delta u = u_N - u_C$ and $\Delta v = v_N - v_C$, respectively. Here, u_N and v_N are surface parameters at the next point. When searching along the expected profile, Δu and Δv should satisfy Eq. 11.

$$\Delta v = -\frac{\nabla Q_{u,i}(u_C, v_C)}{\nabla Q_{v,i}(u_C, v_C)}\Delta u \tag{11}$$

Here, $\nabla Q_{u,i}(u_C, v_C)$ and $\nabla Q_{v,i}(u_C, v_C)$ represent the components of gradient $\nabla Q_i(u_C, v_C)$ at the current point. Expand the checking surface with a first-order Taylor series at point $S_i(u_C, v_C)$ as

$$S_i(u,v) \approx S_i(u_C, v_C) + \frac{\partial S_i(u_C, v_C)}{\partial u}(u - u_C) + \frac{\partial S_i(u_C, v_C)}{\partial v}(v - v_C) \tag{12}$$

To keep a constant search step length L_S, the next point should satisfy

$$|S_i(u,v) - S_i(u_C, v_C)| = L_S \tag{13}$$

The variation Δu is calculated with Eq. 14 when substituting Eqs. 11 and 13 into Eq. 12, while Δv is calculated with Eq. 11 in view of Eq. 14. Then the current point is updated by $u_N = u_C + \Delta u$ and $v_N = v_C + \Delta v$.

$$\Delta u = \frac{L_S}{\left|\frac{\partial S_i(u_C, v_C)}{\partial u} - \frac{\nabla Q_{u,i}(u_C, v_C)}{\nabla Q_{v,i}(u_C, v_C)}\frac{\partial S_i(u_C, v_C)}{\partial v}\right|} \tag{14}$$

As shown in Fig. 7, the dot product $Q_i(u, v)$ at the new point is then calculated to evaluate the visibility using Eq. 2. If this point is not on the profile, a series of iterative adjustments are conducted along the gradient direction of $Q_i(u, v)$. The gradient components of dot product at this point in u and v directions are calculated with Eqs. 7 and 8, respectively. Then the new surface parameters in one adjustment can be calculated as

$$\begin{cases} u_A = u_C - \frac{\nabla Q_{u,i}(u_C, v_C) Q_i(u_C, v_C)}{\nabla Q_{u,i}^2(u_C, v_C) + \nabla Q_{v,i}^2(u_C, v_C)} \\ v_A = v_C - \frac{\nabla Q_{v,i}(u_C, v_C) Q_i(u_C, v_C)}{\nabla Q_{u,i}^2(u_C, v_C) + \nabla Q_{v,i}^2(u_C, v_C)} \end{cases} \tag{15}$$

Conduct this adjustment iteratively until the dot product reduces to the desired limitations or the iteration number exceeds a defined number. In this way, a new critical point is searched with a constant step length. All critical points on this profile are searched point by point like this. Here, if a new calculated point is updated to be outside of one boundary, the related point on this boundary that has the same parameter is selected to replace this one. The following adjustments at this point are conducted along this boundary, which are the same as introduced in searching a start point

on a boundary. All searched points and the related start point are then stored in set S_{CP} as the critical points of this profile.

Before searching another profile with a new start point, the new start point should be checked whether it is already an end point of a searched profile, such as point P_{EP} in Fig. 3. This checking process is mainly used to avoid searching the same profile again. If it is, skip this start point and check the next one; otherwise, use this start point to search a new profile. When start points in set S_{CP} are all checked, the search process to solve critical points on this checking surface is finished.

4 Algorithm to solve accessible regions

4.1 Data processing

The searched critical points determine tool orientations relative to the CL point. They are mapped on a unit sphere centered at this CL point. A spherical coordinate system is introduced to present the mapped points. As shown in Fig. 8, (x_C, y_C, z_C) is the orthogonal coordinates of a critical point in a translated xyz coordinate system whose origin is located at the CL point P_{CL}, while (r, θ, ϕ) is the spherical coordinates of the same point in $r\theta\phi$ coordinate system. Here, $r(r = 1)$, $\theta(0 \le \theta < 2\pi)$, and $\phi(0 \le \phi \le \pi)$ are radial distance, azimuthal angle, and polar angle, respectively.

As introduced in Section 2, the searched critical points are stored in set S_{CP} in segments. They are first processed in four steps. To introduce these steps clearly, the segments solved from the same original blade/hub surface are treated as one group.

Step 1 Rearrange the sequence of segments in the same group. To recognize whether two segments should be connected or not, the orthogonal coordinates of two endpoints of solved segments are compared. If the coordinates are the same, these two segments should be connected at the same endpoint. The new sequence of all segments in each group is arranged in this way.

Step 2 Connect all segments in the same group as closed curves. To connect two adjacent segments, if the last points of these solved segments are the same, the critical points in the second segments are reversed by storing them from the last point to the first point. One of the same points is removed to reduce the multiplicity of this connection point to one. When connecting segments in the group where the CL point is located, the tangent plane at this CL point is also related to the accessible region. The mapped curve of this plane on the unit sphere is a great circle. Then part of this circle and stored segments in this group can form a closed curve.

Step 3 Assign a positive direction for each closed curve. The positive direction is defined to distinguish the accessible and inaccessible regions. When walking on the outer surface of the sphere along this direction, checking surfaces (also the obstacles and inaccessible region) should be located on the left-hand side, while the accessible region is on the right-hand side. Suppose two adjacent critical points are points P_i and P_{i+1}, while a test point closed to point P_i on a checking surface is point P_T. If D_P in Eq. 16 is less than zero, the current direction is positive; otherwise, reverse it.

$$D_P = (P_T - P_i) \cdot ((P_{CL} - P_i) \times (P_{i+1} - P_i)) \quad (16)$$

Step 4 Divide closed curves into nonintersecting segments. The critical points on each closed curve can form a polygon, which is expressed by azimuthal angle θ and polar angle ϕ in two-dimensional space. Then the intersection points of theses polygons are calculated and inserted into the closed curves to divide them into nonintersecting segments.

Following such a procedure, the closed curve to present an accessible region is expressed by a group of nonintersecting segments. When walking along the positive direction of this curve on the unit sphere, the accessible region is located on the right-hand side. When more than one closed curve is solved in a group, they are treated as a pair to divide a space into accessible and inaccessible regions. In this case, the new accessible regions keep unchanged, while the new inaccessible region is the intersection of the former ones in this group.

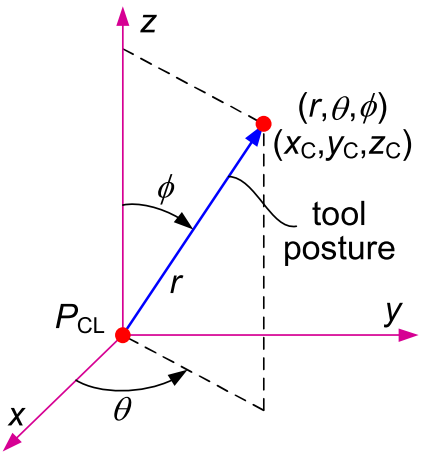


Fig. 8 Definition of the spherical coordinate system

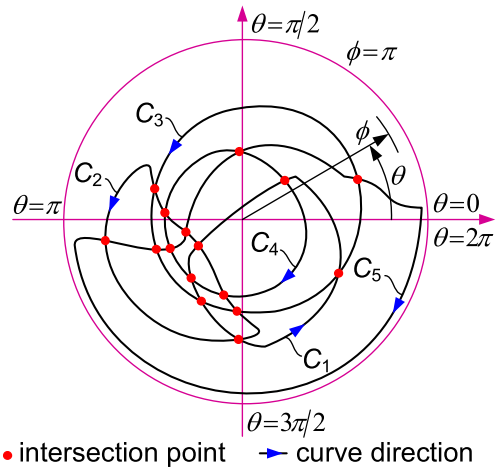


Fig. 9 Definition of the polar coordinate system

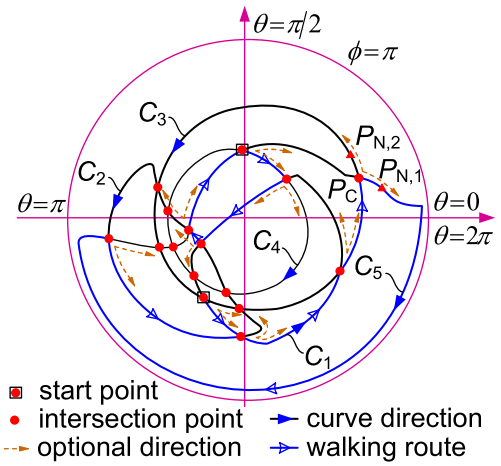


Fig. 10 Accessible region solving

4.2 Combine accessible regions

In this part, the effects of all original blade/hub surfaces are considered together by combining different accessible regions into simply connect regions. Since a complex figure on a sphere surface is apt to cause ambiguity when it is drawn on a plane, the concept of two-dimensional C-space is adopted to show the idea clearly. The closed curves solved in the last section are drawn in $\theta\phi$ polar coordinate system, shown as curves C_1–C_5 in Fig. 9. When walking along each curve direction, the accessible region is located on the right-hand side. In this example, the accessible regions presented by curves C_4 and C_5 are located inside of each closed curve, which are opposite to the others. However, it makes no difference because the curve direction defines the "inside" and "outside" of a closed curve in two-dimensional space. Curves C_3 and C_4 are from the same group as a pair. The region between them is their inaccessible region.

A walking and selecting algorithm is proposed to combine accessible regions efficiently. As shown in Fig. 10, it looks like a man walking along the positive direction of each closed curve. Whenever a new intersection point is encountered, the positive direction of each followed segment provides an optional direction to go ahead. Then the rightmost segment is selected as the new walking route. Here, the rightmost segment is defined as follows. Suppose there are two segments following a current intersection point P_C along each positive direction. From point P_C, the next critical points on the first and second segments are points $P_{N,1}$ and $P_{N,2}$, respectively. If D_R in Eq. 17 is greater than zero, the first segment is the rightmost one; otherwise, the second one is.

$$D_R = (P_C - P_{CL}) \cdot \big((P_{N,1} - P_C) \times (P_{N,2} - P_C)\big) \qquad (17)$$

The algorithm starts at an arbitrary intersection point. In Fig. 10, two intersection points are selected as the start points to show examples. In the walking route, all encountered intersection points are marked to avoid the walk falls into an endless loop. When an intersection point is encountered at twice, the walk stops and the critical points between the twice appearances of this point are stored as a new closed curve. The accessible region is located on the right-hand side when walking along these points. After that, another intersection point is selected to start a new walk. Whenever a marked point is selected or encountered, the new walk stops without storing any points because the following route has been walked. When all intersection points are checked like this, the accessible regions are finally solved. The solved accessible regions are shown in Fig. 11. Starting at any intersection points in Fig. 10, the solved accessible region will be one of them.

The accessible regions solved with this algorithm are usually the final results. In very rare cases, some regions might be wrongly included. Thus, a simple checking algorithm is added after the walking and selecting algorithm to

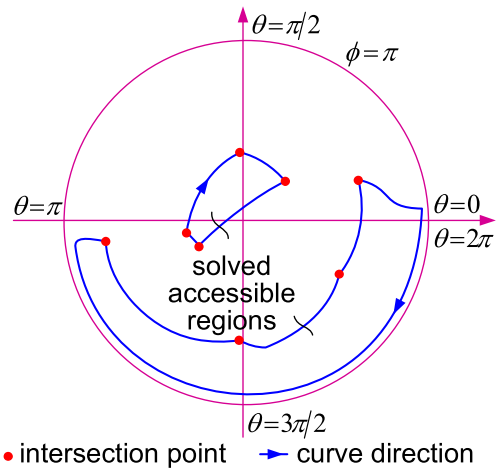


Fig. 11 Solved accessible regions

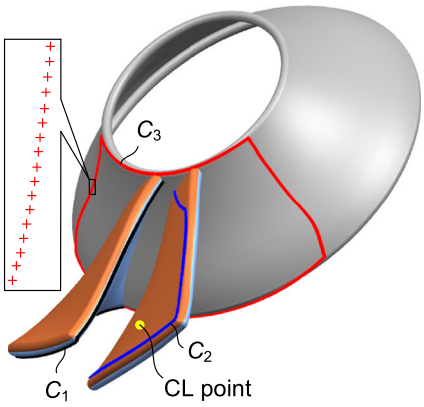
check the validity of all solved accessible regions. To check the validity of a region, an arbitrary point located inside of this region is compared with the inaccessible region of each original blade/hub surface. A line through this point intersects with the polygon of each inaccessible region. Then the parity of intersection points on both sides of this point is used to check whether this point is located inside of each region. If this point is located in none of these inaccessible regions, the current region is valid; otherwise, the current region is removed. This checking algorithm is almost unused in our simulations. However, it is added to guarantee the robustness of the algorithm.

5 Algorithm verification

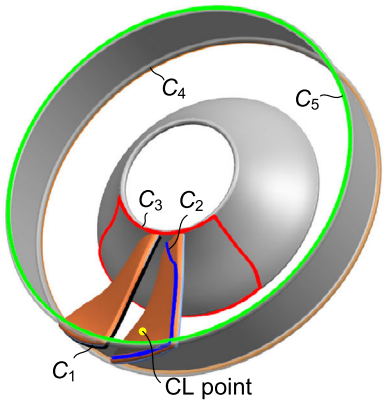
5.1 Applications

The proposed algorithm is applied to calculate the accessible regions for an open blisk and a closed blisk. The open blisk in applications is shown in Fig. 1, while the closed blisk is converted from this open blisk by adding an outer hub to demonstrate the influence of checking surface on the accessible regions. These blisks are 166 mm high with 15 blades. The diameter of the outer hub is 910.2 mm, while the diameter of the inner hub varies from 324.1 to 596.6 mm. The cutter in applications is a ball-end mill with diameter of 20 mm. The suction surfaces, pressure surfaces, and (inner) hub surface are related to the accessible region for the open blisk. They are offset along the inward direction of a channel in Siemens NX 7.5. The suction surfaces, pressure surfaces, inner hub surface, and outer hub surface are offset for the closed blisk. The offsets of these surfaces are exported into an IGES (Initial Graphics Exchange Specification) file as "rational B-spline surface" entities. Then they are used in MATLAB to calculate accessible regions.

The small positive quantity ε for dot product Q_i is set to be 1.745×10^{-5} in these applications. It represents that when the angle between view direction and surface normal at a point is within $(90 \pm 10^{-3})°$, this point is considered to be a critical point. The constant step length L_S is set to be 0.1 mm. The searched critical points for the two blisks are imported into Siemens NX 7.5 to show them in Fig. 12. Notations C_1, C_2, and C_3 represent the critical points searched on pressure surface, suction surface, and inner hub surface for the open blisk, respectively. For the closed blisk, a pair of closed curves can be obtained from the outer hub surface, which are denoted as C_4 and C_5. The region between them is their inaccessible region. The searched critical points are then mapped into $\theta\phi$ coordinate system and drawn in Fig. 13 with MATLAB. Here, notations C_1–C_5 are still used to represent



(a) Open blisk



(b) Closed blisk

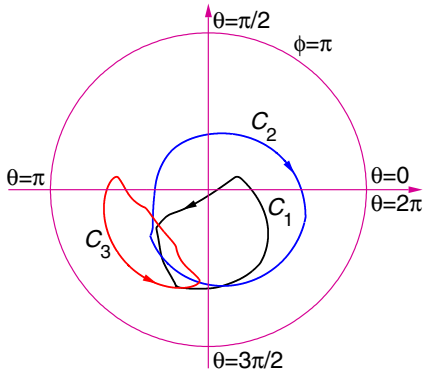
Fig. 12 Searched critical points in applications

the mapped points in $\theta\phi$ coordinate system. The arrow on each curve demonstrates the positive direction of this closed curve.

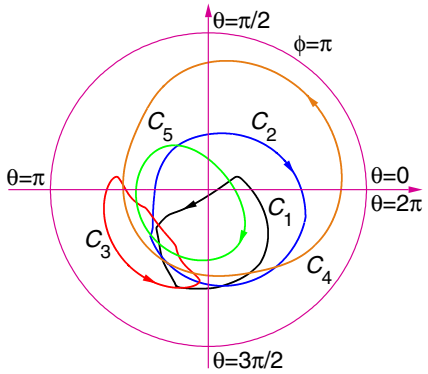
The final accessible regions solved with the proposed method are shown in Fig. 14. There is one accessible region for the open blisk in this example, while there are two accessible regions for the closed blisk. The only geometrical difference in two examples is that the closed blisk has an outer hub. On the one hand, it cuts off the accessible region for the open blisk into two parts; on the other hand, it removes part of accessible region when it acts as an obstacle in the closed blisk. When walking along the direction shown with the arrows in Fig. 14, each accessible region is located on the right-hand side.

5.2 Analysis

The proposed method is implemented in MATLAB R2014a in a 64-bit operating system with an Intel(R) Core(TM) i5-3320M CPU with 2.60 GHz processor. Running time and calculation errors in the examples are used to evaluate the proposed algorithm in solving accessible regions. The total running time for the open blisk and closed blisk is 24.3 and

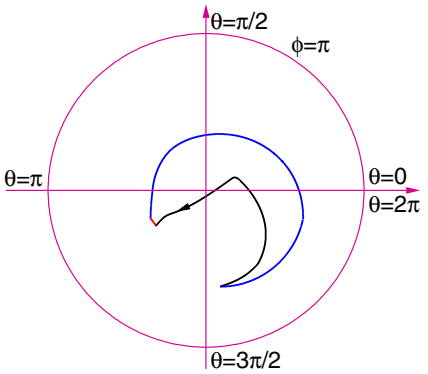


(a) Open blisk

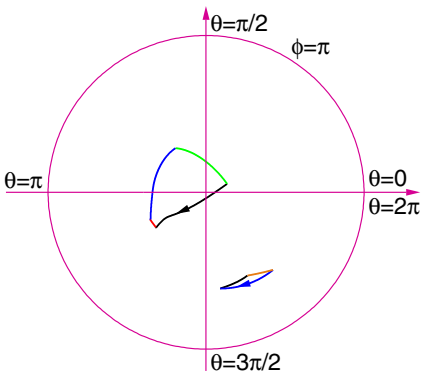


(b) Closed blisk

Fig. 13 Critical points expressed in $\theta\phi$ coordinate system



(a) Open blisk



(b) Closed blisk

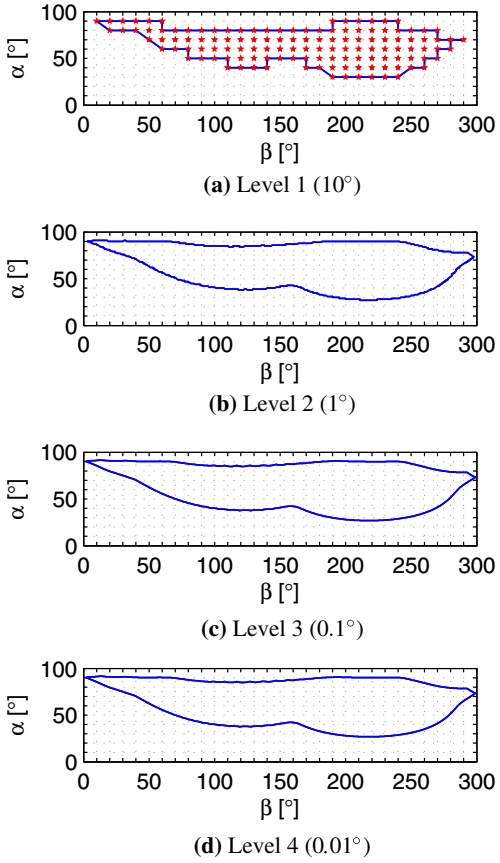
Fig. 14 Solved accessible regions in applications

86.7 s, respectively, as shown in Table 1. The algorithm to search critical points is the key part in the proposed method. In the examples, 35,751 points are searched as the critical points for the open blisk in 7.8 s, which is 32.1 % of the total running time. By contrast, 93,534 points are searched for the closed blisk in 12.6 s, which is 14.5 % of the total running time. The running time to combine the accessible regions takes 0.4 and 0.6 s for the open blisk and closed blisk, respectively. The most time-consuming part is the algorithm to find and calculate intersection points for all segments. Although the vectorization codes are used in MATLAB, it still consumes 12.1 and 68.5 s to check the existence of an intersection point for each line pair in two examples, respectively. When enlarging the constant step length, the running time will be shortened because the searched points are reduced. The boundary points solved with the proposed method to represent the accessible regions are 11,326 and 15,467 for the open and closed blisk, respectively.

The calculation errors are evaluated in two aspects: distance error in searching with constant step length and angle error in calculating dot product. Since the critical points searched for the open blisk are all included in those for the closed blisk, the calculation errors are checked for the latter one. One thousand pairs of random points are selected to calculate the distance error. The mean error, maximum error, and standard deviation are 2.26×10^{-7}, 4.63×10^{-5}, and 1.45×10^{-5} mm, respectively. It can be seen that the distance error between adjacent searched points in this algorithm is at most 0.046 % of the specified constant step length (0.1 mm). The actual step lengths can be considered constant as expected. The angle error in calculating dot product reflects the accuracy of solved noninterference tool

Table 1 Results of the proposed method

	Searched points (num.)	Boundary points (num.)	Running time (s)		
			Searching points	Combining regions	Total
Open blisk	35,751	11,326	7.8	0.4	24.3
Closed blisk	93,534	15,467	12.6	0.6	86.7



(a) Level 1 (10°)

(b) Level 2 (1°)

(c) Level 3 (0.1°)

(d) Level 4 (0.01°)

Fig. 15 Accessible regions at different levels solved with the referenced method

orientations. When the angle between the view direction and the surface normal is equal to 90°, the checked point will be located on a profile. The mean error, maximum error, and standard deviation of the angles calculated at 2000 critical points are (6.56×10^{-5}), (8.92×10^{-4}), and $(4.27 \times 10^{-4})°$, respectively. The checked angle error is less than the specified limitation $\pm(10^{-3})°$, which is also the round-off error for most machining centers.

5.3 Comparisons

The proposed approach is compared with the edge detection algorithm presented by Jun et al. in reference [28]. In

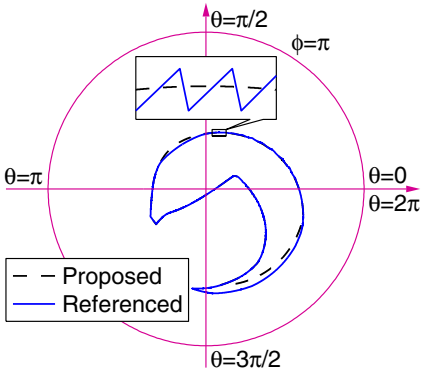


Fig. 16 Comparison of accessible regions solved with two methods

their work, the accessible region was first constructed by collision checking between a cutter and triangular facets. Then the boundaries of accessible region were searched by turning right when the candidate point is feasible, while turning left when not feasible. To compare two approaches, the same open blisk and cutter are used to implement the referenced approach in MATLAB R2014a. The original surfaces of the open blisk shown in Fig. 12 are exported from Siemens NX 7.5 into an STL (STereoLithograhpy) file to express them as triangular facets. The number of exported facets is 44,836 when the triangle tolerance and adjacency tolerance are all set as 0.0025 mm. The (α, β) domain is divided into four levels of grids, with angle interval of 10, 1, 0.1, and 0.01°, respectively. Here, $\alpha(0° \leq \alpha \leq 90°)$ is the tilt angle, while $\beta(0° \leq \beta < 360°)$ is the yaw angle. They were defined in reference [28] to represent tool orientations in the C-space. The collision is detected by checking the distances between the tool axis and parts of triangular facets. These facets are under the projection of the inclined cutter as introduced in reference [28]. The tolerance used in distance checking is 10^{-5} mm. The accessible regions at different levels solved with the referenced method are shown in Fig. 15. Here, the feasible orientations at the first level are shown with pentagons. The difference between the third and fourth levels is hard to be distinguished in view of the presented size of figures.

Running results of the referenced algorithm are listed in Table 2. When the angle interval is set as 10° at the first

Table 2 Results of the referenced method

	Angle interval (°)	Collision detection (num.)	Boundary points (num.)	Running time (s)		
				Detecting collision	Searching boundaries	Total
Level 1	10	3.25×10^2	6.70×10^1	0.5	0.2	1.1
Level 2	1	1.38×10^4	6.63×10^2	12.5	11.5	27.4
Level 3	0.1	1.48×10^5	6.78×10^3	226.7	831.4	1,079.8
Level 4	0.01	1.97×10^6	6.80×10^4	3,714.1	74,018.2	78,409.2
Final	0.01	2.13×10^6	6.80×10^4	3,953.8	74,861.3	79,517.5

Table 3 Comparisons of two methods

	Collision detection (num.)	Boundary points (num.)	Running time (s)	Angle tolerance (°)	Maximum angle error	
					Absolute (°)	Relative
Referenced method	2.13×10^6	6.80×10^4	79,517.5	10^{-2}	1.41×10^{-2}	141.0 %
Proposed method	0	1.13×10^4	24.3	10^{-3}	8.92×10^{-4}	89.2 %

level, 3.25×10^2 tool orientations are checked in collision detection in 0.5 s, while 6.70×10^1 points are found as the boundaries of accessible region in 0.2 s. Total running time at this level is 1.1 s. When the angle interval is refined as 0.01° at the fourth level, the collision detection is conducted along 1.97×10^6 tool orientations in 3,714.1 s at this level, while 6.80×10^4 points are found in 74,018.2 s to construct the boundaries of accessible region. This procedure takes a long time because for-loops are used in MATLAB to check tool orientations in the stored feasible C-space and determine whether to turn right or left. Total running time at this level is 78,409.2 s. However, to get the results at this level, previous levels need to be conducted to get the relative coarse grids. The final running time when the angle interval is set as 0.01° should include all running time at previous levels, which is 79,517.5 s.

The final solved accessible region with the referenced method is shown in Fig. 16 in $\theta\phi$ coordinate system to compare it with the method proposed in this paper. It can be seen that the results of two methods have a little deviation in some regions. This is because the local facets around cutter contact point affect the distance checking with the referenced method in our implementation. Since local interference is not considered in this paper, the parts of boundaries that are obviously free of local interference are selected to compare the results of two methods. The angle between the searched tool orientations and related accurate ones are also calculated at 2000 boundary points with the referenced method. The mean error, maximum error, and standard deviation are (8.32×10^{-3}), (1.41×10^{-2}), and $(1.07 \times 10^{-2})°$, respectively. The relative angle error is used to compare the accuracy of two methods by dividing the absolute angle error by the angle interval (or angle tolerance). As shown in Table 3, the maximum relative angle error with the referenced method is 141.0 % in our implementation, while it is 89.2 % with the proposed method. The defined angle tolerance is not guaranteed with the referenced method since the angle interval is greater than the defined angle tolerance when angles α and β are both different in two adjacent tool orientations. By contrast, the defined angle tolerance is used as the abort condition in the iterative search process with the proposed method, which guarantees the searched tool orientations will satisfy the defined angle tolerance. With the referenced method, 2.13×10^6 tool orientations are checked in collision detection even though the layer-by-layer refining process is adopted. The proposed method calculates accessible region without collision detection, which avoids the contradiction of efficiency and accuracy as mentioned before. Fewer points (1.13×10^4) are used in the proposed method to present the boundaries of accessible region, even with a smaller angle tolerance. They are more efficient in expressing the boundaries of accessible region because they are actually the sample points on these boundaries, while those searched by the referenced method are used to approximate the boundaries with polygonal chain, as shown in Fig. 16. Running time of the proposed method (24.3 s) is obviously shorter than the referenced method (79,517.5 s), which are compared in Table 3. Once a searched point is stored in an array with the proposed method, it is rarely used again. On the contrary, plenty of tool orientations are used time after time to conduct collision detection and query their feasibility in a large array within for-loops, which takes a long time. By comparing the calculation errors, the number of collision detection, the number of solved boundaries points, and running time, the proposed method has its advantages in efficiently and accurately solving the accessible regions in multi-axis milling. It is worth pointing out that these comparisons are based on our understanding and implementation of the two methods. The results might be different when improved codes are used.

6 Conclusion and outlook

An original approach is proposed to solve accessible regions in multi-axis milling of blisks with a ball-end mill. Compared with discretization methods and subdivision methods, only the critical points that are related to the accessible regions are concerned in this method. The main contributions of this work include that (a) a search approach to solving accessible regions without interference detection, (b) a two-stage algorithm to solve critical points with a constant step length, and (c) a walking and selecting algorithm to combine accessible regions. The proposed method has been verified to be efficient and accurate in solving accessible regions. This method can be applied in solving accessible regions of impellers, marine propellers, or other complex parts in industry. The future work will be

dedicated to dealing with the case that a profile does not start at a boundary to improve the robustness of the proposed algorithm. Nevertheless, the proposed method is still a novel perspective to search the accessible regions without interference detection.

Acknowledgments This work was supported by the National Science and Technology Major Project on CNC Machine tool, China (No. 2014ZX04012013).

References

1. Rao N, Ismail F, Bedi S (1997) Tool path planning for five-axis machining using the principal axis method. Int J Mach Tools Manuf 37(7):1025–1040
2. Gray P, Bedi S, Ismail F (2003) Rolling ball method for 5-axis surface machining. Comput-Aided Des 35(4):347–357
3. Gray PJ, Ismail F, Bedi S (2007) Arc-intersect method for $3\frac{1}{2}\frac{1}{2}$-axis tool paths on a 5-axis machine. Int J Mach Tools Manuf 47(1):182–190
4. Chiou JCJ, Lee YS (2005) Optimal tool orientation for five-axis tool-end machining by swept envelope approach. J Manuf Sci Eng 127(4):810–818
5. Warkentin A, Ismail F, Bedi S (2000) Multi-point tool positioning strategy for 5-axis machining of sculptured surfaces. Comput-Aided Geom Des 17(1):83–100
6. Fan W, Wang X, Cai Y, Jiang H (2012) Rotary contact method for 5-axis tool positioning. J Manuf Sci Eng 134(2):1–6
7. Kim YJ, Elber G, Barto M, Pottmann H (2015) Precise gouging-free tool orientations for 5-axis CNC machining. Comput-Aided Des 58:220–229
8. Chen KH (2011) Investigation of tool orientation for milling blade of impeller in five-axis machining. Int J Adv Manuf Technol 52(1):235–244
9. Liang Y, Zhang D, Chen ZC, Ren J, Li X (2014) Tool orientation optimization and location determination for four-axis plunge milling of open blisks. Int J Adv Manuf Technol 70 (9):2249–2261
10. Han FY, Zhang DH, Luo M, Wu BH (2015) An approach to optimize the tilt angle of indexable table for nonorthogonal four-axis milling of impeller. Int J Adv Manuf Technol 76 (9):1893–1904
11. Lin Z, Shen H, Gan W, Fu J (2012) Approximate tool posture collision-free area generation for five-axis CNC finishing process using admissible area interpolation. Int J Adv Manuf Technol 62(9):1191–1203
12. Chu CH, Huang WN, Li YW (2012) An integrated framework of tool path planning in 5-axis machining of centrifugal impeller with split blades. J Intell Manuf 23(3):687–698
13. Chen L, Xu K, Tang K (2015) Collision-free tool orientation optimization in five-axis machining of bladed disk. J Comput Des Eng 2(4):197–205
14. Ding S, Mannan MA, Poo AN (2004) Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces. Comput-Aided Des 36(13):1281–1294
15. Tang TD, Bohez ELJ, Koomsap P (2007) The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC machining. Comput-Aided Des 39(11):1012–1024
16. Wang QH, Li JR, Zhou RR (2006) Graphics-assisted approach to rapid collision detection for multi-axis machining. Int J Adv Manuf Technol 30(9):853–863
17. Bi QZ, Wang YH, Ding H (2010) 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 48(4):1105–1124
18. Ding H, Bi Q, Zhu L, Xiong Y (2010) Tool path generation and simulation of dynamic cutting process for five-axis NC machining. Chin Sci Bull 55(30):3408–3418
19. Chuang LC, Young HT (2007) Integrated rough machining methodology for centrifugal impeller manufacturing. Int J Adv Manuf Technol 34(11):1062–1071
20. Wu B, Zhang D, Luo M, Zhang Y (2013) Collision and interference correction for impeller machining with non-orthogonal four-axis machine tool. Int J Adv Manuf Technol 68(1):693–700
21. Balasubramaniam M, Sarma SE, Marciniak K (2003) Collision-free finishing toolpaths from visibility data. Comput-Aided Des 35(4):359–374
22. Lee YS, Chang TC (1995) 2-Phase approach to global tool interference avoidance in 5-axis machining. Comput-Aided Des 27(10):715–729
23. Hsueh YW, Hsueh MH, Lien HC (2007) Automatic selection of cutter orientation for preventing the collision problem on a five-axis machining. Int J Adv Manuf Technol 32(1):66–77
24. Wang N, Tang K (2007) Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath. Comput-Aided Des 39(10):841–852
25. Li LL, Zhang YF (2006) Cutter selection for 5-axis milling of sculptured surfaces based on accessibility analysis. Int J Prod Res 44(16):3303–3323
26. Morishige K, Kase K, Takeuchi Y (1997) Collision-free tool path generation using 2-dimensional C-space for 5-axis control machining. Int J Adv Manuf Technol 13(6):393–400
27. Lu J, Cheatham R, Jensen CG, Chen Y, Bowman B (2008) A three-dimensional configuration-space method for 5-axis tessellated surface machining. Int J Comput Integr Manuf 21(5):550–568
28. Jun CS, Cha K, Lee YS (2003) Optimizing tool orientations for 5-axis machining by configuration-space search method. Comput-Aided Des 35(6):549–566
29. Lasemi A, Xue D, Gu P (2010) Recent development in CNC machining of freeform surfaces: a state-of-the-art review. Comput-Aided Des 42(7):641–654