Title:

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

Authors:
Rahnuma Islam Nishat, rnishat@ryerson.ca, Ryerson University, Canada
Yeganeh Bahoo, bahoo@ryerson.ca, Ryerson University, Canada
Konstantinos Georgiou, konstantinos@ryerson.ca, Ryerson University, Canada
Robert Hedrick, bob.hedrick@camufacturing.com, CAMufacturing Solutions Inc., Canada
R. Jill Urbanic, jurbanic@uwindsor.ca, University of Windsor, Canada

Keywords:
Collision Avoidance, Mesh, Shortest Path, Graph Theory

DOI: 10.14733/cadconfP.2022.297-301

Introduction:

Additive manufacturing (AM) is a process family by which complex components are fabricated in layers from the bottom up. Therefore, the possibility of collision between the object/surface being printed and machine parts increases with time as the material is being added, inflicting damages to the object or the machine parts. To avoid such unwanted scenarios, special care needs to be taken while designing the tool-path, i.e., the path that the deposition head follows during the manufacturing process. With many contemporary AM systems, a planar build strategy is employed. 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 (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.

Existing algorithms to design tool-paths for AM processes do not ensure that the tool-path is collision-free. 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 collision for 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 material being built up, we implement algorithms from the literature [7, 3], and also use a third party software licensed under 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 the next two sections, we discuss some related work on collision detection and avoidance, and the algorithms we use for detecting collision between the printing surface/object and the tool holder. Next, we present our algorithm for computing conflict-free toolpath; discuss 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 with a summary of our contributions and future directions of research.

Related works:

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, see [8] for a detailed survey. Among these methods, *visibility and accessibility-based* methods, which we describe below, are related to our approach.

Balasubramanium et al. [1] 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. [2] 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 [9] 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. [10] applied a similar approach. Li and Zhang [5, 6] used point cloud-based approach, which requires huge computer resources in terms of space and time, similar to the algorithms of Balasubramanium et al.

Detecting intersection between geometric objects in 3D, especially intersection of triangles, has been extensively studied. The algorithms by Möller [7] and Held [4] are the most popular among them. Guigue and Devillers [3] gave an algorithm based on the algorithms in [7, 4], 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 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.

Computing Collision-Free Tool-Path:

We presented the surface as a triangular mesh . Instead of using the actual holder model for the rotationally symmetric holder, we create an approximate geometric model of the tool-path holder and generated a triangular mesh from the approximate model, see Fig. 1.

We try two geometric approaches in detecting the collisions. Our first approach is based on collision detection between triangles. We implement Möller's algorithm [7] and Devillers and Guigue's algorithm [3]

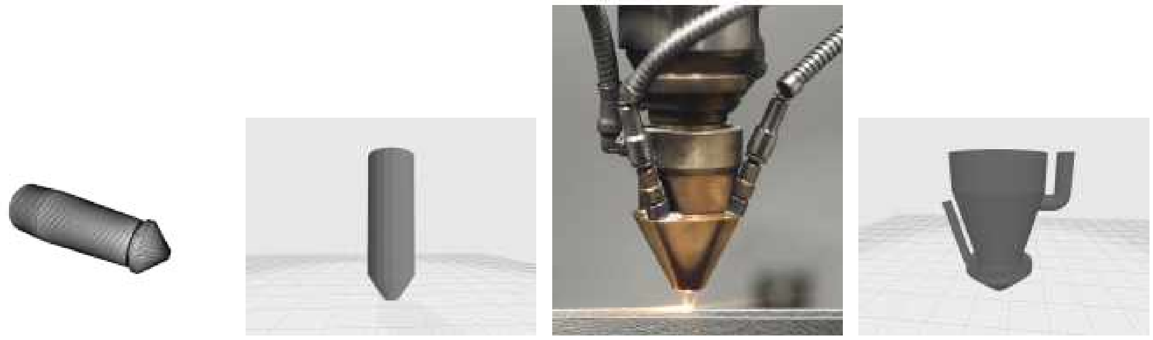

Fig. 1: From left to right: (a) rotationally symmetric holder, (b) approximate model of (a) with 80 triangles, (c) rotationally asymmetric holder, and (d) CAD model of (c).

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 build a configuration graph with tool vectors as vertices and then apply graph algorithms such as breadth-first search or Dijkstra's shortest path algorithm. For a toolpath 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 toolpath. 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. 2. Clearly, an s-t path in G corresponds to (collision-free) tool-vectors that are compatible with the heads specs along the tool-path.

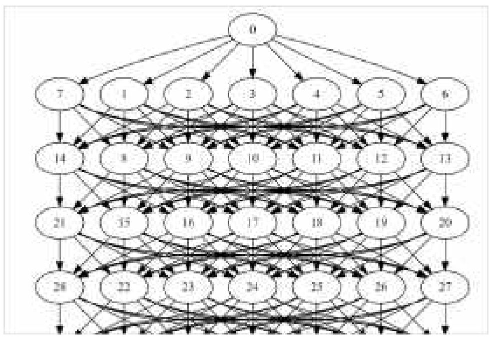

Fig. 2: Part of a configuration graph where 7 different configurations are considered for each toolpath point, and vertex 0 is the auxiliary vertex s.

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.

Experimental Result:

We use APlus software that integrates with Mastercam to generate AM specific toolpaths and visualize them. 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)

Fig. 3 shows two sets of experiments where our algorithm creates a collision-free toolpath from a toolpath with collisions. In Fig. 3(a)–(c), the rotationally symmetric holder model from Figure 1(b) is used. Fig. 3(d)–(f) show the second set of experiments where, the asymmetric holder model from Figure 1(d) is used. The running time for our different approaches are listed in Table 1. This approach is promising, and will be extended to include more complex case studies. As well, the search spaces will be refined and optimized to reduce processing time.

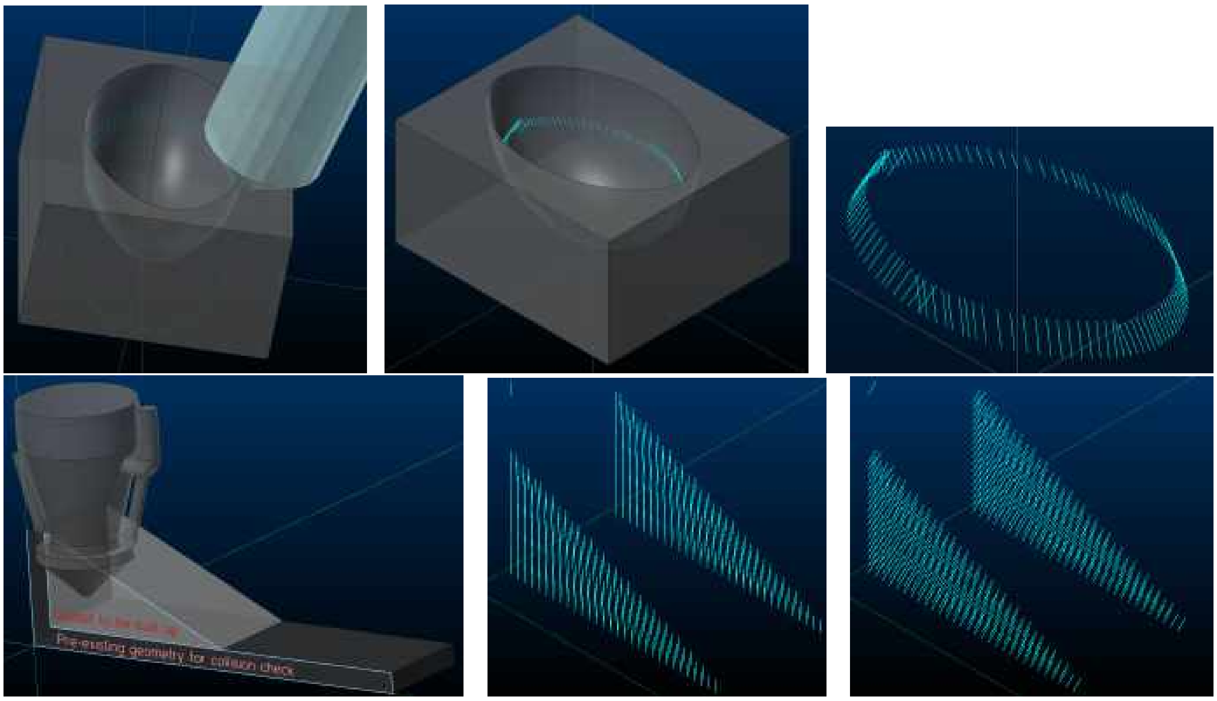

Fig. 3: Generating collision-free tool-path from a given tool-path. Top row from left to right: (a) toolpath with 128 points creates collision between the surface and the holder model in Figure 1(b); (b) modified toolpath for (a) returned by our algorithm; (c) modified tool-path for (a) without the part. Bottom row from left to right: (d) tool-path with 566 points creates collision between the surface and the holder model in Figure 1(c); (e) original tool-vectors for (d); (f) Modified tool-path for (d).

Conclusions:

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

Algorithm	Execution policy	Experiment I time	Experiment II time
Moller's algorithm	Parallel	63932 ms	35649 ms
Guigue and Deviller's algorithm	Parallel	66003 ms	36034 ms
Commercial third-party library	Sequential	4373 ms	8996 ms

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

extra support structures that needs to be removed later [11]. This introduces new process planning 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.

References:

[1] Balasubramaniam M.; Laxmiprasad P.; Sarma S.; Shaikh Z.: Generating 5-axis NC roughing paths directly from a tessellated representation. Computer-Aided Design, 32(4), 200, 261—277. https://doi.org/10.1016/S0010-4485(99)00103-7

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

[3] Devillers O.; Guigue P.: Faster Triangle-Triangle Intersection Tests. Technical Report RR-4488, INRIA, June 2002. https://hal.inria.fr/inria-00072100

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

[5] 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. 10.1080/16864360.2005.10738368

[6] 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), 2006, 249—258. 10.1080/16864360.2006.10738462

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

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

[9] Wang N.; Tang K.: Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath. Computer-Aided Design, 39(10), 2007, 841—852. https://doi.org/10.1016/j.cad.2007.04.003

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

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