The International Journal of Advanced Manufacturing Technology (2022) 118:1843–1858
https://doi.org/10.1007/s00170-021-07954-y

ORIGINAL ARTICLE

Nesting of 3D irregular shaped objects applied to powder-based additive manufacturing

Hong-Tzong Yau [1,2] · Che-Wei Hsu [1]

Received: 19 April 2021 / Accepted: 25 August 2021 / Published online: 23 September 2021

Abstract

This paper proposes a 3D nesting algorithm which combines the flower pollination algorithm (FPA) with the oriented bounding box (OBB) collision detection to solve the 3D packing of irregular shaped objects for powder-based additive manufacturing. In the powder-based 3D printing process, stacking multiple models most compactly in the build volume is an important task because no support structure is needed. Given a fixed printing area, the post-nesting build height directly impacts the printing cost and efficiency. To reduce the printing cost, 3D models must be packed as closely as possible and the build height must be minimized. This has been found to be a combinatorial optimization and an NP-hard problem. We propose to use the most recent and effective meta-heuristic optimization algorithm FPA, combined with the collision detection of printed objects set up as the optimization constraint using the OBB tree, to find the near-optimal 3D nesting solution. In FPA optimization, a significant amount of time is spent on collision detection. This paper uses the safety clearance distance to adaptively reduce the OBB tree subdivision, hence significantly reducing the computation of collision detection. As a result, the proposed method finds the model positions and rotations of the global solution, and generates the near-optimal solution in reasonable time. Finally, our method is compared with state-of-the-art commercial softwares. The results show that the proposed method produces lower build height with better efficiency. For real-world complex engine parts (30 different STL models and a total of 192,018 triangles), the computation time is only 160 s, and the build height is 12.4% and 13% better than the results from Netfabb and Magics, respectively.

Keywords 3D nesting · Additive manufacturing · SLS · 3D printing · FPA

1 Introduction

Additive manufacturing (AM), also known as 3D printing or rapid prototyping, is a layer-by-layer production method that aims at producing highly complex parts suitable for prototyping or customized production. Due to different types of materials, AM can be classified into solid-, liquid-, or powder-based printing methods. In addition, based on different printing principles, AM parts can be printed with or without support structures. Among different AM technologies, binder jetting (BJ), selective laser sintering (SLS), and color jet printing (CJP) are powder-based printing methods that do not need support structures and can lay models on top of others to print more parts in one setup. More recently, HP introduced multi jet fusion (MJF) that uses HP's high-speed print head for spraying infrared light sintering agent before solidification, hence greatly increasing the production speed. Mass customization becomes possible when direct printing of various parts in one batch can be realized without the tedious post-processing work of removing supports. AM with 3D packing in one container batch can significantly increase production capacity and efficiency particularly for multiple customized parts. However, the 3D layout planning before printing will affect the printing time and production cost. Therefore, effective nesting or packing of 3D models in the build volume to reduce the build height becomes a practical and important research subject. An emphasized characteristic of AM technology is that it does not employ any dedicated physical tooling such as moulds, cutting implements or dies [1, 2]. Since there are no physical tools, parts can be manufactured cost-effectively in small batches, or even just

✉ Hong-Tzong Yau
imehtyccu@gmail.com

[1] Department of Mechanical Engineering, National Chung Cheng University, Chia-Yi, Taiwan, Republic of China

[2] Advanced Institute of Manufacturing with High Innovation, National Chung Cheng University, Chia-Yi, Taiwan, Republic of China

a single customized product [3]. As a concurrent manufacturing process, AM enables the parallel manufacture of different geometric products in a single build volume [4]. This creates a build volume 3D nesting problem during the machine setup process, which should be solved computationally rather than manually by experienced technicians [5, 6]. Baumers et al. [7, 8] have shown that the effectiveness of solving this problem in AM execution constitutes an important determinant of manufacturing cost. In the absence of physical tooling, it has been argued that AM should be considered as a flexible manufacturing system [9]. Therefore, the build time for a volume depends upon the build height and the number of build tasks needed for a set of products, which is under the direct impact of the 3D nesting result [7].

Because AM is a layered printing process, contours of sliced models can be generated and processed in each layer, hence processing multiple models and parts at the same time. In this sense, AM can be considered as an ideal concurrent manufacturing process. With the fixed build volume as a constraint, optimization methods can be utilized to lay out the printed models as compact as possible to maximize printing efficiency and reduce production time and cost. Considering the fixed build volume inside an AM container, there are two primary packing or nesting approaches: 2D nesting (single stacking) and 3D nesting (multiple stacking). This is because in some AM methods such as fused deposition modeling (FDM), stereolithography (SLA), and electron-beam melting (EBM), support structures are needed and therefore it is not recommended to lay models on top of each other. For these AM methods, 2D nesting of models is usually employed. On the other hand, part support in 3D space for powder-based AM methods such as SLS, BJ, and MJF comes from the powder itself; therefore, no additional support structure is required. Models can lay on top of lower level models, creating more work space for printing parts in one setup.

Traditionally, 3D nesting has been operated manually by experienced engineers or technicians. When nested objects are complex in geometry and large in quantity, processing time becomes longer and build height becomes difficult to minimize. As a result, printing time and production cost are increased and manufacturing efficiency is reduced. Therefore, automating and optimizing 3D nesting problem in 3D printing becomes an important and practical research topic. In the process of 3D nesting, positions and orientations of different 3D objects need to be carefully arranged so that collisions are avoided and the empty space or the build height is minimized. When object geometry becomes more complicated and the number of objects becomes larger, the level of complexity in 3D nesting increases exponentially. Finding a near-optimal 3D nesting solution in a reasonable and acceptable time frame poses a real challenge in real-world manufacturing situations.

Garey and Johnson [10] have shown that 3D nesting is an NP-hard problem which can only be solved by numerical methods to obtain the near-optimal solution. Meta-heuristic algorithms have been proposed to approach the problem. In the following paragraphs, 3D nesting and related solutions are reviewed before our proposed method is presented. Ikonen et al. [11] adopted a genetic algorithm (GA) approach to solve the 3D nesting problem for SLS process. For collision detection, part geometry is approximated by feature points and interference checking is performed on bounding boxes. Hur et al. [5] also reported a GA-based method for the SLS process. They proposed a modified bottom-left(BL) approach to search for the best part orientation and sequence with genetic algorithm. In contrast to bounding box collision detection, they voxelized part geometry and used the voxel structure for interference checking. However, different part orientation/position requires the generation of a new voxelization. Hence, it is time consuming and memory demanding for higher resolutions and complex geometries. Furthermore, from our later discussion, it is found that GA is inefficient compared with nature-inspired heuristic algorithms. Stoyan et al. [12] used mathematical modeling of convex polytopes for packing 3D objects into a parallelepiped. However, the geometric entities are usually rather simple and the mathematical modeling approach has difficulty handling complex models. Also, there is no triangulated mesh and also no rotation but only translation of objects. Gogate and Pande [13] proposed a multi-objective optimization approach for the general nesting problem in AM. The objective functions include build height, average staircase error, total support structure volume, and contact areas between parts and support structures. Weights can be arbitrarily assigned to interpolate between the different objective functions before they are combined to find the overall optimization goal. Similarly, they also used the BL approach and GA to find the solution. Voxelization of complex geometric models is also used to simplify interference checking. Egeblad et al. [14] proposed a heuristic algorithm for 3D nesting. Likewise, they dealt with translation only and packed objects into a parallelepiped. Their objective was to minimize the volume between overlapped objects. In comparison, their result was better than Stoyan et al. [12] by 14% under the same circumstances. However, utilizing only translations but no rotations means they cannot deal with highly complex or concave objects. Wu et al. [15] proposed another GA-based algorithm and improved the BL method to solve the 3D nesting problem and packed objects into a rectangular container. They built an octree for the voxelization of 3D objects. For optimization, they divided the GA algorithm into 2 parts. The first part deals with the selection of the object orientation and the packing sequence. The second part uses an improved BL method to place the objects in the best positions. Still, their method is limited by the use of GA-based algorithm and the voxelization approach. Liu et al. [16] proposed a new constructive algorithm called HAPE3D to place all objects into a rectangular

container. The method is also a heuristic approach and aims at finding the minimum build height when dealing with various geometric shapes. Their orientation angles increase in increment from 45, 90, to 180°, altogether with 24 possible orientations. Since it is a heuristic approach in nature, it can also be combined with other meta-heuristic algorithms such as a simulated annealing (SA) algorithm. However, since their optimization algorithm is still based on SA, which is not very efficient, the computation time for real-world cases (where the number of polyhedrons > 50) is still in the order of hours.

As stated, the 3D nesting problem is NP-hard and not easily modeled [10]. Hence, most of the existing solutions are based on heuristics or meta-heuristics. Still, there were some researchers who tackled the 3D nesting problem by mathematical modeling [12, 17, 18]. However, the geometric entities are usually rather simple and the mathematical modeling approach finds it difficult to handle complex models. Another shortcoming of mathematical modeling is that in the AM context, a model is usually provided as a three-dimensional mesh in the stereo lithography (STL) format [19]. Mathematical modeling with simple geometric entities does not cover the most general representation of STL models.

It is quite clear from existing literature that most of the published works use meta-heuristic algorithms to approach the combinatorial 3D nesting optimization problem. Genetic algorithm (GA) is perhaps the most used meta-heuristic algorithm in 3D nesting. Holland [20] first proposed this algorithm based on Darwin's natural selection principle. GA simulates the chromosome crossover, mutation, and selection steps. By the mutation in each iteration step, local minimum zone is escaped and the global minimum can be found by the most fitness value of the best survival. Another common method is the simulated annealing (SA) algorithm proposed by Kirkpatrick et al. [21] that mimics the annealing process of metal crystallization. Although GA or SA had been used quite often for solving 3D nesting problem, efficiency consideration has always been a concern for this type of optimization algorithms. To tackle this, quite a few nature-inspired meta-heuristic algorithms, such as BAT [22], firefly [23], and ant colony optimization [24], have been proposed in recent years. These algorithms improve in efficiency and robustness in finding the near-optimal solution. Yang et al. [25] proposed flower pollination algorithm (FPA) which simulates the pollination process in which pollen is taken from one plant to another so that new plant seeds can be produced. This new meta-heuristic algorithm has good convergence rate and mechanism for finding the global optimal value. It outperforms popular meta-heuristic algorithms such as GA, SA, or other nature-inspired algorithms in single value or multi-value optimization tests. Nabil [26] tried to further improve FPA by combining the colonial selection algorithm (CSA) with FPA. He tested the algorithm using 23 different functions and compared the result with those from 5 other algorithms (FPA, GA, SA, BAT, and firefly algorithms). The result shows that both FPA and MFPA outperform the other 4 algorithms. MPFA could sometimes perform better than FPA when using properly tuned parameters. The drawback of MPFA is that there are many parameters to set and their appropriate values are not easy to find.

From literature, most of the heuristic approaches in 3D nesting-adopted GA or SA methods [11, 13, 15, 27, 28]. It is clear that the computational efficiency and accuracy of these traditional heuristic approaches are not comparable to the most recent nature-inspired algorithms. Furthermore, among the nature-inspired algorithms, FPA has the advantage in efficiency and accuracy over traditional GA or SA algorithms, and is easy to implement with fewer parameters compared with MPFA.

To pack complex 3D models in space without overlapping, collision detection or interference checking between 3D models needs to be carried out for different part orientations and positions. In the past, two major approaches have been taken to deal with this problem. One is using detection of approximate bounding box interference which is popular in computer graphics applications; the other is using voxelization of objects to check interference by finding shared occupied voxels between models. Voxelization of STL models can be achieved using marching cubes [29]. However, different part orientation/position requires the generation of a new voxelization. Furthermore, it is time consuming and memory demanding for higher resolutions and complex geometries. In contrast, bounding box decomposition approaches, no matter if it is axis-aligned bounding box (AABB) or oriented bounding box (OBB), need to be done only once and then rely on efficient bounding box detection for different part orientations and positions. Gottschalk et al. [30] proposed the OBB tree (OBBT) data structure for decomposing triangulated meshes in order to reduce computation time for collision detection. OBBT can be applied to complex STL models and still have efficient collision detection efficiency. Hierarchical structure of a binary tree can be used to remove objects with no collision at a high efficient rate. Only nearby objects will traverse to deeper nodes for interference checking. Separating axis theorem (SAT) [30] is used to quickly check the interference between two OBBs with high efficiency. Bergen [31] compared collision detection efficiency between AABB tree and OBB tree by using 3 different models. It is found that an OBB tree is always faster than an AABB tree. Although the OBB tree construction time is longer than AABB tree, it takes only one construction for all models. In contrast, AABB needs to be built every time the part orientation changes.

Last but not the least, in the 3D packing of irregular objects into a build volume without overlapping, the minimum allowable distance between objects is usually assigned as a process parameter. There has been no study to address the effect of this

parameter on the result of 3D nesting. In this work, we show that it is a very useful property to decide and influence the optimization efficiency.

2 Proposed 3D nesting solution

From existing literature, it is found that most researches in the past used GA or SA to find the global optimization solution when dealing with the 3D nesting problem. Computational efficiency of GA and SA is relatively low and this makes the waiting time unacceptable in some real-world AM applications. From a literature survey of the most recent meta-heuristic algorithms, it is found that the flower pollination algorithm (FPA) has a better performance on the global optimization problem. However, as far as the authors know, FPA has never been applied to the 3D nesting problem. Therefore, in this work, for the first time, FPA is combined with 3D collision detection using OBBT to minimize the build height and printing time for AM applications. The goal of this work is to introduce a reasonable and useable 3D nesting solution for practical and real-world AM usages. The following are the advantages and innovations of the proposed method:

a. Using OBB to find a suitable initial rotation angle for object orientation strategy.
b. Using OBB to find the largest print area and the smallest print height for each object and sort the packing sequence accordingly.
c. Using FPA to efficiently find the near-optimal model positions and orientations.
d. Using OBBT to replace triangle meshes for interference checking to increase the collision detection efficiency. In particular, specified clearance distance or part interval between models can be used to adaptively reduce OBBT decomposition levels and increase computational efficiency.
e. For real-world AM applications involving 3D nesting of a large number of complex geometric models, the proposed method can reach better and faster solutions than GA- or SA-based methods.

Figure 1 shows the flow chart of the proposed method. First, the models to be nested are read. Based on the specified safety clearance distance between models, OBBT is computed for each model, and the initial orientation angle is also obtained before FPA. During the FPA 3D nesting optimization, the minimum build height is set as the primary objective function; the secondary objective function is to move and orient the model as close to the origin of the container as possible. The iterations continue until the convergence criterion is met and the result is output for 3D printing. In the following, we will describe in detail how to use FPA for 3D nesting, and how to integrate 3D collision detection (including how to reduce unnecessary interference tests to increase efficiency) to create a complete 3D nesting process.

2.1 Combinatorial optimization using FPA

3D nesting is categorized as a combinatorial optimization problem. The goal is to lay out a set of irregular 3D models into a fixed-size container or the so-called build volume. As described previously, combinatorial optimization is an NP-hard problem [10]. For the packing of a set of models, the sequence of model selection and the position/orientation of each model all need to be sorted out to find the optimization solution which gives the lowest build height. As the number of models increases, the computation time will increase exponentially. A brute-force search for the best solution is computationally expensive if not impossible. Heuristic algorithms seek the near-optimal solution by randomly searching the variable space to reduce computation. By setting boundary conditions in the search space, heuristic algorithms can narrow down the search and speed up the solution finding. However, because of the limited search space, heuristic algorithms often end up with the search in the local minimum zone. Meta-heuristic algorithms provide a remedy to this problem by using escaping functions to jump out of the local traps to arrive at the global optimum. This is the basic principle of the FPA. In the following section, we will briefly review the FPA algorithm.

Inspired by the flower pollination process in nature, Yang et al. [25] proposed PFA as a new meta-heuristic algorithm. It can be applied to non-deterministic polynomial problems with a single objective or multiple objectives often found in industrial problems, for example, the welded beam design optimization in the design of structures [32]. FPA has the following advantages: few parameter setting, easy implementation, good convergence rate, and high computation efficiency. It aims to find the global optimization solution through the global pollination process which changes its parameters to approach the better solution in the next iteration steps. In FPA, 4 rules are specified to simulate the flower pollination process [25]:

1. When insects such as bees or butterflies pollinate among different flowers, they fly around the flowers in the mode of Levy flights. This is considered as global pollination.
2. Non-insect or self-pollination is regarded as local pollination.
3. Flower constancy can be considered since the reproduction probability is proportional to the similarity of the two flowers involved.
4. Switching between global and local pollination is decided by a random parameter $p \in [0,1]$. When $p = 1.0$, it is equivalent to the global pollination.

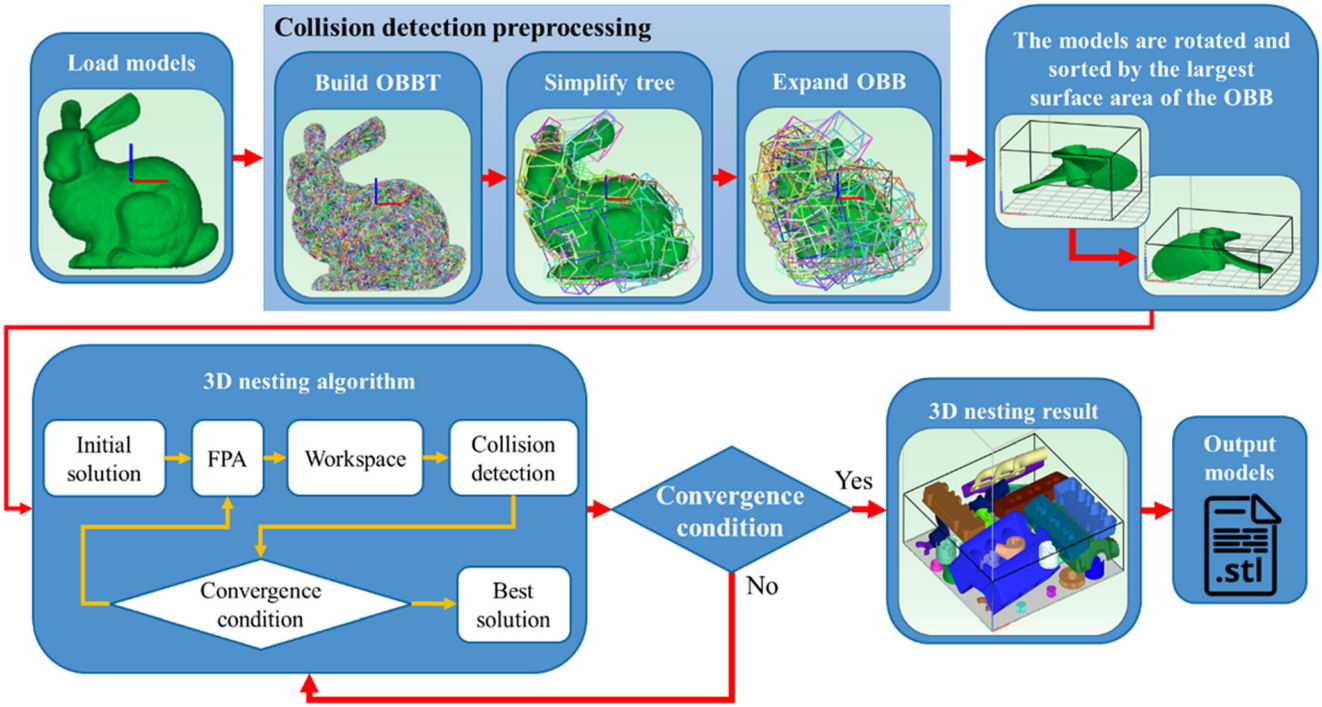

Fig. 1 A flow chart of the proposed 3D nesting solution

The above rules can be converted to iterative updating equations. First, pollinators such as bees or butterflies can carry the gametes for a long distance. This can be considered as global pollination and by combining rule 1 and rule 3 the updating equation can be written as:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)\left(g^* - x_i^t\right) \tag{1}$$

x_i^t is the pollen i at iteration t, and g^* is the best solution at the current iteration. γ is a scaling factor to control the step size. $L(\lambda)$ is the iteration step distance based on Lévy flight which corresponds to the pollination strength. Since insects usually can move over a long distance, Pavlyukevich [33] proposed their travels be modeled by Lévy flight. The Lévy distribution can be formulated as:

$$\mathrm{L} \sim \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{\mathrm{s}^{1+\lambda}} \tag{2}$$

In Eq. (2), $\Gamma(\lambda)$ is the standard Gamma function. Mantegna [34] suggested to use the following equation to obtain s:

$$\mathrm{s} = \frac{\mathrm{U}}{|\mathrm{V}|^{1/\lambda}}, \quad \mathrm{U} \sim \mathrm{N}\left(0, \sigma^2\right), \quad \mathrm{V} \sim \mathrm{N}(0,1) \tag{3}$$

U and V are random normal distribution and σ^2 can be calculated by

$$\sigma^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right]^{1/\lambda} \tag{4}$$

In parameter setting, Mantegna [34] found λ=1.5 to be the most appropriate through many tests.

In self-pollination (combining rule 2 and rule 3) which simulates local optimization, the updating equation can be formulated as:

$$x_i^{t+1} = x_i^t + \epsilon\left(x_j^t - x_k^t\right) \tag{5}$$

where x_i^t and x_i^{t+1} are pollen from different flowers of the same plants. This simulates the behavior of a local walk among the same species if ϵ is drawn from a uniform distribution in [0,1].

In practice, pollination can occur in local or global scales. Rule 4 can be simulated by using a switching probability p to switch between the two. An initial value of p = 0.5 can be used. However, Draa [35] used the CEC 2013 database of 28 functions to show that p = 0.2 will produce the best results. This basically states that there is about 80% of probability that local pollination will occur because adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those far away.

2.2 Collision avoidance in 3D nesting

In this paper, the proposed 3D nesting algorithm aims at finding the best object positions and orientations in AM through the FPA optimization, at the same time avoiding object interferences and keeping a safety clearance distance between

adjacent objects. In other words, collision detection can be viewed as the constrained conditions in the FPA optimization.

Collision detection has been widely used in video games. In interactive gaming such as car racing and wall climbing, collision detection is often used to trigger responding events [36]. In recent years, collision detection has also been increasingly applied to industrial applications related to virtual manufacturing simulation such as multi-axis machining and robotic manipulation.

Most geometric models for 3D printing are nowadays represented by triangle-based STL format. A complex STL model can sometimes consist of more than several hundred thousand triangles. It would be very computationally expensive to calculate interferences of all triangles between models. To reduce the computation burden, geometric models are usually approximated by the closing bounding volumes (BV) to construct a tree-structured bounding volume hierarchy (BVH).

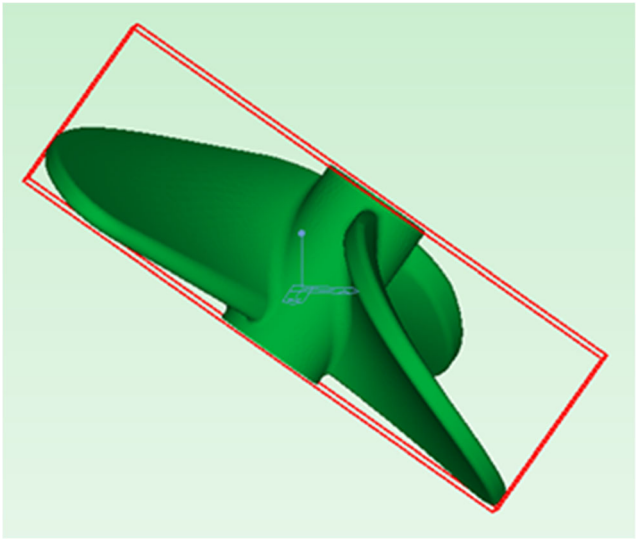

Fig. 2 OBB of a 3D fan model

2.2.1 Bounding volume

Bounding volume is a sphere or a simplified polygonal box that encloses the complex geometric model. When interference checking is carried out, bounding volumes are used to quickly rule out untouched volumes or models. Although spheres are easy to construct and quick for interference checking, spherical bounding volumes are too large and cannot adequately enclose 3D objects. In comparison, axis-aligned bounding box (AABB) and oriented bounding box (OBB) are more often used to approximate complex geometry. The advantage of using AABB is that it is simple and easy to construct and quick to test. However, it is axis dependent. If we rotate objects in a fixed (container) coordinate system, AABBs will change their sizes and need to be computed every time we rotate the objects. In contrast, OBB is axis-independent and is the minimum of AABBs. It can more adequately describe or enclose the object (see Fig. 2). The principal axes of an OBB can be obtained by finding the eigenvectors of the covariance matrix. The steps to find the principal axes of an OBB are outlined below:

1. Find the mean positions for all of the 3D object's vertices.

$$\left(\overline{\mathrm{X}},\overline{\mathrm{Y}},\overline{\mathrm{Z}}\right)=\left(\frac{\sum x_{\mathrm{i}}}{\mathrm{n}},\frac{\sum y_{\mathrm{i}}}{\mathrm{n}},\frac{\sum z_{\mathrm{i}}}{\mathrm{n}}\right) \tag{6}$$

2. Find the covariance of each pair of axes using Eq. (7).

$$\mathrm{cov}(\mathrm{x},\mathrm{y})=\frac{\sum\left(\mathrm{X}-\overline{\mathrm{X}}\right)\left(\mathrm{Y}-\overline{\mathrm{Y}}\right)}{\mathrm{n}} \tag{7}$$

3. Construct the covariance matrix using Eq. (8). Find the eigenvalues and eigenvectors of the covariance matrix using singular value decomposition (SVD). The primary principal axis is the eigenvector that corresponds to the largest eigenvalue.

$$\mathrm{C}=\begin{bmatrix}\mathrm{cov}(\mathrm{x},\mathrm{x}) & \mathrm{cov}(\mathrm{x},\mathrm{y}) & \mathrm{cov}(\mathrm{x},\mathrm{z})\\ \mathrm{cov}(\mathrm{x},\mathrm{y}) & \mathrm{cov}(\mathrm{y},\mathrm{y}) & \mathrm{cov}(\mathrm{y},\mathrm{z})\\ \mathrm{cov}(\mathrm{x},\mathrm{z}) & \mathrm{cov}(\mathrm{y},\mathrm{z}) & \mathrm{cov}(\mathrm{z},\mathrm{z})\end{bmatrix} \tag{8}$$

4. Project all the points onto the principal axes and find the limiting positions to construct the boundary values of OBB.

It is noted that when an OBB is rotated, the size and shape of the OBB do not change and we only need to rotate the 8 vertices of the OBB.

2.2.2 Bounding volume hierarchy

Bounding volume hierarchy (BVH) is a tree structure, widely used in collision detection, ray tracing, and viewing frustum culling of computer graphics. Based on different BVs, different BVHs can be constructed such as AABBT or OBBT. As pointed out, OBB needs to be constructed only once as compared to many instances of AABB. Furthermore, OBBT can approximate complex geometric models with less number of levels than AABBT [31]. Since the 3D nesting problem requires many interference checking between complex geometric

models, it saves more computation time to adopt OBBT for our hierarchical data structure. Therefore, in this work, we choose to build the binary OBB tree in a top-down hierarchical structure. The root or the top node is the OBB of the entire model. The geometric centers of all the triangles inside the OBB are projected onto the primary principal axis of the OBB. The principal axis is divided into two equal halves with triangles on each side forming two subdivided OBBs (see Fig. 3). At each level of the OBBT, this subdivision process continues and every leaf node is divided into two until the clearance distance of the model is larger than the minimum height of the OBB (see the discussion in Section 2.2.4), or the leaf node becomes a single triangle (see Fig. 4). During collision detections, if the parent node does not have interference, the child nodes of the next level need not be further tested, hence saving much computation effort.

2.2.3 Interference test between OBBs

To determine if two OBBs have interference, Gottschalk et al. [30] proposed a very efficient method—the separating axis theorem (SAT). In Fig. 5, it is shown that if OBB A and OBB B are projected onto any of the 3 axes of each OBB, or onto any of the axis that is orthogonal to any pair of the two axes from OBB A and OBB B, we can quickly determine if there is interference. Altogether, there will be 15 such cases for testing and for any of them, if Eq. (9) stands, there is no interference.

$$|\mathrm{T} \cdot \mathrm{L}| > \mathrm{r_A} + \mathrm{r_B} \tag{9}$$

2.2.4 Reduction of OBBT

One nice feature about OBBT is that if we divide the tree all the way to the final leaves, they can test all the STL triangles. However, it is of too much computation and it would not be necessary if we can terminate the unnecessary tests early to save time. In real-world AM applications, no matter SLS, BJ, or MJF, there will always be a requirement to maintain a safety clearance distance between printed parts to prevent sticking or breaking of parts. This part interval is usually about 3–5 mm dependent on the part size. When using SVD to find the principal axes of an OBB, we can find the largest surface area of the object with the minimum height (along the third primary axis). The subdivision of an OBBT node can terminate when the minimum height is equal to or smaller than the clearance distance. We can use this nice property to adaptively reduce the OBBT levels to save computation time and still meet the collision detection goal. Figure 6 shows different levels of OBBT (drawing only leaf nodes) with respect to different values of clearance distance. In addition, Fig. 7 shows the curve of the trend of OBB node reduction. It is observed that the number of OBB nodes drops very quickly after just few millimeters (Table 1 shows the number of leaf nodes versus clearance distance). Using this property, we can greatly reduce the levels and leaf nodes of OBBT and also the computation cost of collision detection.

2.2.5 Expansion of OBB

To ensure that the safety clearance distance is protected during 3D nesting, intervals between OBBs during interference testing need to be maintained. This can be achieved by extending the principal axes of the OBBs with half of the clearance distance in each direction (as shown in Fig. 8) so that when some STL triangles completely fall on the sides of OBBs the safety clearance distance can still be properly maintained.

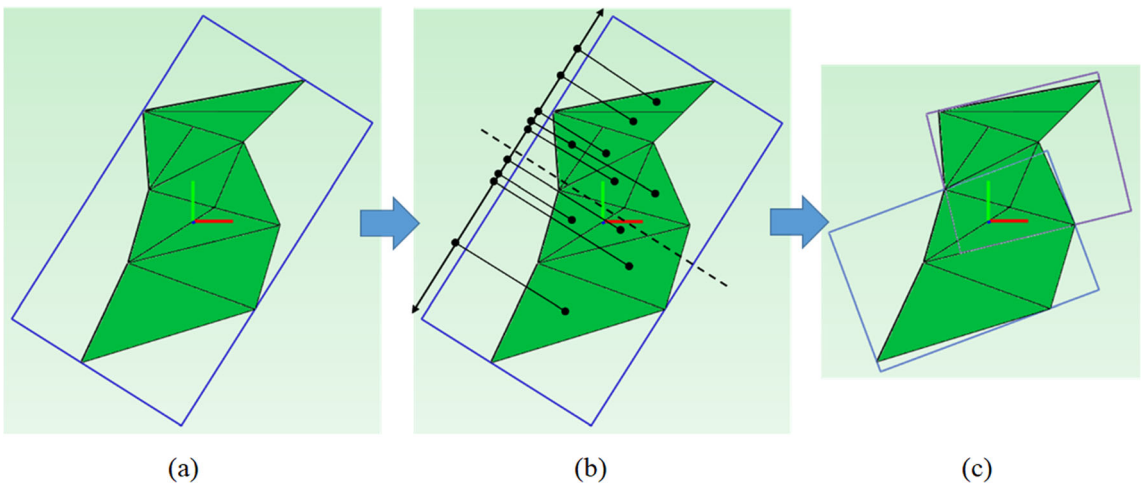

Fig. 3 Subdivision of an OBB: **a** root node, **b** geometric centers projected unto the primary axis, and **c** subdivided into two OBBs

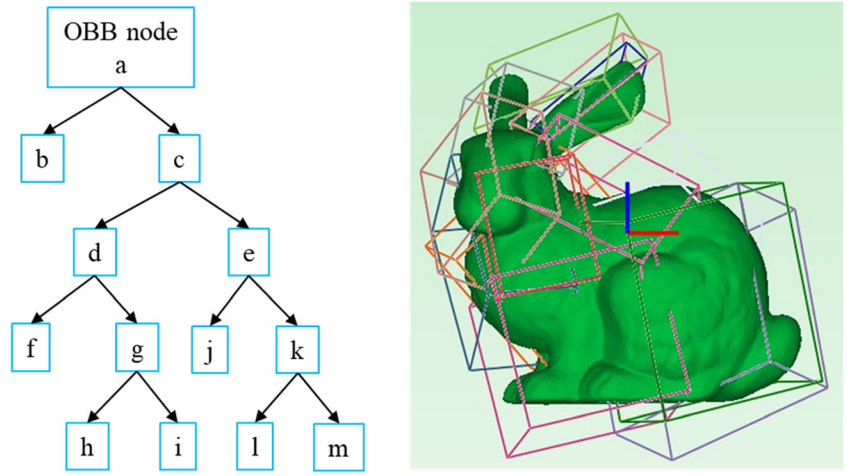

Fig. 4 BVH (leaf nodes: b, f, h, i, j, l, m)

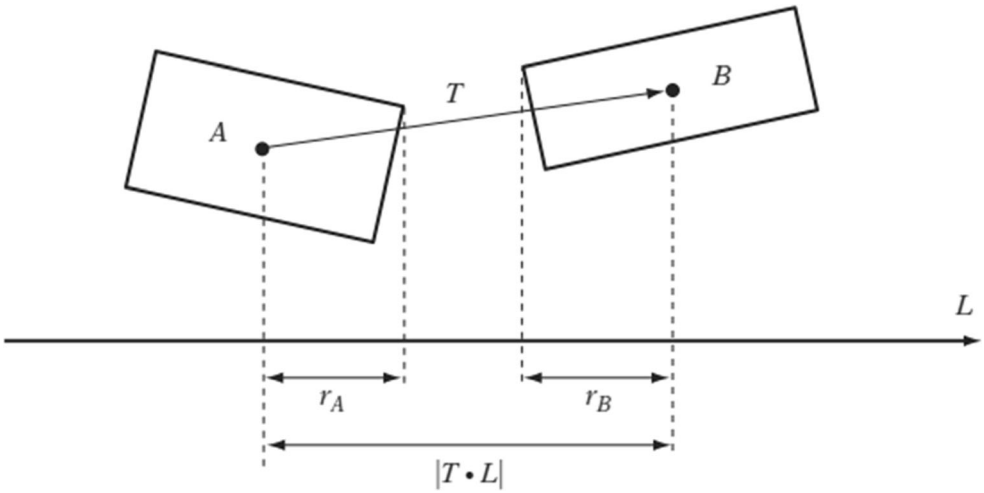

Fig. 5 Separating axis theorem [36]

2.3 Integration of FPA and collision detection by OBBT

To achieve the desired efficiency of 3D nesting of irregular shaped objects for AM technologies, we propose the integration of FPA optimization with 3D collision detection using OBBT. The software code is developed using C++ and OpenGL under Microsoft Visual Studio 2019 IDE. For verification of the algorithm efficiency, we will compare the 3D nesting results with that of commercial software such as Netfabb [37] and Materialise [38] in Section 3.

2.3.1 Process flow of the 3D nesting solution

As previously described, 3D nesting of complex 3D irregular shaped objects is an NP-hard optimization problem that requires the use of meta-heuristic algorithms. In literature, it is found that FPA is more efficient than traditional GA or SA algorithms. In addition, interferences of irregular shaped objects or complex geometric models present the constraints that need to be avoided during the optimization process. In this work, OBBT is adopted to efficiently test the interference conditions. The choice of OBBT has several advantages.

Table 1 Total number of OBBT nodes with different safety clearance distance

Safety clearance distance (mm)	0	1	2	3	4	5	6	7	8	9	10
Total number of OBBT nodes	39975	22537	6211	2341	1191	739	505	353	255	201	173

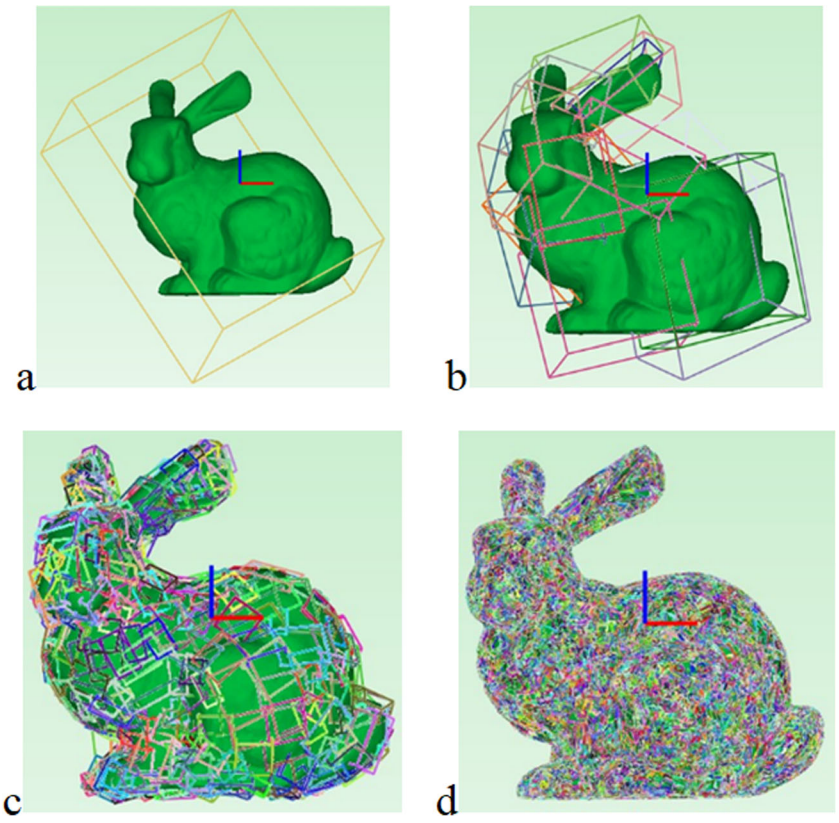

Fig. 6 OBBT with different safety clearance distance (only drawing leaf nodes): **a** 25mm, **b** 15mm, **c** 2mm, **d** 0mm

First, it can model any complex STL models and it only needs to be constructed once (as opposed to many times of voxelization in earlier studies). Second, the binary structure of OBBT allows the subdivision of OBBT until the minimum height of the OBB is smaller than the safety clearance distance. This feature makes good use of the fact that there is always a safety clearance between all the printed objects in SLS, BJ, or MJF AM systems. It allows the reduction of unnecessary OBBT subdivision levels and also the computation time. Third, OBB is not only used to find the principal axes of the object, it can also be used to find the largest area of the object with the minimum height. Therefore, it is convenient to use this orientation as the initial state to start the optimization iterations. Here, we outline the steps of the proposed 3D nesting algorithm (see Fig. 1):

1. Set the convergence conditions: The users can choose to have a single or multiple convergence conditions including specified print height, specified computation time, specified number of iteration, and relative error of print height as expressed in Eq. (10).

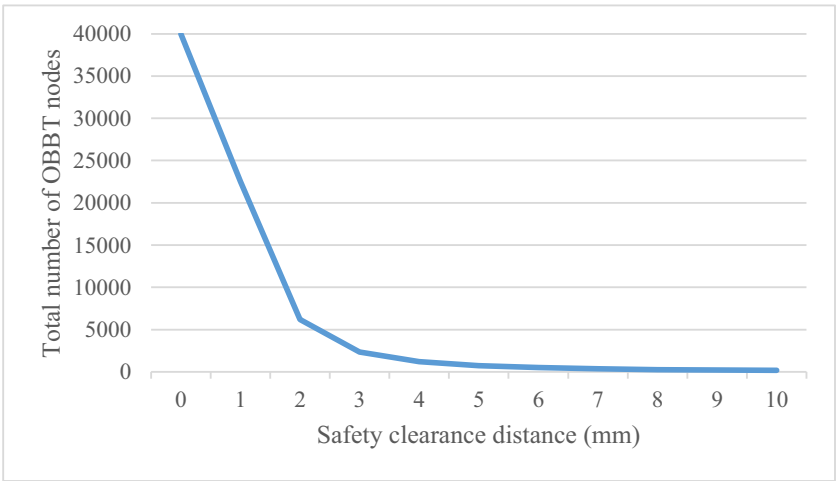

Fig. 7 Total number of OBBT nodes with different safety clearance distance

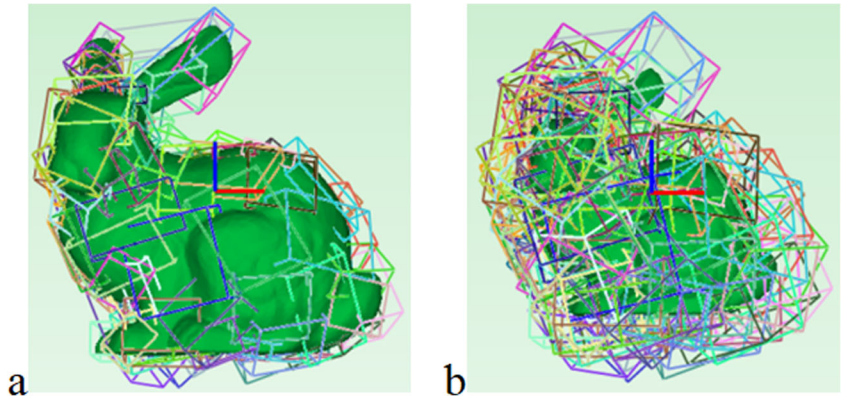

Fig. 8 Expanded OBB comparison: **a** without expansion and **b** expansion (model safety distance is 10mm)

$$\text{relative height error} = \frac{|A-B|}{A}100\% \tag{10}$$

A the print height of current iteration
B the print height of previous iteration

2. Read all the STL models, and construct the OBBT of each model based on the safety clearance distance.
3. Set up the sequence by sorting the OBB surface areas in descending order. The object with the largest OBB print area will be placed first then those with smaller print areas.
4. After the first object is placed, the rest of the objects will be oriented and positioned by FPA to find the optimized rotations and positions following the sequence in step 3.
5. Check the current print height and compare it with the last print height. If it is smaller, update the print height and record rotations and positions of all of the objects.
6. Check if any of the convergence conditions is met. If not, go back to step 4 and repeat the iterations.
7. If convergence is reached, record the best print height and rotate/translate all objects to the nesting orientations/positions.
8. Display the 3D nesting result. Output all models in the nesting orientations/positions in one STL file to the slicing software of the 3D printer.

Inside the 3D nesting algorithm, FPA is used to find the best position and orientation of the next nested model. The integration of OBB collision detection sets up the constraint conditions in the FPA optimization. The control flow chart of the FPA process integrated with OBB collision detection is shown in Fig. 9. The details are outlined in the following steps:

1. Set up the FPA parameters including number of population states, switching probability p, scaling factor γ, convergence criteria (relative percentage error or a fixed number of iteration), and safety clearance distance.
2. Set up the initial values for the population states, using these values to find the value of the objective function.

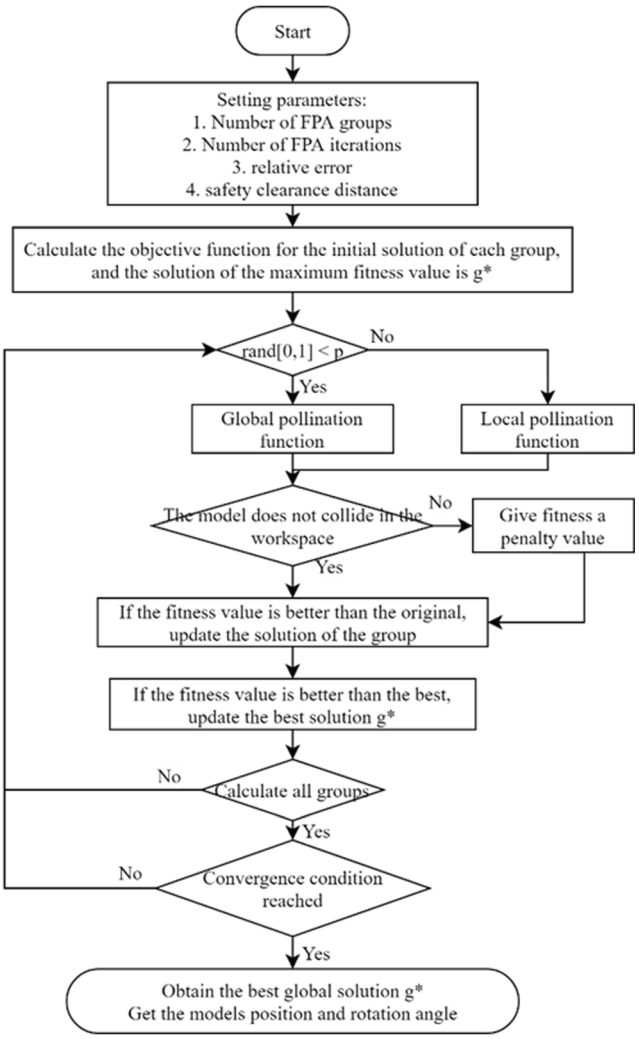

Fig. 9 The flow chart of combining FPA and collision detection.

Table 2 Algorithm parameters for the proposed 3D nesting algorithm

Relative error	Scaling factor γ	Switching probability p	Safety clearance distance
10^{-6}	0.1	0.2	3 mm

Note that the principal axes of OBB are adequately used to set the initial orientation angles.

3. Generate a random number between 0 and 1; compare it with the switching probability p. If it is smaller than p, take the global pollination route Eq. (1); otherwise switch to the local pollination route Eq. (5).
4. Check if the new states pass the interference test. This includes if the positioned models have interferences and if the placed model exceeds the boundary of the container. If the test fails, go back to step 3.
5. Substitute the new states into the objective function; if the resulting value (the print height) is smaller, keep the value and update the solution g^* to the new states.
6. Check if all states are visited. If not, go to step 3.
7. Check if the convergence condition is met? If not, go to step 3.
8. Output the model position and orientation to the 3D nesting iteration steps.

3 Implementation and comparison tests

In this section, we demonstrate two sets of 3D nesting test cases (a single model case and a multiple model case) and compare the result with that obtained from commercial softwares. The proposed 3D nesting algorithm is implemented in C++ with OpenGL. It is compiled using Microsoft Visual Studio 2019. The program was executed on a PC with I5-3570 3.4 Ghz CPU (single processor) with 16 GB memory.

3.1 Comparison tests with commercial software

In setting up the parameters of FPA, we followed the suggestion of [35] and used the scaling factor γ= 0.1 and the switching probability p = 0.2. The convergence criterion of the relative height was set to 10^{-6} or 10^{-4}%. The safety clearance distance was set at 3 mm which was used to maintain part intervals between 3D objects. The FPA parameter setting is listed in Table 2. We compare the 3D nesting performance with the results using commercial software Autodesk Netfabb Premium 2019 [37] and Materialise Magics 23.1 [38]. In Netfabb, there is no setting of rotation angles, and we used the choice of any rotation. In Materialise Magics, however, the default rotation angle is 90°, and the minimum rotation angle is 15°. We, likewise, used the same default rotation parameter setting. The print volume and safety clearance distance were the same. All the tests were run on the same computer. The key comparison indicators are the print height and the running time.

Table 3 Comparison of performance with and without OBBT reduction

	Without reduction (A)	With reduction (B)	Difference ((A-B)/A) 100%
Total number of OBB nodes	111900	60300	46%
Print height (mm)	67.9	69.7	-2.6%
Computation time (s)	102	102	0%

Table 4 Computation time and proportion of each part (single model)

	Calculation time (s)	Proportion of total time
OBBT construction	0.4	0.4%
FPA calculation	6.7	6.6%
Collision detection	94.9	93%

3.2 Results

3.2.1 Single model case

Given the specified safety clearance distance, the OBBT construction can be reduced to a reasonable level to reduce the collision detection time. Here, we use a fan model [39] to compare the result with and without OBBT reduction. The build space was set at 200×200×200 mm^3 to pack 20 fan models as shown in Fig. 12. With the safety clearance distance set at 3mm, and a fixed number of 100 iterations, it is found that the OBB nodes are reduced by 46%, the computation time reduced by 24%, and the increase of print height is only 3% (see Table 3). Table 4 shows the computation time divided into 3 major parts, namely OBBT construction, FPA calculation, and OBB collision detection. It is found that collision detection takes most of the computation time (93%).

Table 5 Comparison of the proposed method with commercial softwares (single model)

	Netfabb (A)	Magics (B)	Our method (C)	Difference with Netfabb ((A-C)/A)	Difference with Magics ((B-C)/B)
Print height (mm)	75.5	71	69.7	7.7%	2%
Computation time (s)	102	102	102	0%	0%

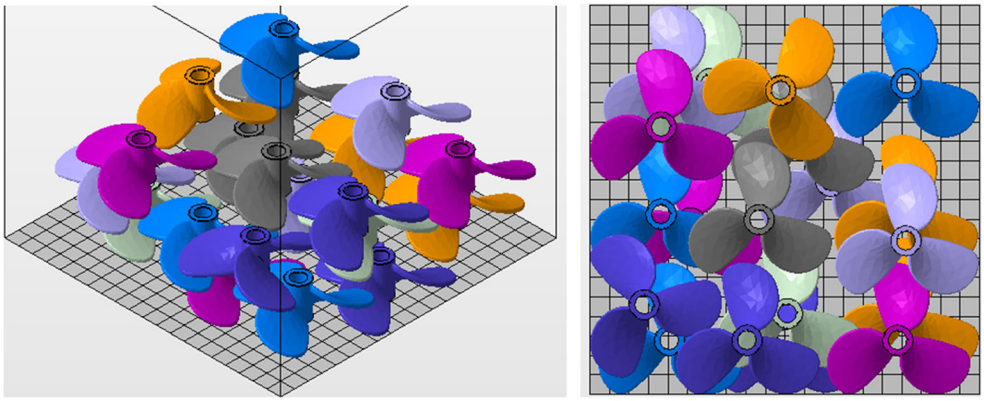

Fig. 10 Netfabb nesting result (single model case)

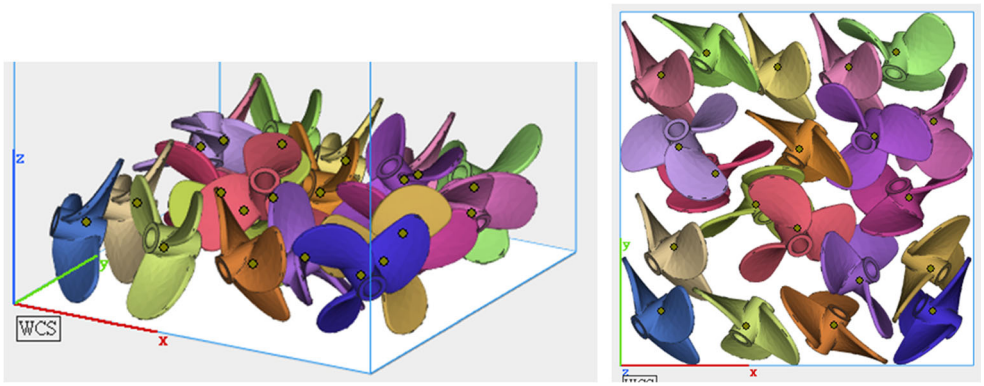

Fig. 11 Magics nesting result (single model case)

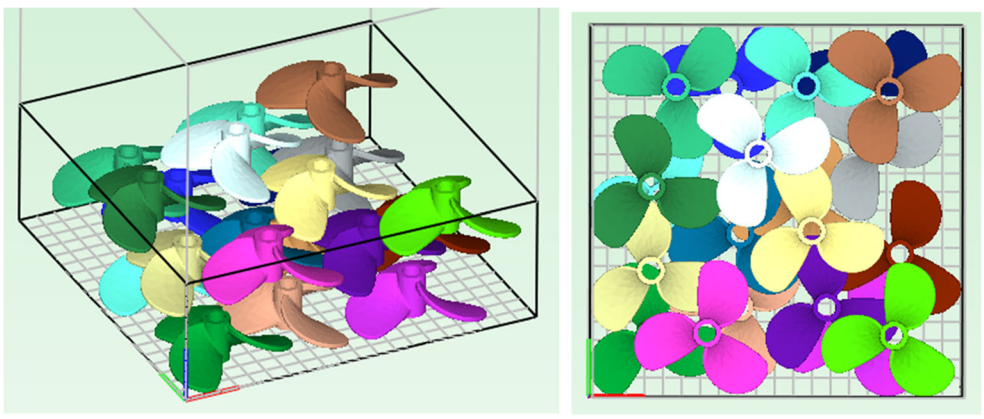

Fig. 12 The result of our proposed method (single model case)

Table 6 Comparison of the proposed method with commercial software (multiple models)

	Netfabb (A)	Magics (B)	Proposed method (C)	Difference with Netfabb ((A-C)/A)	Difference with Magics ((B-C)/B)
Print height (mm)	325.1	327.5	284.7	12.4%	13%
Computation time (s)	303	160	160	47.2%	0%

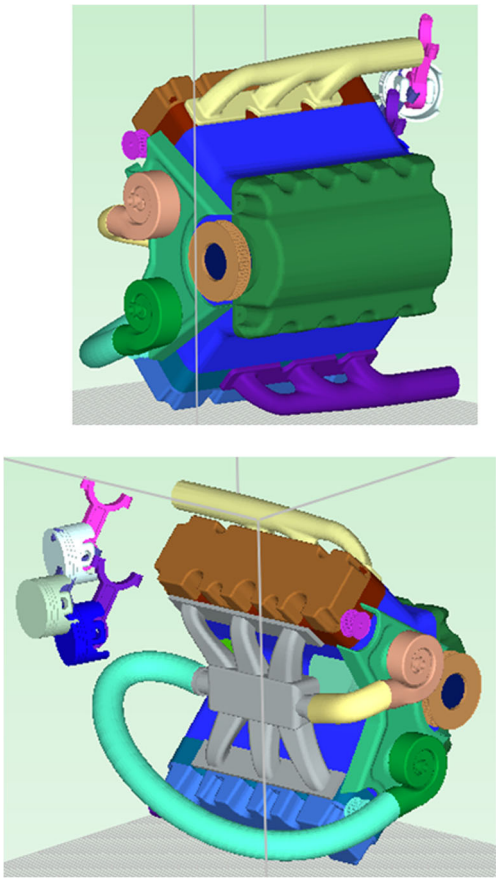

Fig. 13 V6 engine model (192,018 triangular meshes)

Therefore, the reduction of OBBT is much meaningful in helping to reduce the overall computation time.

In Netfabb, there is no setting of rotation angles and we use the choice of any rotation. In Materialise Magics, however, the default rotation angle is 90°. We used the same default rotation parameter setting. The build volume and safety clearance distance were the same. All the tests were run on the same computer. The key comparison indicators are the print height and the running time. In the single fan model test, Netfabb terminated the iterations at 102 s. Therefore, we purposely stopped the Magics and our proposed 3D nesting algorithm at the same time. Table 5 shows the comparison of results. Our proposed method has the better performance. Figures 10, 11 and 12 display the nested models using different software.

3.2.2 Multiple model case

In this section, we show the case of packing multiple parts of a V6 engine [40] (see Fig. 13). It consists of 30 different STL models and a total of 192,018 triangles. The build space is $1000 \times 1000 \times 1000$ mm^3. In this case, Netfabb terminated at 303 s. and Magics terminated at 160 s. Therefore, we purposely stopped our proposed method at 160 s. Table 6 shows the comparison result. Our proposed method has a better print height (12.4% better than Netfabb and 13% better than Magics). Figures 14, 15, and 16 show the graphical display of the nested models from 3 different methods.

4 Discussion

It is beyond question that the 3D nesting of irregular 3D objects is an important determinant to the cost reduction and control of powder-based AM processes. One of the major benefits of using AM is its capability to handle complex parts. However, from [41], it is found that most researches facing this problem in the past tend to use relatively simple cases for their demonstrations which cannot properly represent real-world problems. Even if the work used complex triangulated meshes rather than elementary geometric entities such as sphere, cubic, or convex polyhedrons for demonstrations, the number of faces or triangles are often less than 30,000. Yet, the computation time can go up to several hours sometimes [16, 18]. In this work, we show that in order to deal with real-world complex cases with efficiency, we need to pay attention to several key factors. First, we know the 3D nesting problem is NP-hard and not easily modeled. Some researchers tackled the problem by mathematical modeling. However, the geometric entities are usually rather simple, and the mathematical modeling approach finds difficulties handling

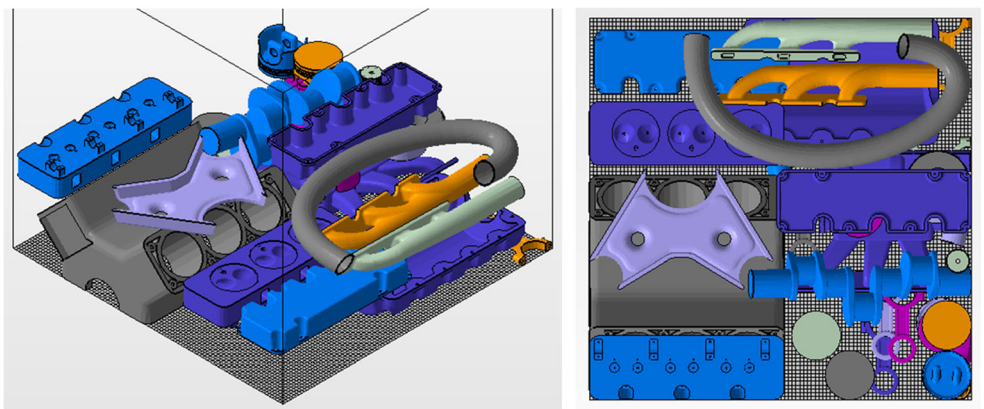

Fig. 14 Netfabb nesting result (multiple model case)

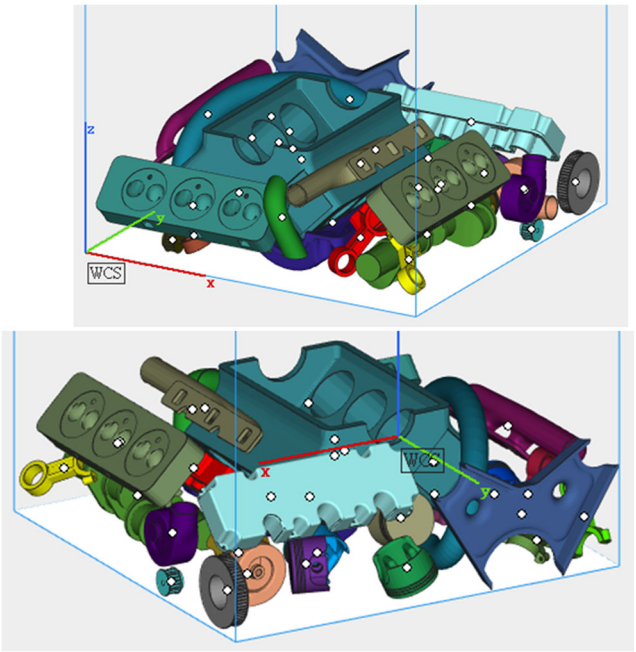

Fig. 15 Magics nesting result (multiple model case)

complex models. Hence, most of the existing solutions are based on heuristics or meta-heuristics. From literature, most researchers who used heuristic approaches in 3D nesting-adopted GA or SA algorithms. From Nabil [26], it is clear that the computational efficiency and accuracy of these traditional heuristic approaches are not comparable to the most recent nature-inspired algorithms. Furthermore, among the nature-inspired algorithms, FPA has the advantage of better efficiency and accuracy over traditional GA or SA algorithms, and it is easy to implement with only a few parameters compared with MPFA. Therefore, in this work, we choose to use the effective meta-heuristic optimization algorithm FPA, combined with the collision detection of printed objects set up as the optimization constraint using OBB tree to find the near-optimal 3D nesting solution. The result shows that for real-world complex engine parts (30 different STL models and a total of 192,018 triangles), the computation time is only 160 s, and the build height is 12.4% better than the result from Netfabb and 13% better than that from Magics.

Second, to pack complex 3D models in space without overlapping, collision detection or interference checking between 3D models needs to be carried out for different part orientations and positions. In the past, voxelization of objects to check interference by finding shared occupied voxels between models has been frequently used. However, different part orientation/position requires the generation of a new voxelization. Furthermore, it is time consuming and memory demanding for higher resolutions and complex geometries. In contrast, bounding box decomposition approaches, particularly oriented bounding box (OBB), need to be done only once and rely on efficient bounding box detection for different part orientations and positions. In this work, we demonstrate that

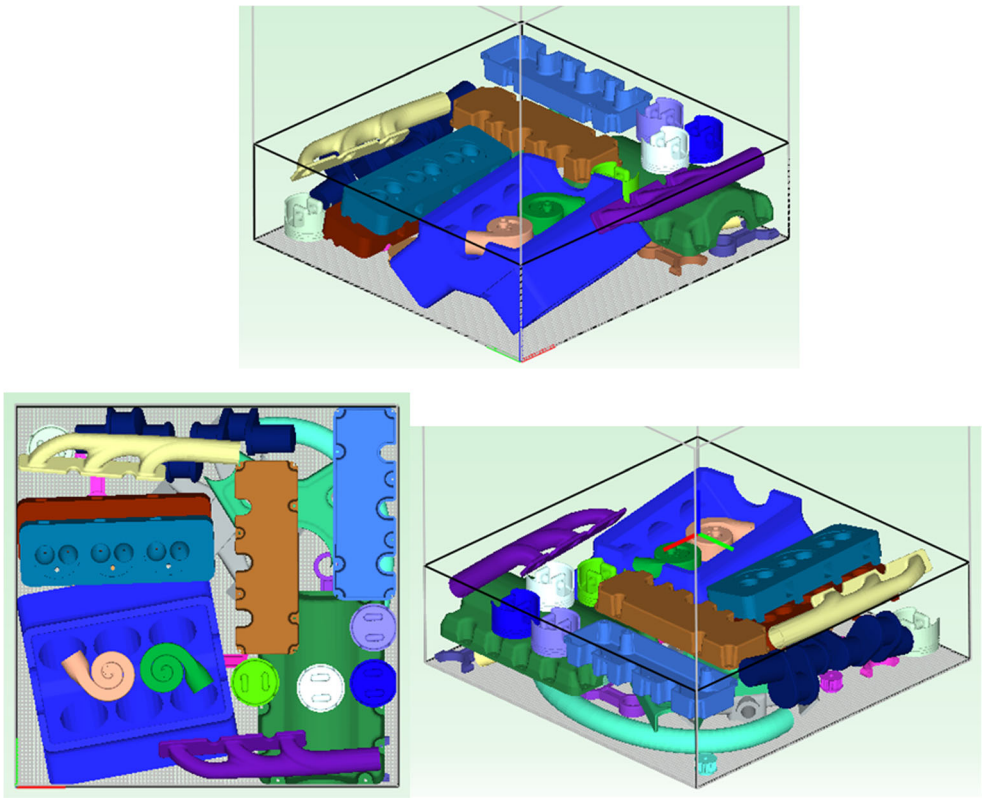

Fig. 16 The result of the proposed method (multiple model case)

the OBB tree can effectively detect collisions for real-world complex cases.

Last but not the least, in the 3D packing of irregular objects into a build volume without overlapping, minimum allowable distances between objects are usually assigned as a process parameter. In this work, we show that it presents a very useful property to reduce the number of OBB nodes and reduce the collision detection time, which constitutes most of the computation time due to its role as the optimization constraint.

5 Conclusions

AM has proved to be a versatile and agile production method for modern manufacturing. It is gaining more acceptance from the industry every day. It is particularly suitable for the production need of small quantity yet diverse variety of parts. 3D nesting of these various parts in one batch AM production hence becomes almost like an everyday routine work. This paper addresses the problem of 3D nesting of complex geometric objects for AM technologies. Through the integration of the FPA optimization method with collision detection using OBBT, this proposed method has shown to be an effective 3D nesting solution. The proposed method has several distinctive advantages. First, FPA has been accepted as a more efficient meta-heuristic optimization algorithm compared to traditional GA or SA algorithms. In this paper, it is the first time FPA is used to deal with the 3D nesting problem and it is proven to be more efficient compared with existing commercial software packages. Second, the use of OBBT can efficiently and accurately compute the interferences of complex CAD models. The active use of clearance distance between printed objects can effectively reduce the levels of OBBT and the computation time. Third, SVD is used to find the primary axes of an OBB and the largest area of the object with the minimum height (along the third primary axis). We can align the bottom of OBB with the base plane of the build volume. This gives a good initial orientation of FPA with the minimum height.

Since there will be increasing need for processing 3D nesting in AM production, it is necessary to further speed up the computation to reduce the production wait time in our future work. It is noted that more than 90% of computation takes place in OBB collision detection. Parallel computing will be a good solution since the OBBT structure allows independent computation in all the subdivided branches. Furthermore, 3D collision detection is not new to video game industry, so the use of GPU for hardware acceleration is also a reasonable choice for future implementation. In this work, our focus is on the optimization of the build height to reduce the cost of the AM process. Another future work is the optimization of multiple objectives considering additional factors such as dimensional accuracy [42] to find out the best parametric combination that can simultaneously optimize all performance measures [43].

Author contribution Both authors contribute equally to the theoretical development and experimental design and implementation of this work.

Funding The authors received the financial support the Ministry of Science and Technology of Taiwan (Grant number [MOST 109-2221-E-194-006]).

Data availability Data and material can be available upon request.

Code availability Code can be available upon request.

Declarations

Ethics approval The authors declare that the content of this work is original and there is no conduct of violating the principle of ethics in this work.

Consent to participate The authors were granted and approved to participate in this work.

Consent for publication The authors give full consent to the publisher for the publication of this work.

Conflict of interest The authors declare no competing interests.

References

1. Hague R, Mansour S, Saleh N (2004) Material and design considerations for rapid manufacturing. Int J Prod Res 42(22):4691–4708
2. Gibson IG (2014) Additive manufacturing technologies 3D printing, rapid prototyping, and direct digital manufacturing. Springer, Berlin, p 414
3. Tuck CJ, Hague RJ, Ruffo M, Ransley M, Adams P (2008) Rapid manufacturing facilitated customization. Int J Comput Integr Manuf 21(3):245–258
4. Ruffo M, Hague R (2007) Cost estimation for rapid manufacturing'simultaneous production of mixed components using laser sintering. Proc Inst Mech Eng, Part B: J Eng Manuf 221(11):1585–1591
5. Hur SM, Choi KH, Lee SH, Chang PK (2001) Determination of fabricating orientation and packing in SLS process. J Mater Process Technol 112(2-3):236–243
6. Nyaluke A, Nasser B, Leep HR, Parsaei HR (1996) Rapid prototyping work space optimization. Comput Ind Eng 31(1-2): 103–106
7. Baumers M, Tuck C, Wildman R, Ashcroft I, Rosamond E, Hague R (2013) Transparency built-in: energy consumption and cost estimation for additive manufacturing. J Ind Ecol 17(3):418–431
8. Baumers M, Beltrametti L, Gasparre A, Hague R (2017) Informing additive manufacturing technology adoption: total cost and the impact of capacity utilisation. Int J Prod Res 55(23):6957–6970
9. Weller C, Kleer R, Piller FT (2015) Economic implications of 3D printing: Market structure models in light of additive manufacturing revisited. Int J Prod Econ 164:43–56

10. Garey MR, Johnson DS (1979) Computers and Intractability - a guide to the theory of NP-completeness. W. H. Freeman & Co., New York, pp 109–118
11. Ikonen I, Biles WE, Kumar A, Wissel JC, & Ragade RK (1997) A genetic algorithm for packing three-dimensional non-convex objects having cavities and holes. in *ICGA*, pp591-598.
12. Stoyan YG, Gil NI, Scheithauer G, Pankratov A, Magdalina I (2005) Packing of convex polytopes into a parallelepiped. *Optimization* 54(2):215–235
13. Gogate AS, Pande SS (2008) Intelligent layout planning for rapid prototyping. Int J Prod Res 46(20):5607–5631
14. Egeblad J, Nielsen BK, Brazil M (2009) Translational packing of arbitrary polytopes. Comput Geom 42(4):269–288
15. Wu S, Kay M, King R, Vila-Parrish A, & Warsing D (2014)Multi-objective optimization of 3D packing problem in additive manufacturing. In IIE annual conference. Proceedings. pp 1485.
16. Liu X, Liu JM, Cao AX (2015)HAPE3D—a new constructive algorithm for the 3D irregular packing problem. Front Inform Tech Elect Eng 16(5):380–390
17. Chernov N, Stoyan Y, Romanova T (2010) Mathematical model and efficient algorithms for object packing problem. Comput Geom Theory Appl 43(5):535–553
18. Litvinchev I, Pankratov A, Romanova T (2019) 3D irregular packing in an optimized cuboid container. *IFAC-PapersOnLine* 52(13): 2014–2019
19. Gardan J (2016) Additive manufacturing technologies: state of the art and trends. Int J Prod Res 54(10):3118–3132
20. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press. pp 1-16.
21. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
22. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, SCI 284, pp. 65-74.
23. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84
24. Blum C (2005) Ant colony optimization: Introduction and recent trends. Phys Life Rev 2(4):353–373
25. Yang XS, Karamanoglu M, He X (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237
26. Nabil E (2016) A modified flower pollination algorithm for global optimization. Expert Syst Appl 57:192–203
27. Canellidis V, Dedoussis V, Mantzouratos N, Sofianopoulou S (2006)Pre-processing methodology for optimizing stereolithography apparatus build performance. Comput Ind 57(5):424–436
28. Canellidis V, Giannatsis J, & Dedoussis V (2010) Effective nesting of layer manufacturing fabricated parts using a genetic algorithm and a bottom-left ray casting procedure. In IEEM, IEEE International Conference, pp 547-551.
29. Lorensen WE, Cline HE (1987) Marching cubes: A high resolution 3D surface construction algorithm. ACM Siggraph Comp Graph 21(4):163–169
30. Gottschalk S, Lin MC, & Manocha D (1996). OBBTree: a hierarchical structure for rapid interference detection. In Computer Graphics (SIGGRAPH 96 Proceedings), pp 171–180.
31. Bergen GVD (1997) Efficient collision detection of complex deformable models using AABB trees. J Graph Tools 2(4):1–13
32. Ray T, Liew KM (2002) A swarm metaphor for multiobjective design optimization. Eng Opt 34(2):141–153
33. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
34. Mantegna RN (1994) Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phys Rev E 49(5):4677–4683
35. Draa A (2015) On the performances of the flower pollination algorithm–qualitative and quantitative analyses. Appl Soft Comput 34:349–371
36. Ericson C (2004)Real-time collision detection. CRC Press, New York, pp 1–2
37. Autodesk Netfabb. https://www.autodesk.com/products/netfabb.
38. Materialise Magics. https://www.materialise.com/en/software/magics.
39. Fabpilot. https://www.fabpilot.com/.
40. Atef A (2018) V6 Engine. GrabCAD Community Library. https://grabcad.com/library/v-6-engine-12-valve-1/details?folder_id=4802705
41. Araújo LJ, Özcan E, Atkin JA, Baumers M (2019) Analysis of irregular three-dimensional packing problems in additive manufacturing: a new taxonomy and dataset. Int J Prod Res 57(18):5920–5934
42. Mansaram MV, Chatterjee S, Dinbandhu, Sahu AK, Abhishek K, Mahapatra SS (2021) Analysis of dimensional accuracy of ABS M30 built parts using FDM process. Recent Adv Mech Infrastruct: Proceedings of ICRAM 2020:173–181
43. Sahu AK, Mahapatra SS, Chatterjee S (2017) Optimization of electrical discharge coating process using MOORA based firefly algorithm. Proceedings of the ASME 2017 Gas Turbine India Conference, Bangalore, pp 1–16