Int J Adv Manuf Technol (2015) 81:1247–1258
DOI 10.1007/s00170-015-7293-x

ORIGINAL ARTICLE

A new collision avoidance strategy and its integration with collision detection for five-axis NC machining

Tran Duc Tang[1] · Erik L. J. Bohez[2]

Received: 20 July 2014 / Accepted: 8 May 2015 / Published online: 20 May 2015

Abstract This paper presents a new collision avoidance strategy and its integration with the collision detection for five-axis NC machining that improves upon the earlier collision detection algorithm developed by the present authors [T.D. Tang, Erik L.J. Bohez, Pisut Koomsap (2007) The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC machining. Computer-Aided Design 39(11):1012–1024]. The proposed algorithm automatically detects and corrects the collision based on the biggest collision boxes. The collision detection algorithm is based on the bounding volume and the sweep plane approach. The collision is firstly detected by using the bounding sphere algorithm, and the colliding spheres are then further checked with the sweep plane algorithm. The change of the workpiece geometry is included in the detection process. After the collision detection, the collision data are stored. Only the biggest collision boxes (the boxes with the biggest edge in the X, Y, or Z direction) for each type of collisions are stored. The collision avoidance algorithm corrects the biggest collision based on heuristic strategy. With this strategy, when the biggest collision is corrected, most of other collisions will disappear automatically. Therefore, the time complexity of the collision avoidance strategy is considerably reduced. The algorithm has been implemented in Visual C++ and OpenGL, demonstrated for five-axis machine with two rotary axes on the table (Deckel MAHO 600e) and can be customized to apply for any five-axis CNC machines.

Keyword Five-axis NC machining · Collision detection · Collision avoidance · Machine simulation

✉ Tran Duc Tang
tangtd@mta.edu.vn

[1] Le Quy Don Technical University, 236 Hoang Quoc Viet Str., Cau Giay Dist., Hanoi, Vietnam

[2] Asean Institute of Technology (AIT), P.O. Box 4, Klongluang, Pathumthani 12120, Thailand

1 Introduction

A five-axis machine has two extra rotational axes over a three-axis machine. These two rotational axes provide users with great flexibilities and make it possible to machine high quality complicated free-form shapes which were previously un-machinable [1]. Therefore, five-axis CNC machines are widely used in machining of sculptured surfaces such as turbine blades, impeller, aerospace parts, molds, and dies [2]. However, because of the two additional rotation axes, collisions are prone to occur in five-axis machining [3]. These collisions may occur between the tool (or toolholder) and the workpiece, fixtures (or machine components), between the workpiece and machine components, or between moving machine components. These collisions are considered to be serious as accidents may happen if such collision occurs; besides the unacceptable workpiece machined, it may damage the cutter, fixtures, and the machine structure itself [4]. Therefore, the collision detection and avoidance is one of the important problems in five-axis NC machining; to take full advantage of five-axis machining such collision detection and avoidance problem must be solved.

There have been a lot of research done on collision detection and collision avoidance, attempting to solve the problem:

Based on the curvature matching technique, Chen et al. [5] detected the local gouging and then determined an optimal cutter orientation by matching the instantaneous cutting profile of the cutter and the machined surface as close as possible.

Rear gouging detection and avoidance were implemented by calculating the intersection between the offset cylinder of the cutter and the offset surface of the machined surface for a simple smooth surface. Whereas for the complex shaped surfaces, a search for possible rear gouging is conducted by firstly dividing the machined surface into a set of triangular facets and then classifying the relative position of the bottom plane of the cutter and the vertices of the triangular facets. If one of the vertices of the triangles under the cutter shadow is above the bottom plane of the cutter, rear gouging will occur. If so, the cutter orientation is adjusted to eliminate gouging. Similarly, Du et al. [6] used the method of exact curvature matching between the cutter and part surface to avoid local cutter gouging at the cutter contact point. Rear gouging detection is performed by accurately judging the position of the each valid checking point relative to the bottom plane of the cutter. The method can be used in three-, four-, or five-axis machining. Wang et al. [7] proposed a new three-dimensional (3D) curvature match and curvature gouge detection and elimination method for five-axis CNC machining of sculpture surface. The method can be applied to all three types of commonly used milling cutter (end, torus and spherical mills), and all concave curved surfaces.

In Kim et al.'s research [8], the cutter interference free orientation is selected by the collision detection between the axis line of the cutter and the offset surface (offsetting the design model by the amount of the cutter radius). This is a simpler process than a collision check between the cutter model and the design model. However, it can only be applied for a ball endmill.

Gain et al. [9] used the concept of open regions and vector fields to determine the appropriate tool orientation that prevents collisions in five-axis machining of cavity regions with undercut areas.

Morishige et al. [10] apply the configuration space (C-space) method to five-axis machining. The idea of the C-space method is to represent a moving object as a point in an appropriate space in which the obstacles are mapped. This mapping transforms the complex problem of planning the motion of an object into a simpler one of planning the motion of a point in the C-space. In Morishige's approach, the cutting tool is the moving object and the workpiece is the obstacle, and collision-free access can be inferred by simply navigating the point in the C-space around the obstacles in the C-space. To deal with the collision avoidance problem in five-axis machining, Morishige employed a 2D C-space to enable automatic judgment of the possibility of collision avoidance at each cutting point. In the case where the whole definition area on the C-space is occupied by the projected collision areas, collision avoidance is considered to be impossible at any cutting point. On the other hand, if some free areas exist on the C-space, the process of collision avoidance is immediately carried out. The point corresponding to the tool with collision is modified to the nearest point on the boundary between the collision area and the free area in the C-space. This modified point indicates a new pair of parameters of the tool orientation for which the tool is free of collision.

Zhiwei et al. [11] utilized the idea of admissible area interpolation for the whole designed free-form surface and proposed an algorithm to generate tool posture collision-free area for the free-form surface during five-axis CNC finishing period. The algorithm consists of two phases. In the first phase, a few points are picked on the surface, and the admissible area of tool posture is calculated at each point. In the second phase, the admissible area of the sampling points is interpolated with cubic B-surface interpolation technique, forming an expression as $Q(u,v)$, through which, when the parameters u and v are assigned, the global collision-free area for the corresponding surface point can be easily calculated. The algorithm can detect any kind of global collision between the cutting tool and a workpiece surface and can be extended to any kind of cutting tool.

Based on the distance calculation method, You and Chu [12] firstly subdivide the surfaces to be machined into discrete sample points. Then, the tool interference detection is conducted between discrete tool positions, and these sample points on the machined surface by calculating the relative distance from the sample points and the tool axis. Before the calculation, the tool geometry is projected onto the XY plane, and the sample points which are covered in the shadow of the tool projection are considered as possible interference points. To locate these points, a bounding box is calculated to approximate the tool projection area, and only sample points in this bounding box need further interference detection. In this method, since the interference detection is conducted only at discrete sample points, not for the polyhedral model, therefore, it is possible that the cutter does not interfere with the discrete sample points while it protrudes into the surface between the sample points.

Kiswanto et al. [13] have detected gouges by projecting the triangles of the faceted model and the cutting tool to a projection plane (e.g., the XY plane of the workpiece coordinate system). On the projection plane, the region covering all the triangles which are intersected by the shadow of the extended cutting tool projection is called possible interference region (PIR). All triangles in the PIR are then transferred to the tool coordinate system. Only triangles which are in the interference region are really checked for gouging at the rear-bottom side of the cutting tool and near the CC (cutter contact) point. In the proposed algorithm, if five-axis milling is using constant tool orientation, then gouging is eliminated by simply lifting the tool immediately when the gouging occurs. If varying optimal tool orientation is applied, the cutting tool will firstly try to find the minimum rotation angle to avoid gouging. If the gouging still occurs, e.g., when the specified maximum inclination angle at a CC point is exceeded, then the tool is lifted. In both methods, the tool is lifted until it is gouge free.

Wang et al. [14] proposed a graphics-assisted approach to rapid collision detection for multi-axis machining by integrating machining environment culling and a two-phase collision detection strategy. In this approach, the machining environment is initially subdivided and organized hierarchically using a binary space partitioning (BSP) tree structure. The machining environment is culled conservatively to remove the irrelevant geometries from further checking steps. Then, in the precise collision detection stage, a two-phase collision detection strategy is used. In the first phase, based on the extracted portions of machining environment and the viewing volumes which are modeled according to the cutting tool's geometry, a collision map is generated to determine the collision. The second phase of detection will be invoked only for some ambiguous cases, where vector-based calculation is adopted to precisely check the collision between the cutting tool geometry and a couple of ambiguous triangles which have been located by phase one. Since this approach utilizes the graphics processing unit in graphics hardware to assist computation in its main steps, the computational efficiency of the algorithm is improved. However, checking the accuracy of the algorithm is subject to the size of the cutting tool and the graphics hardware's specification. In addition, the algorithm only detects collisions between the cutting tool and workpiece or machine parts, collisions among machine parts was not taken into account.

Today, many commercially available CAM software for five-axis machining such as Mastercam, UGS NX, CATIA, and Delcam's PowerMILL provide gouge detection and correction. However, an intensive user interaction is still needed while using this software to avoid collisions. These CAM systems are unable to accomplish the collision avoidance autonomously.

MasterCAM does not perform collision avoidance on the machine level. Actually, the software only avoids collisions between the tool and the stock or chuck by simply defining an area around the stock or chuck to allow the tool to retract to a safe position outside of this area.

UGS NX offers the full 3D motion of the machine tool and allows the system to detect and respond to collisions with the machining part and check the geometry of the part. If a collision is detected, it may modify the tool path to eliminate the collision by lifting along the tool axis to a safe level or alter the tool path to contour around the geometry to avoid the collision while maintaining the desired travel.

In CATIA, collisions between the tool or toolholder and part or fixtures are detected and graphically visualized. The system provides several display options such as the collision point, tool, or toolholder sweep for the cut which caused the collision or the tool position at the start and end of the cut which caused the collision. Collisions are avoided by a dynamic inclination of the tool axis. Users can interactively edit machining operations or modify tool paths.

Delcam's PowerMILL offers gouge checked and detects collisions of the toolholder and machine tool. It tilts the tool away from an obstacle by a specified clearance. Once the obstacle is cleared, the tool returns to the original cutting angle.

VERICUT is a machine simulation and verification software (not real CAM system) that provides realistic 3D simulation of entire CNC machines and detect collisions between all machine tool components such as rotary tables, spindles, workpiece, and fixtures. However, the task of collision avoidance is left to the user.

A state-of-the art review on the algorithms for collision detection and avoidance for five-axis NC machining has been also carried out by the author in the previous work [15]. Through the analysis and comparison of algorithms, it is clear that most of available algorithms in the literature only focused on collision detection and avoidance at the finishing operation; especially, the change of the workpiece geometry during the machining process was not taken into account. Most of the algorithms only detect the collision between the tool and workpiece; possible collisions between the machine and part, the machine and tool, or among moving machine components as well as complete machine modeling have not been considered yet.

In earlier research, the author has proposed the sweep plane algorithm for global collision detection for five-axis NC machining [16]; the collision avoidance has not been proposed yet. This paper presents a new collision avoidance strategy and its integration with the above sweep plane collision detection algorithm. The proposed algorithm automatically detect and correct the collision based on the biggest collision boxes. The algorithm takes into account not only collisions between the tool and the workpiece but also collisions between the other parts of the CNC machine; especially, the change of the workpiece geometry is included in the detection process.

The rest of the paper is organized as follows. An overview of the proposed algorithm is presented in Section 2. Section 3 presents the algorithm for collision detection. A new collision avoidance strategy is presented in Section 4. The time complexity of the proposed algorithm is discussed in Section 5, and Section 6 is the conclusion.

2 Algorithm overview

The flowchart of the proposed algorithm for collision detection and avoidance is presented in Fig. 1. The input data for the algorithm is a NC program that needs to be checked for collisions. The output is a NC program free of collisions or the information about the failure to find collision-free solutions.

The NC program is inputted into the algorithm. The algorithm reads the first block of the NC program and interpolates the swept motion between two consecutive blocks. Then, the algorithm simulates the movement of the machine axes

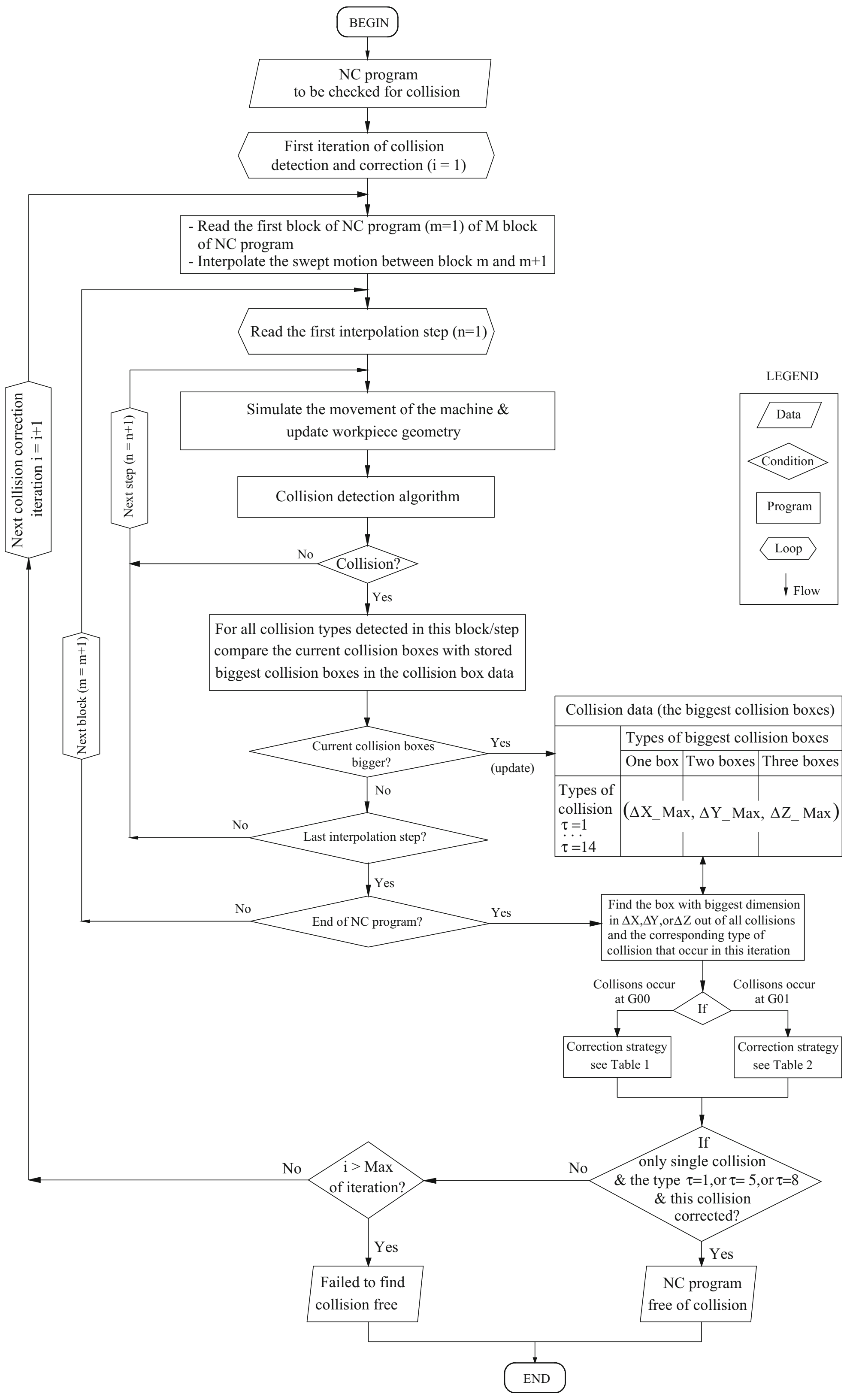

BEGIN
NC program to be checked for collision
First iteration of collision detection and correction (i = 1)
- Read the first block of NC program (m=1) of M block of NC program
- Interpolate the swept motion between block m and m+1
Read the first interpolation step (n=1)
Simulate the movement of the machine & update workpiece geometry
Collision detection algorithm
Collision?
No
Yes
For all collision types detected in this block/step compare the current collision boxes with stored biggest collision boxes in the collision box data
Current collision boxes bigger?
Yes (update)
No
Collision data (the biggest collision boxes)
Types of biggest collision boxes
One box
Two boxes
Three boxes
Types of collision τ=1 ... τ=14
(ΔX_Max, ΔY_Max, ΔZ_Max)
Last interpolation step?
End of NC program?
Next step (n = n+1)
Next block (m = m+1)
Next collision correction iteration i = i+1
Find the box with biggest dimension in ΔX,ΔY,orΔZ out of all collisions and the corresponding type of collision that occur in this iteration
Collisons occur at G00
Collisons occur at G01
If
Correction strategy see Table 1
Correction strategy see Table 2
If only single collision & the type τ=1,or τ= 5,or τ=8 & this collision corrected?
i > Max of iteration?
Failed to find collision free
NC program free of collision
END
LEGEND
Data
Condition
Program
Loop
Flow

Fig. 1 Flowchart of the new proposed algorithm for collision detection and avoidance

according to G-code block and updates the workpiece geometry. At the same time, the collision detection algorithm checks for possible collisions. If there is no collision, the algorithm goes to the next step. If there are collisions detected, the algorithm computes and compares the current collision boxes with stored biggest collision boxes in the collision data.

In case the current collision boxes are bigger, then these collision boxes are updated in the collision data. This process is repeated until the end of the NC program. Then, the algorithm searches for the box with the biggest dimension (ΔX, ΔY, or ΔZ max) and corresponding collision type in the collision data. If the collision occurs in the G-code (G00), the algorithm finds the corresponding solution as presented in Table 1. If the collision occurs in the G-code (G01), the algorithm finds the corresponding solution as presented in Table 2.

In the case that there is only a single collision and the type of collision is $\tau=1$ (the tool collides with *B*-axis), or type $\tau=5$ (the tool collides with *A*-axis), or type $\tau=8$ (the tool collides with the clamping), after the tool is retracted along the *Z*-axis, this single collision will be corrected without any need to further check for other collisions. Otherwise, the loop for checking and correcting the collision is repeated. After several iterations of correction (pre-defined by the user), if there is no collision-free solution, the system informs the user the failure to find collision-free solutions. The user then can interactively correct the collision, and return to the loop to check for collision again, or regenerate the CL file with CAM software.

3 Algorithm for collision detection

The algorithm is divided into two phases. In phase I, the bodies of the machine and workpiece are approximated by an octree of spheres. The collisions among these machine bodies are then conducted by checking interferences between their bounding spheres. If there is no interference between these bounding spheres, there is no collision between the enclosed bodies, and no further collision testing is needed. In contrast, the checking process is continued with phase II.

In phase II, the bodies contained within the colliding spheres are checked by using the sweep plane algorithm. These bodies are sliced by parallel planes called sweep planes. The intersections of the sweep planes with the tool and workpiece are computed to find out the slice polygons of the machined part. The collision detection is then conducted by checking the interference between these polygons. If there is any intersection between these slice polygons, a real collision between the machine bodies is found and finally reported for correction of the NC program.

Concretely, the STL files of all bodies are sliced in the reference position in the horizontal plane of the machine coordinate system (MCS) which is fixed to the machine frame. The workpiece is sliced in the vertical plane of the workpiece coordinate system (WCS) (Fig. 2). Then, each body is approximated by a hierarchy of bounding spheres. For each NC program block, the swept motion is interpolated in the small discrete steps based on the required accuracy that is compatible with the slice distance, and the position of all bodies and slices after each motion step is computed.

To calculate the geometry of the workpiece, a memory plane (M-plane) file is computed based on the CL (cutter location) file. The M-plane file contains all necessary information for computing the intersections of the sweep planes with the tool (the detail of the M-plane file is available in Ref. [17]). Then, at each incremental tool position, the intersections of the tool slices with corresponding workpiece slices are computed, and the slices of the updated workpiece (in WCS) are obtained. The location of the updated workpiece slices in MCS is finally calculated by the direct kinematics.

Most of the slices of the moving bodies stay parallel to each other, and checking the interference between these parallel slices is easily done by using Polygon Clipper [18]. In the case of MAHO 600e five-axis machine, only the slices of the workpiece, clamping, and *A*-axis do not stay parallel to the initial slicing direction (in the horizontal plane); the interference detection between these non-parallel slices is implemented by examining overlaps of their projections on the three perpendicular planes XY, YZ, and ZX. The detail of the sweep plane collision detection algorithm was presented in our previous paper [16].

4 Collision avoidance strategy

4.1 Collision geometry data structure

The collision data is stored as follows: At the current interpolation step, the biggest collision box or boxes (the boxes with the biggest edge in the X, Y, or Z direction) is computed for each type of collision. These current collision boxes are then compared with the biggest collision boxes which were stored over all previous blocks and steps to find out the biggest

Table 1 Collision avoidance strategy for the collisions that occur in G-code G00

Type of collision (1)	ΔX Max (2)	ΔY Max (3)	ΔZ Max (4)
Type of collision τ=1,2,3,5,6,7,8,9,10,11,12,14	Translate *X*-axis by ΔX (Insert extra block G00 X+ΔX Y Z A B)	Translate *Y*-axis by ΔY (Insert extra block G00 X Y+ΔY Z A B)	Translate *Z*-axis by ΔZ (Insert extra block G00 X Y Z+ΔZ A B)
Type of collision τ=4 (Updated workpiece/frame)	Rotate *B*-axis by ΔB		
Type of collision τ=13 (updated workpiece/*B*-axis)	Rotate *A*-axis by ΔA		

Table 2 Collision avoidance strategy for the collisions that occur in G-code G01

Type of collision (1)	ΔX Max (2)	ΔY Max (3)	ΔZ Max (4)
Type of collision τ=1, 5, 8,14 (tool collides with *B*-axis, or *A*-axis, or clamping, or workpiece)	Select alternative solutions of the inverse kinematics/ Change lead (lag) and/or tilt angle/ Combination of change setup and lead/lag and/or tilt angle		
Type of collision τ=2, 6, 9, 11 (toolholder collides with *B*-axis, or *A*-axis, or clamping, or workpiece)	Select alternative solutions of the inverse kinematics/ Change lead (lag) and/or tilt angle/ Combination of change setup and lead/lag and/or tilt angle		Extend the tool length/ Change lead (lag) and/or tilt angle/ Combination of change setup and lead/lag and/or tilt angle
Type of collision τ=3, 7, 10, 12 (*Z*-axis collides with *B*-axis, or *A*-axis, or clamping, or workpiece)	Change clamping position/ Change lead (lag) and/or tilt angle/ Combination of change setup and lead/lag and/or tilt angle		
Type of collision τ=4 (workpiece collides with frame)	Change setup/ Combination of change setup and lead/lag and/or tilt angle		
Type of collision τ=13 (workpiece collides with *B*-axis)	Change clamping position/ Combination of change setup and lead/lag and/or tilt angle		

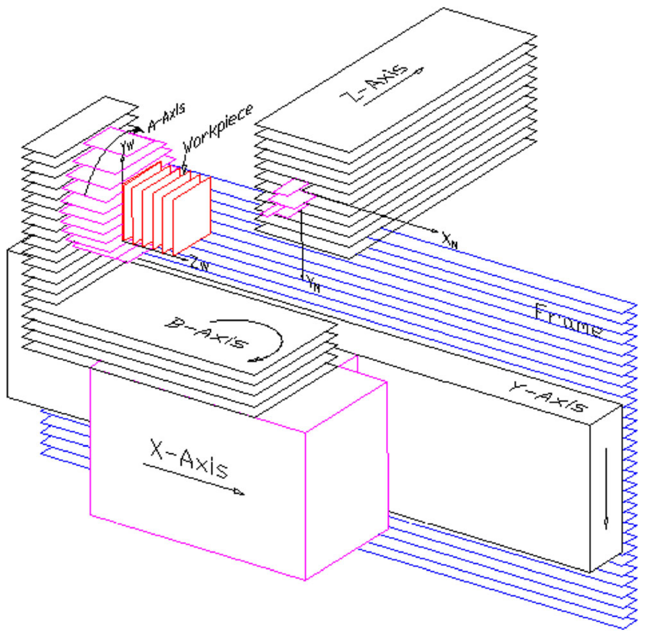


Fig. 2 Reference position and slicing direction

collision boxes up to the present step. The process is repeated until the end of the NC program. Finally, for each type of collision, only the biggest collision boxes are stored, the others are discarded. Therefore, for each type of collision, if collisions occur at multiple locations and steps, only the biggest collision boxes are stored. The time complexity of the collision avoidance strategy will be considerably reduced. This is because in many cases, when the biggest collision is corrected, the other collisions automatically disappear. The algorithm does not have to calculate and correct for all collisions. The pseudo code of the algorithm, to compute and store the biggest collision boxes for each type of collision, is given as follows

```
Pseudo code: computing and storing the collision data of two colliding bodies (denote: body1 and body2)
//body1 and body2 ∈ {machine bodies to be checked for collision}
{
For ($m$=1 to M) //M is the number of blocks of NC program
{
For ($n$=1 to N) //N is the number of steps between two consecutive blocks
{
Collision_Index=0; //Collision_Index is used to number collisions in the current interpolation step
For ($i$=1 to Number_of_slices_ body1)
{
For ($j$=1 to Number_of_slices_ body2)
{
If (slice $S_i$ intersects with slice $S_j$)
```

```
    {
    Store block number (m), step number (n), and X, Y, Z, A, B values, G00/G01;
    If (slice $S_{i-1}$ does not intersect with slice $S_{j-1}$)
    {
    Collision_Index=Collision_Index+1;
    }
    - Store the intersection polygon of two intersecting slices $S_i$ and $S_j$ (or intersecting polygons of the projections of two slices);
    - Calculate and add elements (Min_X_{i,j}, Max_X_{i,j}), (Min_Y_{i,j}, Max_Y_{i,j}), (Min_Z_{i,j}, Max_Z_{i,j}) of the intersection polygon to the set Collision[Collision_Index];
    - Compute the collision box of the intersecting slices of the set
    Collision[Collision_Index]; //this is to determine collision box of each collision
    }
    }
    }
    //After checking possible interferences of all slices at a step
    Compute the biggest collision boxes (in the X, Y, and Z directions) over all collisions (all Collision_Index) for the current step;
    Compare the biggest collision boxes of the current step with the biggest collision boxes which were computed over all previous blocks and steps and store the biggest ones;
    }
    }
    }
```

The storage of the biggest collision boxes for each type of collision is as below: if one collision box with three edges in the X, Y, and Z direction are biggest, then only one collision box is stored. If one box with two edges are biggest, the other edge is not, then two collision boxes need to be stored. In the case of three boxes, each box has one biggest edge (one box has the biggest edge in the X direction, one box has the biggest edge in the Y direction, and the other has biggest edge in the Z direction), then three collision boxes need to be stored. Figure 3 illustrates the case of three boxes.

4.2 Collision avoidance strategy

From the collision data finds the biggest collision box (ΔX, ΔY, or ΔZ max) and corresponding type (τ) of collision (type of collision $\tau \in${(1) Tool collides with *B*-axis, (2) Toolholder collides with *B*-axis, (3) *Z*-axis collides with *B*-axis, (4) Workpiece collides with Frame, (5) Tool collides with *A*-axis, (6) Toolholder collides with *A*-axis, (7) *Z*-axis collides with *A*-axis, (8) Tool collides with clamping, (9) Toolholder collides with clamping, (10) *Z*-axis collides with clamping, (11) Toolholder collides with update workpiece, (12) *Z*-axis collides with update workpiece, (13) Update workpiece collides with *B*-axis, (14) Tool collide with workpiece}). Based on the value of (ΔX, ΔY, or ΔZ max), a strategy for collision avoidance is proposed. There are three cases:

```
    If the biggest collision box is in the X direction ($\Delta X_\tau$ Max)
    {
    The collision avoidance strategy for collision type ($\tau$) is shown in Tables 1 and 2, column 2.
    }
    If the biggest collision box is in the Y direction ($\Delta Y_\tau$ Max)
    {
    The collision avoidance strategy for collision type ($\tau$) is shown in Tables 1 and 2, column 3.
    }
    If the biggest collision box is in the Z direction ($\Delta Z_\tau$ Max)
    {
    The collision avoidance strategy for collision type ($\tau$) is shown in Tables 1 and 2, column 4.
    }
```

The methods to remedy each type of collision corresponding to the biggest of (ΔX, ΔY, or ΔZ max) are presented in the following tables. The methods to correct the collisions that occur in G-code G00 are presented in Table 1. Table 2 shows the methods to correct the collisions that occur in G-code G01.

The collision correction strategy is as follows: suppose at a specific block and interpolation step in the G-code G00 of a NC program, the collisions occur between the workpiece and *B*-axis (type $\tau=13$) and between the tool and the workpiece (type $\tau=14$), as shown in Fig. 4a. It can be seen from Fig. 4a that the biggest collision dimension is ΔX—the intersection of the workpiece with *B*-axis in the *X* direction. This collision gets the highest priority for the correction. After the collision between the workpiece and *B*-axis is corrected, the collision between the tool and the workpiece automatically disappears, as shown in Fig. 4b. Therefore, by this strategy, the number of collisions that need to be corrected will be reduced.

The same strategy is applied for the collisions that occur in G-code G01. The biggest dimension (ΔX, ΔY, or ΔZ max) gets the highest priority for the collision correction. Based on the type of the collision, the corresponding correction method is chosen, as presented in Table 2. Six types of collision avoidance method are used in the algorithm: (1) select alternative solutions of the inverse kinematics, (2) extend the tool length,

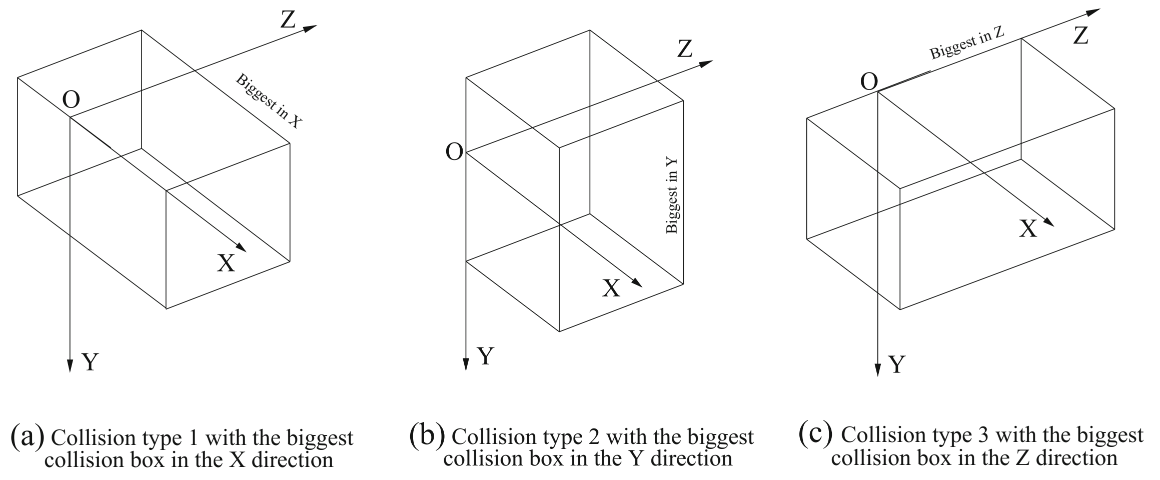

Z
O
Biggest in X
X
Y
(a) Collision type 1 with the biggest collision box in the X direction
Z
O
Biggest in Y
X
Y
(b) Collision type 2 with the biggest collision box in the Y direction
Biggest in Z
Z
O
X
Y
(c) Collision type 3 with the biggest collision box in the Z direction

Fig. 3 Biggest collision boxes (**a–c**)

(3) change clamping position, (4) change lead/lag or tilt angle, (5) change setup, and (6) combination of change setup and change lead/lag and/or tilt angle.

4.3 Example scenario

Suppose that a NC program consists of 1000 blocks, with the number of interpolation steps between two consecutive blocks is 10. Check this NC program for the collision and correct the collision. Suppose also that after the first iteration of the collision check, there are three collisions detected in the initial program, as below:

- At block 500th (m=500), step 3rd (n=3), there are two types of collision (as shown in Table 3):

  – The collision between the updated workpiece and B-axis (type τ=13); this collision has two boxes: the first one with ΔX=4, ΔY=2, ΔZ=5 (mm) and the second box with ΔX=3, ΔY=10, ΔZ=4 (mm).

  – The collision between the tool and the updated workpiece (type τ=14); this collision has one box with ΔX=5, ΔY=1, ΔZ=3 (mm).

- At block 650th (m=650), step 8th (n=8), there is one type of collision: the collision between the toolholder and B-axis (type τ=2); this collision has three boxes: the first one with ΔX=6, ΔY=2, ΔZ=4 (mm), the second box with ΔX=3, ΔY=7, ΔZ=2 (mm), and the third collision box with ΔX=5, ΔY=1, ΔZ=8 (mm), as shown in Table 3.

Before correcting the collisions, the collision table is as in Table 3.

The collision avoidance algorithm finds the biggest collision box in the collision data. The collision box of the collision between the updated workpiece and B-axis (type τ=13) with ΔY=10 is the biggest one. Therefore, this collision gets the highest priority for the avoidance. The solution for this type of collision is to rotate the A-axis (refer to Table 1). After this collision is corrected, the NC program is checked again.

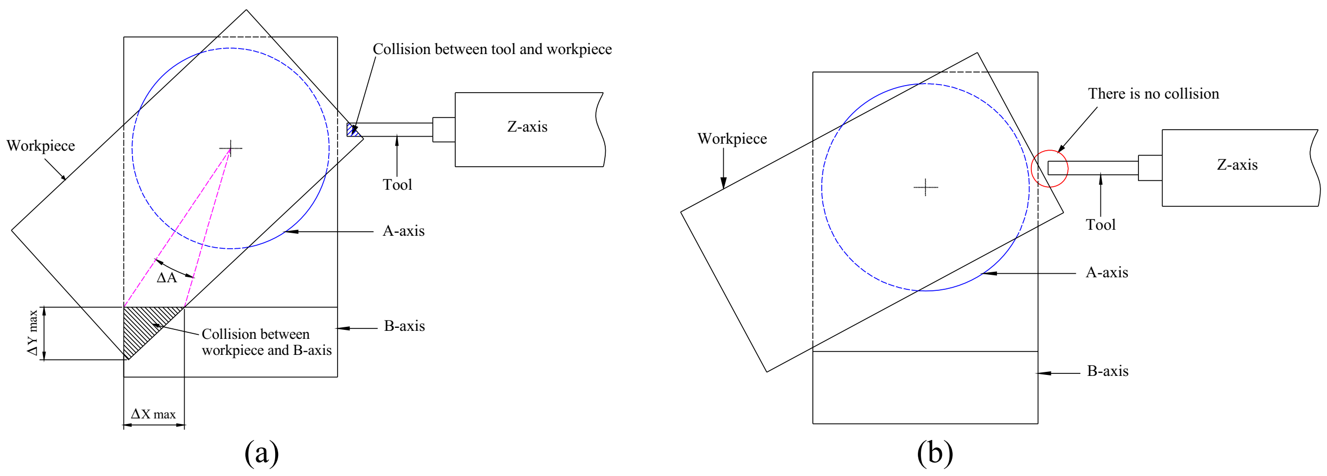
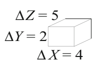
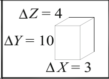
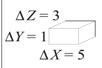
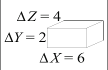
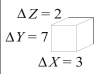
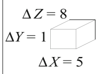


Fig. 4 **a, b** Collision avoidance strategy. **a** Collisions occur at two positions. **b** The smaller collision automatically disappears after the collision between the workpiece and B-axis has been corrected

Table 3 Table of the biggest collision boxes at the iteration $i=1$

Table of the biggest collision boxes (iteration i = 1)										
Block m	Step n	G code	Coordinate	Collision type	Types of the biggest collision boxes					
					One box	Two boxes		Three boxes		
500	3	G00	$X=100$ $Y=45$ $Z=-205$ $A=55$ $B=-5$	$\tau=13$		 $\Delta Z=5$ $\Delta Y=2$ $\Delta X=4$	 $\Delta Z=4$ $\Delta Y=10$ $\Delta X=3$			
				$\tau=14$	 $\Delta Z=3$ $\Delta Y=1$ $\Delta X=5$					
650	8	G01	$X=130$ $Y=85$ $Z=-245$ $A=30$ $B=5$	$\tau=2$				 $\Delta Z=4$ $\Delta Y=2$ $\Delta X=6$	 $\Delta Z=2$ $\Delta Y=7$ $\Delta X=3$	 $\Delta Z=8$ $\Delta Y=1$ $\Delta X=5$

The collision types $\tau=13$ (collision between the updated workpiece and B-axis) and type $\tau=14$ (collision between the tool and the updated workpiece) disappeared. The collision type $\tau=2$ (collision between the toolholder and the updated workpiece) still remains. The collision table after the first correction is shown in Table 4:

The collision correction algorithm finds the biggest collision box in the collision table. The biggest box for this collision is $\Delta Z=8$. And the avoidance algorithm finds the corresponding solution for this type of collision—extend the tool length (refer to Table 2). After the correction of this collision, the NC program is checked again, and in the third iteration, no more collision is detected.

4.4 Numerical example

The same example presented in [16] is continued here, given two consecutive blocks of NC program, N01 and N02:

N01	G01	X134.407	Y80	Z-244.294	A0	B-45
N02	G01	X110.273	Y35.355	Z-231.178	A-45	B-35.264

Its two CL points:

	x	y	z	i	j	k
CL point 1:	80	160	100	1	0	1
CL point 2:	80	130	130	−1	1	1

A collision between the toolholder and the workpiece has been detected at both blocks [16]. The biggest collision box found at block N02, with $\Delta X_{\max}=14.9408$, $\Delta Y_{\max}=51.7563$, $\Delta Z_{\max}=21.1297$, and the biggest dimension of ($\Delta X_{\max}$, $\Delta Y_{\max}$, $\Delta Z_{\max}$) is $\Delta Y_{\max}$. With the proposed collision avoidance strategy, the alternative solution of inverse kinematics is selected first (refer to Table 2). By doing the inverse kinematics, G-code blocks of the NC program for solutions 2, 3, and 4 are obtained as below:

Solution 2	N01	G01	X134.407	Y80	Z-244.294	A-360	B-405
	N02	G01	X110.273	Y35.355	Z-231.178	A-405	B-395.264
Solution 3	N01	G01	X-134.294	Y-80	Z-244.407	A-180	B225
	N02	G01	X-110.181	Y-35.355	Z-231.309	A-225	B215.264
Solution 4	N01	G01	X-134.294	Y-80	Z-244.407	A180	B-135
	N02	G01	X-110.181	Y-35.355	Z-231.309	A135	B-144.736

For the five-axis machine, MAHO 600e in the example, the range of the rotation B is $\pm 105^0$. In this example, the workpiece is set up at the position where the center line of the A-axis perpendicular to the Z-axis; therefore, the range of the rotation B must be $-195^0 \leq B \leq 15^0$, as shown in Fig. 5.

By checking the rotation limit of the B-axis, the solutions 2 and 3 are eliminated, and the alternative solution 4 is selected. The NC program is checked again for collision by using the sweep plane algorithm. The result shows that the collision still occur between the toolholder and the workpiece, as shown in Fig. 6. Therefore, another method needs to be applied. The choice is applied with $\Delta Z_{\max}$ (the second big value), and the tool length extension method is used.

After extending the tool length by $\Delta Z_{\max}$ (21.1297), the NC program is checked again. The result shows that the collision at block N02 has been corrected, and the smaller collision at the block N01 automatically disappears (Fig. 7).

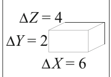
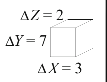
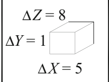
5 Time complexity of the proposed algorithm

The time complexity of the whole collision detection and avoidance algorithm can be calculated as:

$$T = i \times (T_{\Phi} + N_i \times t_c + T_{\Psi}) \quad (1)$$

where i is the number of iterations (as described in Fig. 1), T_{Φ} is the time complexity of the collision detection algorithm, N_i is the number of steps in which a collision is detected in the

Table 4 Table of the biggest collision boxes at the iteration i=2

Table of the biggest collision boxes (iteration i = 2)										
Block m	Step n	G code	Coordinate	Collision type	Types of the biggest collision boxes					
					One box	Two boxes		Three boxes		
650	8	G01	X = 130 Y = 85 Z = -245 A = 30 B = 5	τ=2						

iteration i, t_c is the average time for a collision data storage, and T_Ψ is the time complexity of the collision avoidance method.

The time complexity of the collision detection algorithm is a trade-off between the time complexity of the bounding sphere algorithm and the time complexity of the sweep plane algorithm. It is expressed in the following equation:

$$T_\Phi = T_b + T_u + T_s \tag{2}$$

where T_b is the time complexity of the bounding sphere algorithm, T_u is the time complexity of the workpiece geometry update, and T_s is the time complexity of the sweep plane collision detection algorithm. The relevant terms for calculating these times are described below [1, 19, 20].

The worst time for the bounding sphere algorithm is:

$$T_b = M \times N \times t_\text{b} \times \left(\sum_{j=0}^{m} 8^j\right) \times (n-1)! \qquad \text{(exponential factorial)} \tag{3}$$

where M is the number of blocks of NC program, N is the average number of interpolation steps between two consecutive blocks, t_b is the time for one pair of bounding spheres test, m is the number of levels of the octree, and n is the number of machine bodies that need to be checked for collisions.

The time for the workpiece geometry update is:

$$T_u = M \times N \times t_\text{u} \times S_\text{w} \qquad \text{(linear)} \tag{4}$$

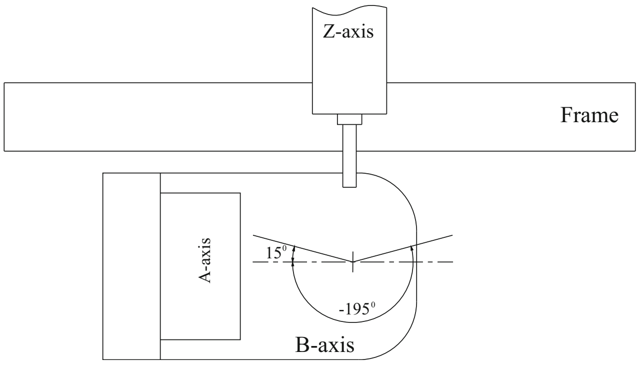


Fig. 5 Rotation limit of the B-axis—MAHO 600e

where M and N are the same as in Eq. 3, t_u is the average time for updating one slice, and S_w is the number of slices of the workpiece.

The worst time for the sweep plane collision detection algorithm is:

$$T_s = M \times N \times t_s \times \sum_{k,l} S_k \times S_\text{l}{}^l \qquad \text{(quadratic)} \tag{5}$$

where M, N are the same as in Eqs. 3 and 4, t_s is the average time for one pair of slices test; S_k and S_l are the number of slices within the colliding sphere of bodies k and l, respectively; k and $l \in \{$*machine bodies that need to be checked for collision*$\}$

The time complexity for collision correction is:

$$T_\Psi = t_\upsilon^\tau \qquad \text{(linear)} \tag{6}$$

where t_υ^τ is the average time for correcting a collision of type τ by method υ; method υ∈ {(1) Translate X, or Y, or Z-axes; (2) Rotate A-axis; (3) Rotate B-axis; (4) Inverse kinematics; (5) Extend tool length; (6) Change clamping position; (7) Change lead/lag and/or tilt angle; (8) Change setup without changing CL file; (9) Combination of change setup and change lead/lag and/or tilt angle} (Tables 1 and 2)

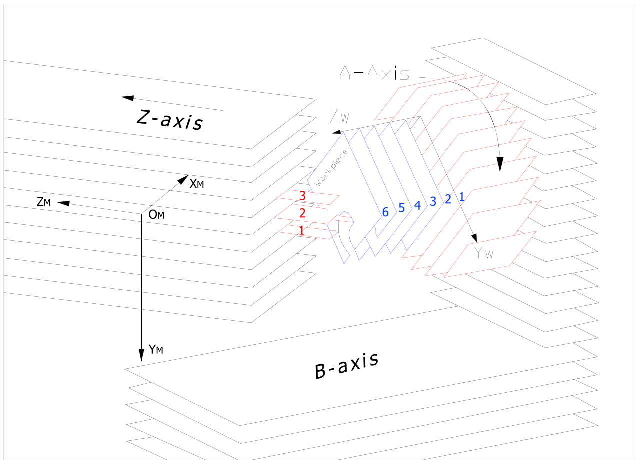


Fig. 6 Collision between the toolholder and workpiece at block: N02 G01 X-110.181 Y-35.355 Z-231.309 A135 B-144.736

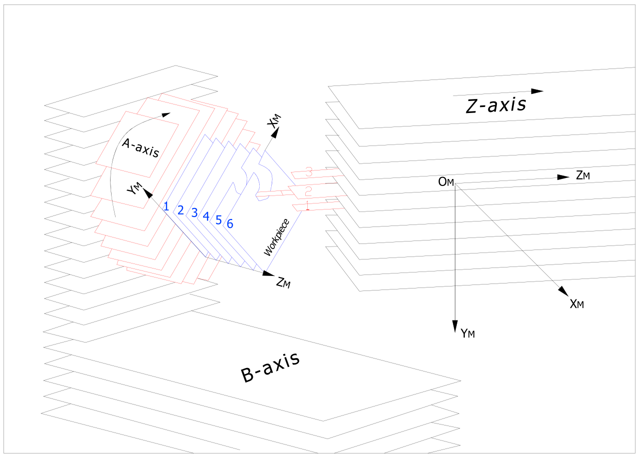


Fig. 7 There is no collision at block: N02 G01 X110.273 Y35.355 Z-231.178 A-45 B-35.264

Equation 3 shows that the time complexity of the bounding sphere algorithm is exponential in the number of octree levels (m) multiplied by a factorial of the number of bodies (n−1). The time for updating the workpiece geometry is a linear function of the number of slices of the workpiece as in Eq. 4. The time for the sweep plane collision detection is quadratic with respect to the number of slices within the colliding spheres as in Eq. 5. However, the worst case will rarely occur for the bounding sphere algorithm; this is because if there is no collision between the bounding spheres, the calculation will terminate at the first level of the octree. If there are a small number of collisions, fewer branches of the octree will need to be processed, thus resulting in shorter processing time. In addition, for the case of the sweep plane collision detection, we just take the number of slices within the colliding spheres; therefore, the number of slices to be tested will be considerably reduced, and the calculation speed of the proposed collision detection algorithm is fast. The integration of the sweep plane with the bounding sphere algorithm considerably reduces the time complexity of the algorithm from an exponential function of the number of octree level m (Eq. 3) to a quadratic function with respect to the number of slices within the colliding spheres of the octree (Eq. 5).

If there are many steps in which a collision is detected and the number of iterations is large, then the time complexity of the whole algorithm will be increased. Nevertheless, by correcting the biggest collision, most of the small collisions are automatically removed. Therefore, the number of steps in which collisions occur will be considerably reduced after each iteration.

For example, in Section 4.4, the number of levels $m=2$ was selected, one block to be checked ($M=1$), and the number interpolation steps between two consecutive blocks is 12 ($n=12$). The number of bodies that needs to be checked for collisions in five-axis CNC machine-Maho 600e is 7 ($n=7$). The time for the bounding algorithm in this case is $T_b=1\times12\times t_b\times8^2\times(7-1)!=552960\times t_b$. The number of slices of the workpiece is 6 ($S_w=6$), and the time for the workpiece geometry update is $T_u=1\times12\times t_u\times6=72\times t_u$. A collision occurred between the first octant bounding sphere of the workpiece (the number of slices within this sphere is 2 ($S_k=2$)) and the fifth octant bounding sphere of the toolholder (the number of slices within this sphere is 2 ($S_l=2$)); therefore, the time for the sweep plane collision detection is $T_s=1\times12\times t_s\times2\times2=48\times t_s$. The number of steps in which the collision detected is 12 ($N_i=12$); hence, the time for storage of collision data is $12\times t_c$. There are two iterations ($i=2$) needed to check and correct the collision (the first one corrects the collision by an alternative solution of inverse kinematics; the second one corrects the collision by extending the length of the tool). The time of the collision correction algorithm is $T_\Psi=t_4^T+t_5^T$ (where t_4^T and t_5^T are the time to correct the collision by the inverse kinematics method and extending the tool length at the first and second iteration). In the above equations t_b, t_u, t_s, t_c, and t_υ^T depend on the computer configuration and are of the order of microsecond.

6 Conclusion

A new collision correction strategy and its integration with the collision detection for five-axis NC machining has been presented. The proposed algorithm automatically detect and correct for the collisions. The collision detection algorithm is based on the bounding volume and the sweep plane approach. After the collision is detected, the biggest collision boxes for each type of collisions are stored and the others are discarded. The collision correction based on the heuristic strategy where the biggest collision boxes are corrected first. This newly proposed strategy reduces the time complexity of the collision avoidance algorithm since most of the other collisions automatically disappear when the biggest collision is corrected. The proposed algorithm can be customized to apply for any type of five-axis CNC machines.

Acknowledgments This research is funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.04-2011.09.

References

1. Ding S, Mannan MA, Poo AN (2004) Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces. Computer-Aided Des 36(13):1281–1294
2. Jun C-S, Cha K, Lee Y-S (2003) Optimizing tool orientations for 5-axis machining by configuration-space search method. Computer-Aided Des 35(6):549–566
3. Lin T, Lee J-W, Bohez ELJ (2009) A new accurate curvature matching and optimal tool based five-axis machining algorithm. Journal of Mechanical Science and Technology 23:2624–2634

4. Lauwers B, Dejonghe P, Kruth JP (2003) Optimal and collision free tool posture in five-axis machining through the tight integration of tool path generation and machine simulation. Computer-Aided Des 35:421–432
5. Chen T, Ye P, Wang J (2005) Local interference detection and avoidance in five-axis NC machining of sculptured surfaces. Int J Adv Manuf Technol 25:343–349
6. Du J, Yan X-G, Tian X-T (2012) The avoidance of cutter gouging in five-axis machining with a fillet-end milling cutter. Int J Adv Manuf Technol 62:89–97
7. Wang YJ, Dong Z, Vickers GW (2007) A 3D curvature gouge detection and elimination method for 5-axis CNC milling of curved surfaces. Int J Adv Manuf Technol 33:368–378
8. Kim S-J, Lee D-Y, Kim H-C, Lee S-G, Yang M-Y (2006) CL surface deformation approach for a 5-axis tool path generation. Int J Adv Manuf Technol 28:509–517
9. Gian R, Lin TW, Lin AC (2003) Planning of tool orientation for five-axis cavity machining. Int J Adv Manuf Technol 22:150–160
10. 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 13:393–400
11. Zhiwei L, Hongyao S, Wenfeng S, Jianzhong F (2012) Approximate tool posture collision-free area generation for five-axis CNC finishing process using admissible area interpolation. Int J Adv Manuf Technol 62:1191–1203
12. You C-F, Chu C-H (1997) Tool-path verification in five-axis machining of sculptured surfaces. Int J Adv Manuf Technol 13:248–255
13. Kiswanto G, Lauwers B, Kruth JP (2007) Gouging elimination through tool lifting in tool path generation for five-axis milling based on faceted models. Int J Adv Manuf Technol 32:293–309
14. Wang Q-H, Li J-R, Zhou R-R (2006) Graphics-assisted approach to rapid collision detection for multi-axis machining. Int J Adv Manuf Technol 30:853–863
15. Tang TD (2014) Algorithms for collision detection and avoidance for five-axis NC machining: a state of the art review. Computer-Aided Des 51:1–17
16. Tang TD, Bohez ELJ, Koomsap P (2007) The sweep plane algorithm for global collision detection with workpiece geometry update for 5-axis NC machining. Computer-Aided Des 39(11):1012–1024
17. Bohez ELJ, Minh NTH, Kiatsrithanakorn B, Natasukon P, Huang R-Y, Son LT (2003) The stencil buffer sweep plane algorithm for 5-axis CNC tool path verification. Computer-Aided Des 35(12): 1129–1142
18. Murta A (1997–1999) Generic Polygon Clipper, A new algorithm for calculating the difference, intersection, exclusive-or or union of arbitrary polygon sets. Advanced Interfaces Group, University of Manchester
19. Larsson T, Tomas A-M (2006) A dynamic bounding volume hierarchy for generalized collision detection. Computer & Graphics 30(3):450–459
20. Gottschalk S (2000) Collision queries using oriented bounding boxes. Ph.D dissertation, The University of North Carolina at Chapel Hill