www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Collision-Free Multi-Axis Tool-Path for Additive Manufacturing

Rahnuma Islam Nishat[1], Yeganeh Bahoo[1], Konstantinos Georgiou[1], Robert Hedrick[2] and R. Jill Urbanic[3]

[1] Ryerson University, Canada, {rnishat, bahoo, konstantinos}@ryerson.ca
[2]CAMufacturing Solutions Inc., Canada, bob.hedrick@camufacturing.com
[3]University of Windsor, Canada, jurbanic@uwindsor.ca

Corresponding author: Rahnuma Islam Nishat, rnishat@ryerson.ca

Abstract. Additive manufacturing (AM) is a process by which complex components or assemblies are fabricated in layers. As materials are deposited over time, the possibility of collision increases, inflicting damage to the components being printed or the machine parts. Therefore, special care needs to be taken while designing the tool-path (the path that the deposition head follows during the manufacturing process).

We provide an algorithm to modify a given tool-path to avoid collision between the printing surface and the tool holder using a geometric approach. For each point on the tool-path, we generate multiple tool vectors that are collision-free. A tool vector at a point of the tool-path shows the direction of the deposit head at that point. Using these collision-free tool-vectors we build an edge-weighted graph, where an edge is added between two tool-vectors if they are associated with adjacent points on the tool-path. The weight of an edge denotes the change in angle between two tool-vectors. Then using shortest path algorithm, we generate a collision-free tool-path such that the difference in tilting angle is minimized between adjacent tool-path points. We present experimental results to show the performance of our algorithm.

Keywords: Collision Avoidance, Mesh, Shortest Path, Graph Theory, Geometric objects
DOI: https://doi.org/10.14733/cadaps.2023.1094-1109

1 INTRODUCTION

With many contemporary additive manufacturing (AM) systems, a planar build strategy is employed. A thin-walled component example is shown in Fig. 1, where a machining or a combination of machining and welding would be costly, time consuming, and have much waste materials. On the other hand, an AM solution is trivial as the designer would select the slicing and fill strategies then initiate the build.

With a planar '$2\frac{1}{2}D$' build strategy, collision avoidance issues between the component and AM heat source solution (e.g., a laser, or a material deposition nozzle) do not exist. However, the directed energy deposition

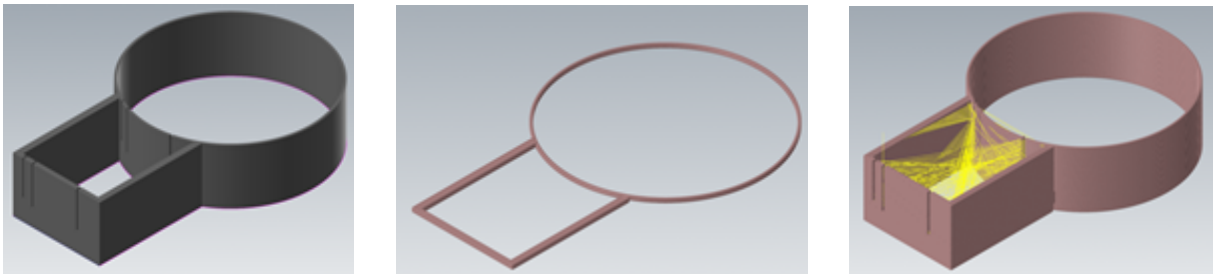

Figure 1: Left to right: (a) Thin walled component, (b) sample layer, (c) virtual build model with tool paths.

(DED) processes utilize a heat source and material feeding system mounted on a multi-axis CNC system or a robot to deposit beads side by side to fill a layer. With leveraging non-planar slicing and multi-axis tool paths, the DED process can be used for fabricating a new part without support structures, repairing a damaged part, and surface coatings. This multi-axis solution introduces potential collision issues that need to be addressed.

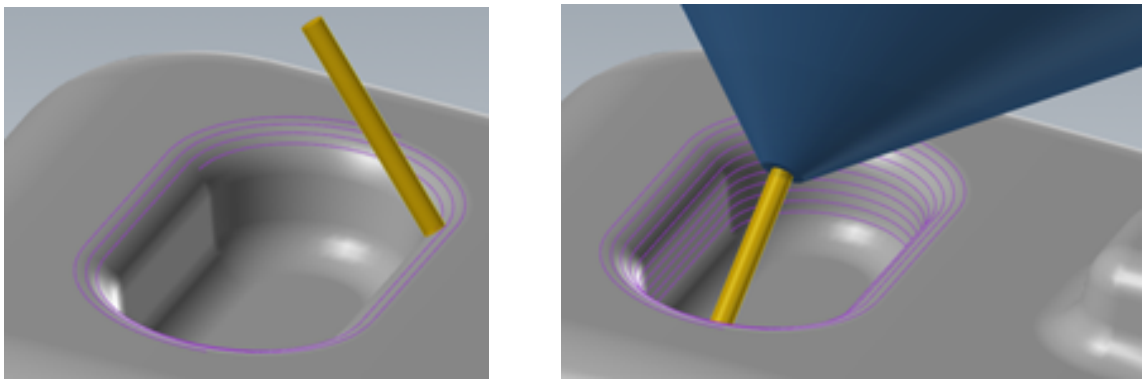

Figure 2: Left to right: (a) Tool depositing material onto a recessed area, (b) showing the holder and a boss.

Existing algorithms to design tool-paths for AM processes do not ensure that the tool-path is collision-free for multi-plane/multi-axis build scenarios, which is typical for turbine blade repair, surface coating, arbitrary volume build cases (see Fig. 2). Collisions between the machine parts and the *workpiece* (i.e., the object being printed) may cause damage to the machine part and/or the workpiece which would be expensive both financially and time-wise. Moreover, if collision cannot be avoided automatically when generating the tool-path, manual intervention would be required more frequently during the virtual verification processes (i.e., hand editing tool-paths) prior to engaging in manufacturing the component. Such overhead can be avoided by automating the designing of collision-free tool-paths for AM processes.

Although collision avoidance in CNC machining has attracted the attention of researchers for some time (see Section 2 for details), it is relatively a new direction of research in the field of additive manufacturing. As mentioned above, the possibility of collision increases over time in additive manufacturing in contrast to machining, and hence, avoiding collisions in bead based AM deposition processes are of utmost importance.

However, since there is a large difference in the working environment and objects of CNC machine tools and the AM machine tools, the approaches applied in machining to avoid collisions are not readily applicable to AM processes. While we do study algorithms in the literature on avoiding collisions in CNC machining, we do not compare the performance of our algorithm with any such algorithm.

1.1 Our Contribution

In this paper, we give an algorithm to modify a given tool-path to a collision-free tool-path, where a tool-path is a path defined by a sequence of points in 3D along with tool vector for each tool-path point which is a vector representing the direction of the deposit head at that point, and change in the angle of the tool vectors of the two consecutive points of the tool path does not exceed some given threshold.

Avoiding collisions in a tool-path: *Given a surface, a tool-path, and the AM tool holder specifications, the goal is to propose a collision-free tool-path.*

In our algorithm, we represent the surface and machine model using a 3D triangle-mesh. To detect collisions between the deposition head and the workpiece, we implement algorithms from the literature [13, 4], and also use a third party software licensed under an NDA for comparison. Using these tools, we build a configuration graph where different tool vectors for each tool-path point is represented as vertices. We then compute a collision-free tool-path by applying graph algorithms to compute a path in the configuration graph containing a collision-free tool-vector for each tool-path point.

The rest of the paper is organized as follows. In Section 2, we discuss some related work on collision detection and avoidance. Section 3 describes our algorithm for computing conflict-free toolpath. In Section 4, we present the results of our experimental runs, and compare them with state-of-the-artwork on collision-free tool-path design for CNC machining. We conclude in Section 6 with a summary of our contributions and future directions of research.

2 RELATED WORK

Several approaches for detecting avoiding collisions have been investigated for CNC machining for both *local* (i.e., local gouging, rear gouging, etc.) and *global* collisions. Researchers have applied variety of methods including correction vector-based method [8], where the interference surface normal can be leveraged avoid the interference region; C-space-based method [14]; local surface geometry-based method [12]; visibility and accessibility-based method [2, 3]; graphics-assisted method [19]; methods based on physical modeling of the machining tool [7], and many other approaches to solve the problem. See [17] for a detailed survey.

Balasubramanium et al. [2] presented an algorithm to generate tool-path using visibility and accessibility-based method. The algorithm works in three stages. First, it computes *visibility cones* (a set of angles from which the given tool-path point is visible from an observer outside) for every point in the tool-path. However, since visibility does not ensure accessibility (i.e., the machining tool cannot access the point without colliding), the second stage checks for accessibility of the tool for each of the angles in the visibility cone; and computes the set of angles which are *valid* or collision-free. Finally, a continuous tool-path is computed from the sets of valid angles for each tool-path point. Balasubramanium et al. [3] presented another algorithm in 2003 using similar approaches to deal with global collision detection, where they apply a rotational or translational correction when a collision occurs based on the location of the colliding points in the local coordinates of the tool. Note that, in both the above algorithms, the object being machined is presented as a *cloud of points* and thus, computationally expensive.

Wang and Tang [19] proposed an algorithm using the concept of *discretized visibility map (VMap)* to identify the set of valid orientations or configurations by inspecting the valid area with all the gouging constraints. Although the algorithm can handle a general type of tool including the flat-end tool and arbitrarily complicated obstacles, the computation, and storage of the VMaps have a high space and time complexity. Xu et al. [20] applied a similar approach. Li and Zhang [9, 11, 10] used point cloud-based approach, which requires huge computer resources in terms of space and time, similar to the algorithms of Balasubramanium et al.

Aliyu and Al-Sultan [1] gave linear programming algorithm to detect collisions between moving objects, where the objects are represented as polyhedral sets in $\mathbb{R}^2$ and $\mathbb{R}^3$. Tang et al. [16] gave a sweep plane based algorithm to detect global collisions in 5-axis NC machining. They first check for collision between spheres

bounding the workpiece and machine part. If intersection is found then they slice both the objects, and detect for collision in 2D planes (i.e., XY, YZ or XZ planes) using the workpiece slices and machine component slices. They followed up this work with an algorithm to avoid collision in 5-axis NC machining [18].

Schumann et al. [15] gave an algorithm to compute a hull around moving machine parts (e.g., tool holder) based on the maximum speed and deceleration values of the specific part. They argued that computing this hull beforehand will allow to prevent collision in real time.

In this paper, we use the *convex hull* of a coarse approximation of the tool holder (i.e. the smallest convex geometry enclosing the approximated holder model) to check for collision with the printing surface. Previously, Lee and Chang [8] presented a collision avoidance algorithm by using convex hull of the sculpted surface (i.e., workpiece). Wang [19] proposed a graphics-assisted approach to compute admissible tool orientations for collision avoidance in five-axis ball-end milling. The algorithm relies on graphics-assisted cubic mapping, and instantaneous visibility and accessibility cones computation. However, image resolution determines correctness of the approach. To the best of our knowledge, graph algorithms (e.g., shortest path algorithms) has not been yet employed to compute collision-free *smooth* toolpath.

Detecting an intersection between geometric objects in 3D, especially the intersection of triangles, has been extensively studied. The algorithms by Möller [13] and Held [5] are the most popular among them. Guigue and Devillers [4] gave an algorithm based on the algorithms in [13, 5], and improved the time complexity by around 20% using floating-point computation.

The AM problem space has unique challenges to overcome: (i) the nozzle or deposition head is not always rotationally symmetric, unlike the rotary tool holders and cutting tools utilized in machining; (ii) the AM process adds material to the workspace, which means that the collision detection probability increases as the build progresses; and (iii) the build shape cannot be accurately represented due to the basic imprecision of the DED AM process. This paper presents a foundation for addressing these issues.

3 COMPUTING A COLLISION-FREE TOOL-PATH

We present the surface as a triangular mesh. Instead of using the actual holder model, we export an approximate model of the tool-path holder from Mastercam, and then compute the *convex hull* of the approximated tool holder. We use the triangular mesh representation of the convex hull in our calculation as it contains less triangles and improves the performance of the collision detection algorithm. We used two different holder models in our experiments: rotationally symmetric and asymmetric; see Figs. 3 and ‘4, respectively, for the constructed convex hulls of the approximated models.

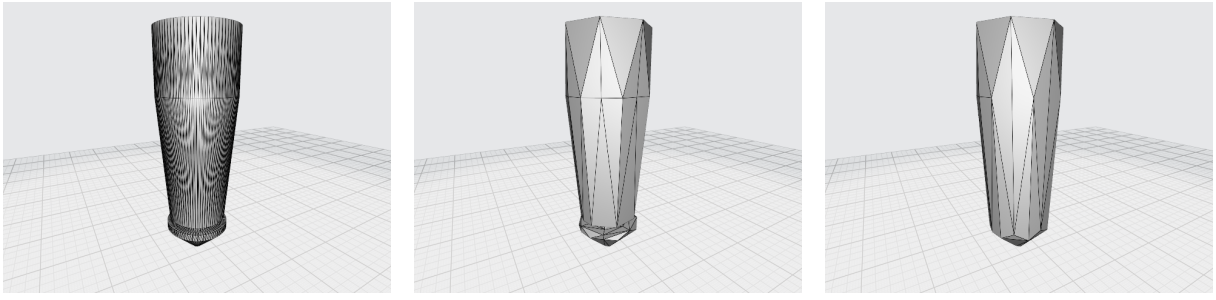

Figure 3: From left to right: (a) rotationally symmetric holder (1,446 triangles), (b) approximate model of the holder (144 triangles), (c) convex hull of the approximated model (72 triangles).

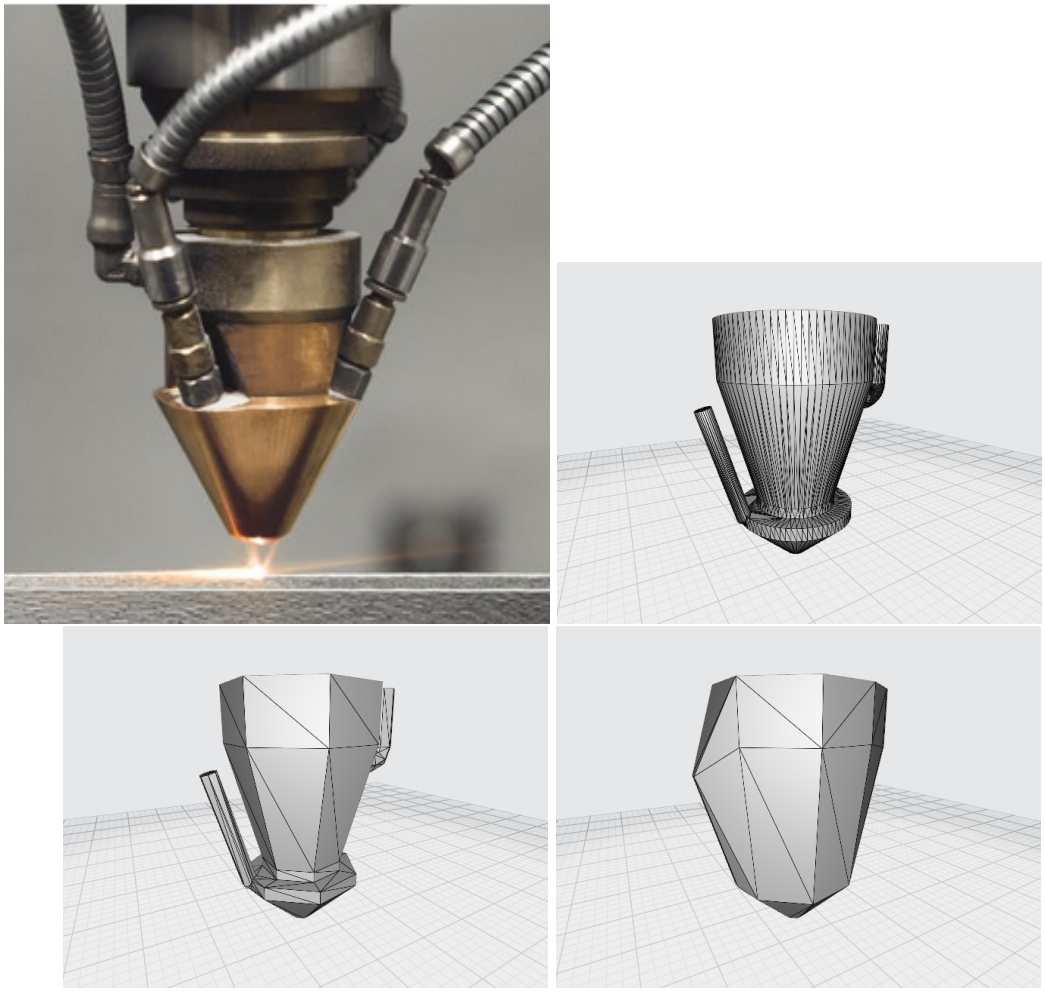

Figure 4: From left to right: (a) rotationally asymmetric holder, (b) the CAD model (6,122 triangles), (a) approximate model of the holder (340 triangles), (c) convex hull of the approximated model (140 triangles).

We try two geometric approaches in detecting the collisions. Our first approach is based on collision detection between triangles. We implement Möller's algorithm [13] and Devillers and Guigue's algorithm [4] to detect collision between two triangles. We check for collisions between each pair of triangles from the surface mesh and the mesh of the deposit head. Additionally, we applied parallel calculation to improve the computation time further.

Our second approach is based on collision detection between solids using a clash detection algorithm supplied in a commercial third-party library. In this case, we check for collision between the whole solid surface and the solid that represents the model of the holder. The clash detection algorithm applies multithreading to detect collision.

To compute a collision-free tool-path, we apply the following three steps, which are described in the next three sections.

1. Generating different tool vectors for each tool-path point.
2. Building the configuration graph.
3. Computing a collision-free tool-path from the configuration graph.

3.1 Generating Different Tool Vectors for Each Tool-path Point

We generate unit tool vectors on a unit sphere where the center of the sphere is a point on the tool-path as shown in Figure 6. Here, θ is the maximum allowable change of angle for the tool between adjacent tool-path points, and ϕ is the maximum tilt angle for the tool/holder. As the figure shows, the distance between consecutive cycles is θ, as is the distance between two consecutive tool vectors on the same circle.

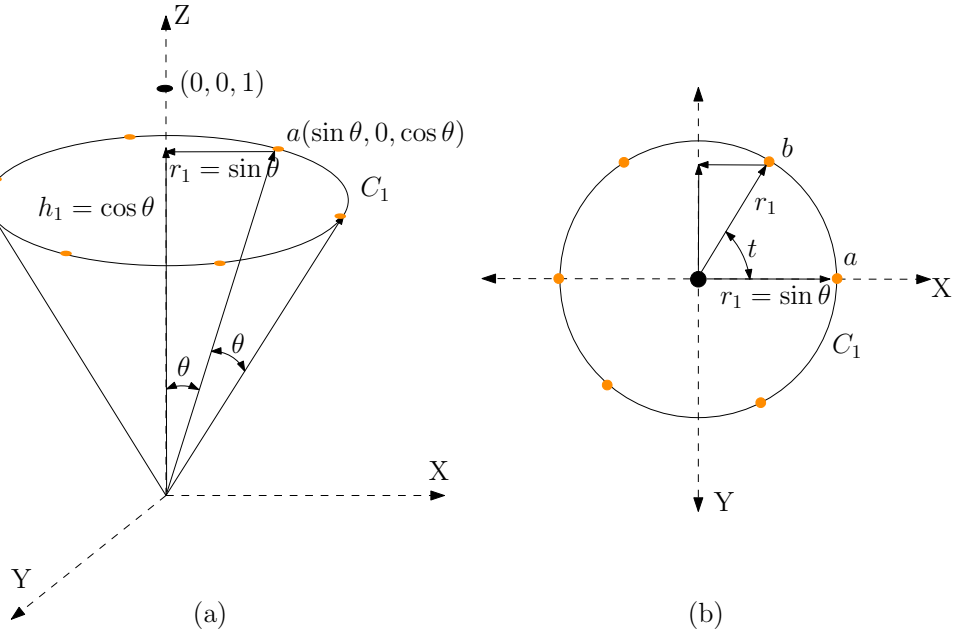

Figure 5: Generating points on a circle on the unit sphere.

We first compute circles on the unit sphere, as shown in Fig. 5, such that the angular distance between two consecutive circles is θ. Let C_1 be the closest circle to Z-axis on the sphere, and let a be the point on the XZ plane on C_1. Then the coordinate of a is $(\sin\theta, 0, \cos\theta)$. Let r_1 and h_1 respectively be the radius and height of the circle C_1. Then $r_1 = \sin\theta$ and $h_1 = \cos\theta$; see Fig. 5(a). Let C_1, C_2, ... be the circles with increasing angles $\theta, 2\theta, \ldots$ with the Z-axis. Then the radius r_q and height h_q of any such circle is given by the following formulae.

$$r_q = \sin(q\theta) \tag{1}$$

$$h_q = \cos(q\theta) \tag{2}$$

We now compute points on C_1 such that the corresponding vectors to the consecutive points on C_1 have angular distance θ. All such point will have Z coordinate equal to h_1, we have to compute the X and Y coordinates. We take a projection of C_1 on the XY plane as shown in Fig. 5(b). Let the angular different of the points on the circle be δ_1, and let b be the counterclockwise neighbor of a. Since the length of the

vector from the center of the circle to b is r_1, the coordinates of the point b will be $(r_1 \cos\delta_1, r_1 \sin\delta_1, h_1)$, or $(\sin\theta\cos\delta_1, \sin\theta\sin\delta_1, \cos\theta)$. Therefore the pth point counterclockwise from a will have the coordinate:

$$(\sin\theta\cos p\delta_1, \sin\theta\sin p\delta_1, \cos\theta)$$

We compute the value of δ_1 from the dot product of the vectors $\overrightarrow{a}$ and $\overrightarrow{b}$:

$$\overrightarrow{a} \cdot \overrightarrow{b} = |\overrightarrow{a}||\overrightarrow{b}|\cos\theta$$

$$\cos\theta = \frac{r_1^2\cos\delta_1 + 0 + h_1^2}{|\overrightarrow{a}||\overrightarrow{b}|}$$

$$\cos\theta = \frac{r_1^2\cos\delta_1 + h_1^2}{1\cdot 1}$$

$$\cos\delta_1 = \frac{\cos\theta - h_1^2}{r_1^2}$$

$$\delta_1 = \cos^{-1}\frac{\cos\theta - h_1^2}{r_1^2}$$

Generalizing for any circle C_q, the angle δ_q between consecutive points on the XY projection of the circle is:

$$\delta_q = \cos^{-1}\frac{\cos\theta - h_q^2}{r_q^2} \tag{3}$$

Therefore, the number of points on circle C_q would be $\frac{2\pi}{\delta_q}$, where $q = 1, 2, \ldots$. Fig. 6 shows all the circles around the normal along Z-axis, where ϕ is the maximum tilt angle of the tool holder.

3.2 Building the Configuration Graph

We build a configuration graph with tool vectors as vertices. We then apply graph algorithms such as breadth-first search or Dijkstra's shortest path algorithm on . For a tool-path with n points, we construct a configuration graph G as follows. G is a directed layered graph (directed edges appear only between consecutive layers) with each layer corresponding to a point of the tool-path. Every layer of G contains vertices corresponding to tool vectors of the corresponding tool-path point. We introduce a directed edge between two vertices in that appear in consecutive layers exactly when the two corresponding tool vectors are compatible with the head's specifications. Finally, we introduce two auxiliary vertices, a source s and a target t. We introduce a directed edge from s to every vertex of the first layer of G, and directed edges from the last layer of G to t, see Fig. 6. An s-t path in G corresponds to (collision-free) tool-vectors that are compatible with the heads specs along the tool-path.

3.3 Computing a Collision-free Tool-path from the Configuration Graph

To find a collision-free tool-path, we first apply the breadth-first search (BFS) algorithm to find an s-t path in the *unweighted* configuration graph. However, the path returned by BFS may have jittering issues, where the direction of the tool vectors changes frequently, making the tool-path unsuitable for a 5-axis machine tool or robot. We address this issue by considering a *weighted* configuration graph, where the edges have weights proportional to the angular difference between their two corresponding tool vectors of their endpoints. We then find the shortest path from the source to the sink node using Dijkstra's shortest path algorithm. The shortest path makes a minimal change to the tool vectors between consecutive tool-path points.

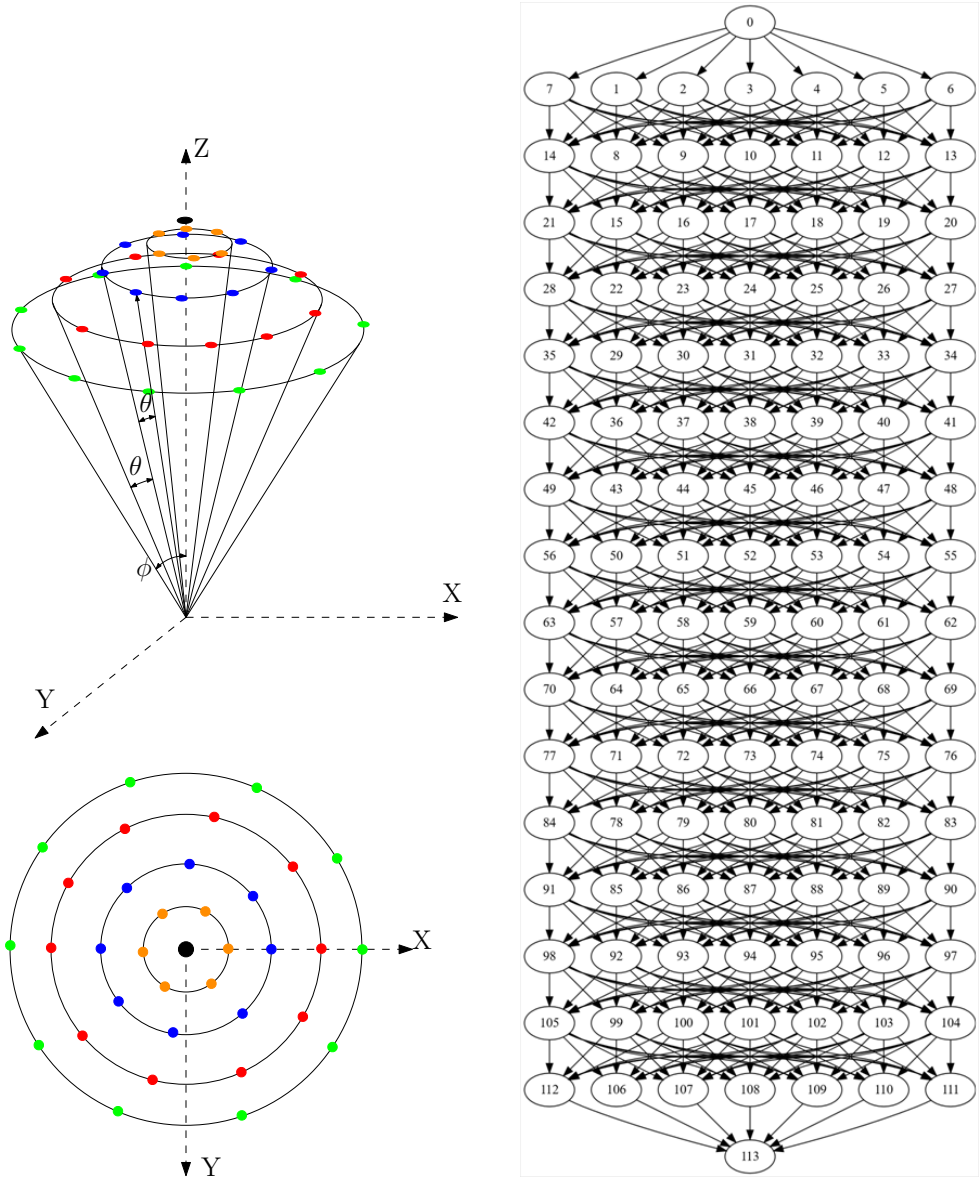

Figure 6: Left: Generating tool vectors on a unit square withing the given conic volume around a normal along Z-axis; isometric view (above) and top view (below) are shown. Right: A configuration graph with 16 tool-path points, where 7 different configurations are considered for each tool-path point. Vertices 0 and 113 are the auxiliary vertices s and t, respectively.

4 EXPERIMENTAL RESULT

We use APlus software that integrates with Mastercam to generate AM specific toolpaths and visualize them. Our algorithm takes the tool-path generated by the APlus software as an input, as well the workpiece and

the holder geometry as STL (Standard Tessellation Language) files, and generates a collision free tool-path if such a path exists. If no collision free tool-path is found by our algorithm, we return the original tool-path recieved as input with a notification that the path has possible collision between the holder and workpiece.

We conducted our experiment on two solid parts shown in Fig. 7. We nicknamed the solids in Fig. 7(a) and (b) as *Bathtub* and *L-bracket*. The two workpieces were chosen because the tool-path generated by the APlus software causes collision between the holder and workpiece. Our goal was to generate tool-paths by modifying the current tool-path that will be collision free and will not introduce mechanical issues such as jittering.

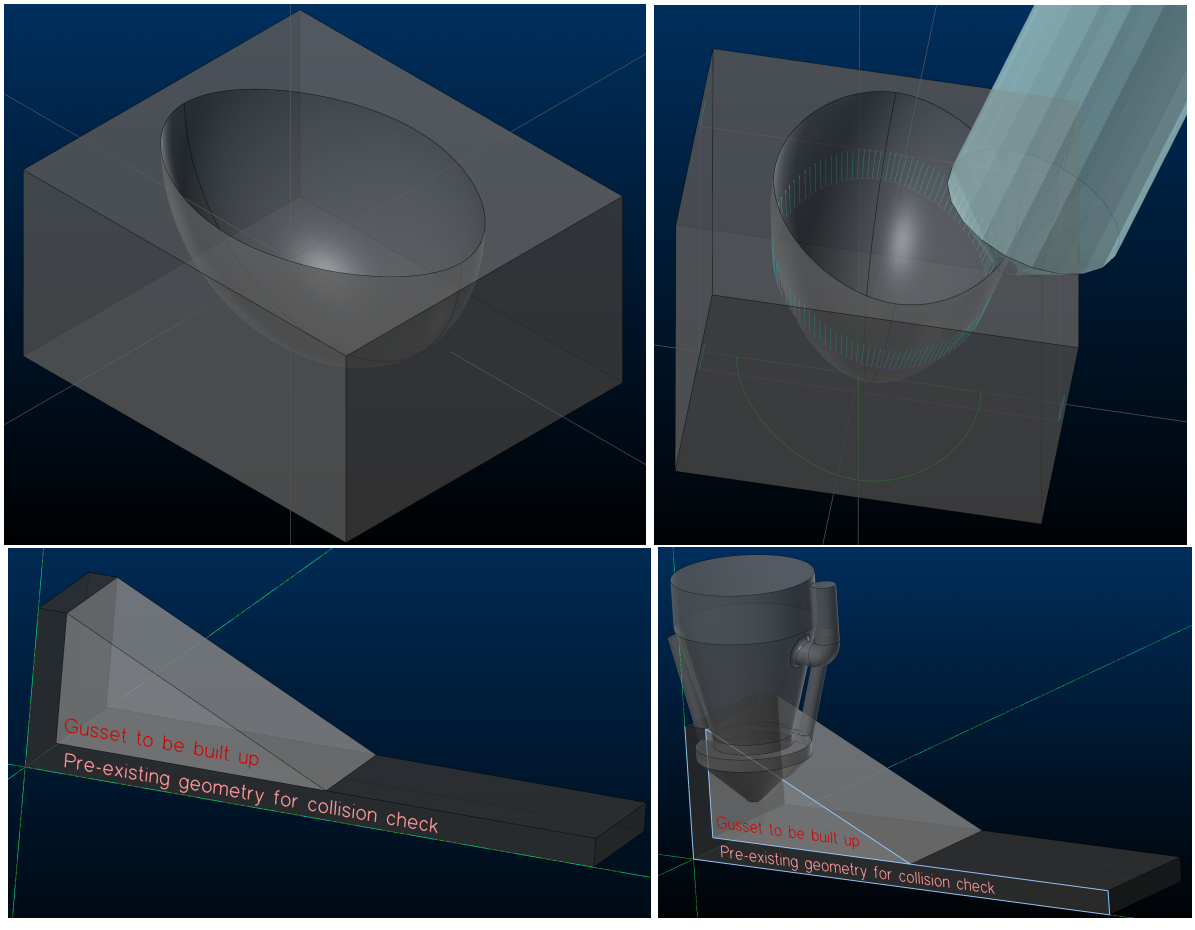

Figure 7: The two solid parts used in our experiments, and collisions with tool holder. Above from left to right: (a) Bathtub, and (b) collision with symmetric holder approximation. Below from left to right: (a) L-bracket, and (b) collision with asymmetric holder.

4.1 System Specification

Our testing system specifications are given below:

- CPU: Intel(R) Core(TM) i9-10885H CPU @2.40GHz

- Memory: 32GB 2933MHz DDR4
- Storage: 953.86GB (Model: SKHynix_HFS001TD9TNI-L2B0B)

4.2 Tool-paths Generated by our Algorithm

Fig. 8 shows the modified tool-path generated by our algorithm for the Bathtub solid, and Fig. 9 shows the modified tool-path generated by our algorithm for the L-bracket solid. In both the tool-paths generated by our algorithm, the tool vectors have been modified to avoid-collision. In Figure 8, one can observe that additional points have been added to the tool-path so that the difference between the tool vectors of adjacent tool-path points do not exceed some given threshold. Figure 9 shows another efficient feature of our algorithm, i.e., only the tool vectors for the tool-path points that causes collision have been modified.

To make sure that the tool-paths do not induce jittering to the holder or machine, we ran machine simulations in Mastercam.

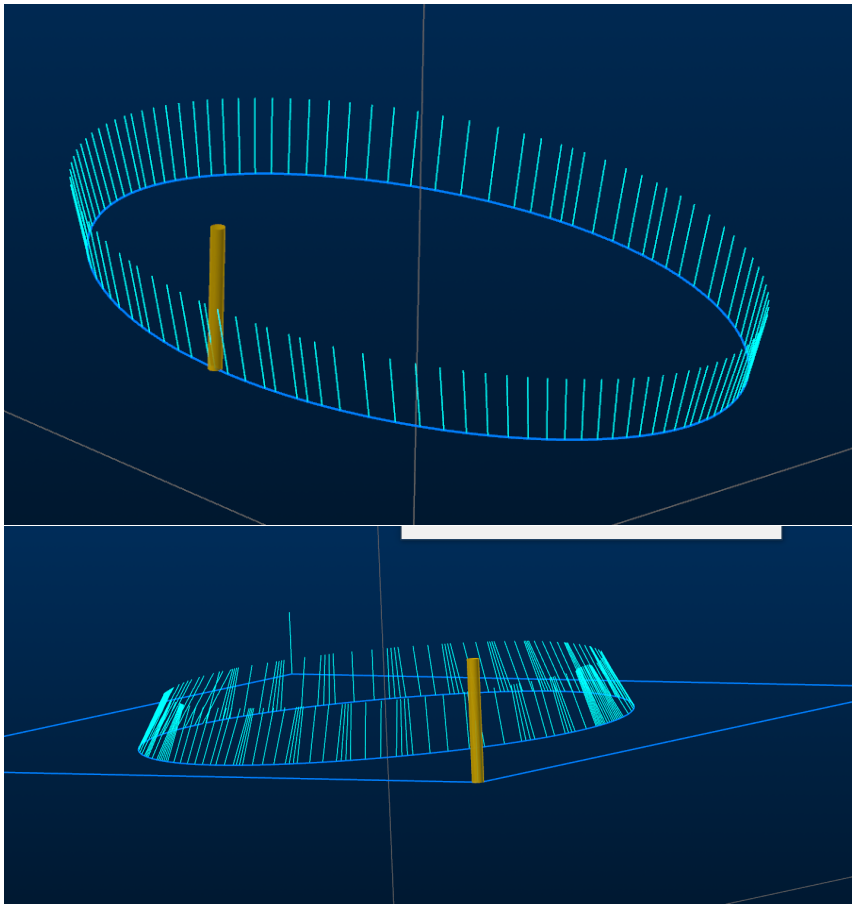

Figure 8: Above: Original tool-path with collisions generated by APlus for the Bathtub;below: the tool-path that is collision-free, generated by our algorithm from the above tool-path.

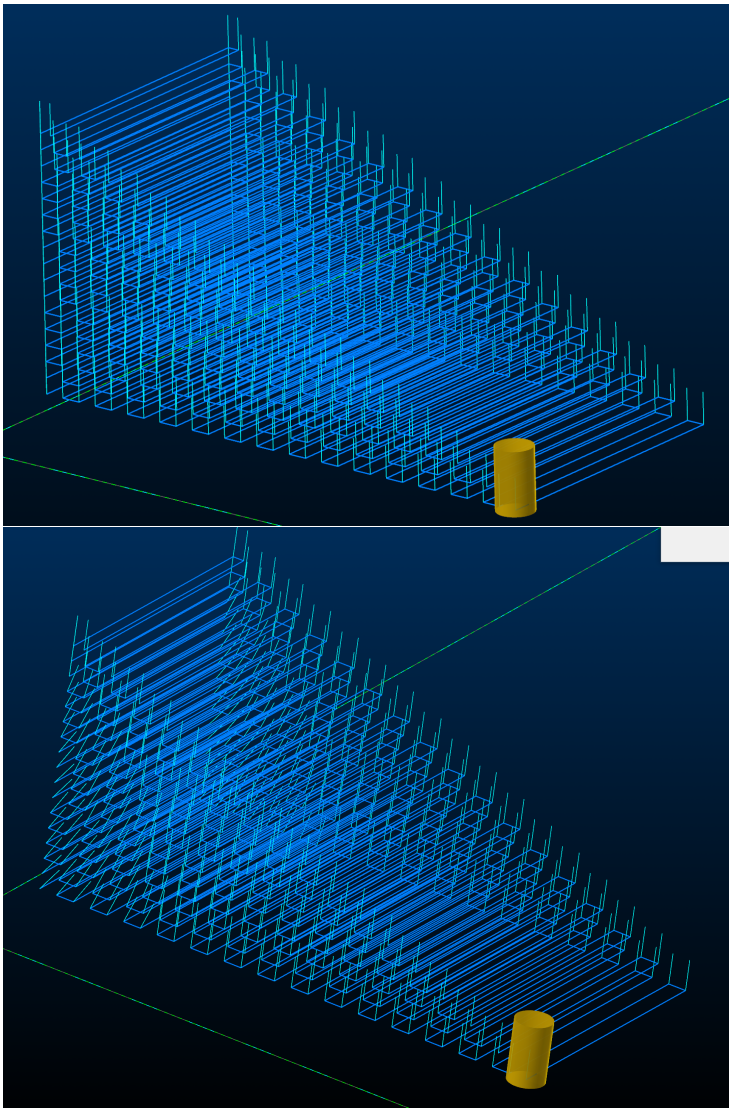

Figure 9: Above: the original tool-path causing collisions for the L-bracker; below: the modified collision-free tool-path that was generated by our algorithm.

4.3 Comparing Collision Detection Algorithms

We first compared different collision detection algorithms to determine their performance on the two solids.

Experiment I: Bathtub and symmetric holder model. We ran Moller's algorithm with parallel execution on the Bathtub using the symmetric holder model, and a collision-free tool-path was generated in 64 seconds. For the same set up, Guigue and Deviller's implementation took 66 seconds. However, even with sequential execution the commercial third-party library generated the desired tool-path in 4.4 seconds.

Experiment II: L-bracket and asymmetric holder model. Using parallel execution, both Moller's algorithm

and the algorithm by Guigue and Deviller generated collision-free tool-paths in approximately 36 seconds. The commercial third-party library took 9 seconds using sequential execution policy.

The running time for our two experiments are listed in Table 1. Since the commercial third-party library was the fastest, we conducted further experiments using this library.

Algorithm	Execution policy	Symmetric holder & Bathtub	Asymmetric holder & L-bracket
Moller's algorithm	Parallel	64 s	36 s
Guigue and Deviller's algorithm	Parallel	66 s	36 s
Commercial third-party library	Sequential	4.4 s	9 s

Table 1: Comparing our approaches in computing collision-free toolpaths.

4.4 Comparing Holder Model Approximations

Using the commercial third-party library, we measured performance of our collision avoidance algorithm for different approximations of the holder models from Figs. 3 and 4 in Mastercam with APlus.

L-bracket. The L-bracket tool-path contains 566 points. For the actual symmetric holder model generated from Mastercam ($1,446$ triangles), the algorithm takes 145 seconds; for the approximation (144 triangles), also exported from Mastercam with a higher error tolerance, it takes 22 seconds; with the convex hull of the approximated holder (72 triangles) the running time comes down to 13 seconds.

With the asymmetric holder model ($6,122$ triangles) the running time is 400 seconds, with a coarser approximation (340 triangles) exported from Mastercam it is 35 seconds, and with the convex hull of the coarse approximation the time required is 15 seconds.

Bathtub. The Bathtub tool-path contains 273 points. For the actual symmetric holder model generated from Mastercam ($1,446$ triangles), the algorithm cannot generate a collision free tool-path; for the approximation (144 triangles), a collision-free tool-path is generated in 19 seconds; with the convex hull of the approximated holder (72 triangles) the running time comes down to 12 seconds.

With the asymmetric holder model ($6,122$ triangles) the running time is 78 seconds, with a coarser approximation (340 triangles) exported from Mastercam it is 27 seconds, and with the convex hull of the coarse approximation the time required is 14 seconds.

The summary of our results are shown in Table 2, which shows that using the convex hull of a coarse approximation improves the running time substantially.

5 FUTURE WORK

While building our configuration graph, we generate a number of different tool vectors for each of the tool-path points. We then check for each such tool vectors, whether there is a collision between the holder and the printing surface using a collision detection algorithm. Therefore, the performance of our algorithm depends on the following factors:

1. Efficiency of the collision detection algorithm we choose.
2. Size of the configuration graph. There are two sub factors.

Test surface	Symmetric holder	Asymmtric holder	Symmetric approximation	Asymmetric approximation	Symm convex hull	Asymm convex hull
L-bracket (566 point)	145 s	400 s	22 s	42 s	13 s	15 s
Bathtub (273 point)	n/a	78 s	19 s	27 s	12 s	14 s

Table 2: Comparing performance of the collision avoidance algorithm using different holder models.

(a) Number of tool-path points

(b) Number of tool vectors (i.e., configurations) considered for each point on the tool-path.

3. Efficiency of the algorithm to calculate the optimal collision-free tool-path from the configuration graph.

Since we are using Dijkstra's shortest path algorithm for the last step, which is computationally optimal, we focus on the remaining factors to improve the performance of our collision-avoidance algorithm presented in the paper.

5.1 Improving Efficiency of Detecting Collision

We implemented collision-detection algorithms that checks for collision between each pair of triangles from the surface mesh and the holder mesh. Here the number of triangles in the two meshes are usually huge. In an effort to reduce the number of triangles in the holder mesh, we exported a coarse approximation of the holder model from Mastercam and then computed its convex hull. As our experiments show, the performance of the algorithm has improved using convex hulls. However, we have not changed the surface mesh. We believe that by applying remeshing techniques, the size of the surface mesh can also be reduced, which can improve the running time of our algorithm further.

Another technique that we are considering is using the *slicing and clipping* algorithm presented in [16, 18] for collision detection. The idea is to compute 'slices' of both the holder and the printing surface, where each slice is a 2D polygon. Then for each pair of slices, one from the holder and one from the printing surface, we compute the intersection of the polygons to see if there is any collision.

5.2 Using Less Number of Tool-path Points in the Graph

We design a heuristic algorithm as follows. Instead of using all the points on the tool-path when creating the configuration graph, we use some of the points. For example, we can take every tenth point to create the graph, which will be much smaller in size. In that case, if the maximum change of angle between two consecutive tool vectors for two adjacent tool-path points is θ, the maximum change we allow in the graph will be $10 \times \theta$. If we cannot find a continuous tool-path from the graph that respects the requirement of maximum change of angle between adjacent points on the tool-path, we first find the bottleneck point; we then add more points from the ignored tool-path points that are before or after the bottleneck point; then we try to find a continuous tool-path in the modified (and slightly bigger than before) configuration graph. We recursively apply modification to the graph as long as we do not find a smooth and continuous tool-path from the graph.

5.3 Reducing the Number of Tool Vectors for Each Point

The idea here is to apply some local correction before computing the configuration graph. Given the initial tool-path, we apply a local correction similar to the approach in [8] whenever we find a collision. However, these local modifications might give us a tool-path where the maximum angle between two adjacent tool vectors might be bigger than the threshold. Therefore, we create a configuration graph after applying the local corrections, but in this case we consider a smaller conic volume around the new surface normal (not around Z-axis as we do in this paper) at the tool-path points. The graph will be smaller in size and improve the time to calculate a collision-free continuous tool-path.

6 CONCLUSION

The AM process family is expanding from the $2\frac{1}{2}$ D build domain into simultaneous 5 axis motions for the DED process, allowing us to manufacture structures (e.g., overhangs beyond certain angle limits) without extra support structures that needs to be removed later [6]; see Fig. 10. This introduces new process planning

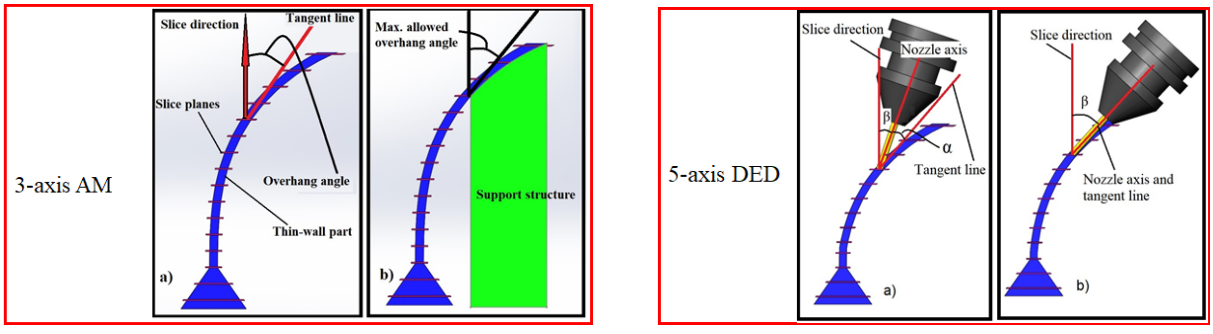

Figure 10: Maximum allowed over hang angle for 3-axis and 5-axis machines [6].

challenges related to both heat management and collision avoidance. New tools to assist in developing viable build strategies need to be developed. This research focuses on developing a foundation for fast, robust collision avoidance. Multiple approaches are being explored. The running time of our algorithm greatly improves with multithreading in the commercial third-party library. However, we believe that it can be improved even more by using OpenCL to include GPU in the computation. In the future, we would like to test our algorithm for complex multilayer tool-paths. Adding rapid moves (where no material is deposited) in the tool-path to avoid collision could be another interesting direction.

ACKNOWLEDGEMENTS

This research is part of a project funded by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ontario Centre of Innovation (OCI).

REFERENCES

[1] Aliyu, M.D.S.; Al-Sultan, K.S.: Lp-based algorithms for detecting the collision of moving objects. The Journal of the Operational Research Society, 46(7), 854–866, 1995. ISSN 01605682, 14769360. http://www.jstor.org/stable/2583968.

[2] Balasubramaniam, M.; Laxmiprasad, P.; Sarma, S.; Shaikh, Z.: Generating 5-axis nc roughing paths directly from a tessellated representation. Computer-Aided Design, 32(4), 261–277, 2000. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/S0010-4485(99)00103-7.

[3] Balasubramaniam, M.; Sarma, S.E.; Marciniak, K.: Collision-free finishing toolpaths from visibility data. Computer-Aided Design, 35(4), 359–374, 2003. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/S0010-4485(02)00057-X.

[4] Devillers, O.; Guigue, P.: Faster Triangle-Triangle Intersection Tests. Tech. Rep. RR-4488, INRIA, 2002. https://hal.inria.fr/inria-00072100.

[5] Held, M.: Erit - a collection of efficient and reliable intersection tests. Journal of Graphics Tools, 2, 25–44, 1998. http://doi.org/https://doi.org/10.1080/10867651.1997.10487482.

[6] Kalami, H.: Supportless Fabrication, Experimental, and Numerical Analysis of the Physical Properties for a Thin-Walled Hemisphere. Ph.D. thesis, University of Windsor, 2020.

[7] Lacharnay, V.; Lavernhe, S.; Tournier, C.; Lartigue, C.: A physically-based model for global collision avoidance in 5-axis point milling. Computer-Aided Design, 64, 1–8, 2015. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/j.cad.2015.02.003.

[8] Lee, Y.S.; Chang, T.C.: 2-phase approach to global tool interference avoidance in 5-axis machining. Computer-Aided Design, 27(10), 715–729, 1995. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/0010-4485(94)00021-5.

[9] Li, L.L.; Zhang, Y.F.: Flat-end cutter accessibility determination in 5-axis milling of sculptured surfaces. Computer-Aided Design and Applications, 2(1-4), 203–212, 2005. http://doi.org/10.1080/16864360.2005.10738368.

[10] Li, L.L.; Zhang, Y.F.: Cutter selection for 5-axis milling of sculptured surfaces based on accessibility analysis. International Journal of Production Research, 44(16), 3303–3323, 2006. http://doi.org/10.1080/00207540500444720.

[11] Li, L.L.; Zhang, Y.F.: An integrated approach towards process planning for 5-axis milling of sculptured surfaces based on cutter accessibility map. Computer-Aided Design and Applications, 3(1-4), 249–258, 2006. http://doi.org/10.1080/16864360.2006.10738462.

[12] Lo, C.C.: Efficient cutter-path planning for five-axis surface machining with a flat-end cutter. Computer-Aided Design, 31(9), 557–566, 1999. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/S0010-4485(99)00052-4.

[13] Möller, T.: A fast triangle-triangle intersection test. Journal of Graphics Tools, 2, 25–30, 1997. http://doi.org/https://doi.org/10.1080/10867651.1997.10487472.

[14] Morishige, K.; Kase, K.; Takeuchi, Y.: Collision-free tool path generation using 2-dimensional c-space for 5-axis control machining. The International Journal of Advanced Manufacturing Technology, 13, 393–400, 1997.

[15] Schumann, M.; Witt, M.; Klimant, P.: A real-time collision prevention system for machine tools. Procedia CIRP, 7, 329–334, 2013. ISSN 2212-8271. http://doi.org/https://doi.org/10.1016/j.procir.2013.05.056. Forty Sixth CIRP Conference on Manufacturing Systems 2013.

[16] Tang, T.; Bohez, E.L.; Koomsap, P.: The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis nc machining. Computer-Aided Design, 39(11), 1012–1024, 2007. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/j.cad.2007.06.004.

[17] Tang, T.D.: Algorithms for collision detection and avoidance for five-axis nc machining: a state of the art review. Computer-Aided Design, 51, 1–17, 2014. http://doi.org/https://doi.org/10.1016/j.cad.2014.02.001.

[18] Tang, T.D.; Bohez, E.L.J.: A new collision avoidance strategy and its integration with collision detection for five-axis nc machining. The International Journal of Advanced Manufacturing Technology, 81, 1247–1258, 2015.

[19] Wang, N.; Tang, K.: Automatic generation of gouge-free and angular-velocity-compliant five-axis tool-path. Computer-Aided Design, 39(10), 841–852, 2007. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/j.cad.2007.04.003.

[20] Xu, X.J.; Bradley, C.; Zhang, Y.F.; Loh, H.T.; Wong, Y.S.: Tool-path generation for five-axis machining of free-form surfaces based on accessibility analysis. International Journal of Production Research, 40(14), 3253–3274, 2002. http://doi.org/10.1080/00207540210150643.