As-continuous-as-possible Extrusion-based Fabrication of Surface Models

ANONYMOUS AUTHOR(S)

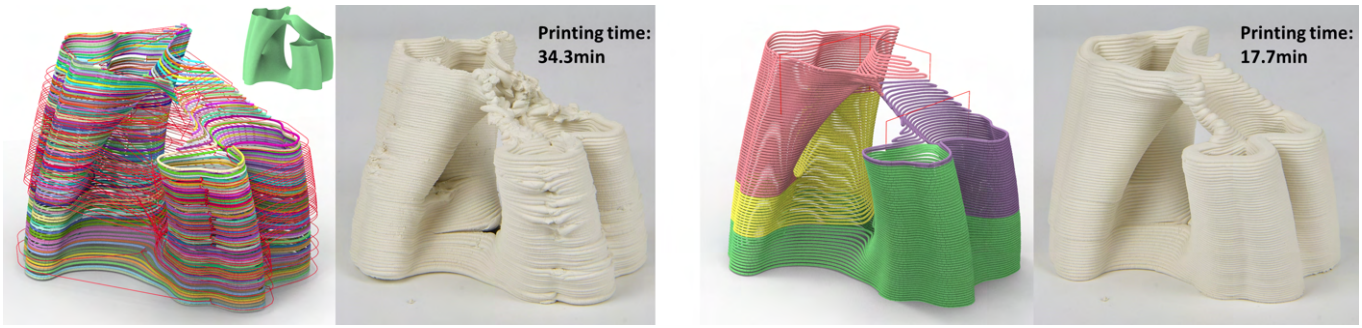


Fig. 1. Our framework significantly improves the efficiency of fabrication and the quality of surface models of the product with *as-continuous-as-possible* toolpaths for printing. We also propose a geometric criterion called the "one-path patch" (OPP) to decompose the input surface into a minimal number of continuous printing patches with flat and curved collision-free paths. Compared with 832 disconnected printing toolpaths with only flat slicing layers (left), generated by the Ultimaker Cura software (different colors indicate different connected paths and the red lines represent the transfer moves), our framework produces four continuous paths of deposition together with flat and curved slicing layers to ensure a much higher quality of printing (right) and an efficient fabrication process (34.3 mins vs. 17.7 mins).

In this study, we propose a computational framework for optimizing the continuity of the toolpath in fabricating surface models on an extrusion-based 3D printer. Toolpath continuity is a critical issue that influences both the quality and the efficiency of extrusion-based fabrication. Transfer moves lead to rough and bumpy surfaces, where this phenomenon worsens for materials with large viscosity, like clay. The effects of continuity on the surface models are even more severe in terms of the quality of the surface and the stability of the model. We introduce a criterion called the "one-path patch" (OPP) to represent a patch on the surface of the shell that can be traversed along one path by considering the constraints on fabrication. We study the properties of the OPPs and their merging operations to propose a bottom-up OPP merging procedure to decompose the given shell surface into a minimal number of OPPs, and to generate the "as-continuous-as-possible" (ACAP) toolpath. Furthermore, we augment the path planning algorithm with a curved-layer printing scheme that reduces staircase defects and improves the continuity of the toolpath by connecting multiple segments. We evaluated the ACAP algorithm on ceramic and thermoplastic materials, and the results showed that it improves the fabrication of surface models in terms of both efficiency and surface quality.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; *Graphics systems and interfaces*;

Additional Key Words and Phrases: Toolpath planning, shell models, extrusion-based printing

0730-0301/2022/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Anonymous Author(s). 2022. As-continuous-as-possible Extrusion-based Fabrication of Surface Models. *ACM Trans. Graph.* 1, 1 (October 2022), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Surface or shell models are widely used in structural design because they are efficient in representing the shape of the structure and have desirable functionalities, like a small weight and suitable thermal conductivity. Pottery in the shell form has been produced since the Stone Age. In the modern context of additive manufacturing (AM), shells are more cost effective than solid forms in terms of both the amount of material consumed and the fabrication time. In this paper, we focus on the surface model, particularly on a thin shell with the thickness of a single path. It can be either open or closed, because of which the single layers might have multiple connected components.

The continuity of the toolpath is a fundamental problem in material extrusion-based AM. The continuity of the movement of the nozzle and the extrusion of the material directly influence the quality of the surface, stability of the model, and efficiency of fabrication. The continuity of the toolpath plays a more critical role for the extrusion-based fabrication of surface models than that of solid models. Transfer moves (the movement of the nozzle with no extrusion) also induce extra forces on the printed shell surface to weaken the stability of the model such that it may sag or collapse with the accumulation of forces.

The fabrication of surface models with clay is becoming particularly popular owing to the rapid progress in ceramics printing techniques. The most feasible and cost-effective technique for the 3D printing of clay is direct ink writing (DIW), which shares its architecture with fused deposition modeling (FDM) but has a larger opening nozzle that provides more efficient material extrusion for

Fig. 2. Left: a strictly-continuous path generated by [Hergel et al. 2019]. We tailor this method to surface models by slightly thickening the original surface into a model with a valid volume. However, the redundant non-model structure still needs to be used to connect separate components of the model (see the top view). Right: Our toolpath is fully continuous due to the use of curved layers, without requiring an extra path.

highly viscous clay than thermoplastics [Chen et al. 2019b]. As a natural material, clay is environmentally friendly and durable. Because semi-liquid pastes have a high rate of deposition, surface models are the most popular 3D-printed objects. Nevertheless, the effects of artifacts on the quality of the surface due to transfer moves cannot be neglected.

Current path planning methods can be grouped into two categories. One category of methods focuses on optimizing the infilling patterns in each layered section [Zhai and Chen 2019; Zhao et al. 2016] to obtain a continuous toolpath even in layers with complicated contours. However, surface models with no interior cannot take advantage of this technique. The other group of methods of path planning is applicable to surface models, and involves optimizing sequences of contours [Lensgraf and Mettu 2017, 2018; Yoo et al. 2020] based on search algorithms to achieve the minimal "extrusionless travel distance." Nevertheless, this criterion of motion is not equivalent to the continuity of the toolpath in terms of the number of transfer moves. The optimal continuous toolpath of a surface model is essentially a tailored problem of surface decomposition that considers the constraints on fabrication.

Hergel et al. [2019] recently proposed a path planning method for the extrusion-based printing of ceramics that produces strictly continuous paths of deposition that eliminate transfer moves. This method works well with a single contour for each layer but cannot print multiple components per layer without adding non-model structures. Therefore, it cannot be easily adapted to shell models because it is nearly impossible to "hide" the non-model intermediate structures without affecting the appearance of the surface, especially open surfaces; see Figure 2.

Manufacturing with curved layers is regarded as effective for removing staircase defects [Etienne et al. 2019] and improving the strength of the material by aligning filaments along directions with high stresses [Fang et al. 2020]. Curved layers have the unique advantage for surface models whereby multiple components may be printed in a connected toolpath.

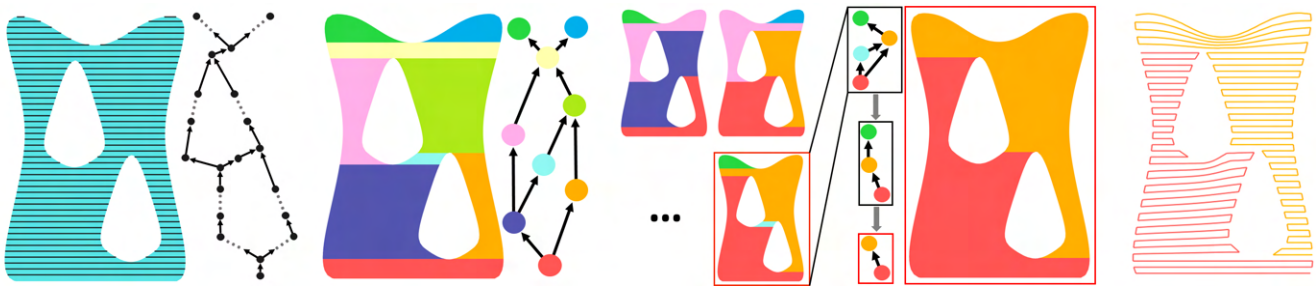
Using the curved layers-based scheme for Cartesian 3D printing encounters two constraints related to fabrication. First, the thickness of the layer is adjustable but is bounded by the extent of extrusion. Second, the slope of the curved toolpath cannot be too steep because the possibility of collision between the nozzle and the printed model must be considered. Considering the constraints of fabrication, the above problem can be regarded as a *precedence-constrained minimum path cover problem (PC-MPC)*. To the best of our knowledge, no polynomial-time algorithm is available to optimally solve this problem.

We target the development of an "as-continuous-as-possible" (ACAP) toolpath for surface models such that the number of transfer moves is minimized. The key idea is our proposal of a "one-path patch" (OPP) criterion to represent a surface patch that can be printed in a continuous toolpath on a combination of flat and curved layers. We propose a bottom-up OPP merging algorithm to decompose the given shell model into a minimal number of OPPs to generate the ACAP toolpath. This paper makes the following contributions to the literature:

- We introduce an original, constraints on fabrication-aware criterion called the "one-path patch" (OPP) for representing the surface patch of a shell that can be printed along one path in the context of printing schemes based on both flat and curved layers.
- We propose an algorithm for decomposing the given shell surface into a minimal number of OPPs and generating the "as-continuous-as-possible" (ACAP) collision-free toolpath.
- We adapt our technique as a general computation framework for printing shell models in an ACAP manner on three-axis extrusion-based printing platforms.

2 RELATED WORK

Slicing and Path Planning. In AM, the slicer is used for converting a model into toolpaths. Currently available slicing algorithms divide the model into a stack of flat layers by using geometric operations. [Lensgraf and Mettu 2016, 2017, 2018; Yoo et al. 2020] proposed a series of optimization algorithms to minimize the total *extrusionless travel distance* (wasted motion or print time) in the space of feasible toolpaths. We represent the precedence-related constraints as a dependency graph in a similar manner. In contrast to the above, however, we define the OPP criterion, reform the optimization into a more compact dependency graph, and obtain a more efficient optimization framework. We can thus achieve the optimal results rather than approximated ones for most cases. Many studies have sought to optimize the toolpath in terms of continuity, filling rates, and mechanical properties [Xia et al. 2020; Zhai and Chen 2019; Zhao et al. 2016]. Strategies for the adaptive width control of the toolpath can reduce under- and over-filling artifacts [Hornus et al. 2020; Kuipers et al. 2020]. Hergel et al. [2019] proposed a method for generating strictly continuous and self-supporting paths of deposition for extrusion-based ceramic printing. This method performs



(a) Dependency relationship (b) Initial decomposition (c) Merging with flat layers (d) Merging with curved layers (e) Toolpath generation

Fig. 3. Overview of the ACAP algorithm. Given a surface model and a feasible orientation, we horizontally slice along the Z-axis and represent the dependency relationship with a graph (a), where each node represents a *segment* or a *contour* and each edge represents the dependency between nodes. The input model is initially decomposed, and each decomposed patch can be printed continually. Their dependency relationships are shown in (b). Following this, we merge the small patches into larger, single, and printable patches, where this may yield multiple optimal merging solutions (c). We further reduce the number of patches for each solution by applying curved layers (d). For the optimal merging solution with the least number of patches, we generate the toolpath for each patch and assign the order of printing between them (e).

well on water-tight geometric models but cannot be directly applied to shell models, as illustrated in Figure 2.

Curved Layer Printing. Compared with the traditional flat layers, curved layers contain dynamic z-values within individual layers and have excellent properties for use in AM. They include alleviating staircase defects, improving surface smoothness, strengthening the printing model, and reducing the printing time. The first method proposed for printing curved layers was called curved-layer fused deposition modeling (CLFDM) [Chakraborty et al. 2008]. Following this, [Allen and Trask 2015; B.Huang and S.Singamneni 2012; Llewellyn-Jones et al. 2016] performed experiments on FDM printers to demonstrate these properties. The industry-standard slicing software Ultimaker Cura [Ultimaker 2021] also involves printing curved layers in the surface model, instead of a solid model, to produce spiralized outer contours of the mesh, where this works well on simple shapes like a vase or a cylinder. [Ezair et al. 2018] proposed an algorithm that generates covering curves based on the geometric characteristics of a given volume. [Etienne et al. 2019] used a different approach that optimizes the parameterization to obtain tops with smooth surfaces. The toolpaths thus produced are mapped back into the initial domain without requiring splitting or re-ordering. We apply this method in our method for merging curved OPPs. Researchers have recently applied curved layer-based printing to multi-axis printers. [Dai et al. 2018; Li et al. 2021; Xu et al. 2019] designed a curved toolpath by using additional DOFs to fabricate solid models in a support-free manner. [Chen et al. 2019a] proposed a CLFDM slicing algorithm that allows layers of variable thickness. [Fang et al. 2020] introduced a field-based optimization framework to generate curved layers to reinforce 3D-printed models. We focus on three-axis printer platforms in this paper.

Fabrication of Thin Shells. Fabricating thin shells is gaining increasing attention in the area as it shortens the fabrication time compared with closed models. Lightweight shell models have thus been widely applied. This advantage is further enhanced when fabricating viscous slurry materials, like clay and concrete, with a large volume of extrusion amount and a high rate of deposition. The continuity of material deposition is critical for shell models due to the artifacts caused by transfer moves. Most studies have used multi-axis platforms and incorporated curved layers for printing shells. [Mitropoulou et al. 2020] proposed a method to design non-planar, layered paths for the robotic FDM printing of single-shell surfaces. [Bhatt et al. 2020] proposed a layer slicing and toolpath planning algorithm to build thin parts of the shell on a three-DOF building platform and a three-DOF extrusion tool. The printing of concrete shells has attracted research interest in the interdisciplinary field of digital fabrication and architecture. [Burger et al. 2020] used single shells as molds for concrete casting, and [Anton et al. 2019] proposed a design tool for producing bespoke concrete columns that used a curved layer for continuous extrusion. [Bhooshan et al. 2020] also emphasized interactive shell modeling and integrated modeling for toolpath generation.

Decomposition for Fabrication. Many studies have focused on model decomposition for fabrication. The objectives of model decomposition include fabricating a model that satisfies the constraints, improving the surface quality, reducing the number of support structures or avoiding them altogether, and reducing the printing time. [Luo et al. 2012] proposed a solution to decompose the model into smaller parts such that every part can fit into the printing platform. Structural soundness and aesthetics are the objectives of decomposition in addition to the volume of printing. [Hildebrand et al. 2013] generated a partition and computed the optimal direction of slicing of the subparts to improve the surface quality. [Hu et al. 2014] decomposed a given shape into as small a number of approximate pyramidal parts as possible on the premise that this shape is well suited to fabrication. [Vanek et al. 2014; Wei et al. 2018] decomposed shell models into small parts to reduce the amount of support material needed and the printing time. Manual assembly was required after having printed all the shells in this case. To avoid the need for supporting materials, [Wu et al. 2017, 2020] considered the collision-free constraint and the sequence of printing in their approach to decomposition. They printed models in a multi-DOF 3D printing

system so that manual assembly was not required. [Herholz et al. 2015; Muntoni et al. 2018] decomposed general 3D geometries to satisfy the constraint on the height field. To minimize the number of setups of the cutter for finish-stage machining in CNC, [Zhao et al. 2018] developed an algorithm to perform surface decomposition with the accessibility constraint. The above methods do not consider the continuity of the printing path as a criterion for model decomposition. [Mahdavi-Amiri et al. 2020] proposed carvability criteria for continually carving a connected domain, where this requires both visibility and monotonicity. However, they did not consider curved slicing layers. In such a situation, the key difference between our OPP criterion and the carvability criteria is that the former does not require visibility but supports the fabrication of curved layers.

Ceramic Printing. Ceramic materials have attracted interest in AM in recent years [Chen et al. 2019b; Zocca et al. 2015]. They can reduce both the number of processes and the resources required to produce geometrically complex shapes in the traditional ceramics industry, and thus provide new ideas and applications for architectural decorations [Chan et al. 2020] and the arts. Researchers are also developing advanced engineering ceramics, such as metal oxides, carbides, and nitrides, to cater to specific engineering demands [Peng et al. 2018]. Current work has mainly focused on studying formulations of the water-to-clay ratio and some additives as well as the physical analysis of sintered models in terms of compression and thermal stability [Ordoñez et al. 2019; Revelo and Colorado 2018]. Even though slicing and toolpath planning for DIW ceramic printing share both in constraints and objectives with FDM, they involve additional constraints due to the viscosity of clay. Recent studies have considered path planning for closed models [Hergel et al. 2019], integrated modeling and path generation for simple shapes [Zhong et al. 2020], in addition to enhancing the stability of shell models [Xing et al. 2021]. No effective path planning method is currently available for general shell models.

3 OVERVIEW

Given a thin shell model M with a feasible orientation that satisfies the constraints on the support structure, our algorithm aims to achieve maximal path continuity, i.e., it decomposes M into the minimum number of surface patches, where each patch can be printed consecutively. For narrative convenience, we define a printable surface patch as a patch that can be printed by using a single path.

For the PC-MPC problem, the key idea here is to apply an over-segmentation followed by a bottom-up merging procedure. We first slice M by using uniformly distributed flat planers (Section 4.1). Each sliced element can be considered to be a single printable surface patch. Multiple printing paths from mutually contiguous sliced elements can be connected into a single path by using a set of short connecting paths. This means that we can reduce the number of printable patches by merging small, initial patches (Section 4.2). We also observe that curved slicing layers can be exceptionally effective in generating continuous printing paths for multiple, separate printable surface patches of flat layers. The number of printable patches can be further reduced by replacing as many flat layers with curved slicing layers as possible (Section 4.3).

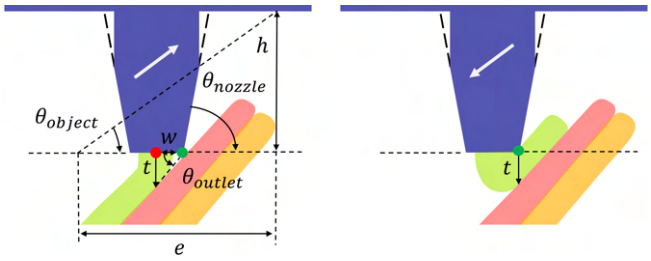


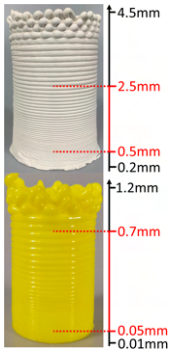
Fig. 4. Constraint on the slope angle of the curved path. The nozzle is a combination of a cylinder and a truncated cone (purple). We conservatively simplify it to a truncated cone to calculate the angles. θ_{nozzle} indicates the angle between the tip of the nozzle and the horizontal. $\theta_{object} = \tan^{-1}\frac{h}{e}$, where h is the vertical distance between the carriage and the tip of the nozzle, and e is the maximum XY extent of the already printed object. $\theta_{outlet} = \tan^{-1}\frac{t}{w}$, where w is the radius of the nozzle , t represents the thickness of a reference layer, and (left) and (right) represent the nozzle going uphill and downhill, respectively.

The merging criterion is the main challenge to the processes of bottom-up merging and the replacement of the curved layers. We need to formulate "subsurface patches" that can be merged into a single printable patch or replaced by curved layers. We introduce the "one-path patch" (OPP) as the merging criterion (Section 3.2). In the final step, we plan the path of each OPP and then subject it to post-optimization processing to improve its smoothness and spacing (Section 4.4). Note that we avoid potential global collisions between the printer and the printed layers by considering the size of the nozzle (Section 5). The pipeline of our algorithm is shown in Figure 3.

3.1 Constraints on Fabrication

We consider three constraints on fabrication in this work. The first bounds the feasible range of thickness of the layers. The second is intended to avoid collisions between the extrusion device (nozzle, extruder, and carriage) and the printed parts while operating the tool along its path on curved layers. The third constraint describes the geometric requirements for the strategy of decoupling-based fabrication of the intact model and the support structures.

Constraint on thickness. Very thin layers tend to squeeze together because they are affected by the fluidity of the material. Such over-stacking results in artifacts on the surface of the material (see inset). Very thick layers, on the contrary, result in under-stacking such that adjacent layers are not well bonded. We use t_{min} and t_{max} to represent the minimum and maximum thicknesses of the layers, respectively. We use $t_{min} = 0.5mm$ and $t_{max} = 2.5mm$ for ceramic printing, and $t_{min} = 0.05mm$ and $t_{max} = 0.7mm$ for FDM printing. These ranges are used as constraints when generating the curved layers.



Constraint on the slope angle . In [Etienne et al. 2019], the constraint on collision was modeled as an inverted cone to forbid already

printed parts from entering it. A local constraint on the slope angle of the printing paths was extracted from the forbidden cone as $\theta_{max} = \min(\theta_{nozzle}, \theta_{object})$ (see Figure 4 for the detailed formulation). Instead of regarding the printer as a pointed conical nozzle by overlooking the flat outlet of the latter, we propose making this formulation more precise by representing the nozzle as a combination of a cylinder and a truncated cone.

We make an interesting observation that the possible collisions are different when the nozzle moves uphill and downhill; see Figure 4. When it moves uphill (left), it may collide with the printed part because the point on the right side of the outlet (green dot) is closest to the layer printed below. This collision can be avoided by restricting the slope angle of the path to θ_{outlet}. When the nozzle moves downhill (right), it needs to be raised by t to avoid collision between layers at the point on the right side of the outlet (green dot) and the current layer. There is no need to define any slope angle for the case of downhill movement because the nozzle is raised. Finally, we set the upper bound of the slope angle of the path to $\theta_{max} = \min(\theta_{nozzle}, \theta_{object}, \theta_{outlet})$. Note that this is a local constraint. The global collision caused because the height of the printed model exceeds the length of the nozzle is not considered. The solution to this is provided in Section 5.

Support structure constraint. We impose restrictions on the support structures for the surface models because they degrade the quality of the surface [Hergel et al. 2019]. For surface models that are not self-supporting, we decouple the fabrication of the intact model from that of the support structures, i.e., we pre-print the support structures and install them during fabrication. This requires that the support structures be located on the ground, and form a volume of the height field related to the orientation of printing. This is because the support structures placed on the model may induce too large a weight for the printed shell to bear. We decompose such models into multiple patches and assemble them after fabrication (an example is shown in Figure 21).

3.2 One-path patch (OPP)

Recall that we aim to maximize continuity by decomposing the input model of the shell into the minimal number of printable "one-path patches" (OPPs). We first define a *segment* as: a component of a layer with two endpoints, and a *contour* as: a closed component without endpoints. With respect to the direction of printing, a manifold surface patch is a printable OPP iff 1) there exists a set of slicing layers, where each layer orthogonal to the direction of printing intersects the patch to form a single *segment*, or (*contour*) 2) the resulting intersecting *segments*/*contours* satisfy the constraints on fabrication. We deliberately choose the height of a low-resolution layer to render the printing paths more visible. Three types of OPP can be defined according to the slicing layers: (I) only flat layers, (II) only curved layers, and a (III) combination of I and II. They are called I-OPP, II-OPP, and III-OPP, respectively, and are shown in Figure 5. Curved layers of the OPPs consider the constraints on thickness and the slope angle.

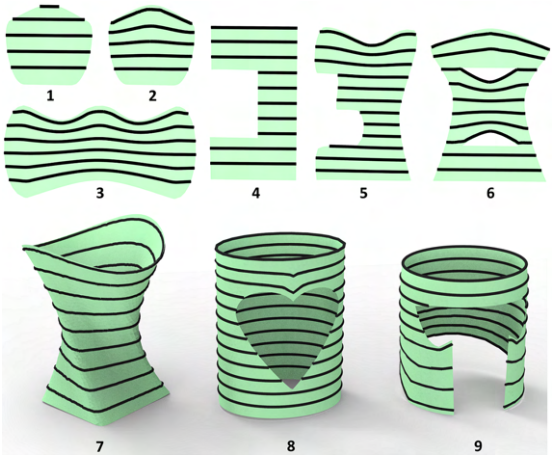


Fig. 5. Illustration of OPPs that can be printed along a single toolpath generated from flat layers (1,4), curved layers (2,3,5,7), and combined flat and curved layers (6,8,9). An OPP can be sliced by using different strategies (1,2).

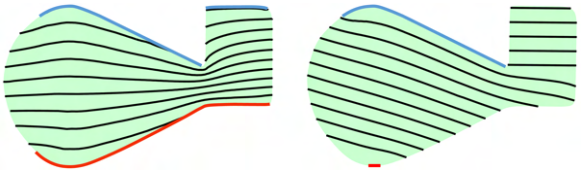


Fig. 6. An illustration that staircase minimization is not equivalent to continuity maximization. The left part visualizes the results of CurviSlicer with target flat areas (initial attempt to flatten all areas under a specific slope angle; red segments). It could flatten only a part of the target flat areas to avoid violating the constraint on the thickness of the layers. The resulting layers could not produce OPP layers. The right part shows the OPP layers and the specific target flat areas of our method. Note that in contrast to CurviSlicer, the downward-facing areas can be taken as target flat areas due to the constraint on the support structures (Section 3.1).

What geometric properties should an OPP have? For type I, the OPP criterion is equivalent to its *monotonicity*[1], as exemplified by (1,4) in Figure 5. For type II, the intrinsic geometric properties of an OPP are challenging to determine, and *monotonicity* becomes a sufficient but unnecessary condition. In Figure 5, (3,5,7) is an OPP but not *monotonic*. This demonstrates that a single OPP with curved layers can cover regions in which multiple I-OPPs are applied.

We can directly extract the curved layers and assess the two criteria of fabrication to determine whether P is a II-OPP. CurviSlicer seems a perfect match for this because it seeks to flatten as many areas as possible to minimize staircases [Etienne et al. 2019]. However, staircase minimization is not always equivalent to continuity maximization, and is sensitive to the target flat areas taken as the input to CurviSlicer as shown in Figure 6. Two I-/II-OPPs can be merged to a single II-OPP (details in Section 3.3). A bottom-up OPP

[1]A 2D polygon P is *monotonic* with respect to a straight line L if every line orthogonal to L intersects P at most twice. A 3D manifold surface patch is *monotonic* in direction L if all cross-sections orthogonal to L are single section [Toussaint 1985].

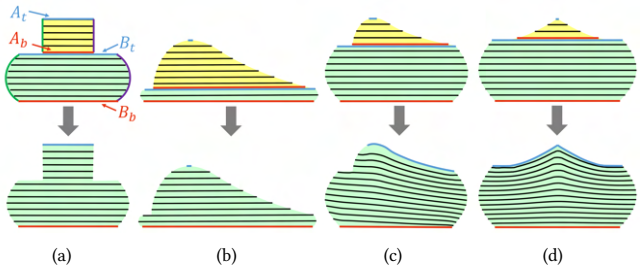


Fig. 7. Illustration of merging two OPPs by using the *stacking* operation (a)(b) and the *curving* operation (c)(d). (a) and (b) cannot be merged via curved layers because they would violate the constraints on the slope angle and the thickness, respectively. The red and blue segments indicate the top and bottom target flat areas (A_t, A_b, B_t, B_b). The green and purple lines indicate two oblique polylines.

merging procedure is introduced for a shell model to minimize the number of OPPs. During this process, III-OPPs (as shown by (6,8,9) in Figure 5) are generated (Section 4.3).

To the best of our knowledge, the OPP criterion has not been explored before. An OPP possesses three key properties as the elementary element of path generation to maximize the continuity of extrusion-based printing:

(1) The OPP criterion is defined with respect to a specific direction of printing. An OPP along a given direction of printing (left) may not be an OPP along another direction (right), as shown in the inset.
(2) Even with the same direction of printing, the valid slicing strategies of an OPP may not be unique. In Figure 5, (1) and (2) show two kinds of slicing layers to which flat and curved slicing layers can be applied. Curved layers always produce fewer staircases associated with higher priority than flat layers.
(3) Two OPPs can be merged to a single OPP. Two OPPs (A and B) can be merged via two operations, *stacking* and *curving*, as shown in Figure 7. *Stacking* indicates that 1) the bottom layer A_b of OPP A is located above the top layer B_t of its neighboring OPP B, and 2) A_b and B_t can be connected by its two endpoints. *Curving* indicates the merging operation of Section 3.3. Two operations were used in Section 4.3 to minimize the number of OPPs.

3.3 Curving Operation

The input to this operation consists of two I/II-OPPs (A and B) that can be originally merged via *stacking*. The *curving* operation outputs the merged OPP (C) along with its slicing layers. Suppose that OPP A is placed above B, as shown in Figure 7. Note that *curving* cannot be applied to two OPPs with only closed *contours*.

The basic idea here is to generate a modified version of CurviSlicer to extract curved layers for the merged OPP C. There are two key questions: 1) how do we determine the top/bottom target flat areas of C? 2) how do we guarantee that the constraints on fabrication are satisfied? We answer the first question by combining the top/bottom target flat areas of A and B. We detect the constraint on the slope angle on the top/bottom target flat areas of C that have been determined, and then extract the curved layers in between them by using our modified CurviSlicer such that the constraints on fabrication are satisfied.

We define the projection of the top (bottom) layer of an OPP as its top (bottom) target flat areas (A_t, A_b, B_t, B_b). We begin traversing from the top layer of A; two oblique polylines can be obtained by connecting the two endpoints on both sides of each layer with the bottom layer. If both oblique lines violate the constraint on the slope angle, the two OPPs cannot be merged via curved layers. If not, we generate the top target flat areas of C by combining 1) the top target flat areas (A_t) of OPP A, 2) the difference between the top target flat areas of B and the bottom target flat areas of A ($B_t - A_b$), 3) ensure that the oblique lines satisfy the constraint on the slope angle, and choose the bottom target flat areas of C from those of B.

Following this, we call CurviSlicer to specify both the top and the bottom target flat areas. CurviSlicer formulates two key terms in its objective function: a flat term to determine whether the target area can be flattened based on the constraints on the slope angle and thickness, and a smooth term to smoothen the generated curved layer. Using CurviSlicer with the smooth term is time consuming, and this is unnecessary for our case because the *curving operation* is called frequently and the generated layers are not used to generate the final toolpath. We thus use only the flat term while applying CurviSlicer. The merge is executable if the top and bottom target flat areas are successfully flattened without violating the constraints on fabrication. Note that CurviSlicer works only for watertight 3D models. We convert the surface model into an approximate watertight model with a shell of minimal thickness that can be used as the input to CurviSlicer (details in Appendix B).

4 ACAP METHOD

This section describes our algorithm in detail. For clarity of exposition, we explain the methodology for open 2D patches with only *segments* in each layer. The extension to 3D is discussed in Section 6. As introduced in Section 3, the basic idea of our algorithm is a bottom-up OPP merging process based on a unified graph-based representation of the OPP and a set of operations to merge graphical nodes. The OPP graph encodes the surface decomposition and its dependencies during the process of bottom-up OPP merging. The node merging operations of the OPP are formulated through the flat and curved slicing layers.

4.1 Building the Dependency Graph

With an orientation that meets the constraint on the support structure (see Appendix A for details), we uniformly slice the model with flat planers vertical to the direction of printing by layer thickness (1 mm for ceramic printing and 0.2 mm for FDM printing). We then build a directed acyclic graph, called the *dependency OPP graph* G_{depend}, to describe the dependency relationships, where each node represents a sliced element (*segment*) and each directed edge represents a dependency relationship between neighboring nodes, and the closest horizontal distance between them is shorter than the path width (6 mm for ceramic printing and 1.5 mm for FDM printing). This is shown in Figure 8(a). If node N_1 has a directed

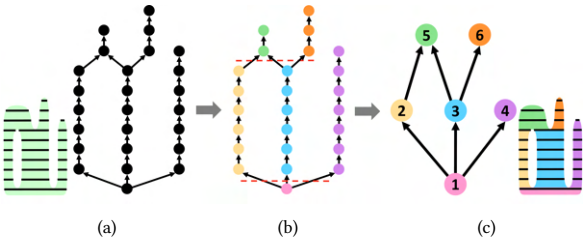


Fig. 8. (a) *Dependency graph* of (G_{depend}). Each node represents a sliced element and each edge represents a dependency relationship between nodes. (b) We temporarily ignore edges pointing to (from) nodes with an in-degree (out-degree) not smaller than two (crossed by the dashed red lines), and compute the connected components. (c) *Initial OPP graph* of (G_{init}). Each connected component of (b) acts as a node, and the edges are inherited from G_{depend}.

edge pointing to N_2, this indicates that (1) N_2 can be printed only after N_1 and (2) the two OPP nodes can be merged through the *stacking* operation. A node can be printed only if all nodes on which it depends have been printed.

4.2 OPP Merging through Flat Layers

This section aims at the maximal continuity provided by flat slicing, that is, merging the flat, sliced elements of G_{depend} into a minimal number of I-OPPs. Each sliced element can be seen as a I-OPP, and can be merged by the *stacking* operation (Section 3.2). The merged I-OPPs should maintain the dependency relationships formulated in G_{depend}. Such a merging process involves finding a path cover for G_{depend} with the fewest paths by considering the dependency relationships.

We propose two key steps for the merging process: 1) merge the nodes of G_{depend} that must appear on the same path in any minimum path cover in advance, and then build an *initial OPP graph* G_{init} to reduce the size of G_{depend}; and 2) merge the nodes of G_{init} further by solving the path cover problem with dependency constraints. Rather than running an approximation algorithm, we propose a search-based method with a pruning strategy to explore the possible solutions.

Initial OPP Graph. To build a simplified graph G_{init} from G_{depend}, we traverse all nodes of G_{depend}. If two or more edges point to the same node or start from the same node, we delete these edges temporarily (crossed by the dashed red lines in Figure 8(b)). We then compute the connected components. Each component acts as a node of G_{init}, which can be considered to be a larger I-OPP merged by *stacking*. Such a merging process maintains optimality, i.e., the subnodes of a G_{init} node must belong to a single path of the optimal solution[2]. The dependencies of G_{init} are inherited from G_{depend}. As shown in Figure 8(c), the number of nodes is significantly reduced.

Path Cover of Initial OPP Graph. The OPP nodes of G_{init} can be further merged via *stacking*, as shown by (1,2), (1,3,6), and (3,5) in

[2]This can be proved by contradiction. If two adjacent subnodes belong to two paths of an optimal solution, they must be terminal nodes of the paths. The two paths can be further connected, which shows that this solution to the path cover problem is not optimal.

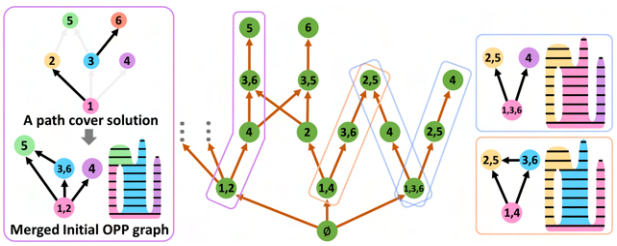


Fig. 9. (a) The nodes of G_{init} are merged based on a path cover solution. (b) The space of the path cover solution ($G_{solution}$), where the nodes (green dots) encode paths (merged I-OPPs) of the path cover solutions, The directed edges encode the order of printing of the merged I-OPPs, and each path cover solution corresponds to a sequence of directed edges and nodes. Different sequences of $G_{solution}$ may produce the same path cover solution but in a different order of printing (the two blue boxes). (c) Three sequences (two blue boxes and one orange box) produce two optimal path cover solutions associated with the minimal number of nodes of G_{flat}.

Figure 8(c). A merged *flat OPP graph* G_{flat} of G_{init} can be generated as a result of a path cover solution of G_{init}, where each path determines a merged I-OPP. Each node of the merged OPP comprises a sequence of I-/II-OPPs (denoted by *subOPPs*), where the edges of the sequence indicate the order of printing. Figure 9(a) shows a merged G_{flat} with four merged I-OPPs based on the path cover solution {(1,2), (4), (3,6), (5)}. Note that the dependency relationships of G_{init} are preserved. Such a path cover solution can be generated from a specific depth-first search (DFS) starting from the root nodes of G_{init}. In contrast to a general DFS, it may not search as deep as possible in the unexplored node before backtracking. The search has to stop while the dependent nodes of a given node have not been explored. For instance, (1,2,5) is not valid in that (3) has not been explored.

The space of the path cover solution of G_{init} can be represented by a specific directed graph ($G_{solution}$), where the root node is set to an empty node (an additional virtual node), the non-root nodes indicate the corresponding merged I-OPPs of the path cover solutions, and the directed edges encode the order of printing of the merged I-OPPs. A path cover solution is presented as a finite sequence of directed edges and nodes of $G_{solution}$, starting from its root node (see the sequence of $\{(1,2), (4), (3,6), (5)\}$ in Figure 9(b)).

Exploring the Space of Path Cover Solutions . To merge G_{init} into the minimal number of I-OPPs, we search for the shortest sequences while exploring $G_{solution}$. We propose two techniques to speed-up such exploration by pruning the solution space. First, starting from the root node of $G_{solution}$, we apply a beam search procedure with the branch-and-bound technique to explore $G_{solution}$ level by level, where the width of beam search was set to $W = 10^4$ in our implementation. For each level of $G_{solution}$, we sort its candidate nodes by the number of nodes of G_{init} included. This implies that we tend to pick as many nodes of $G_{solution}$, including those of G_{init}, as possible. Note that the nodes of $G_{solution}$ point to the same node in the next level if their sequences include the same nodes of G_{init}. For instance, {(1,2), (4)} and {(1,4), (2)} both point to (3,6) and (3,5).

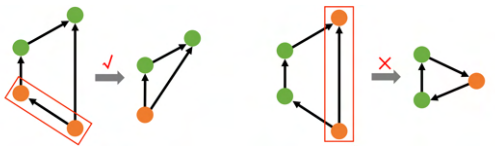


Fig. 10. Illustration of the printing dependency deadlock. The orange nodes represent a pair of selected nodes in the DAG. Left: Only one path exists between the two nodes and no deadlock occurs after merging. Right: Two or more paths between the two nodes may produce deadlock loops after merging, i.e., the nodes are inter-dependent. Thus, merging is prohibited.

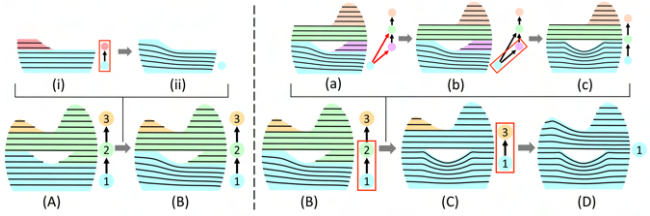


Fig. 11. A mesh is decomposed into three I-OPPs corresponding to G_{curved} with three nodes (A). We first merge subOPPs within each OPP in the Initial Merging Process by using the *curving* operation. This is indicated in (A, B), where (i, ii) representing the merging of subOPPs within OPP 1. Then, we iteratively merge pairs of nodes to one III-OPP node (B~D) in the OPP Merging Process. a~c show the merging process in the inner loops of (B) to (C), and only the three nodes connected by the red edges in (a) are selected for merging. Because the three subOPPs of (c) accept *stacking* into one OPP, nodes 1 and 2 can be successfully merged.

Second, we use a greedy strategy for generating path cover solutions with the DFS to generate candidate nodes in each iteration of the beam search. Each traversal seeks to go as deep as possible. For instance, the traversal (1,3) does not terminate at node (3) because node (6) can be added to (1,3,6), as shown in Figure 8(c). We have proved the optimality of this strategy[3]. The proposed method of exploring path cover solutions produces multiple instances of the optimal G_{flat} with the least number of OPP nodes, as shown in Figure 9(c). The pseudo-code is presented in the Appendix D.

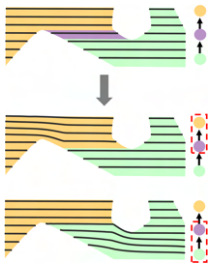
4.3 Merging OPPs through Curved Layers

We have thus far obtained a set of unique instances of G_{flat} with maximal continuity via flat slicing. Can we further merge its OPP nodes? Recall that curved slicing layers can be exceptionally effective in generating continuous printing paths for multiple, separate printable surface patches of flat layers. Driven by this insight, we apply the *curving* operation (defined in Section 3.3) *as much as is possible* to further reduce the number of OPPs. This can be done in two steps. First, we apply the *curving* operation to the subOPPs of each OPP node (**Initial Merging Process**), where this enlarges the target flat areas and is beneficial to the subsequent process of merging OPPs. Second, we apply the *curving* operation to merge multiple OPPs (**OPP Merging Process**). We propose below a general pairwise merging procedure for both the initial and the OPP merging processes. We set G_{flat} as an initial *curved OPP graph* G_{curved}, and iteratively merge G_{curved} to implement the two merging processes. II-OPPs and III-OPPs are generated accordingly.

General Pairwise Merging Procedure. OPP graphs are directed acyclic graphs (DAGs). We propose an iterative pair-wise merging procedure for a general DAG. For each iteration, we randomly select an edge and try to merge two related nodes based on specific merging criteria. If they can be merged, we update the graph by 1) erasing the edges of the two nodes, 2) merging the pair of nodes to one node, and 3) connecting other, related edges of the nodes with the merged node. The terminal condition is that no pair of nodes can be merged.

Merging Criteria for OPP Graphs. The directed edges of the OPP graph indicate the dependency relationships among OPPs. While merging a pair of nodes, one necessary criterion is that there are no multiple paths between the nodes because this results in a deadlock in dependency after merging, as shown in Figure 10(right). The second criterion is that the *curving* operation can be applied to the two OPP (subOPP) nodes. The two merging processes are demonstrated in Figure 11 (A~B) and (B~D). For each iteration of the **OPP Merging Process**, we aim to merge two OPP nodes with the *curving* operation, which is an inner loop of the **OPP Merging Process** (see a~c in Figure 11).

Inner Loop of OPP Merging Process. A DAG can be formulated for the inner loop based on the two candidate OPP nodes for merging: 1) We take their sequences of subOPPs, where each subOPP is a node, and maintain edges representing their dependencies. 2) We add the associated dependency edges between the sequences from G_{init}. While applying the pairwise merging strategy to the resulting DAG (Figure 11(a~c)), we select only edges that benefit the merging of the two sequences. Specifically, we first label the nodes that have edges across the two subOPP sequences (the two red edges shown in (a)) and then select the edges over the labeled nodes. For each pair of nodes, we call *curving* operation to merge them (Section 3.3). The pseudo-code is presented in the Appendix D.



Optimality of Proposed Method. For different G_{curved}, the final number of nodes after merging may be different. We randomly select the G_{curved} with the least number of nodes because we consider only the criterion of the number of OPPs. Different instances of G_{curved} and orders of merging of the OPPs based on *curving* may result in curved layers with different distributions. In other words, different subOPPs of the final III-OPP may be obtained as shown in the inset. Similarly, because the order of selecting nodes at the two levels is random, we cannot guarantee a global optimal solution. Figure 23 shows an example, and more details are discussed in Appendix C.

[3]In path traversal, if we terminate the path to which a node can be added, the node must be the starting point of another path. In the same way as in the last proof, the number of paths of the path cover solution, in this case, is at least one more than in our strategy, and thus it is not optimal.

4.4 Connection between Layers

We use the bottom-up process of merging OPPs to obtain an optimal OPP decomposition. We now describe path planning for each OPP by converting their slicing layers into a continuous toolpath and generating transfer moves between OPPs. Note that we removed the most time-consuming smooth term of CurviSilcer while applying the *curving* operation during OPP merging (Section 3.3). Here, we add it back and rebuild the smoother, curved layers of related OPPs for planning the toolpath. Following this, we connect the layer to form a single path for each OPP and determine the order of fabrication of these paths.

Inter-layer Connection Path. Because the 2D models contain only *segments*, they can be connected using a zig-zag pattern. Of the two terminal points of a *segment*, if one is the entry point, the other is the exit point. We select entry points for all *segments* to minimize the total length of the path between the entry and exit points of adjacent *segments*. We then set a Euclidean distance threshold D to determine whether the two terminal points of two neighboring layers can be connected directly with a straight segment. If the distance between *segments* exceeds D, an extra path is added (see the inset). We then call the terminal point in the current layer P_{source} and the terminal point to be connected in the next layer P_{target}, print along the current printed layer with a layer of thickness t_{min} from P_{source} to the position closest to P_{target}, and then print along a straight line to P_{target} (the orange path in the inset). The subsequent path optimization in Section 6 improves the spatial distribution of the generated extra path.

Sequence of OPPs . With knowledge of the dependency relationships of these paths of OPPs, we apply the method in Section 4.2 to produce a feasible order of fabrication. Following this, we plan transfer moves between OPP paths by withdrawing the nozzle to a safe distance above the printed objects to avoid collisions (see Figure 1). Finally, a G-code file is generated to transfer the toolpath to the printer.

5 GLOBAL COLLISION-RELATED CONSIDERATIONS

The algorithm in Section 4 does not consider the global collision caused by the printed parts exceeding the length of the nozzle, as shown in Figure 12(a). This often occurs when printing a lower layer after the parts in a higher layer. To extend our method to this consideration, the key challenge is to represent the constraint on global collisions in our proposed bottom-up OPP merging algorithm. We formulate such constraints as a new type of directed edges of the OPP graph, called "collision dependency edges." Similar to the original directed edges, the new edges represent printing-related dependencies. The difference is that the OPP nodes of the new edges cannot be merged through *stacking* operations. We first clarify the generation of these collision dependency edges and then introduce the modifications to the algorithm due to them.

Generation of Collision Dependency Edges. To add the novel edges into G_{depend}, we apply the method to model the printing nozzle in Section 3.1. For each node pair of G_{depend}, we add a collision

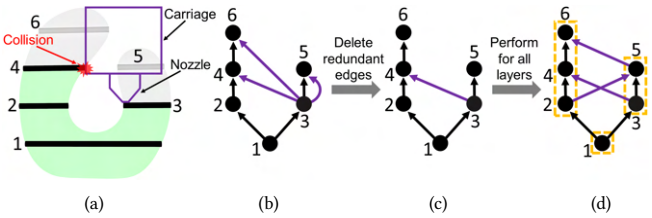


Fig. 12. Illustration of collision detection and building the *dependency graph*. We traverse layer "3" to determine whether it collides with the other layers (a). (b) Layer "3" collides with three layers; thus, three candidate collision dependency edges (purple) are supposed inserted into G_{depend}. (c) We delete two redundant candidate collision dependency edges. (d) We repeated the method for all layers, and finally added three collision dependency edges to G_{depend}. The orange boxes show the results of merging for G_{init} without the addition of the collision dependency edges.

dependency edge between them if a collision occurs during printing. As in case of layers "3" and "4" in Figure 12(a), the collision indicates that layer "4" must be printed after layer "3." Note that we maintain G_{depend} as a Hasse diagram [Pemmaraju and Skiena 2003] in which redundant dependency edges do not exist. For example, if two edges A→B and B→C exist, then A→C is a redundant edge. The example in Figure 12(b) shows that the three candidate collision dependency edges, starting from node "3," have been compressed into a single edge (c).

Modifications to the Algorithm . 1) Given Section 3.3, we need to add a requirement for applying the *curving* operation to guarantee that there is no collision dependency between the top and bottom target flat areas of the OPPs. 2) For Section 4.2, the method of exploring path cover solutions remains the same as above. However, if a node in the solution space of the path cover ($G_{solution}$) includes two subOPP nodes that are connected with "collision dependency edges," we need to split this node and make sure that the two subOPP nodes are not merged into the same node. 3) For Section 4.3, when we add the collision dependency edges, we observe that there are many more OPP nodes (six nodes) in G_{init} than in the original G_{init} (three nodes, indicated by the orange boxes), as shown in Figure 12(d). To solve the problem of efficiency that arises due to the increase in the number of OPP nodes, especially in the OPP merging procedure with curved layers, we add a step before the *Initial Merging Process*. This involves applying the *stacking* operation to the subOPPs of each OPP node according to G_{init}, which is generated without considering global collisions.

6 EXTENSION TO 3D

Extending our algorithm from 2D to 3D does not require extra effort in most steps, except when dealing with *contours* in the construction of G_{init} and extending the generation of the toolpath to the 3D case. We describe these extensions in this section.

Initial OPP Graph. For 3D cases with *contours*, we first build a G_{init} using the method in Section 4.2. If an OPP node of G_{init} has both *segments* and *contours*, we divide it into pure *segment* nodes and pure *contour* nodes. Such classification is conducive to the next step of merging via *curving*, which allows only the input of two OPPs

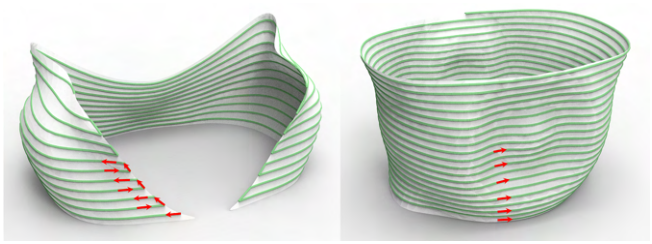


Fig. 13. Zig-zag path connecting the *segments* (left) and spiral path connecting the *contours* (right).

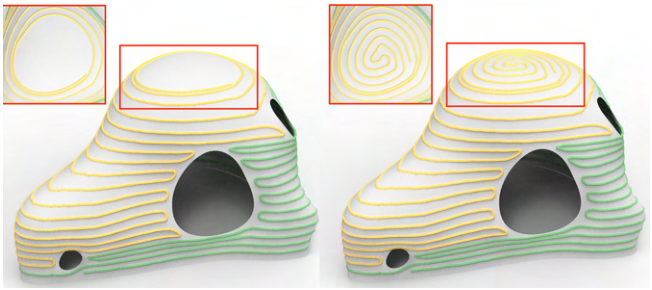


Fig. 14. Under-fills appear at the top of the model because of inadequate slicing in the low-slope region (left). The top of model can be printed completely by filling the area with connected Fermat spirals (right).

composed of *segments*, or one OPP composed entirely of *segments* and the other entirely of *contours*.

Contour Spiralization. To spiralize the *contours* (C_1, ..., C_n), we first need to determine a connecting point for each *contour*. As in Section 4.4, we aim to minimize the total distance between the connecting points of adjacent *contours*. We design a dynamic programming algorithm to choose the appropriate connecting points for each *contour* and discretize each *contour* with m sampling points, where P_{ij} is a sampling point in C_i. We denote by d_{ij} the minimum length between neighboring *contours* C_1, ..., C_i when choosing P_{ij} as the connecting point. The equation of transition is as follows:

$$d_{ij} = \begin{cases} 0, \ i = 1 \\ \min\limits_{k=1\cdots m} \left(d_{(i-1)k} + \left\|P_{(i-1)k}, P_{ij}\right\|_2\right), \ i \neq 1 \end{cases}$$

Then, we connect these *contours* by a spiral path as shown in Figure 13. For each pair of adjacent *contours*, we interpolate all sampling points starting from the connecting points:

$$P_i = w \cdot P_a + (1 - w) \cdot P_b, \quad w = S_{current}/S_{total}$$

where P_b is a sampling point of the lower *contour* (if $S_{current} = 0$, P_b is connecting point of the current *contour*), P_a is the nearest point to P_b in the higher *contour*, S_{total} is total length of the lower *contour*, $S_{current}$ is the geodesic distance moved by P_b from the connecting point along the lower *contour*, and P_i is the interpolation point. Note that the top *contour* as a boundary is not spiralized.

Filling the Low-slope Area. Inadequate layers over low-slope regions inevitably result in under-fills during slicing, as shown in Figure 14(left). Such under-fills over small areas can be removed by

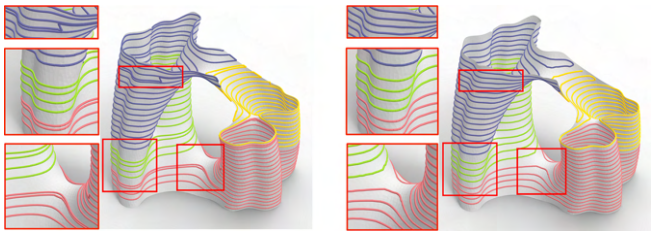


Fig. 15. Effectiveness of optimization of the global path. Left: Without global path optimization, some paths are too close or too far from one another. Right: Global path optimization leads to a uniform path. The optimization of the toolpath cannot guarantee a perfectly uniform solution due to the area around the saddle point.

a post-path optimization process (see Figure 15 left bottom). We propose filling large under-filled areas by using connected Fermat spirals [Zhao et al. 2016]. In our implementation, we set a threshold of 20° to detect the region of the Fermat spiral . We traversed each pair of adjacent *contours*, generated the matching edges between the sampling points of *contours* by using the minimal Euclidean distance, and measured their angles with respect to the horizontal plane. If all angles were smaller than the threshold, we added the surface patch between the *contours* to the region of the Fermat spiral. Figure 14(right) shows the path of filling over the original 3D surface.

Toolpath Optimization. Because we separately generate a toolpath for each OPP, adjacent paths may be too close or too far from one another (center of Figure 15(left)). Some inter-layer paths of connection (Section 4.4) and the filling paths of low-slope areas (bottom of Figure 15(left)) may be not uniformly distributed (top of Figure 15(left)). To solve these problems, we use a method similar to that in [Zhao et al. 2018] to optimize the final toolpath. The authors iteratively evolved a single toolpath by considering the constraints on spacing and smoothing. In contrast to the consideration in this method, our input is not a single path but multiple continuous paths. Figure 15 shows the results before and after optimization.

7 RESULTS AND DISCUSSION

This section details the results of OPP decomposition of the surface models of varying geometric complexity, the results of printing, and comparisons with the results given by Ultimaker Cura 4.9.1 software. We evaluated our ACAP algorithm on DIW-based ceramic printing and FDM platforms, with clay and thermoplastics as the materials, respectively.

7.1 Implementation and Parameters

Our algorithm was implemented in C++, running on a PC with an Intel Core i7-9700 CPU @ 3.0 GHz and 32 GB of memory. For the printing experiments, we used a three-axis DIW ceramic printer *Eazao Mega 5* with a printing volume of 470×370×390mm^3 (Figure 16) and an FDM printer *Hori Z560* with a printing volume of 360×350×500mm^3. Note that the parameters in the parentheses below refer to those of the FDM. We used a 90 (8)-*mm*-long nozzle, with a diameter of 5.2 (1.0) *mm* for the ceramic (FDM) printer (Figure 16

shows the nozzle). The speed of the nozzle was set to 25 (25) mm/s. The width of the printing path was set to 6 (1.5) mm. In Section 3.1, we used the average range of thickness of the layers, i.e., t =1.5 (0.35) mm, to calculate the constraint on the slope angle, which was 30° (35°) according to the formulation. We set the thickness of the flat layer to 1.0 (0.2) mm for slicing as this yielded the best surface quality in our experiments. We also used this as the thickness of the slicing layer of the slicing models to generate G_{depend}. The distance threshold D was 5 (2) mm in the connecting layers (Section 4.4).

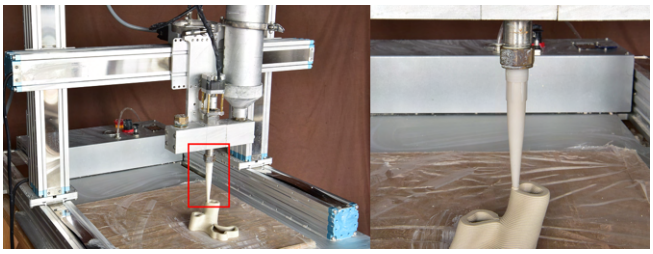


Fig. 16. The three-axis printer *Eazao Mega 5* used in the experiments (left) with a slender nozzle (right).

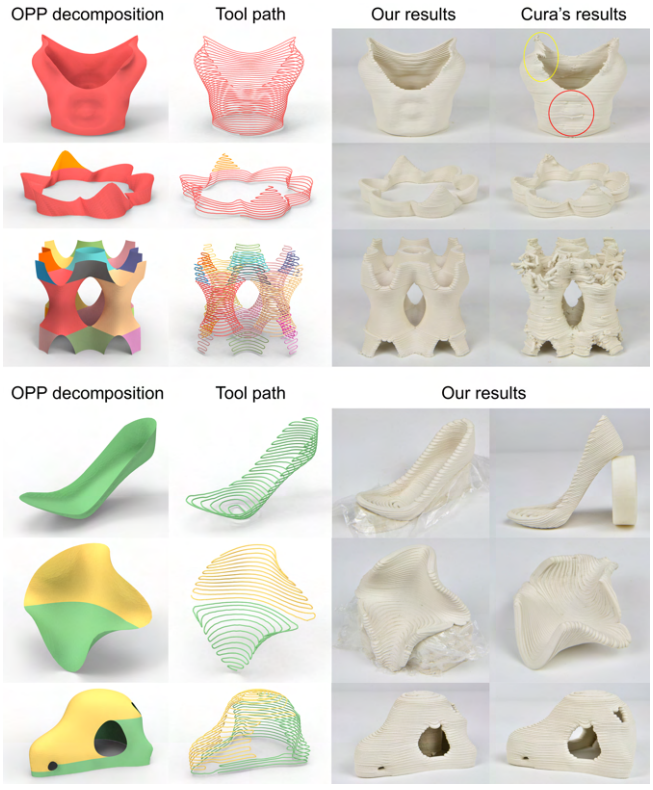


Fig. 17. Some results of ceramic 3D printing. The models in the top three rows are self-supporting. The OPP decomposition, toolpath, and results of printing obtained by using our method and Cura are shown in each row. The last three rows show models with supports. We built the support structures in advance and inserted them manually during fabrication.

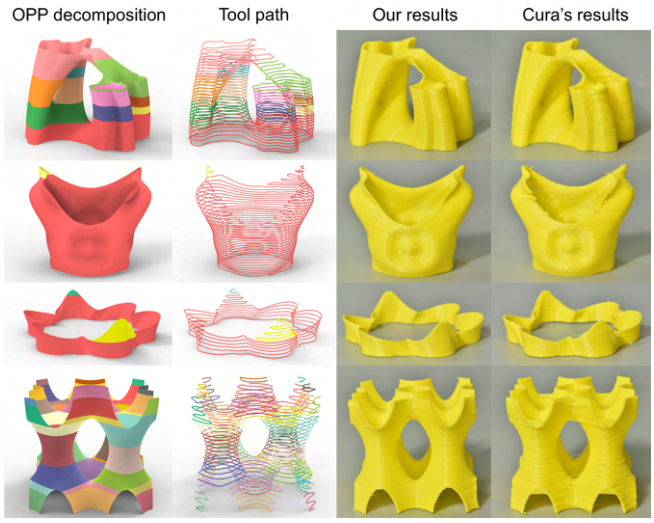


Fig. 18. Some results of FDM printing. Left-to-right: the OPP decomposition, toolpath, and results of printing by using our method and Cura. Compared with those of ceramic printing, the results of FDM had more OPPs owing to the shorter nozzle to avoid collisions. The surface quality was better than that of ceramic printing using both methods.

Fig. 19. The thickness was non-uniform in the outer boundary of the curved layers because the material deformed in different ways when extruded in the downhill or uphill directions.

7.2 Results of Fabrication

Figure 1, Figure 17, and Figure 18 show the results of the two printing processes (ceramic printing and FDM), including the results of OPP decomposition, path of printing, and the printed models (using our method and Cura, respectively). We refer to Table 1, which lists the height of each model, the number of OPPs produced by our algorithm, and the printing times of our method and Cura. Specifically, the table lists the numbers of OPPs after merging based on flat layers (#OF) and curved layers (#OO) to illustrate the effectiveness of the *stacking* and *curving* operations. Note that we scaled the models to half their size in the FDM experiments to save computation time. Because the nozzle used for FDM printing was much shorter than that for ceramic printing, our method partitioned more OPPs to avoid global collisions.

Comparison with Cura. The top three models in Figure 17 and the four models in Figure 18 were all self-supporting. We chose the *Surface Mode* in the settings for Cura, enabled the *Spiralize Outer Contour* and *Retraction* functions, and used the same speed of the nozzle as in our method. Our method outperformed Cura on shell models containing multiple *contours* or *segments* in terms

Model	H	#OF	#OO	#OC	T_{ours}	T_{cura}	T_{save}
Julia vase	81	8	4	832	17.7	34.3	48%
Grail	120	2	1	1541	20.1	26.7	25%
Crown	38	6	2	116	8.3	12.5	34%
TPMS	95	18	18	3332	37.3	58.5	36%
Shoe	52	2	1	N/A	20.7	N/A	N/A
Ocean	79	4	2	N/A	59.5	N/A	N/A
Mask	54	6	2	N/A	8.6	N/A	N/A
Julia vase	41	15	11	741	19.8	35.0	43%
Grail	60	3	2	1487	24.0	31.4	24%
Crown	19	7	3	311	8.8	12.7	31%
TPMS	48	44	44	2971	34.5	51.4	33%

Table 1. Statistics of the results. The gray icons represent the results of ceramics printing (Figure 1 and Figure 17), and the yellow icons represent the results of FDM printing (Figure 18). H is the height of the model (mm). #OF is the number of OPPs of our method with only flat layers. #OO is the number of OPPs using both flat and curved layers. #OC is the number of OPPs for Cura. T_{ours} and T_{cura} indicate the printing times (minutes) of our method and Cura, respectively. T_{save} is the percentage of time saved by our method compared with Cura. For the three models *Shoe, Ocean, Mask*, we did not use Cura as they needed support structures.

	Section 4.1			Section 4.2				Section 4.3			4.4	5	All
Model	#DN	#DE	T_d	#IN	#IE	#BS	T_f	#SO	#CU	T_c	T_l	T_o	T_t
	81	209	ϵ	17	19	8	ϵ	128	768	315	246	326	887
	120	146	ϵ	4	3	2	ϵ	2	4	3	497	438	938
	38	91	ϵ	9	8	6	ϵ	16	96	39	313	144	496
	95	598	ϵ	38	48	18	3	96	0	3	1	455	462
	52	64	ϵ	3	2	2	ϵ	2	2	2	84	53	139
	79	97	ϵ	7	9	4	ϵ	2	8	7	419	94	520
	57	122	ϵ	12	15	6	ϵ	121	741	274	290	242	806
	205	1254	ϵ	443	1182	15	3	48	288	127	274	495	899
	300	415	ϵ	107	157	3	ϵ	2	2	1	440	524	965
	190	1043	ϵ	384	971	7	ϵ	28	196	131	298	257	686
	240	3836	1	905	3295	44	37	1	0	ϵ	1	573	612

Table 2. Statistics of algorithmic performance. A term beginning with "#" indicates the amount of this term, "T" indicates run time (s), and "ϵ" indicates that the run time was negligibly short (<0.1 s). The table shows the number of nodes (#DN), edges in G_{depend} (#DE, including the dependency edges and collision dependency edges), nodes in G_{init} (#IN), edges in G_{init} (#IE), the depth of beam search (#BS), the number of optimal solutions of beam search (#SO), and the calling of CurviSlicer without a smooth term (#CU). In the context of the run time, the table shows the times needed to build G_{depend} (T_d) and G_{flat} (T_f), merge G_{curved} (T_c), execute CurviSlicer with the smooth term and the layer connection (T_l), and to optimize the toolpath (T_o) as well as the total run time (T_t).

of both surface quality (fewer artifacts and slighter deformation) and efficiency of fabrication (saved 24%~48% in printing time). As shown in the *Grail* model printed with Cura (ceramic printing), a large number of transfer moves (1,541) induced deformation (see the red circle in Figure 17). These transfer moves increased the forces on the printed part, and can lead to collapse (see the yellow circle). By contrast, the ACAP toolpath for this model had one only transfer move in FDM and none in ceramic printing. This strengthened the model during fabrication and avoided the above problems. In general, our algorithm improved the efficiency of printing more significantly for models with multiple branches, like *Julia vase* or the *TPMS* model. Moreover, the benefit of using curved layers is evident as they reduced the number of OPPs by around 50% and alleviated staircase defects (see the upper part of the *Crown* model). The *TPMS* model had no curved layer due to the constraint on the slope angle.

For models that required support structures, we built the supports in advance and added them before printing the models. See the last three models in Figure 17. To make the support structures easy to remove, we added a membrane to their contact surfaces before printing the model.

Discussions on the printing quality. In general, the quality of ceramic printing was more sensitive to toolpath continuity for the surface models than that of FDM printing. Therefore, the results of the ACAP algorithm for clay were significantly superior to those of Cura in terms of the quality of the model. However, we still observed artifacts in the clay printouts, such as over-extrusion in the zig-zag turning areas, seams at the junction of the OPPs, and sagging at the concave corner. They occurred because the DIW-based ceramic printing technique with clay is less mature than FDM with thermoplastics. The volume of extrusion could not be precisely controlled due to material inertia. Moreover, there were many uncertainties during the fabrication process, such as the humidity of the material, air pressure, and problems arising from the coupling of the hardware and the material.

We also observed a defect in the curved layers (see Figure 19): The thickness of the layer was non-uniform along the outer boundary, and appeared to have the staircase defect. This is because curved slicing layers with adaptive thickness were used, and the material thus exhibited different deformation behaviors when extruded in the downhill or uphill directions. This artifact can be removed by fine-tuning the flow of the material or the height of the nozzle, but requires that the clay material have stable properties, the simulation of deformation of the clay be accurate, and control over the rate of extrusion of clay in the printing platform be precise.

7.3 Algorithmic Performance

Effect of nozzle length. The size of the nozzle, especially its length, also influenced the number of OPPs as a shorter nozzle make collisions between the model and the printing platform more likely (see Figure 20. We tested three nozzle lengths ranging from 5 mm to 30 mm on the *Julia vase* model.

Run time. Table 2 shows the run time statistics along with the numbers of nodes and edges of the OPP graph during each step of our ACAP algorithm. For most input models, our algorithm was highly efficient in terms of the generation of G_{depend}, G_{init}, and G_{flat} because our algorithm did not involve any geometric computation in this phase, but only some graph operations. The only exception was the *TPMS* model (FDM). It took a much longer time to build

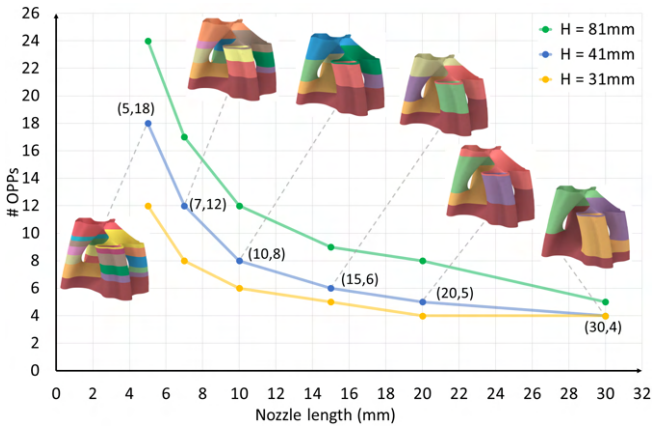


Fig. 20. The number of decomposed OPPs with models of different sizes (H indicates model height) and nozzles of different lengths for the *Julia vase* model in the FDM printing setup. As the length of the nozzle increased, the number of OPPs decreased significantly as fewer global collisions occurred.

G_{flat} than the other models because it had a large number of nodes and edges in G_{init} (905 and 3,295, respectively) owing to the large number of iterations of beam search (44). The run time of OPP merging based on curved layers varied with the number of instances of G_{flat} (#SO) and calls to CurviSlicer (#CU). CurviSlicer was the bottleneck in this step. Similarly, a larger number of connections between layers (T_l) was time consuming as CurviSlicer needed to be used with the smooth term for each decomposed OPP patch. Toolpath optimization also took a long time.

8 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

Path continuity significantly impacts the surface quality and duration of printing in extrusion-based 3D printing, especially for shell models. In this study, we proposed the concept of the OPP to quantify path continuity and developed a method to decompose a shell model into as few OPPs as possible by considering manufacturing-related constraints on a standard three-axis printer platform. We verified the performance of our methods on various models, and they yielded superior results to prevalent methods in terms of surface finish and printing time.

Limitations and Future Work. The toolpath considered here can be significantly improved. The inter-layer connecting path might be outside the model by a certain distance t_{min}, and avoiding such an artifact cannot be guaranteed through post-optimization of the global toolpath. A constant path width helped maintain a uniform thickness of the horizontal wall of the shell model. However, the thickness of the shell in the direction normal to the surface varied, which was an unintended consequence.

Our framework allows only for the geometry of the surface model to be printed. Extending the OPP criterion and the ACAP algorithm to general, solid models should be pursued in future work. A key issue in this vein is to adapt the OPP criterion to different interior structures or infilling patterns. Moreover, techniques of shell reinforcement, like adding ribs to it [Gil-Ureta et al. 2020] or modulating its thickness [Xing et al. 2021], are a promising direction of research to fully explore the applications of surface models. The contours of the surface models of varying thickness can thus be used. The ACAP algorithm should be adapted to combine thin regions of nozzle size with wider ranges of multiple nozzles of different sizes.

Fig. 21. We manually decomposed the kitten model, which did not satisfy the constraint on the support structure, into three patches that satisfied this constraint (left). We separately printed them using our ACAP toolpaths (middle) and assembled them afterward (right).

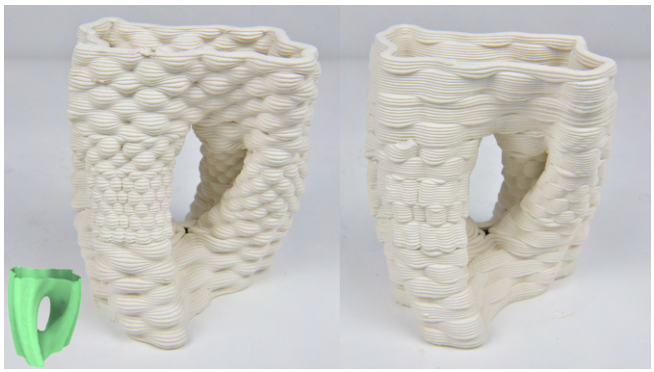


Fig. 22. The Julia vase model was decomposed into four OPPs with different textures.

The constraint on the support structure limits the feasible range of surface shapes for fabrication, i.e., the model is either self-supporting, or has support structures that are located on the ground. Thus, we manually decomposed models with no orientation that satisfies the constraint on the support structure and assembled them afterward, as shown in Figure 21. Different degrees of shrinkage of the material may leave noticeable seam lines between parts after solidification. In future work, we plan to study the generalization of complex surface models by considering decoupling strategies between the support and the intact structures. Moreover, we plan to examine scheduling strategies for printing support structures in situ with the intact model. An automated pre-decomposition process that can balance the number of decomposed patches with path continuity is also a desirable and natural objective.

Because our method generates the toolpath and the related G-code files, it is natural for us to consider research on fine-tuning

the parameters of printing, like the height of the extruder and the volume of extrusion [Takahashi and Miyashita 2017], and positions in a local range [Yan et al. 2021], to achieve a variety of sophisticated geometric features without violating the continuity of the toolpath (see Figure 22, where each OPP can be treated as an independent unit to embed different textures on the fly).

Finally, another problem worth exploring is to extend the proposed algorithm to multi-axis printing setups. Accordingly, the constraint on the slope angle can be ignored, the support structures have fewer restrictions on them, and thus there is more freedom regarding the toolpath. We believe that these directions of research should be pursued in future work in the area.

REFERENCES

Robert J.A. Allen and Richard S. Trask. 2015. An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilising a parallel deposition robot. *Additive Manufacturing* 8 (oct 2015), 78–87. https://doi.org/10.1016/j.addma.2015.09.001

Ana Anton, Angela Yoo, Patrick Bedarf, Lex Reiter, Timothy Wangler, and Benjamin Dillenburger. 2019. Vertical Modulations. Computational design for concrete 3D printed columns. In *ACADIA 19: Ubiquity and Autonomy. Proceedings of the 39th Annual Conference of the Association for Computer Aided Design in Architecture*, Kory Bieg, Danelle Briscoe, and Clay Odom (Eds.). Association for Computer Aided Design in Architecture (ACADIA), s.l., 596 – 605.

Prahar M. Bhatt, Rishi K. Malhan, Pradeep Rajendran, and Satyandra K. Gupta. 2020. Building free-form thin shell parts using supportless extrusion-based additive manufacturing. *Additive Manufacturing* 32 (mar 2020), 101003. https://doi.org/10.1016/j.addma.2019.101003

Shajay Bhooshan, Tom Van Mele, and Philippe Block. 2020. Morph & Slerp: Shape description for 3D printing of concrete. *Symposium on Computational Fabrication* (nov 2020). https://doi.org/10.1145/3424630.3425413

B.Huang and S.Singamneni. 2012. Alternate slicing and deposition strategies for fused deposition modelling of light curved parts. *Journal of achievements in materials and manufacturing engineering* 55 (2012), 511–517.

Joris Burger, Ena Lloret-Fritschi, Fabio Scotto, Thibault Demoulin, Lukas Gebhard, Jaime Mata-Falcón, Fabio Gramazio, Matthias Kohler, and Robert J. Flatt. 2020. Eggshell: Ultra-Thin Three-Dimensional Printed Formwork for Concrete Structures. *3D Printing and Additive Manufacturing* 7, 2 (apr 2020), 48–59. https://doi.org/10.1089/3dp.2019.0197

Debapriya Chakraborty, B. Aneesh Reddy, and A. Roy Choudhury. 2008. Extruder path generation for Curved Layer Fused Deposition Modeling. *Computer-Aided Design* 40, 1 (feb 2008), 235–243. https://doi.org/10.1016/j.cad.2007.10.014

Shareen S.L. Chan, Ryan M. Pennings, Lewis Edwards, and George V. Franks. 2020. 3D printing of clay for decorative architectural applications: Effect of solids volume fraction on rheology and printability. *Additive Manufacturing* 35 (oct 2020). https://doi.org/10.1016/j.addma.2020.101335

Lufeng Chen, Man-Fai Chung, Yaobin Tian, Ajay Joneja, and Kai Tang. 2019a. Variable-depth curved layer fused deposition modeling of thin-shells. *Robotics and Computer-Integrated Manufacturing* 57 (jun 2019), 422–434. https://doi.org/10.1016/j.rcim.2018.12.016

Zhangwei Chen, Ziyong Li, Junjie Li, Chengbo Liu, Changshi Lao, Yuelong Fu, Changyong Liu, Yang Li, Pei Wang, and Yi He. 2019b. 3D printing of ceramics: A review. *Journal of the European Ceramic Society* 39, 4 (apr 2019), 661–687. https://doi.org/10.1016/j.jeurceramsoc.2018.11.013

Chengkai Dai, Charlie C. L. Wang, Chenming Wu, Sylvain Lefebvre, Guoxin Fang, and Yong-Jin Liu. 2018. Support-free volume printing by multi-axis motion. *ACM Transactions on Graphics* 37, 4 (aug 2018), 1–14. https://doi.org/10.1145/3197517.3201342

Jimmy Etienne, Nicolas Ray, Daniele Panozzo, Samuel Hornus, Charlie C. L. Wang, Jonàs Martínez, Sara McMains, Marc Alexa, Brian Wyvill, and Sylvain Lefebvre. 2019. CurviSlicer: Slightly curved slicing for 3-axis printers. *ACM Transactions on Graphics* 38, 4 (jul 2019), 1–11. https://doi.org/10.1145/3306346.3323022

Ben Ezair, Saul Fuhrmann, and Gershon Elber. 2018. Volumetric covering print-paths for additive manufacturing of 3D models. *Computer-Aided Design* 100 (jul 2018), 1–13. https://doi.org/10.1016/j.cad.2018.02.006

Guoxin Fang, Tianyu Zhang, Sikai Zhong, Xiangjia Chen, Zichun Zhong, and Charlie C. L. Wang. 2020. Reinforced FDM: multi-axis filament alignment with controlled anisotropic strength. *ACM Transactions on Graphics* 39, 6 (nov 2020), 1–15. https://doi.org/10.1145/3414685.3417834

Francisca Gil-Ureta, Nico Pietroni, and Denis Zorin. 2020. Reinforcement of General Shell Structures. *ACM Transactions on Graphics* 39, 5 (sep 2020), 1–19. https://doi.org/10.1145/3375677

Jean Hergel, Kevin Hinz, Sylvain Lefebvre, and Bernhard Thomaszewski. 2019. Extrusion-based ceramics printing with strictly-continuous deposition. *ACM Transactions on Graphics* 38, 6 (nov 2019), 1–11. https://doi.org/10.1145/3355089.3356509

Philipp Herholz, Wojciech Matusik, and Marc Alexa. 2015. Approximating free-form geometry with height fields for manufacturing. *Computer Graphics Forum* 34, 2 (2015), 293–251. https://doi.org/10.1111/cgf.12556

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2013. Orthogonal slicing for additive manufacturing. *Computers & Graphics* 37, 6 (2013), 669–675. https://doi.org/10.1016/j.cag.2013.05.011

Samuel Hornus, Tim Kuipers, Olivier Devillers, Monique Teillaud, Jonàs Martínez, Marc Glisse, Sylvain Lazard, and Sylvain Lefebvre. 2020. Variable-width contouring for additive manufacturing. *ACM Transactions on Graphics* 39, 4 (jul 2020). https://doi.org/10.1145/3386569.3392448

Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition Ruizhen. *ACM Transactions on Graphics* 33, 213 (nov 2014), 1–12. https://doi.org/10.1145/2661229.2661244

Tim Kuipers, Eugeni L. Doubrovski, Jun Wu, and Charlie C.L. Wang. 2020. A Framework for Adaptive Width Control of Dense Contour-Parallel Toolpaths in Fused Deposition Modeling. *Computer-Aided Design* 128 (nov 2020), 102907. https://doi.org/10.1016/j.cad.2020.102907

Samuel Lensgraf and Ramgopal R. Mettu. 2016. Beyond layers: A 3D-aware toolpath algorithm for fused filament fabrication. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109/icra.2016.7487546

Samuel Lensgraf and Ramgopal R. Mettu. 2017. An improved toolpath generation algorithm for fused filament fabrication. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109/icra.2017.7989141

Samuel Lensgraf and Ramgopal R. Mettu. 2018. Incorporating Kinematic Properties into Fused Deposition Toolpath Optimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. https://doi.org/10.1109/iros.2018.8594398

Yamin Li, Kai Tang, Dong He, and Xiangyu Wang. 2021. Multi-Axis Support-Free Printing of Freeform Parts with Lattice Infill Structures. *Computer-Aided Design* 133 (apr 2021), 102986. https://doi.org/10.1016/j.cad.2020.102986

Thomas Llewellyn-Jones, Robert Allen, and Richard Trask. 2016. Curved Layer Fused Filament Fabrication Using Automated Toolpath Generation. *3D Printing and Additive Manufacturing* 3, 4 (dec 2016), 236–243. https://doi.org/10.1089/3dp.2016.0033

Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-Printable Parts. *ACM Transactions on Graphics* 31, 129 (nov 2012), 1–9. https://doi.org/10.1145/2366145.2366148

Ali Mahdavi-Amiri, Fenggen Yu, Haisen Zhao, Adriana Schulz, and Hao Zhang. 2020. VDAC: volume decompose-and-carve for subtractive manufacturing. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

Ioanna Mitropoulou, Mathias Bernhard, and Benjamin Dillenburger. 2020. Print Paths Key-framing. In *Symposium on Computational Fabrication*. ACM. https://doi.org/10.1145/3424630.3425408

Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Transactions on Graphics* 37, 5 (nov 2018), 1–15. https://doi.org/10.1145/3204458

Edisson Ordoñez, Jhon M. Gallego, and Henry A. Colorado. 2019. 3D printing via the direct ink writing technique of ceramic pastes from typical formulations used in traditional ceramics industry. *Applied Clay Science* 182 (dec 2019). https://doi.org/10.1016/j.clay.2019.105285

Sriram Pemmaraju and Steven Skiena. 2003. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press. https://doi.org/10.1017/CBO9781139164849

Erwin Peng, Danwei Zhang, and Jun Ding. 2018. Ceramic Robocasting: Recent Achievements, Potential, and Future Developments. *ADVANCED MATERIALS* (oct 2018). https://doi.org/10.1002/adma.201802404

Carlos F. Revelo and Henry A. Colorado. 2018. 3D printing of kaolinite clay ceramics using the Direct Ink Writing (DIW) technique. *Ceramics International* 44 (apr 2018), 5673–5682. https://doi.org/10.1016/j.ceramint.2017.12.219

Haruki Takahashi and Homei Miyashita. 2017. Expressive Fused Deposition Modeling by Controlling Extruder Height and Extrusion Amount. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/3025453.3025933

Godfried T Toussaint. 1985. Movable separability of sets. *Machine Intelligence and Pattern Recognition* 2 (1985), 335–375.

Ultimaker. 2021. Ultimaker Cura 4.9.1. (2021). https://ultimaker.com/software/ultimaker-cura

J. Vanek, J. A. Garcia Galicia, B. Benes, R. Mech, N. Carr, O.Stava, and G.S. Miller. 2014. PackMerger: A 3D Print Volume Optimizer. *Computer Graphics Forum* 33, 6 (sep 2014), 322–332. https://doi.org/10.1111/cgf.12353

Xiangzhi Wei, Siqi Qiu, Lin Zhu, Ruiliang Feng, Yaobin Tian, Juntong Xi, and Youyi Zheng. 2018. Toward Support-Free 3D Printing: A Skeletal Approach for Partitioning Models. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*

24, 10 (oct 2018), 2799–2812. https://doi.org/10.1109/TVCG.2017.2767047
Chenming Wu, Chengkai Dai, Guoxin Fang, Yong-Jin Liu, and Charlie C.L. Wang. 2017. RoboFDM: A robotic system for support-free fabrication using FDM. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109/icra.2017.7989140
Chenming Wu, Chengkai Dai, Guoxin Fang, Yong-Jin Liu, and Charlie C. L. Wang. 2020. General Support-Effective Decomposition for Multi-Directional 3-D Printing. *IEEE Transactions on Automation Science and Engineering* 17, 2 (apr 2020), 599–610. https://doi.org/10.1109/tase.2019.2938219
Lingwei Xia, Sen Lin, and Guowei Ma. 2020. Stress-based tool-path planning methodology for fused filament fabrication. *Additive Manufacturing* 32 (mar 2020), 101020. https://doi.org/10.1016/j.addma.2019.101020
Yu Xing, Yu Zhou, XinYan, Haisen Zhao, Wenqiang Liu, Jingbo Jiang, and Lin Lu. 2021. Shell thickening for extrusion-based ceramics printing. *Computers & Graphics* 97 (jun 2021), 160–169. https://doi.org/10.1016/j.cag.2021.04.031
Ke Xu, Yingguang Li, Lufeng Chen, and Kai Tang. 2019. Curved layer based process planning for multi-axis volume printing of freeform parts. *Computer-Aided Design* 114 (sep 2019), 51–63. https://doi.org/10.1016/j.cad.2019.05.007
Xin Yan, Lin Lu, Andrei Sharf, Xing Yu, and Yulu Sun. 2021. Man-made by Computer: On-the-Fly Fine Texture 3D Printing. *Symposium on Computational Fabrication* (oct 2021). https://doi.org/10.1145/3485114.3485119
Chanyeol Yoo, Samuel Lensgraf, Robert Fitch, Lee M. Clemon, and Ramgopal Mettu. 2020. Toward Optimal FDM Toolpath Planning with Monte Carlo Tree Search. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109/icra40945.2020.9196945
Xiaoya Zhai and Falai Chen. 2019. Path Planning of a Type of Porous Structures for Additive Manufacturing. *Computer-Aided Design* 115 (oct 2019), 218–230. https://doi.org/10.1016/j.cad.2019.06.002
Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics* 35, 4 (jul 2016), 1–10. https://doi.org/10.1145/2897824.2925958
Haisen Zhao, Hao Zhang, Shiqing Xin, Yuanmin Deng, Changhe Tu, Wenping Wang, Daniel Cohen-Or, and Baoquan Chen. 2018. DSCarver: decompose-and-spiral-carve for subtractive manufacturing. *ACM Transactions on Graphics* 37, 4 (aug 2018), 1–14. https://doi.org/10.1145/3197517.3201338
Fanchao Zhong, Wenqiang Liu, Yu Zhou, Xin Yan, Yi Wan, and Lin Lu. 2020. Ceramic 3D printed sweeping surfaces. *Computers & Graphics* 90 (aug 2020), 108–115. https://doi.org/10.1016/j.cag.2020.05.007
Andrea Zocca, Paolo Colombo, Cynthia M. Gomes, and Jens Günster. 2015. Additive Manufacturing of Ceramics: Issues, Potentialities, and Opportunities. *Journal of the American Ceramic Society* 98, 7 (jul 2015), 1983–2001. https://doi.org/10.1111/jace.13700

A ORIENTATION DETERMINATION

Given the input model M, this step extracts feasible printing orientations by uniformly sampling orientations over the Gaussian sphere, and then choosing one that satisfies the *support structure constraint*. For each extracted orientation, apply a similar method as [Hergel et al. 2019] to detect the support areas and validate the *support structure constraint*. First, slice the model and if a layer sampling point requires support, cast a ray downward. If intersect with the model itself (except adjacent layer), it indicates violating the *support structure constraint*.

B CURVISLICER FOR SURFACE MODELS

CurviSlicer takes the watertight triangle mesh, its tetrahedral mesh, and target flat areas as input, which cannot be directly applied to the surface model, a surface patch of watertight triangle mesh. The key challenge is that it's not straightforward to extend the gradient formulation of the vertical coordinates within each tetrahedron (Sec. 4.2 of [Etienne et al. 2019]) to that formulation within each triangle.

We propose to generate a tetrahedral mesh to approximate the original surface patch of watertight triangle mesh with a minimal shell thickness. In our implementation, we set it to 0.3% of the longest diagonal of the bounding box of the input model. We do not suggest directly offsetting the input surface model along the horizontal direction to form a watertight mesh and generate the corresponding tetrahedral mesh. The self-intersection problem raised by offsetting makes it hard for tetrahedralization. Our solution is described below. For each triangle, offset its centroid by a minimal distance along the triangle's normal direction, then connect the resulting point with the three points of the triangle to form a tetrahedron. For each edge of the target flat boundaries (a set of edges of triangles), offset its midpoint by a minimal distance along the horizontal direction starting, then connect the resulting point with the two endpoints to form a triangle.

C DIFFERENT MERGING RESULTS BY CURVING

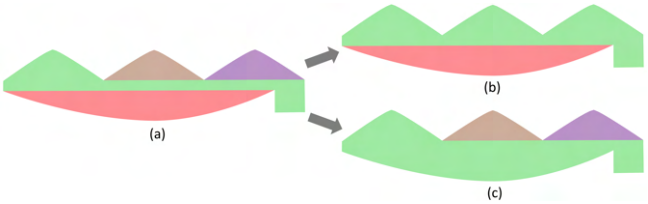


Fig. 23. Four I-OPPs can be decomposed by solving PC-MPC (a). If give priority to *curving* green, brown and purple OPPs, they can merge in sequence and finally merge into two OPPs (b). However, if firstly *curving* green and red OPPs, the remaining OPPs cannot merge due to the layer thickness constraint, and the final number of OPPs is three (c).

When choosing two OPPs *curving*, different orders may lead to different results. Figure 23 is an extreme example. The middle green OPP of (a) meets the layer thickness constraint when *curving* with only the upper or lower OPP. However, merging with both upper and lower OPPs will violate the constraint. The two different orders of OPP *curving* (b, c) result in a different number of final OPPs.

D THE PSEUDO-CODE

Algorithm 1 OPP Merging through Flat Layers

Input: G_{init}; The beam search width W;
Output: A set of optimal *flat OPP graphs* $\mathbb{O}$;
// Data structure setting
A directed acyclic graph $G_{solution}$ to encode the solution space;
A nodes vector $\mathbb{P}$ of the previous depth of beam search;
A nodes vector $\mathbb{C}$ of the current depth of beam search;
// Data structure initialization
$\mathbb{O} \leftarrow \emptyset$; $\mathbb{P} \leftarrow \emptyset$; $\mathbb{C} \leftarrow \emptyset$;
Set $G_{solution}$'s root node to an empty virtual node;
Add $G_{solution}$'s root node to $\mathbb{P}$;
// Solution space exploration with the beam search strategy
while $\mathbb{P} \neq \emptyset$ **do**
    // Generate candidate nodes of current depth from $\mathbb{P}$
    Declare a set of node sequences $\mathbb{S}$; $\mathbb{S} \leftarrow \emptyset$; $\mathbb{C} \leftarrow \emptyset$;
    for each node ♦ of $\mathbb{P}$ **do**
        Get the node sequence s_i from $G_{solution}$'s root node to ♦;
        Get the corresponding *flat OPP graph* G_{flat} of s_i;
        Get the node set M_i of G_{init} which are included in s_i;
        if M_i == all nodes of G_{init} & $G_{flat} \notin \mathbb{O}$ **then**
            $\mathbb{O} \leftarrow \mathbb{O} \cup \{G_{flat}\}$;
            Continue;
        // Generate candidate nodes from ♦
        Get $\overline{G_{init}}$ from G_{init} by deleting related edges to M_i;
        for each node $\overline{n_j} \in \overline{G_{init}}$ **do**
            if !($\overline{n_j}$'s indegree == 0 and $\overline{n_j} \notin M_i$) **then**
                Continue;
            Explore all traversal paths $\overline{P_j}$ starting from $\overline{n_j}$ with the greedy strategy to explore $\overline{G_{init}}$ as deep as possible;
            // Update $G_{solution}$ and $\mathbb{C}$
            for each traversal path $\overline{p_k} \in \overline{P_j}$ **do**
                Take $\overline{p_k}$ as a new candidate node ■ of $G_{solution}$;
                $b \leftarrow$ whether a sequence exists in $\mathbb{S}$ with the same G_{init}'s nodes as $(s_i, ■)$;
                Declare the last node of selected sequence ▲;
                if b & (■ == ▲) **then**
                    Add an edge from ♦ to ▲ in $G_{solution}$;
                else
                    Add ■ and an edge from ♦ to ■ in $G_{solution}$;
                    $\mathbb{C} \leftarrow \mathbb{C} \cup \{■\}$;
                $\mathbb{S} \leftarrow \mathbb{S} \cup \{(s_i, ■)\}$;
    // Apply the branch and bound technique
    if $\mathbb{O} \neq \emptyset$ **then**
        Return $\mathbb{O}$;
    // Extract W nodes from candidate nodes of current depth
    Sort $\mathbb{C}$ by the number of included G_{init}'s nodes;
    $\mathbb{P} \leftarrow$ the first W nodes of $\mathbb{C}$;

Algorithm 2 OPP Merging through Curved Layers

Input: A G_{flat}, G_{init};
Output: A G_{curved};
Two nodes n_l, n_r of G_{curved};
$G_{curved} \leftarrow G_{flat}$;
// Initial merging process
for each node ♦ of G_{curved} **do**
    Pairwisely merge sub-OPPs of ♦ with *curving* operation;
// OPP merging process
$b \leftarrow True$
while b **do**
    $b \leftarrow False$
    for each edge (n_l, n_r) of G_{curved} **do**
        // Deadlock detection
        if more than one path between n_l and n_r **then**
            Continue;
        // Inner Loop of OPP Merging Process
        Formulate a DAG G_{sub} from sub-OPPs of n_l and n_r:
            Get two sub-OPP sequences of n_l and n_r;
            Add back the edges between two sequences in G_{init};
        Pairwisely merge nodes of G_{sub} with *curving* operation;
            Label the nodes that have edges across the sequences;
            Select edges of G_{sub} over such labeled nodes;
            For each selected edge, call *curving* operation;
        // Update G_{curved}
        if G_{sub} has been merged to a single OPP node **then**
            Update G_{curved} by merging n_l and n_r;
            Update sub-OPP sequences of the merged OPP node;
            $b \leftarrow True$;
            Break;