A chunk-based slicer for cooperative 3D printing

Jace McPherson and Wenchao Zhou
Department of Mechanical Engineering, University of Arkansas, Fayetteville, Arkansas, USA

Abstract

Purpose – The purpose of this research is to develop a new slicing scheme for the emerging cooperative three-dimensional (3D) printing platform that has multiple mobile 3D printers working together on one print job.

Design/methodology/approach – Because the traditional lay-based slicing scheme does not work for cooperative 3D printing, a chunk-based slicing scheme is proposed to split the print job into chunks so that different mobile printers can print different chunks simultaneously without interfering with each other.

Findings – A chunk-based slicer is developed for two mobile 3D printers to work together cooperatively. A simulator environment is developed to validate the developed slicer, which shows the chunk-based slicer working effectively, and demonstrates the promise of cooperative 3D printing.

Research limitations/implications – For simplicity, this research only considered the case of two mobile 3D printers working together. Future research is needed for a slicing and scheduling scheme that can work with thousands of mobile 3D printers.

Practical implications – The research findings in this work demonstrate a new approach to 3D printing. By enabling multiple mobile 3D printers working together, the printing speed can be significantly increased and the printing capability (for multiple materials and multiple components) can be greatly enhanced.

Social implications – The chunk-based slicing algorithm is critical to the success of cooperative 3D printing, which may enable an autonomous factory equipped with a swarm of autonomous mobile 3D printers and mobile robots for autonomous manufacturing and assembly.

Originality/value – This work presents a new approach to 3D printing. Instead of printing layer by layer, each mobile 3D printer will print one chunk at a time, which provides the much-needed scalability for 3D printing to print large-sized object and increase the printing speed. The chunk-based approach keeps the 3D printing local and avoids the large temperature gradient and associated internal stress as the size of the print increases.

Keywords Chunk-based slicing, Cooperative 3D printing, Mobile 3D printing, Swarm 3D printing

Paper type Research paper

1. Introduction

Although additive manufacturing (AM) has become increasingly popular in recent years, it has been significantly limited by its slow printing speed and the size of the object it can print. Cooperative three-dimensional (3D) printing is an emerging technology that aims to address these limitations by having multiple printhead-carrying mobile robots (or mobile 3D printers) working together on the same print job on a factory floor. With the traditional layer-by-layer 3D printing approach as defined by the ASTM F42 committee (ASTM-F42-Committee, 2012), it would be difficult for the mobile 3D printers to cooperate without interfering with the already printed part and with each other, which calls for a different approach of 3D printing.

In traditional 3D printing, a computer-aided design (CAD) model needs to be sliced into layers and the path of the printhead movement needs to be planned to deposit materials for each layer. A slicer usually works by intersecting a plane at different Z-heights with the CAD model and calculating the boundary segments on each layer. The movement path of the printhead is then determined to infill the region within the boundary at each layer. Many different slicers have been developed, such as Slic3r (Ranellucci, 2015), Cura (Braam, 2016), Kisslicer (2016) and Skeinforge (2015). C Kirschman *et al.* developed a parallel slicing algorithm to improve the slicing speed (ASTM-F42-Committee, 2012; Kirschman and Jara-Almonte, 1992). Sabourin *et al.* (1996) presented an adaptive slicing algorithm for layer-based 3D printing that can slice with different layer thickness. S. Lefebvre *et al.* reported a graphics processing unit-accelerated slicer (Lefebvre and Grand-Est, 2013). However, slicing for the emerging cooperative 3D printing technology has not been investigated before.

A slicer is usually accompanied by a visualizer for the user to see the slicing results. A G-code viewer is one of the most common visualizers, such as the built-in viewer in Repetier's study (2017). Because cooperative 3D printing involves multiple robots, a simulator that can visualize the dynamic path of each mobile robots and how the materials are deposited over time will be beneficial for validating the printing path and optimizing the printing strategy for cooperative 3D printing.

The current issue and full text archive of this journal is available on Emerald Insight at: **www.emeraldinsight.com/1355-2546.htm**

Rapid Prototyping Journal

[DOI 10.1108/RPJ-07-2017-0150]

Received 21 July 2017
Revised 25 September 2017
Accepted 27 September 2017

Many different simulators have been developed for mobile robots, such as Gazebo (Koenig and Howard, 2004), EyeSim (Bräunl, 2000), UberSim (Browning and Tryzelaar, 2003) and Simbad (Hugues and Bredeche, 2006). These robot simulators can effectively simulate the interaction of multiple robots in 2D or 3D for evaluation of the design and the behavior of the robots. However, simulators for visualizing the dynamic 3D printing process of mobile 3D printers have not been reported.

In this paper, to address the possible geometric interference arising from the layer-by-layer-based approach with multiple mobile 3D printers, we present a chunk-by-chunk-based slicing approach so that each mobile 3D printer only needs to print a small chunk at a time, which can effectively separate the mobile 3D printers. The key difference of the chunk-based cooperative 3D printing from other types of robotic 3D printing, contour crafting or multi-head/multi-axis 3D printing is that the cooperation is amongst multiple independent AM systems. The ultimate vision is to have a swarm of mobile 3D printers and other specialized robots (e.g. a pick-and-place robot) to work together in an autonomous digital factory. This chunk-based printing can also keep 3D printing localized and therefore potentially avoid the large temperature gradient and internal stress that are common with 3D printing large objects. With proper scheduling of each individual mobile printer, this approach can be scaled to a very large number of mobile printers without interference. To simplify the problem, the slicing algorithm in this paper will be limited to the cooperative 3D printing between two mobile 3D printers that carry a fused deposition modeling extruder. This chunk-by-chunk-based slicing algorithm ensures good bonding between the chunks and smooth transitioning when a mobile robot moves from one chunk to another. It is worth noting that the positioning and alignment of multiple mobile printers in the physical world is a non-trivial problem, which deserves separate research. Therefore, instead of validating the chunk-based slicing algorithm on the physical mobile printers, we created a simulator environment to simulate the dynamic printing process over time and the communication between mobile printers using the sliced results as an input. This simulator environment makes it much easier and less expensive to validate the slicing results, which provides a valuable tool to understand and optimize the printing strategies and save time and cost before submitting a print job. In addition, the simulator environment takes similar inputs as the physical mobile printers, which would make it effortless to submit the printing job to the physical mobile printers after validation in the simulator environment. Our results show that our chunk-based slicer works effectively for the two-robot printing strategy, as validated by the simulator. This new chunk-based slicer and the new simulator environment presented in this paper represent a significant step toward cooperative 3D printing where multiple independent 3D printers can work together.

This paper is organized as follows. A discussion of the overall slicing process is presented in Section 2. Section 3 presents the development of a chunker for the chunk-based slicing. The slicer is discussed in Section 4. Section 5 presents a simulator to simulate the sliced results. Conclusions are given in Section 6 and future work in Section 7.

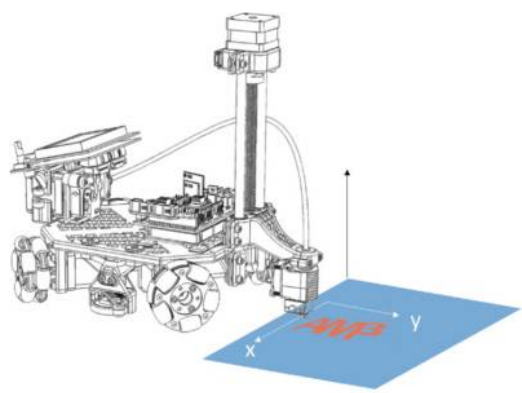
2. Slicing for cooperative three-dimensional printing

At the core of the cooperative 3D printing platform is a mobile 3D printer, as shown in Figure 1, which replaces the XY stage on a regular 3D printer with a set of omnidirectional wheels to translate the printhead in XY direction. This design enables unlimited printing in the X direction, but the Y direction is limited by the distance between the printhead and the front wheels (termed as "build depth" in this paper) if a layer-by-layer-based approach is used because the printed material in the previous layers will block the path of the wheels in Y direction.

In this paper, we propose a chunk-by-chunk-based printing strategy, where the mobile printer finishes all the layers of one chunk before it moves to print another chunk, effectively solving the problem of the blocked path by the printed materials to enable the mobile printer printing unlimited in both X and Y directions. Several methods for portioning CAD models for 3D printing have been developed, but they are mostly used for printing parts that can be assembled into a single model in post-processing. Examples include Chopper (Linjie *et al.*, 2012), curvature-based partitioning methods (Hao *et al.*, 2011) and skeletonization (Xu *et al.*, 2016). While these methods are useful at dividing model meshes for post-process assembly, the chunking method for cooperative 3D printing requires that all chunks be printed such that they are attached without post-processing, requiring a new, different method. One issue that arises is the bonding between the chunks. Our solution to this issue is to use a sloped interface (and/or an angled printhead) to allow more bonding surface between the chunks. A general slicing strategy for cooperative 3D printing is illustrated in Figure 2:

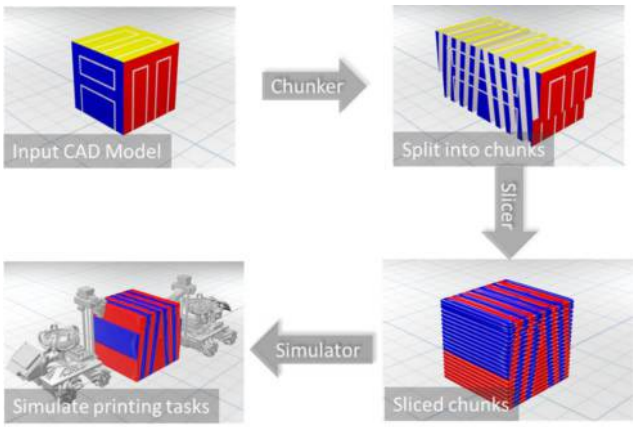
- *Chunker*: A CAD model of the print job will be first input into a "chunker," which splits the CAD model into chunks based on a set of criteria to ensure feasible printing of each chunk and good bonding between chunks.
- *Slicer*: The chunks will then be sliced into layers using a slicer, which generates commands for printing the chunks (e.g. tool paths, material extrusion and temperature control), schedules the sequence of printing the chunks among multiple robots and inserts communication commands to enable necessary communication among multiple robots.

Figure 1 Illustration of a mobile 3D printer



Note: It can print indefinitely in X direction but limited in Y direction

Figure 2 Illustration of the slicing strategy for cooperative 3D printing



Notes: (1) The chunker splits the printing job into chunks and ensures feasible printing of each chunk and good bonding between chunks; (2) the slicer slices the chunks into layers, generates commands for printing the chunks, schedules the sequence of printing the chunks among multiple robots and inserts communication commands to enable necessary communication among multiple robots; (3) the simulator visualizes the dynamic printing process using the commands generated by the slicer

- *Simulator*: The command generated by the slicer is interpreted by a simulator, which visualizes and animates the dynamic printing process over time to provide a tool for evaluating the chunking and slicing parameters and results.

3. Chunker

The objective of chunking is to divide the printing job into chunks such that they can be assigned to as many robots as possible to increase the printing speed. Therefore, the overall chunking strategy is highly dependent on the geometry of the print, the number of available robots and how the robots will be scheduled. To simplify the problem, we will only consider how to chunk for two robots in this paper and leave scaling to many robots for the future. The methodology for splitting a print job into chunks for two robots will generally be applicable for many robots through a "divide and conquer" strategy.

To chunk for two robots, we will split the object into multiple chunks along one direction (Y direction in Figure 1) with sloped planes to ensure good bonding between chunks. Two robots start from the center chunk and print along +Y and –Y direction, respectively, to finish each chunk. To calculate the geometries of these chunks, we simply bisect the original geometry multiple times around multiple planes. Because we have constrained the problem to chunking only in the +Y and –Y directions, each plane can be defined by two things: its slope and Y position.

3.1 Slope determination

A sloped interface between chunks is needed for this chunk-by-chunk-based 3D printing strategy. The angle of the sloped plane needs to be carefully determined because of the conflicting objectives:

- A maximum slope angle will maximize the volume of each chunk and increase printing efficiency.
- A minimum slope angle will maximize the area of the bonding interface and increase the bonding strength.

In addition, the range of the slope angle is limited by the robot parameters, as illustrated in Figure 3, which should be determined by:

$$\theta_{max} = tan^{-1}\left(\frac{nh}{nd}\right) \quad (1)$$

$$\theta_{min} = tan^{-1}\left(\frac{h}{bd}\right) \quad (2)$$

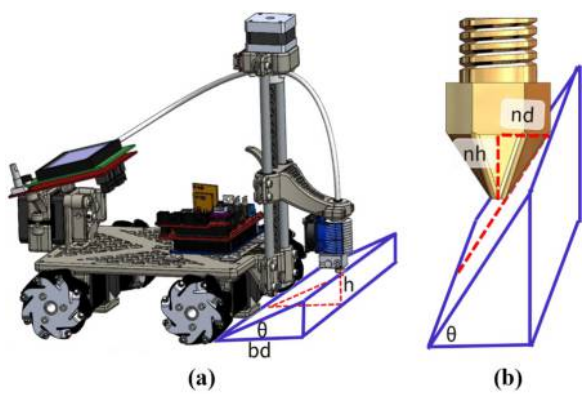
where θ_{max} and θ_{min} are the limits of the slope angle, nh and nd are the nozzle height and nozzle depth, h is the height of the object to be printed and bd is the build depth of the printer, as illustrated in Figure 3.

If the angle is very large or very small, either the front wheels of the robot or the nozzle will interfere with the printed material. It should be noted that the range of the angle is dependent on the printer design and the limits can be easily changed with a tilted nozzle or a printer with a changeable build depth. Tests should be performed to choose an appropriate slope angle. In this paper, we use the calculated θ_{max} for all our tests.

3.2 Chunking plane determination

With a determined slope, we will also need to know where we want to split the object. For the chunking strategy with two robots, we first need a center chunk, which can only be printed by one robot. After the center chunk is completed, the two robots will finish the chunks on the left and the right sides, respectively. The center chunk's chunking planes can both be represented with their normal vector, $\underline{n}$, and any point on the plane, $\underline{p_p}$. This is the most convenient representation because

Figure 3 Illustration of robot build limits



Notes: (a) The smallest slope angle of a chunk depends on the ratio of the object height, h, and the robot build depth, bd; (b) the largest slope angle of a chunk is limited by the ratio of the nozzle height, nh, and the nozzle depth, nd

the bisecting algorithm (specifically, the bisecting algorithm in Blender) we were using requires only these two values to bisect a 3D geometry around the plane. The left and right chunking planes for the center chunk can be determined by:

$$Plane\ L: \underline{n} = \left(\underline{c} \times \left|(0,0,h) + \frac{h}{\tan(\theta)} \perp \underline{c}\right|\right);$$
$$\underline{p_p} = \left(\underline{p_c} + \frac{h}{\tan(\theta)} \perp \underline{c}\right) \quad (3)$$

$$Plane\ R: \underline{n} = \left(\underline{c} \times \left|(0,0,h) - \frac{h}{\tan(\theta)} \perp \underline{c}\right|\right);$$
$$\underline{p_p} = \left(\underline{p_c} - \frac{h}{\tan(\theta)} \perp \underline{c}\right) \quad (4)$$

where $\underline{c}$ is the normal vector of the center line of the object, $\underline{p_c}$ is a point on the center line and θ is the angle of the chunking plane previously determined, and:

$$\perp(x,y,z) := (-y,x,z) \quad (5)$$

After calculating these two planes, we can iteratively shift those planes outward by a shift amount, s, from the center chunk (by iterating $\underline{p_{i+1}} \leftarrow \underline{p_i} + s\underline{c}$). We can use these planes to slice the model into subsequent "left" and "right" chunks. Figure 4 demonstrates the iterative chunking process, starting with the center chunk, and then shifted the chunking planes to the right and the left, respectively.

We have applied this chunking algorithm to two different geometries using different chunking settings to demonstrate its effectiveness, including a cylinder and a car model, as shown in Figure 5. The yellow chunk is the center chunk. As we can see, the chunker works effectively with complex geometries and different settings.

3.3 Speedup discussion

To determine the speedup gains of cooperative 3D printing, we can adapt the Amdahl's law used for parallel computing. The print job can be split up into two parts:

1 a part that can only be printed by one robot (i.e., the center chunk); and
2 a part that can be printed simultaneously by multiple robots (i.e. the rest of the chunks).

Assuming the average build speed is b cm^3/hour, the total volume of the print is V_{total} and the volume of the center chunk as V_{center}, the total printing time with one single printer would be:

$$T_1 = \frac{V_{total}}{b} = \frac{V_{center}}{b} + \frac{V_{total} - V_{center}}{b} \quad (6)$$

As a general discussion, assuming the printing job of the non-center chunks can be split among N printers, the total printing time would become:

$$T_N = \frac{V_{center}}{b} + \frac{V_{total} - V_{center}}{b * N} \quad (7)$$

So the printing time reduction "r" with N "printer" is:

$$r = \frac{T_N}{T_1} = \frac{V_{center}}{V_{total}} + \frac{1}{N}\left(1 - \frac{V_{center}}{V_{total}}\right) \quad (8)$$

And the speedup gain $s = 1/r$. It is clear that the V_{center}/V_{total} needs to be minimized to maximize the speedup gain. In our current two-robot printing scenario ($N = 2$), if we assume $V_{center}/V_{total} = 0.05$ for example, the speedup gain would be:

$$s = \frac{1}{0.05 + \frac{1}{2}(1 - 0.05)} = 1.905 \quad (9)$$

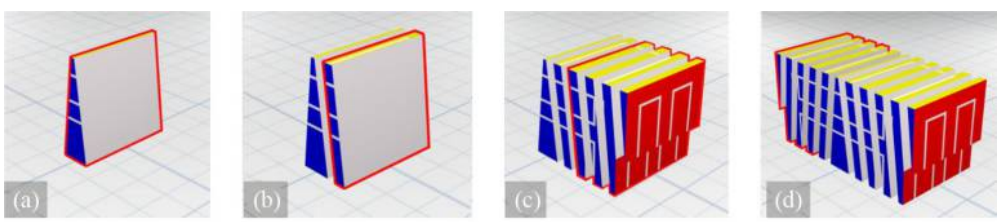
To maximize speedup, we have already set up the calculations of the center chunk to minimize the center chunk volume by using the maximum slope angle possible for the chunk faces θ_{max}.

4. Slicer

The objective of the slicer is to make sure that the robots can work together to finish the printing according to the printing strategy. Unlike a regular slicer that only generates the tool path, the slicer for cooperative printing needs to accomplish three functions:

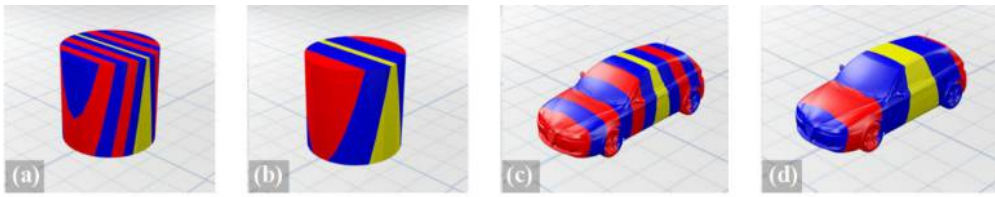
1 assign chunks to each robot and determine their printing sequence;
2 generate tool paths for each chunk and the tool paths for transition between chunks; and

Figure 4 Iterative chunking results



Notes: Planes L and R are reused and shifted to split further chunks on the left and right of the center chunk. (a) Center chunk; (b) shifted plane R to the right by one chunk; (c) shifted plane R to all the right chunks; and (d) shifted plane L to all the left chunks

Figure 5 Chunker results with two different objects, a cylinder and a car model



Notes: (a) Cylinder with short build depth and steep chunking slope; (b) cylinder with deep build depth and moderate slope; (c) car with short build depth and steep chunking slope; (d) car with deep build depth and moderate slope

3 generate commands based on the tool path for the robots to execute and provide a mechanism for the robots to communicate with each other in case one robot's printing task is dependent on the status of the printing task of another robot.

4.1 Printing sequence

To determine the path for a robot to follow, the robot must first know the chunks it will print and their sequence. As we only consider two robots in this paper, we can use the following simple strategy to assign the chunks to the robots, where C_A represents Robot A's chunks and C_B represents Robot B's chunks:

$$C_A = [center\ chunk,\ left\ chunk\ 1,\ left\ chunk\ 2, \ldots] \tag{10}$$

$$C_B = [right\ chunk1,\ right\ chunk\ 2, \ldots] \tag{11}$$

where Robot A is assigned with the center chunk and all the chunks on the left, and Robot B is assigned with all the chunks on the right. The chunks then need to be ordered based on the scheduling strategy for the print job. Because the chunks were generated in order by the chunker, there is no need to order the chunks for the simplified two-robot printing in this paper.

4.2 Tool path generation and transition between chunks

With the ordered chunks assigned to each robot, we need to generate a sequential tool path for each robot to finish its assigned chunks. This task can be accomplished in steps, as illustrated in Figure 6:

- generate tool path for each chunk;
- generate tool path between chunks; and
- combine the tool paths in sequence.

4.2.1 Tool path generation for chunks

The tool path generation for a chunk is similar to what a regular slicer does for a printing object. The general process is illustrated in Figure 7. Instead of starting from an standard triangle language file (or additive manufacturing file format or other file formats), the tool path generation algorithm starts with triangular meshes generated by our chunker. Based on the specified layer thickness, a list of horizontal planes is generated to split the model into multiple layers. The horizontal planes are then intersected with the triangular mesh to calculate the intersection line segments at each layer. Figure 8 shows an algorithm we used to calculate the line segments at each layer. The line segments are then ordered into a ring to form a perimeter for each layer. Infill paths are then generated for the parameters at each layer. Because this process has been well established in current slicers, we are omitting the details here, although the slicing process is by no means a trivial task. Building a general purpose slicer involves accounting for oddities in models, such as multiple isolated meshes, non-closed geometries, holes in closed geometries and more. The development of a proprietary slicer consumed much of the work on this research.

The results of the tool path generation can be seen in Figure 9, which shows that the slicing and infill algorithms are working correctly.

4.2.2 Transition between chunks

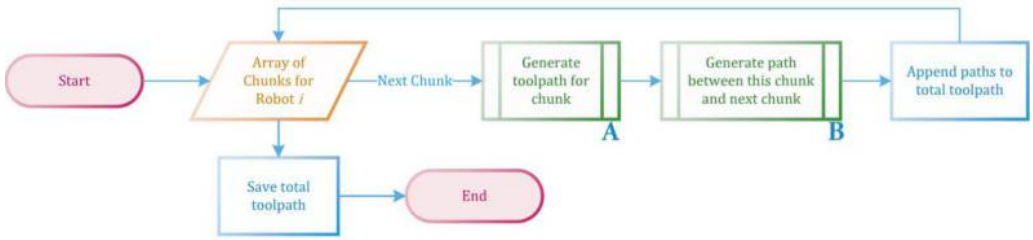
In addition to the print path for each chunk, the robot must have a way of moving from one chunk to the next. A simple direct line would not work, as the robot could knock against previously printed materials. Instead, we generate a separate path from the endpoint of the current chunk to the starting point of the next chunk. There are possible ways to optimize this path to save printing time, but we are using a simple approach that is not optimized but always works. Figure 10 visually demonstrates how we generate this transition path. Four points comprise the path: the endpoint of the current chunk p1, the start point of the next chunk p4 and two points in between. For p2, we simply shift the printhead upwards by a small amount. For p3, we move the extruder to the boundary of the chunk (i.e. to the same x and y positions as p4). The path generation process is detailed in Figure 11.

4.3 Command generation and communication

Till this point, we have generated the entire tool path for each robot, which are organized in a multi-level hierarchical data structure, as illustrated in Figure 12.

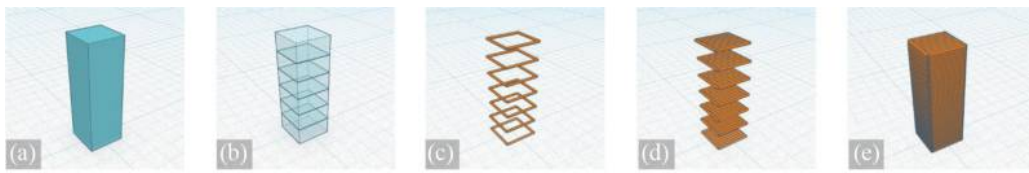
With the entire tool path generated, the slicer needs to generate commands that can be interpreted by the robots to execute the movements. One of the most common types of commands used in a regular 3D printer is G-code commands. In this paper, we use similar commands for our simulator to interpret. For example, we use a "MOVE X Y Z" command, which moves the robot from its current position to the specified position (X, Y and Z). It is equivalent to a G1 command in G-code and thus makes it easy to output the commands to a real

Figure 6 Path generation for each robot



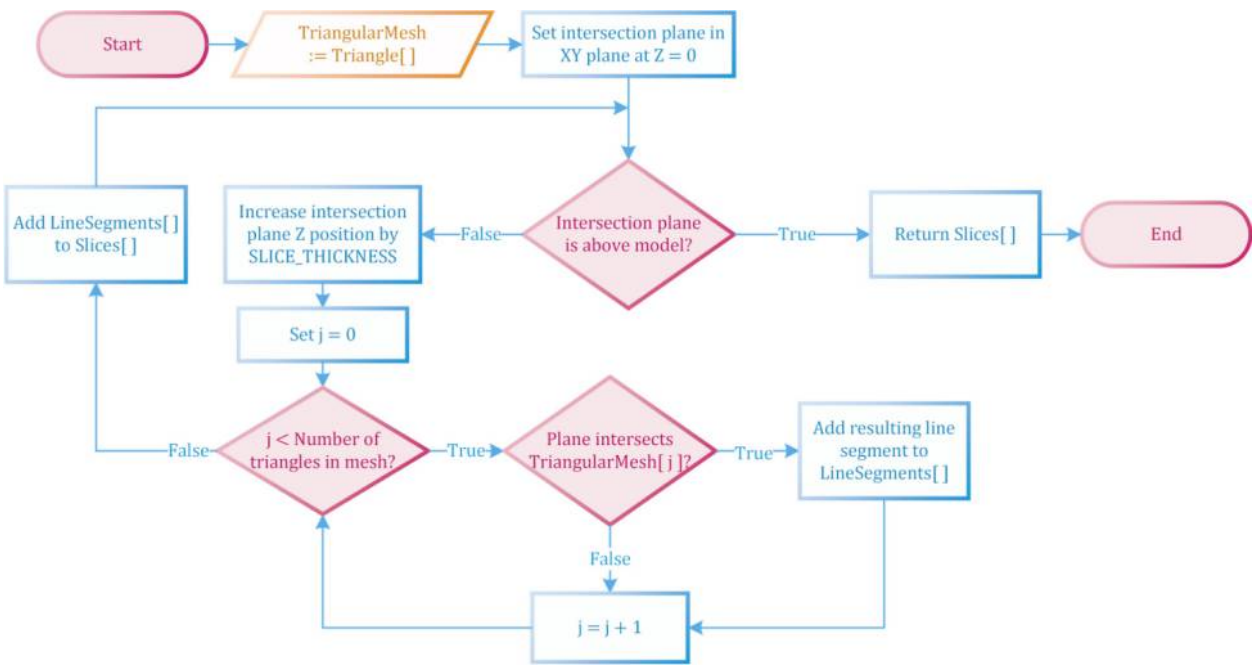
Note: Generate tool path for each chunk and generate path to transition between chunks

Figure 7 Illustration of the tool path generation process



Notes: (a) Original model; (b) model split into horizontal layers; (c) perimeters calculated for each layer by intersecting the triangular mesh with a plane at each layer; (d) generate infill path for the perimeter at each layer; (e) combine all the tool path generated at each layer

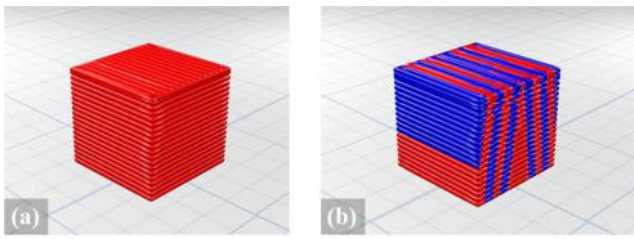
Figure 8 Process for calculating the line segments for the perimeter path of a layer



robot. As the tool path is just a list of line segments, this one command will be sufficient to instruct the robot to move along its tool path.

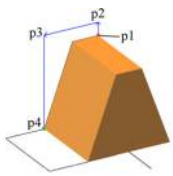
Now that the robots know where to move based on the generated commands, they also need to know when to move. In the situation when one robot's next move is dependent on another robot's printing status, it is necessary to provide a mechanism for the robots to communicate. For example, in our two-robot situation, the second robot must wait until the first robot finishes printing the center chunk to start printing and thus has to know when the first robot finishes printing the center chunk. One direct way is for the robots to have real-time constant communication with each other, but it would significantly increase the complexity when many robots are

Figure 9 Tool path generation from the slicer algorithm (thick filament was used for better visualization)



Notes: (a) Without chunking; (b) with chunking first

Figure 10 Transition path between chunks (p1 → p2 → p3 → p4)



Notes: The printhead moves slightly upward from previously printed materials and navigates to the next chunk. The endpoint of the current chunk is p1 and the start point of the next chunk is p4. p2 and p3 are points used to generate the transition path n

involved. Luckily, the interdependence of the printing tasks can usually be pre-determined in the chunking or slicing stage. Therefore, we can pre-implant a communication command at the stage when communication is needed. In this paper, we implemented a "NOTIFY" command, which is inserted behind a MOVE command to notify the other robot that a certain movement has been finished. Because our situation only involves the center-chunk waiting period, only one NOTIFY command is needed for the entire print. The second robot will not begin along its toolpath until it receives the NOTIFY command from the first robot.

In addition to the MOVE and NOTIFY commands, the robot also needs to know when the materials should be deposited. This is because the robot does not print materials along all the tool paths. For example, when the robot is transitioning for one chunk to another, no materials need to be printed. Therefore, we implemented a "TOOL ON/OFF" command to indicate whether the robot should print materials. When "TOOL ON" command is issued, materials will be printed along all the following tool path until a "TOOL OFF" command is issued. Based on the tool path data and how we want to robot to communicate and print materials, we can translate the tool paths into commands for the robots to execute the printing process using the three commands we have implemented. This process is shown in Figure 13.

5. Simulator

Simulating dynamic processes in 3D is usually challenging because of intensive computation. The objective of the simulator is to animate the command output from the slicer to visualize the dynamic printing process in a computationally effective way so that we can validate the chunking and slicing results and evaluate the overall printing strategy. Two essential functions are needed for the simulator:

1 interpret and visualize each single command from the slicer output; and

Figure 11 Process of generating transition path between two chunks

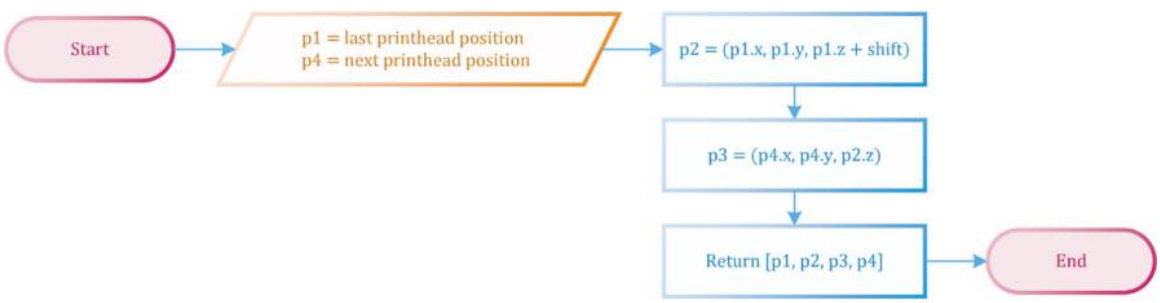


Figure 12 Data structure of the tool paths for the robots

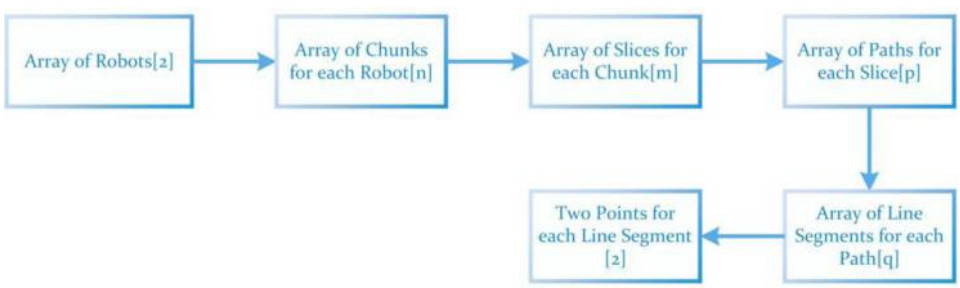
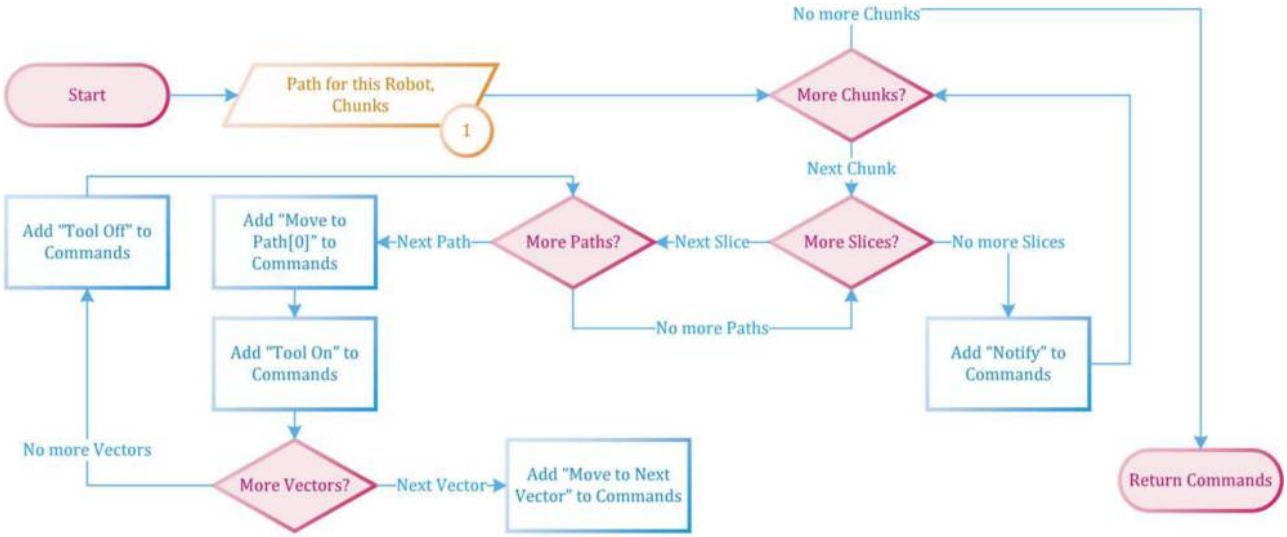


Figure 13 Conversion from the tool paths to robot commands



2 animate the sequential commands in correct timing sequence and speed.

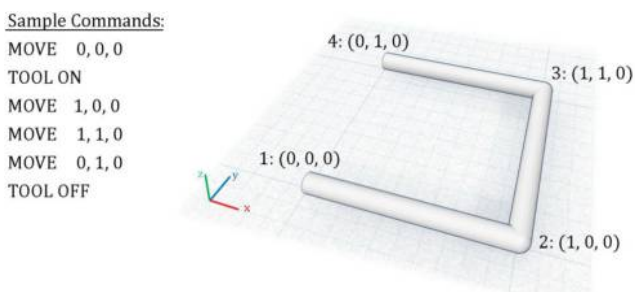
5.1 Command interpretation

The simulator needs to be able to read in the text commands (G-code) generated from the slicer, parse them, and visualize them one by one on the same scene. As we only have three commands (MOVE, TOOL ON/OFF and NOTIFY) with well-defined meanings, it is straightforward to interpret them. Figure 14 demonstrates the interpretation of a small set of commands as an example. A cylindrical tube is used to visualize the printed filament.

5.2 Commands animation

A simple approach to animate the commands is to visualize each command as one video frame. As we step through each command, we will generate frames for a video. However, this naïve approach does not work well. This is because commands take different times to execute in reality but they will all appear to take the same time in the animation, as each command is represented by one frame in the video. For example, if the robot is moving at the same speed, "MOVE 10, 0, 0" will take ten times longer than "MOVE 1, 0, 0" in reality, but they will appear to take the same time in the animation.

Figure 14 Demonstration of interpretation and visualization of a small set of commands



Therefore, we implement the concept of a "time step," which is a defined period of time to be simulated between each frame. Based on the robot's speed and the distance the robot must travel for each command, we can calculate how long the robot should take to execute a given "MOVE" command. To simulate real-time movements, our approach is to execute as many commands in the queue as will fit within the pre-defined time step. If the next command execution results in going over the time step, then we only execute *part* of that command. This requires a linear interpolation operation to calculate where the robot's extruder will be located along the movement path at the end of the time step.

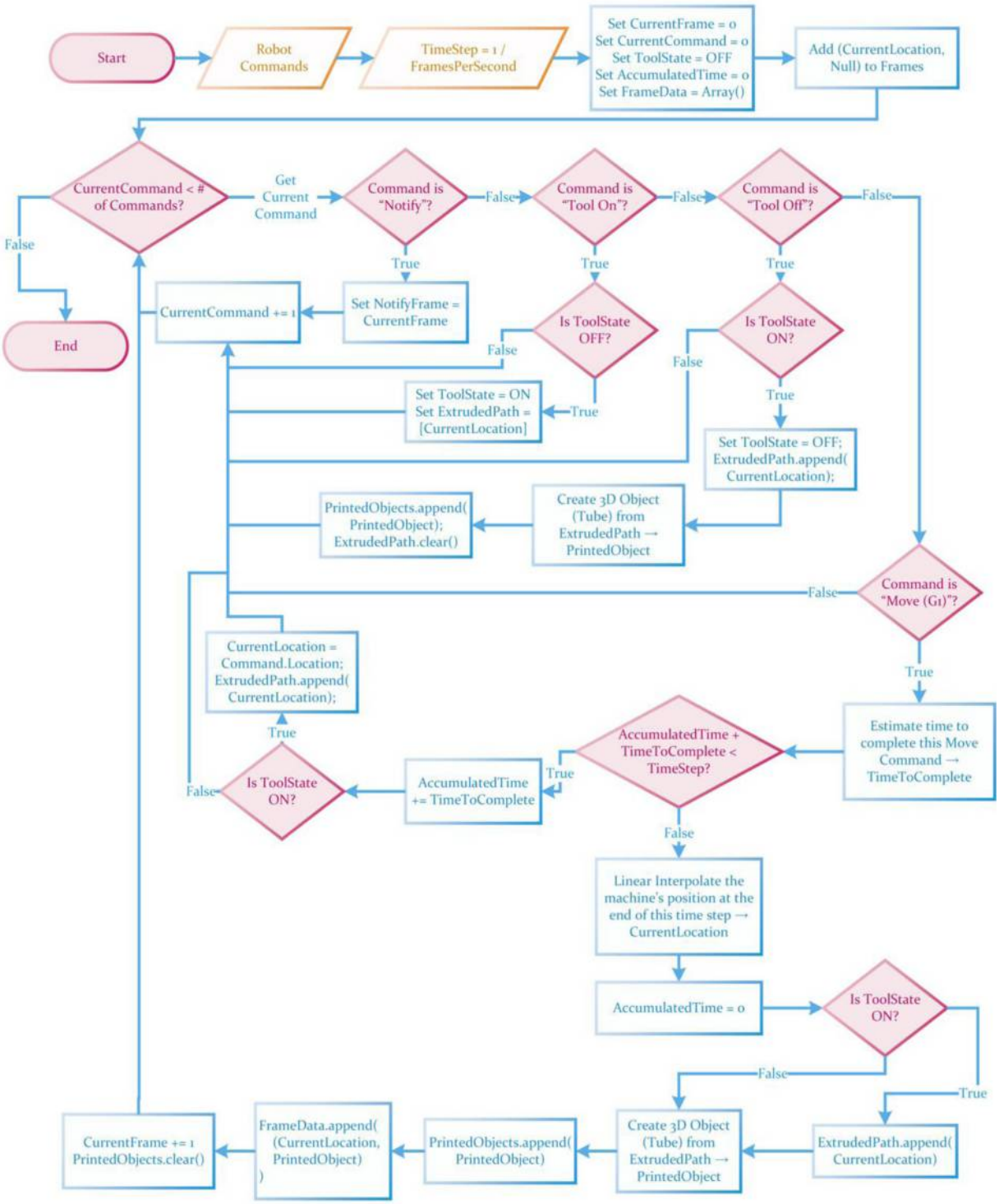
In addition to simulating the robot's movements, we have to simulate the extruded material from the printhead. We accomplish this by creating a Bezier curve (the robot's path) for every time step, then creating a cylindrical tube (for 3D rendering purposes) that follows the path. We only create these tubes if the robot's current state is "TOOL ON." Otherwise, no extruded material is created. Figure 15 includes pseudocode for the path animation function. The function is supplied with a list of commands and the robot's current location and produces a data set for each frame, which includes:

- the robot's new position; and
- a list of extruded material objects (Bezier paths) to display for that frame.

The snapshots of the simulated animation are shown in Figure 16. The results show that the chunk-based slicing works as intended without robot collisions and the simulator also correctly visualizes the slicing results, which provides a tool to visualize the moving path of the mobile robots to evaluate different printing strategies and printing time.

To demonstrate that the developed chunk-based slicer and simulator can handle complex geometry, we also simulated the printing of an airplane model, as shown in Figure 17. As can be seen in the second snapshot, the center chunk was sliced correctly into three distinct pieces in the first few layers because

Figure 15 Pseudo code for animating the commands output from the slicer

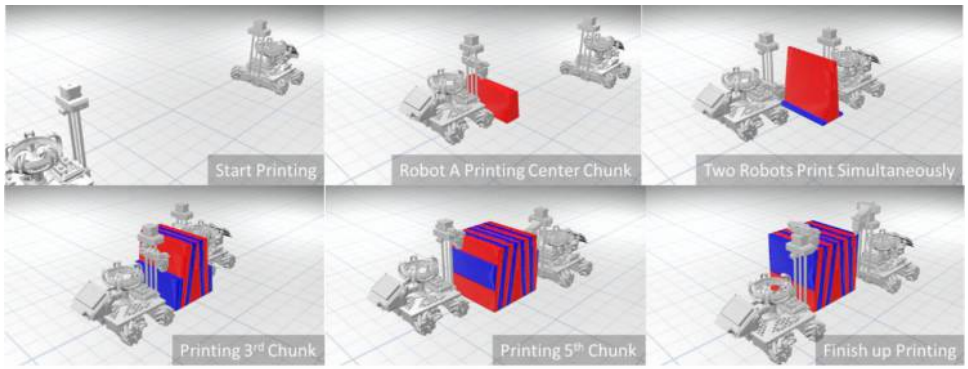


of the complex geometric characteristics of the airplane model. The chunker, slicer and the simulator all behaved correctly with the complex geometry.

6. Conclusion

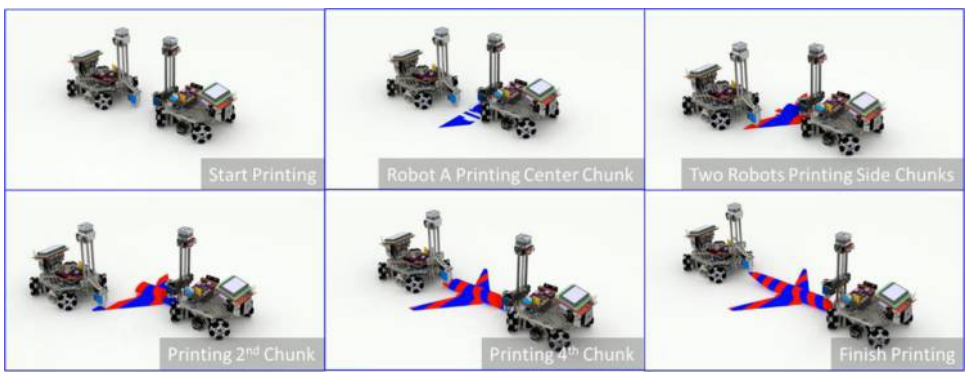
In this paper, we presented a new chunk-by-chunk-based approach to 3D printing for a cooperative 3D printing platform. In contrast to a layer-by-layer-based approach, this new approach enables a plurality of mobile 3D printers to carry out a printing job simultaneously, which can provide the scalability to 3D printing, in terms of both print size and printing time. The chunk-based strategy also keeps printing localized and thus alleviates large temperature gradient and internal stress that are common to large-scale 3D printing.

Figure 16 Snapshots of the roughly simulated animation of two robots printing a cube cooperatively



Notes: During the process, no collisions have occurred. After the process, the intended model has been successfully printed

Figure 17 Snapshots of a realistic simulated animation of two robots cooperatively printing an airplane model



We first presented a general strategy for the chunk-based printing strategy. A chunker is then developed for two-robot printing to demonstrate a general chunking methodology. A slicer is also developed to slice the chunks and coordinate multiple robots for the printing. A simulator is then developed to visualize and animate the dynamic cooperative 3D printing process to validate the developed chunker and slicer. The results show great promise to a new direction of 3D printing that may provide a path to make 3D printing a mainstream manufacturing technology with autonomous printing robots.

7. Future work

As the basic viability of chunk-based cooperative 3D printing has been demonstrated in this paper, more work is needed to establish the complete viability of our vision to realize an autonomous digital factory equipped with a swarm of mobile 3D printers and other specialized robots to manufacture complex parts. One of the next steps is to investigate the scalability of this chunk-based slicing method and develop scheduling strategies for a large number of robots. Although the critical elements of cooperative 3D printing with multiple robots have been addressed in this paper (i.e. chunking, transition between chunks and communication between robots), two mobile printers are not enough to produce at the speed required for mass manufacturing. In order for cooperative 3D printing to reach its maximum potential, many robots must be able to work in parallel to produce parts rapidly. Slicing algorithms for the incorporation of gripper robots that can pick and place pre-manufactured components to be embedded in a 3D printed structure during the 3D printing process can also greatly enhance the capability of 3D printing in fabricating electromechanical devices and other sophisticated products.

References

ASTM-F42-Committee (2012), "Standard terminology for additive manufacturing technologies", in *ASTM International*, West Conshohocken, PA.

Braam, D. (2016), "Cura (software)", *Wikipedia*.

Bräunl, T. (2000), *The EyeSim Mobile Robot Simulator*, CITR, The University of Auckland.

Browning, B. and Tryzelaar, E. (2003), "Ubersim: a realistic simulation engine for robot soccer", *Proceedings of Autonomous Agents and Multi-Agent Systems, AAMAS'03*.

Hao, J., Fang, L. and Williams, R.E. (2011), "An efficient curvature-based partitioning of large-scale STL models", *Rapid Prototyping Journal*, Vol. 17 No. 2, pp. 116-127.

Hugues, L. and Bredeche, N. (2006), "Simbad: an autonomous robot simulation package for education and research", *International Conference on Simulation of Adaptive Behavior, Springer*.

Kirschman, C. and Jara-Almonte, C. (1992), "A parallel slicing algorithm for solid freeform fabrication processes", *Solid Freeform Fabrication Proceedings, Austin, TX*, pp. 26-33.

Kisslicer (2016), available at: www.kisslicer.com/

Koenig, N. and Howard, A. (2004), "Design and use paradigms for gazebo, an open-source multi-robot simulator", *Proceedings, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004*, (IROS 2004), *IEEE*.

Lefebvre, S. and Grand-Est, L.I.N. (2013), "IceSL: a GPU accelerated CSG modeler and slicer", *18th European Forum on Additive Manufacturing (AEFA'13)*.

Linjie, L., Ilya, B., Szymon, R. and Wojciech, M. (2012), "Chopper: partitioning models into 3D-printable parts", *ACM Transactions on Graphics*, Vol. 31 No. 6, Article 129 (November 2012), p. 9.

Ranellucci, A. (2015), "Slic3r: G-code generator for 3D printers", Pàgina web oficial del projecte: available at: http://slic3r.org/about (últim accés 25 October 2013).

Repetier (2017), "Repetier-Host", RepRap Wiki.

Sabourin, E., Houser, S.A. and Helge Bøhn, J. (1996), "Adaptive slicing using stepwise uniform refinement", *Rapid Prototyping Journal*, Vol. 2 No. 4, pp. 20-26.

Skeinforge (2015), available at: http://fabmetheus.crsndoo.com/overview.php, RepRap Wiki.

Xu, W.P., Li, W. and Liu, L.G. (2016), "Skeleton-sectional structural analysis for 3D printing", *Journal of Computer Science and Technology*, Vol. 31 No. 3, pp. 439-449.

Corresponding author

Wenchao Zhou can be contacted at: zhouw@uark.edu