Title:
Voxel-Based Collision Avoidance for 5-Axis Additive Manufacturing

Authors:
Alexandre Desgrees du Lou, alexandre.desgrees_du_lou@mines-albi.fr, IMT Mines Albi, France
Yeganeh Bahoo, bahoo@torontomu.ca, Toronto Metropolitan University, Canada
Robert Hedrick, bob.hedrick@camufacturing.com, CAMufacturing Solutions Inc., Canada
Austin Vuong, austin.vuong@torontomu.ca, Toronto Metropolitan University, Canada

Keywords:
Voxelization, Collision Detection and avoidance, GJK, EPA, Optimization, Additive Manufacturing.

DOI: 10.14733/cadconfP.2024.215-221

Introduction:

Additive Manufacturing (AM) stands as a revolutionary advancement in manufacturing technologies. With this advancement arises a fresh challenge: the potential for collisions during the print process. This research narrows its focus on this collision management challenge, aiming to harness the power of voxel-based [1] solutions to bolster the efficiency and reliability of this nascent technology.

AM has become instrumental in modern production, known for its flexibility in creating complex, customized designs directly from digital models. While traditional AM methods construct objects layer-by-layer, thereby limiting the geometrical freedom and often requiring additional support structures, 5-axis AM transcends these limitations. By introducing two additional rotational axes to the printing head movement, 5-axis AM allows for more intricate product designs, reduces the need for support structures, and can potentially improve the mechanical properties of printed objects by minimizing the anisotropic behavior often seen in traditional AM materials. However, these advancements are not without new challenges.The increased complexity of 5-axis movements means there is a higher risk of collision between the printing apparatus and the object being printed. This risk is particularly pronounced when producing complex or nested geometries, requiring advanced strategies for collision detection and avoidance.

Our approach to manage collisions unfolds in two primary stages: the first involves scrutinizing for possible collisions between the deposition head and the 3D model. If a collision is detected, the second stage kicks in, where the goal is to ascertain the best collision-free deposition head orientation scenario. As the AM process adds material to the work volume, modelling the material deposited by the toolpath is essential.

In the next section, we discuss some related work exploring innovative approaches to collision detection and avoidance in multi-axis additive manufacturing. We then present our input data and test environment. We then describe our first approach, in which we extend the result of Nishat et al. for a voxel framework [2]. This approach evolves towards the adoption of a bounding box strategy [3] associated with the GJK (Gilbert-Johnson-Keerthi) approach [4], to detect collisions. The next step is to manage the collision in the most efficient way. This is where the EPA (Expanding Polytope Algorithm) comes in, generating a solution for the collision-free deposition head axis vector closest to the origin vector.

Related works:

Collision detection and avoidance in multi-axis additive manufacturing have been explored through various innovative approaches. Fang et al. and Plakhotnik et al. focused on collision avoidance by dynamically adjusting tool vectors during the manufacturing process, adding penalties for deviations from optimal orientations [13, 14]. Jiang et al. examined collision avoidance through efficient trajectory planning, using multiple deposition heads to manage complex geometries and workspace constraints [15]. Nishat et al. developed an algorithm for generating collision-free trajectories, focusing on optimizing orientation and trajectory planning to minimize collision risks [2]. The approach of Nishat et al., which models object surfaces using triangular meshes, simplifies collision detection by reducing the number of intersection tests required. Their method for calculating a collision-free trajectory involves generating deposition head vectors, conducting collision tests for each vector using Möller's triangle-triangle intersection algorithm [16], and employing Dijkstra's algorithm to establish a collision-free path from a configuration graph.

Voxelization offers distinct advantages over mesh-based representations, particularly by simplifying collision detection to simple Boolean operations independent among each voxel, which is ideally suited for parallel processing on modern GPUs, thus facilitating real-time collision detection [17]. Hermann et al. developed an optimized framework for collision detection suited to GPU architectures, significantly enhancing the efficiency of collision detection and allowing seamless integration of real-time sensor data [18]. AM presents unique challenges that need to be addressed: (i) managing collisions efficiently; (ii) handling the addition of material to the workspace, which means that the probability of collision detection increases as the build progresses; (iii) ensuring that the collision-free print path does not lead to jittering. This article provides a foundation for solving these issues.

Input Data Details and the Testing Environment:

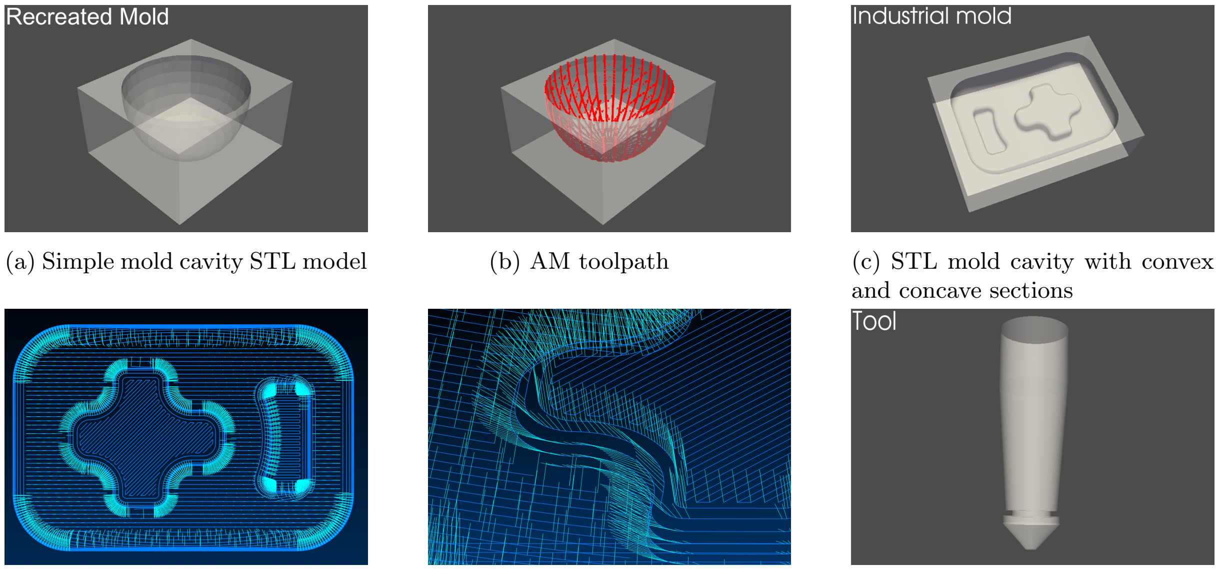
The initial framework for our study was defined by several key components. The toolpath is determined by a sequence of coordinates in $\mathbb{R}^3$, symbolized by the set P. Each point along this path has a distinct deposition head orientation, represented by a normalized vector in the same space. The deposition head and the CAD model are defined by their respective `STL` files, called the deposition head and mold respectively throughout this paper. Along with these, we considered constraints such as the height h,the contact tip to workpiece distance between the deposition head and path point as measured along the deposition head axis, the maximum angular variations of θ between consecutive points on the toolpath, and an overall maximum angular variation of Θ for the deposition head.

Our testing system specifications are as follows: **Chip:** Apple M1, **Memory:** 8GB, and **Storage:** 256GB Flash Storage. Figure 1a displays the 3D model "Recreated Mold", and Figure 1b illustrates the 3D model with red points corresponding to the AM toolpath.

The second path was provided by the industrial partner: CAMufacturing Solutions Inc. (CAMu). The toolpath is a .NC file, which provides the coordinates for each printing point, along with the inclination of the deposition head, as depicted in Figure 1d. The blue line represents the direction of the deposition head for specific points, see Figure 1e. For each point of the toolpath, the deposit head is positioned at a point we will call C, which is offset 6mm along the blue tool axis vector. In the event of a collision, the rotation will be performed from point C. The deposition head is represented as an STL file, which was provided by CAMu for the purpose of our study, as depicted in Figure 1f.

Voxel-based Adaptation for the method of Nishat et al.:

In the approach of Nishat et al. [2], the object surfaces are modelled using a triangular mesh. In this scenario, a collision is equivalent to a triangle intersection between two meshed objects. A simplified convex hull of the deposition head is used for collision detection with 3D model, which increases the algorithm's performance by reducing the number of intersection tests to be performed. Their process for calculating a collision-free trajectory consists of three steps: **1 -** The first step involves generating a



(a) Simple mold cavity STL model (b) AM toolpath (c) STL mold cavity with convex and concave sections

(d) AM toolpath (e) deposition head axis vectors (f) STL of deposition head from industrial partner

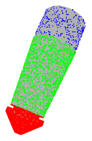
Fig. 1: Visualization of Mold Design, Toolpath, and deposition head Orientation.

variety of deposition head vectors to encompass all tilting possibilities of the deposition head represented by the set S in spherical coordinates, as detailed in [2] Section 3.1. **2 -** For each point in the set P, a collision test is conducted with all vectors in the set S, leading to the construction of a graph. In this graph, points from P represent levels and the collision-avoiding vectors from S serve as vertices. All vertices at one level are linked to those on the next, with weights assigned based on the angular variation of the deposition head, measured between the vector associated with a vertex at the current level and that at the next one. **3 -** Finally, a collision-free path is calculated using Dijkstra's algorithm on the configuration graph.

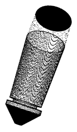
Voxelization Process:

In our approach, we substitute triangular meshes with voxels. However, the overarching objective remains to find the collision free toolpath. It involves the creation of a voxel grid, where each voxel has a density value of 1 if it is part of the mold, and 0 otherwise. The idea is to test, for each position in P and for each orientation in S of the deposition head, whether any of the deposition head's voxels have a value of 1. To do this, we utilized the Python library `Open3D` [7]. This library implements a surface-based voxelization approach, which is significantly faster (0.12 s for the deposition head) compared to volume-based voxelization methods. Despite its speed, a potential limitation of `Open3D`'s method is that it does not fully cover the entire surface, which might result in false positives during collision tests. Nonetheless, for our purposes, the superior performance of `Open3D` in terms of execution time for voxelizing the surface of objects was decisive in choosing it for our upcoming tasks.

We assume that the deposition head's lower part is at a higher risk of collisions compared to its upper part. To address this, we enhance voxel density on the surface of areas prone to risks. Specifically, we divide the deposition head into three sections: the lower section receives a substantially increased voxel density, while the middle and upper sections retain a lower density. This strategy effectively reduces the



(a) Deposition head derived from risk-zone voxelization, distributed as 58%, 35%, and 7% of 13,820 voxels. Execution Time 0.024s.



(b) Voxel model of 35,905 voxels obtained using the `Open3D` voxelization function. Execution Time 0.12s.

Fig. 2: Comparative visualization of tools post-voxelization for voxel size of 0.5 mm.

total number of voxels for analysis, cutting down the execution time of the program. However, it also slightly increases the chance of false positives.

Rotation Mechanism:

In our approach in order to define the set S of possible orientations for the deposition head, we employ Rodrigues' rotation formula [9] for efficient computation. The process begins with the voxelized deposition head oriented along the z-axis, leading to a new voxel model with non-integer coordinates. This inherent characteristic increases the complexity of subsequent collision tests. One idea is to round off the voxel values, this will lead to a modification of the object's shape. Another idea would be to rotate the triangular mesh first and then perform voxelization, thus ensuring that the obtained coordinates are integers while preserving the object's shape. However, doing this for every element in S would take a significant amount of time depending on the voxelization speed.

Experimental Results for voxel adaptation of Nishat et al.:

As we can observe in Table 1, a comparison has been made between the execution times for achieving a collision-free toolpath. This comparison specifically relates to our voxel-based approach under conditions that were closely equivalent to those presented by Nishat et al. in Table 1 of their paper. The scenario considered involves a toolpath comprising 273 points, akin to the bathtub example, here referred to as the recreated mold. The sole difference between our setups lies in the hardware used. For context, the specifications of the testing system used by Nishat et al. are as follows: **Chip**: Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz, **Memory**: 32GB 2933MHz DDR4, and **Storage**: 953.86GB with the model being SKHynix_HFS001TD9TNI-L2B0B. Furthermore, while the solution from Nishat et al. was implemented in C++, our implementation was carried out in Python using PyTorch for parallelization.

Method	Execution Time	Parallelization
Nishat et al. (C++ with Möller's Algorithm)	66 s	Yes
Our Approach with Recreated Mold (Python 3.9 on CPU)	6 s	No
Our Approach with Recreated Mold (Python 3.10 on GPU T4x2)	1.2 s	Yes

Table 1: Comparative Analysis of Execution Times for Computing Collision-Free Toolpaths (273 points)

GJK Algorithm:

The Gilbert-Johnson-Keerthi (GJK) algorithm [3] is a sophisticated method used for detecting the intersection of two convex sets in three-dimensional space. However, the Python library `CoACD` [10] offers a solution by decomposing a non-convex object into a collection of convex sub-objects. Decomposition can take some time, depending on the complexity of the object. The deposition head's form closely mirrors its convex hull, ensuring this approximation doesn't lead to complications. The collision assessment involves applying GJK between the deposition head and a convex sub-object of the mold. This process is repeated until all sub-objects of the mold in proximity to the deposition head (in the bounding box of the deposition head) are examined.

EPA Algorithm:

The Expanding Polytope Algorithm (EPA) serves as a natural progression from the GJK algorithm. Its primary purpose is to identify the minimum displacement vector essential for collision resolution. This vector, which we will refer to as the extraction vector, represents the most efficient path to exit a collision state. Here, a collision is defined as the overlap of two objects. The CoACD library decomposes our object into a sum of n convex sub-objects. Nevertheless, this introduces a new problem: multiple sub-objects collide with the deposition head. Therefore, we have a collision extraction vector $\mathbf{V}_i$ available for each colliding sub-object i. To determine the correct extraction vector to prevent further collisions, we compute a weighted average of the magnitudes of the vectors. We compute the weights w_i for n vectors as $w_i = \frac{\|\mathbf{V}_i\|}{\sum_{k=1}^{n} \|\mathbf{V}_k\|}$ and then calculate the correct extraction vector $\mathbf{E}$ as $\mathbf{E} = \sum_{k=1}^{n} \; w_1 \mathbf{V}_k$.

This method provides an extraction vector, $\mathbf{E}$, which suggests translating the deposition head to exit the collision state. However, any translation is infeasible during 3D printing. Thus, a rotation centered at the tip of the deposition head, in the direction of $\mathbf{E}$ is required. We can therefore iteratively vary the unit orientation $\mathbf{N}$ of the deposition head in the direction $\mathbf{E}$.

Algorithm and Experimental Result for GJK with Bounding Box and EPA:

The operation of the program is as follows: **1 -** The program uses CoACD to decompose the object into convex subparts. **2 -** The STL model of the deposition head is converted into a voxel model, with a distribution of voxels categorized into three zones: low, middle, and high. **3 -** There are two methods to change the orientation of deposition head positioning: Firstly, the initial orientation of the deposition head is aligned with the z-axis, and it will be modified from this starting position. Secondly, the orientation is continuously adjusted throughout the path. For each point, the deposition head's orientation is determined based on the result of the procedure that exits the collision state from the previous point. **4 -** Using the deposition head's bounding box, only the 3D model region in collision with the deposition head is selected. The goal then is to execute the GJK algorithm solely on the convex subparts within this zone. **5 -** The region of the deposition head in contact with the colliding 3D model zone is identified using the bounding box of this zone. This selected region is then used in the orientation adjustment method to conduct collision tests of equivalent density. **6 -** The GJK algorithm is executed on these zones. Subsequently, the extraction vector is determined using the EPA following the GJK. **7 -** Two iterative methods exist for adjusting the orientation: The first method involves obtaining the extraction vector initially and iterating the deposition head's rotations in this direction, which can lead to new collisions. The other option is to recalculate a new extraction vector after each rotation, with the new extraction direction computed using Lami's theorem [11] to retain information about both the previous and new vectors. As demonstrated in Table 2, the total execution time for our second method with GJK is comparable to that of our voxel-based approach, as shown in Table 1.

Aspect	Recreated Mold	CAMu Mold
Execution Time (after summing all program actions)	71.2 s	95.4 s

Table 2: Execution Times for the Molds without parallelization for 5000 points on the toolpath

Updating the Model and Reducing Jittering:

In our 3D printing model, we simulate material extrusion through voxelized spheres, updating their values as the deposition head moves for real-time updates. To ensure smoother transitions and reduce jitter, we've implemented an angular adjustment function using Bezier curves. This smooths out angular changes and segments the toolpath for efficient computation, resulting in continuous, stable motion.

Conclusion and Future work:

We have developed a faster method to detect collisions in toolpaths for AM and generate collision-free alternatives. Although our current methods are promising, they need further refinement for industrial speeds.

References:

[1] Cohen-Or, D.; Kaufman, A.: *Fundamentals of surface voxelization*, Graphical models and image processing, 57(6), 453–461, 1995, Elsevier. https://doi.org/10.1006/gmip.1995.1039.

[2] Nishat, R.I.; Bahoo, Y.; Georgiou, K.; Hedrick, R.; Urbanic, R.J.: *Collision-Free Multi-Axis Tool-Path for Additive Manufacturing*, Computer-Aided Design and Applications, 2023, 1094–1109. https://doi.org/10.14733/cadaps.2023.1094-1109.

[3] Gottschalk, S. A.: *Collision queries using oriented bounding boxes*, The University of North Carolina at Chapel Hill, 2000.

[4] Gilbert, E.G.; Johnson, D.W.; Keerthi, S.S.: *A fast procedure for computing the distance between complex objects in three-dimensional space*, IEEE Journal on Robotics and Automation, 1988, Vol. 4, Issue 2, 193–203. https://doi.org/10.1109/56.2083.

[5] Van Den Bergen, G.: *Proximity queries and penetration depth computation on 3d game objects*, Game Developers Conference, 170, 209, 2001. https://graphics.stanford.edu/courses/cs468-01-fall/Papers/van-den-bergen.pdf.

[6] Cameron, S.: *Enhancing GJK: Computing minimum and penetration distances between convex polyhedra*, Proceedings of International Conference on Robotics and Automation, 4, 3112–3117, 1997.

[7] Zhou, Q.-Y.; Park, J.; Koltun, V.: *Open3D: A Modern Library for 3D Data Processing*, arXiv:1801.09847, 2018.

[8] Liang, K. K.: *Efficient conversion from rotating matrix to rotation axis and angle by extending Rodrigues' formula*, arXiv preprint arXiv:1810.02999, 2018.

[9] Rodrigues, O.: *Des lois géométriques qui règissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire* Journal de mathématiques pures et appliquées, 5, 380–440, 1840.

[10] Wei, X.; Liu, M.; Ling, Z.; Su, H.: *Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search*, ACM Transactions on Graphics (TOG), 41(4), 1–18, 2022, ACM New York, NY, USA. https://doi.org/10.1145/3528223.3530103.

[11] Kumar, A.; Kushreshtha, D.C.: *Engineering Mechanics: Statics and Dynamics*, Section 4.6, pp. 98, Jaypee University of Information Technology, Solan, HP, 2015.

[12] Bézier, P.: *Numerical control-mathematics and applications*, Translated by AR Forrest, John Wily, 1972.

[13] Fang, G.; Zhang, T.; Zhong, S.; Chen, X.; Zhong, Z.; Wang, C.C.L.: *Reinforced FDM: Multi-axis filament alignment with controlled anisotropic strength*, ACM Transactions on Graphics (TOG), 39(6), 1–15, 2020, ACM New York, NY, USA.

[14] Plakhotnik, D.; Glasmacher, L.; Vaneker, T.; Smetanin, Y.; Stautner, M.; Murtezaoglu, Y.; van Houten, F.: *CAM planning for multi-axis laser additive manufacturing considering collisions*, CIRP Annals, 68(1), 447–450, 2019, Elsevier.

[15] Jiang, Z.; Wang, H.; Sun, Y.: *Improved co-scheduling of multi-layer printing path scanning for collaborative additive manufacturing*, IISE Transactions, 53(9), 960–973, 2021, Taylor Francis.

[16] Möller, T.: *A fast triangle-triangle intersection test*, Journal of graphics tools, 2(2), 25–30, 1997, Taylor Francis. https://doi.org/10.1080/10867651.1997.10487472

[17] Jang, D.; Kim, K.; Jung, J.: *Voxel-based virtual multi-axis machining*, The International Journal of Advanced Manufacturing Technology, 16, 709–713, 2000, Citeseer.

[18] Hermann, A.; Drews, F.; Bauer, J.; Klemm, S.; Roennau, A.; Dillmann, R.: *Unified GPU voxel collision detection for mobile manipulation planning*, In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4154–4160, 2014, IEEE.