Proceedings of the IASS Annual Symposium 2025
"The living past as a source of innovation"
27 October– 31 October 2025, México City, México
Juan Gerardo Oliva, Juan Ignacio del Cueto, Elisa Drago (eds.)

Surface-Toolpath Twins of Shell Components in 3D Concrete Printing for Optimized Buildability and Surface Quality

Yefan ZHI[a], Masoud AKBARZADEH[a,b,*]

[a] Polyhedral Structures Laboratory, School of Design, University of Pennsylvania, Philadelphia, USA
Pennovation Center, 3401 Grays Ferry Ave. Philadelphia, PA 19146, USA
*masouda@design.upenn.edu

[b] General Robotic, Automation, Sensing and Perception (GRASP) Lab, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, USA

Abstract

This paper directly links the abstract geometry of structural form-finding to the fabrication-aware design of discrete shells and spatial structures for 3D concrete printing through a bidirectional approach, where it creates surface-toolpath twins for the components, optimizing the buildability of the parts and their surface quality. The design-to-production process of efficient structural systems for 3D printing is often a top-down unidirectional process involving form-finding, segmentation, and slicing, where results face printability challenges due to incompatibility between the initial geometry and the printing system, as well as material constraints. We introduce surface-toolpath twins that can be interconverted and synchronized through efficient slicing and surface reconstruction algorithms to allow the combination of optimizations and modifications on either part of the twin in flexible orders. We provide two core methods for fabrication rationalization: (1) global buildability optimization on the surface mesh by normal-driven shape stylization and (2) local surface quality optimization on toolpath curves through intra-layer iterative adjustments. The result is a bidirectional design-to-production process where one can plug and play different form-finding results, assess and optimize their fabrication schemes, or leverage knowledge in fabrication design, model toolpath curves as sections, reconstruct surfaces, and merge them into form-finding and segmentation in an inverse way. The proposed framework enables the integration of form-finding expertise with fabrication-oriented design, allowing the realization of spatial shell structures with complex topologies or extreme geometrical features through 3D concrete printing.

Keywords: 3D concrete printing, continuous shells, toolpath design, shape stylization, surface reconstruction

1. Introduction

Extrusion-based 3D printing has emerged as a transformative technology for constructing advanced structural systems with increased design freedom [1], [2]. The additive manufacturing process enables the efficient construction of concrete shells and spatial structures with non-planar surfaces and complex topologies. These efficient structural forms are typically segmented into discrete components, 3D printed separately, and assembled on site [3], [4], [5] to integrate different printing orientations, utilize material anisotropy, and embed reinforcements.

While advanced form-finding techniques promise structural efficiency and expression, the design-to-production process faces challenges in buildability and surface quality. The successful printing of the structural components is contingent upon the printing system, material properties, and input geometries. Previous work outlines the criteria for their buildability and provides methods of predicting failure [6], thereby suggesting locations of manual modification.

This paper accelerates this top-down, unidirectional process involving form-finding, segmentation, and slicing by establishing a bidirectional design-to-production approach based on surface-toolpath twins

(Figure 1). It demonstrates how the surface representation and the toolpath representation of a compo-ent form a pair of twins that can be interconverted and synchronized through efficient slicing and surface reconstruction algorithms (Section 2). Two core methods for fabrication rationalization are then introduced: (1) global buildability optimization on the surface mesh by normal-driven shape stylization (Section 3) and (2) local surface quality optimization on toolpath curves through intra-layer iterative adjustments (Section 4). The surface-toolpath twins thus enable the optimizations and modifications on either part of the twin in flexible orders, regardless of the type of input geometry. The proposed methods convert surface design inputs into fabrication-ready toolpaths of optimal buildability and surface quality, while returning synchronized, optimized surfaces for design iterations. Furthermore, by accepting toolpath curves as direct input and connecting the synchronized surface to analysis and visualization methods (and even alternative fabrication methods), the workflow becomes fully bidirectional between surface and toolpath, thereby unifying design freedom and fabrication rationalization (Section 5).

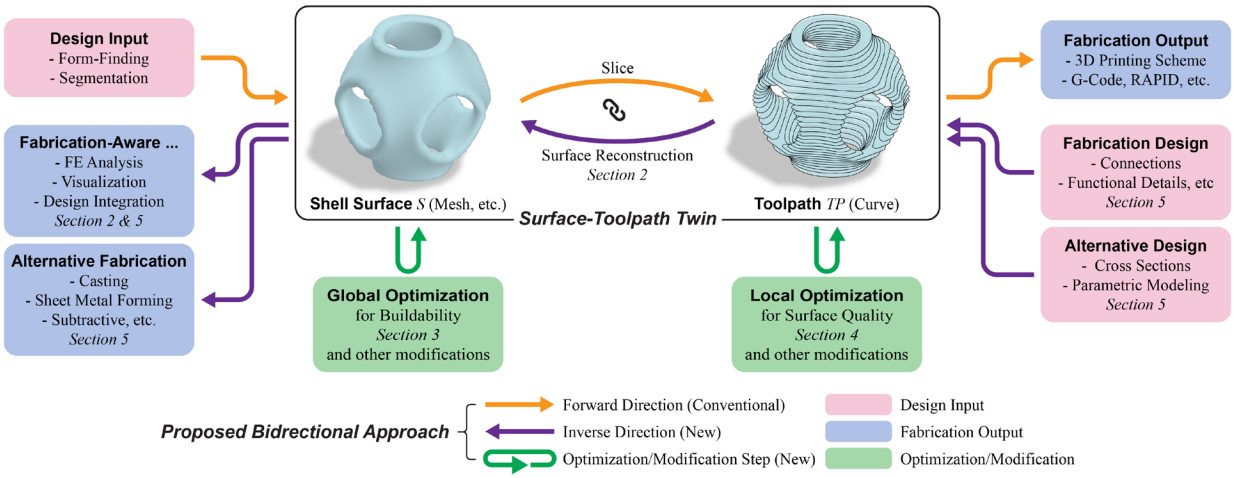

Figure 1: Proposed bidirectional approach highlighting the inputs, outputs, and optimization/operation steps.

The result of this paper is a toolset of analyses and optimizations organized in a bidirectional framework. With the proposed framework, one can combine expertise in form-finding and fabrication-oriented design and realize spatial shell structures with complex topologies or extreme geometrical features through 3D concrete printing.

2. Surface-toolpath twins

Surface. The surface and the volume are intrinsic properties of a 3D solid object, where the surface is a watertight 2D manifold that represents the boundary of the object, enclosing the volume. Architects and structural engineers exploit surfaces to model and communicate lightweight structural designs. They are (i) the direct product of shell form-finding methods such as force density method [7], thrust network analysis [8], polyhedral graphic statics [9], [10], and can also be easily derived from either (ii) volume-based methods such as topological optimization [11] and signed distance fields [12] by extracting the boundaries, or (iii) bar-node models such as polyhedral frames [13] by thickening.

Toolpath. Toolpath refers to a sequence of movement instructions to an end effector, typically an extruder on a robotic arm or gantry in 3D concrete printing. In this paper, a toolpath is defined as a hierarchical data structure of planar curves organized in layers (see [14]). To form surface-toolpath twins, the curves should be free of (self-)intersections and closed. In circumstances where open curves are necessary, they can be temporarily closed to adapt to the surface-toolpath twins framework and reopened afterwards.

2.1. Slicing

Slicing converts a surface into a toolpath. Types of surfaces, including NURBS surfaces [15], polygon meshes [16], and subdivision surfaces [17], can be intersected with equidistantly distributed planes

(height functions) to form the hierarchical data structure of the toolpath. (See supplementary information of [6]. While most toolpaths have parallel layers, the methods introduced in this paper also apply to non-parallel layers with moderate angles.) The 2D manifold in 3D space is thereby represented by a series of 1D manifolds in 2D spaces. The resulting curves are closed, free of intersections, and can be consistently oriented such that the left side is always the interior (see [14]).

2.2. Surface reconstruction for mutual conversions

We introduce an efficient algorithm that converts a toolpath back to a triangular mesh surface. The surface reconstruction from cross-sections problem is comprehensively reviewed by M. Zou, M. Holloway, N. Carr, and T. Ju [18]. While existing methods successfully reconstruct surfaces with complex topologies, a faster algorithm that harnesses the properties of concrete 3D printing toolpath curves is needed to *synchronize* the twins in real-time.

Our method is inspired by J.-D. Boissonnat [19] and adopts an alternative polygonal view of toolpath curves. Since toolpath curves for 3D concrete printing should have moderate curvatures, they can be approximated as a polygon with segments of similar lengths. Given a resolution L_R, a curve of length L is represented by $\lceil L/L_R \rceil$ *sample points* uniformly distributed using the length parameters (Figure 2a). Surface meshes are generated using these vertices between adjacent layers and then joined to form the complete surface, also called the *toolpath mesh*. For faces to have reasonable aspect ratios, we recommend using the layer thickness (also called "height") t as L_R.

2.2.1. A slice of the toolpath mesh between two adjacent layers

A slice of the toolpath mesh is generated by (i) using 3D Delaunay triangulation to populate volumetric tetrahedra that fill the space between the two planes, (ii) selecting those representing the interior, and (iii) extracting their surface faces. In (ii), we introduce the TETRACLASSIFICATION algorithm that examines a tetrahedron T and decides on its *type* (examples in Figure 2b) and *merging threshold* $f(T)$. (Out of T's at most four neighboring tetrahedra, if at least $f(T)$ tetrahedra are interior, then T is also selected to be interior and merged with its neighbors.)

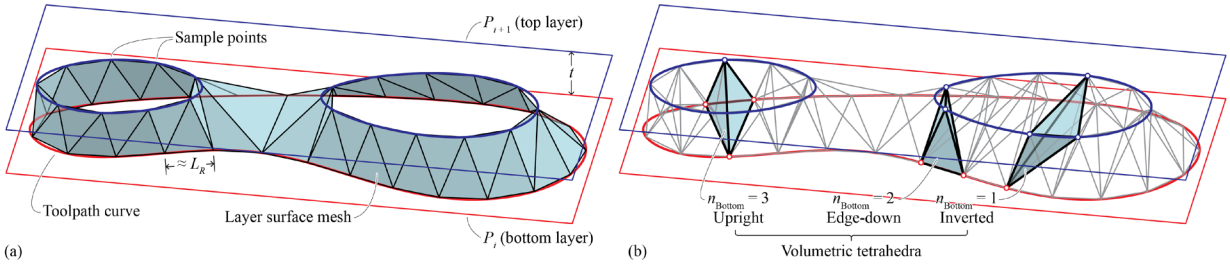

Figure 2: (a) A slice of the toolpath mesh. (b) Types of tetrahedra (showing three interior tetrahedra).

Figure 3 illustrates how TETRACLASSIFICATION investigates the shape of the local neighborhoods on the toolpath curves concerning the four vertices. Key terms used in Figure 3 are defined as follows:

- A vertex u *covers* another vertex v if when v is projected onto u's plane as v', v' is inside the sector defined by the two edges $(u, u.\text{next})$ and $(u, u.\text{prev})$ incident to u connecting neighboring sample points (illustrated in Figure 4). Since the polygon is consistently oriented, the left side of the two edges is the interior.
- An edge or a face of T inside one toolpath layer is *in region* if it is inside the closed interior formed by the toolpath curves on that layer. In TETRACLASSIFICATION: An edge (u, v) is in region if it is a native edge (u, v are neighboring sample points) or u covers v or v covers u. A face(u, v, w) with vertices in counterclockwise order is in region if any edge is native or every vertex covers at least one other vertex.
- The *typical diagonal length* $L_D = (t^2 + L^2)^{1/2}$.

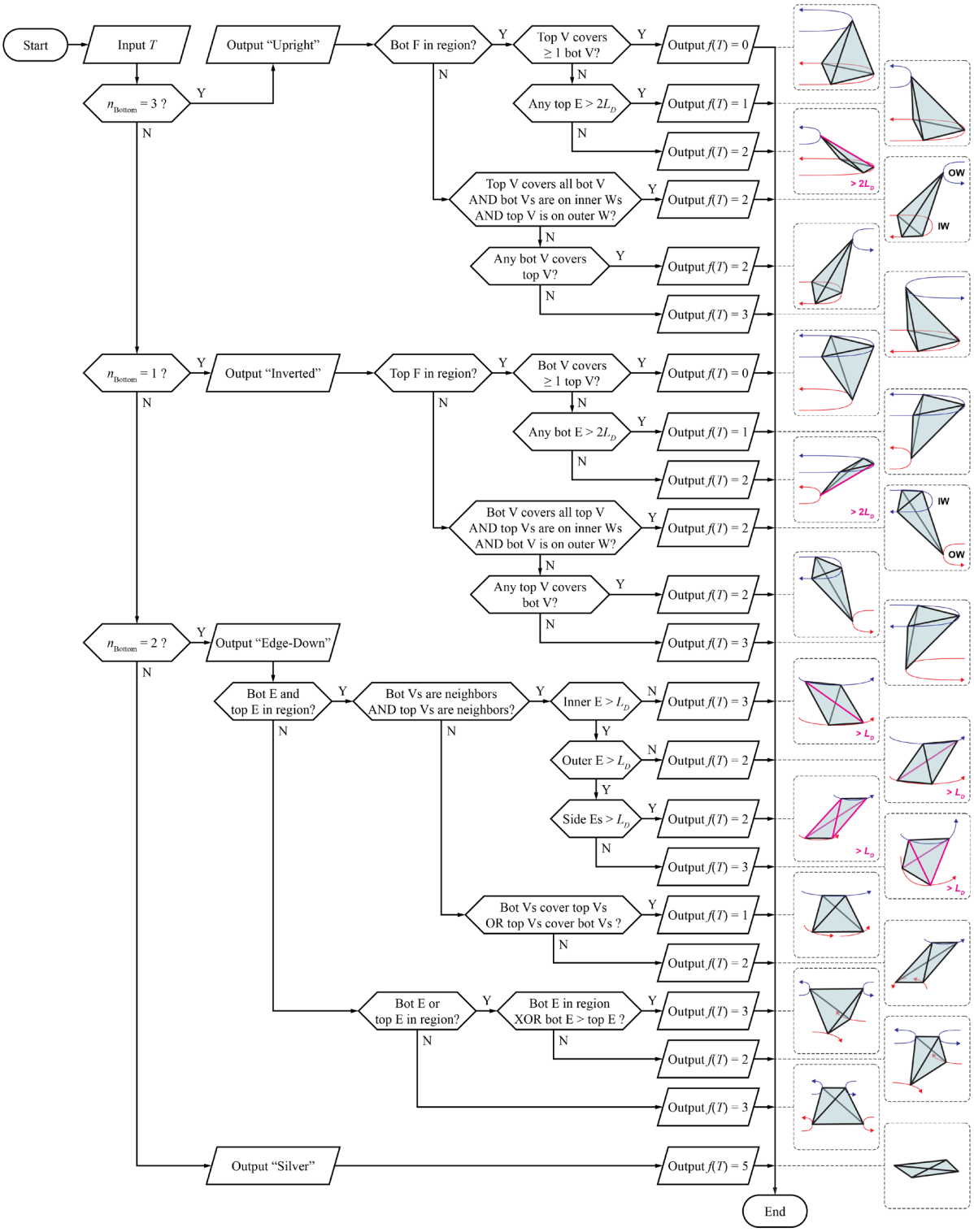

Figure 3: TETRACLASSIFICATION determines the tetrahedron type and the merge threshold $f(T)$.

After TETRACLASSIFICATION has examined every tetrahedra, a depth-first search (DFS) takes place to collect and merge the tetrahedra to form the interior. The algorithm collects all T with $f(T) = 0$, where for each T, it traverses all its neighboring tetrahedra and attempts to merge when they meet the merge threshold. Whenever a T is merged, its neighboring tetrahedra are searched again. Finally, the collected side faces that did not cancel out form the toolpath mesh slice as in Figure 2a.

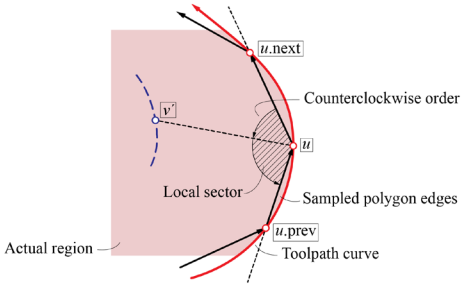

Figure 4: u covers v if relative to u, we have u.next, v', and u.prev in counterclockwise order.

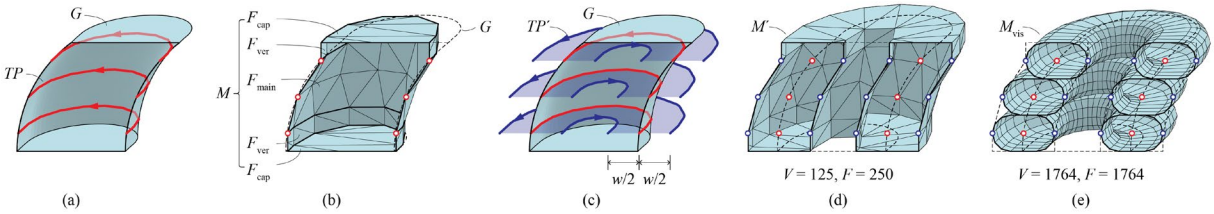

Figure 5: A 3-layer example of surface reconstruction: (a) input geometry G and sliced toolpath TP (b) toolpath mesh M; (c) offsetted toolpath TP' and the extruded regions; (d) toolpath mesh for the extruded volume M'; and (e) visualization mesh M_{vis}.

2.2.2. The complete toolpath mesh

As illustrated in Figure 5a and b, a complete toolpath mesh M is formed by (i) the side faces from all slices F_{main}; (ii) the vertical faces formed by projecting the bottom and top layers F_{ver}; and (iii) the cap faces formed by triangulating naked edge loops F_{cap}.

Representing the extruded volume. Another toolpath mesh can be generated to represent the volume of extrusion for simulation and representation purposes. By offsetting a toolpath TP on both sides by half of the extrusion width $w/2$, we get a pseudo-toolpath TP' whose interior region represents the extrusion (Figure 5c). Its toolpath mesh M' represents the volume of extrusion (Figure 5d) and is compact compared with the visualization mesh (Figure 5e, also see supplementary information of [6]). It serves

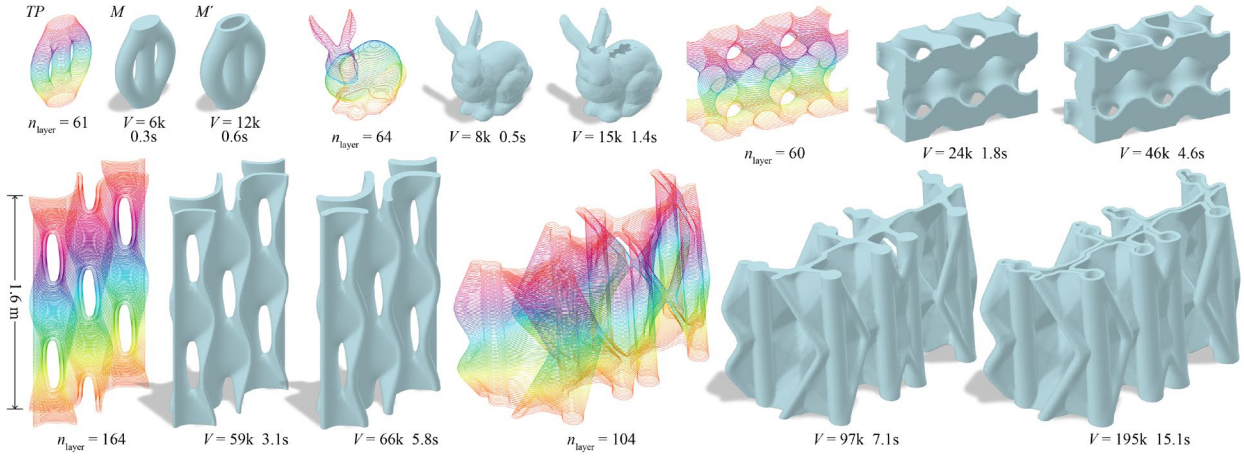

Figure 6: Examples of TP (color-coded by layer), M, and M'. 10 mm layer thickness, on a uniform scale. Showing number of layers n_{layer}, vertex count V, and computation time on a laptop with an Intel Core i9-13950HX CPU and 32 GB of RAM. Implementation discussed in Section 6. Top row: a porous column; the Stanford bunny, ©Stanford 3D Scanning Repository; a Gyroid TPMS brick. Bottom row: a porous panel, after Erwin Hauer [20], also in [6]; a Diamond-TPMS-inspired component for a post-tensioned canopy structure [5].

as the input for simulations such as finite element analysis (FEA). Offsetting can occur only on the outer side if a mesh representing a printed shell with casted infill is desired.

Time complexity. The 3D Delaunay triangulation that computes such tetrahedra is made possible by lifting the vertices into 4D, generating the 4D convex hull, and recovering the tetrahedra from 3-simplex faces [21]. The famous Quickhull algorithm achieves this in $O(n^2)$ time, bounded by the size of the output [22], where n is the size of input vertices, depending on the total length of curves within the same layer. For randomly placed vertices, the convex hull is expected to have $O(n)$ faces, and thus uses $O(n)$ time. While the vertices are contained in two planes, there is no better upper bound than $O(n^2)$ in the output size/time required. TETRACLASSIFICATION on each tetrahedron uses $O(1)$ time. DFS uses $O(n^2)$ time to inspect and merge all $O(n^2)$ tetrahedra. The total time used per layer is thus $O(n^2)$. Each layer can be computed in parallel. Figure 6 shows the result and computation time on five sets of examples. The proposed method might not recreate a smooth, continuous surface when toolpath curves change drastically between layers and thus become unprintable (e.g., see Stanford bunny).

3. Global optimization of the surface for buildability

Researchers have combined self-supporting constraints with topological optimization to generate efficient structural components printed free of supports [23], [24]. However, the generative method offers limited design freedom and does not follow the holistic design of discrete structural systems. To directly optimize the buildability of a given geometry, studies have employed curve-based [25] and voxel-based [26] methods, where results were limited to predefined configurations or saw significant changes from the input shapes and topology. Here we propose a versatile method of buildability optimization that operates on the surface mesh using normal-driven shape stylization. It reads any manifold mesh, takes as input a target overhang angle and a strength parameter, and outputs an optimized mesh with preserved topology and local shapes, where certain parts can also be fixed as boundaries.

3.1. Normal-driven stylization

The local overhang of a toolpath is defined per sample point as the projected offset between the point and its supporter on the previous layer, divided by the layer thickness [6]. On the surface twin, it can be approximated as the tangent of the angle between the sample point's *normal direction* and the world XY plane. Therefore, if we can effectively control the distribution of a surface's normal directions, we can perform overhang optimization on the surface twin. H.-T. D. Liu and A. Jacobson [27] introduced a mesh stylization method known as "spherical shape analogy" that, as illustrated in Figure 7a to d, takes as input a sphere A, a style shape A', an input mesh B and outputs a stylized B' such that "A is to A' as B is to B'" when we compare the normal directions of the same vertex before and after stylization.

3.2. Overhang optimization

Given a desired maximal overhang angle θ, we design A' as a spindle shape with the top and bottom having an overhang angle of θ (Figure 7a). The spherical analogy assigns vertices in B that exceed θ a new desired normal (T , Figure 7b and c). Finally, an iterative optimization deforms the vertices of B to form B' (Figure 7d) with a minimized energy $E = E_R + \lambda E_N$ where E_R is the regularization energy that matches the input mesh, E_N is the normal energy that matches the style, and λ is the strength parameter of the deformation. Case studies (Figure 7e, Figure 8) show that the optimized surfaces have reduced overhang values ($H_{B\prime}$) while the overall topology and the rigidity of local surface patches are preserved. To preserve the contact interface between the component/unit and its neighbors, boundary vertices are locked (highlighted in gray in $D_{B-B\prime}$).

4. Local optimization of the toolpath for surface quality

A printed component should be dimensionally accurate and have clean surface qualities to serve as part of a larger assembly and bond with additional reinforcements. Due to the mortar viscosity and

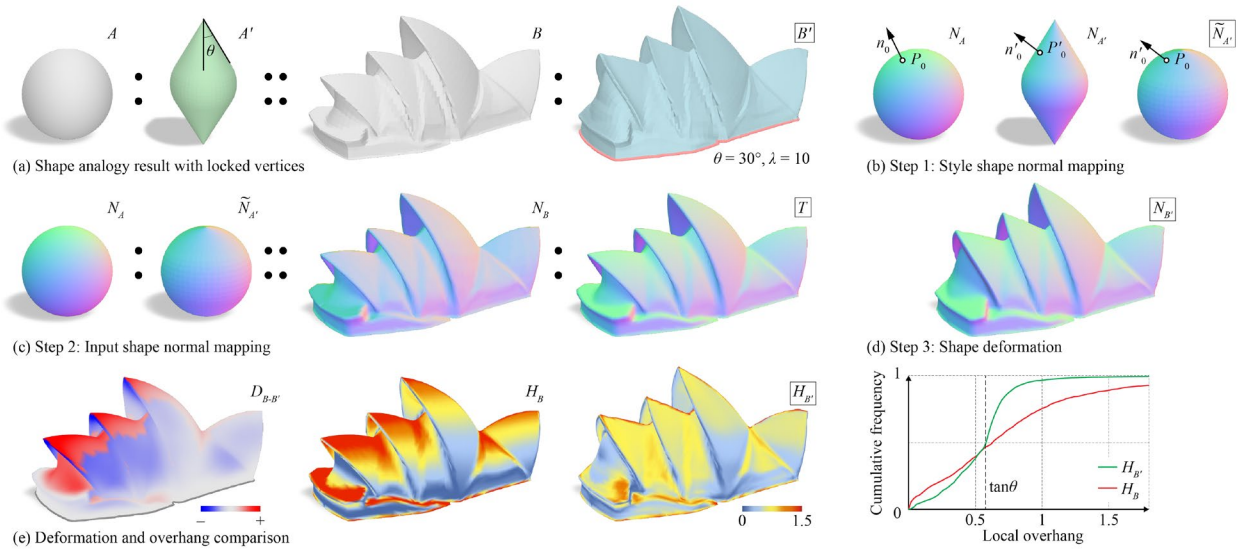

Figure 7: Application of normal-driven spherical shape analogy in overhang optimization. Partly after [27]. Sydney Opera House as an example, with bottom vertices locked (red in B', gray in $D_{B-B\prime}$).

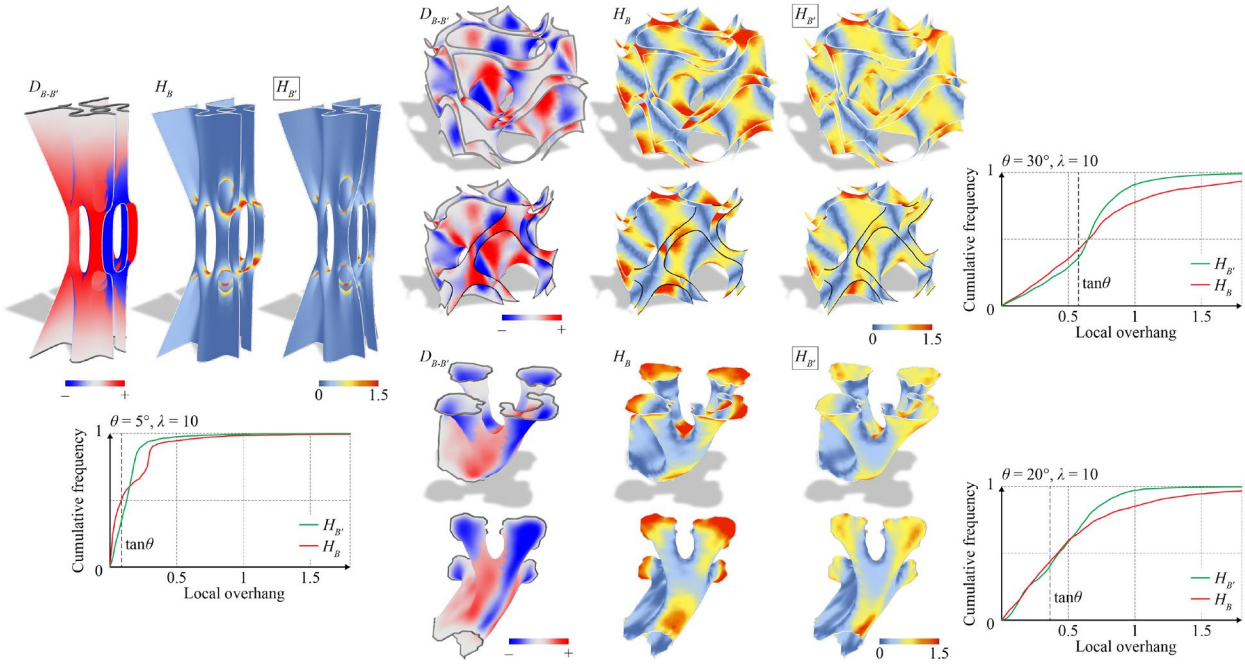

Figure 8: Overhang optimization examples with fixed boundaries (gray in $D_{B-B\prime}$) showing optimization parameters θ and λ, deformation $D_{B-B\prime}$, and overhang distributions of H_B, $H_{B\prime}$. Left: a shellular funicular column [9]. Top right: a double-sided Gyroid TPMS unit. Bottom right: a topologically optimized branching support structure.

the millimeter-to-centimeter-scale extruded section, prominent over/under-extrusion and deviation are easily formed at sharp turning angles [28] or dense toolpath areas and should thus be prevented.

4.1. Curve smoothing with side and curvature constraints

By replacing sharp turning points with fillet arcs, the toolpath curvature can be reduced to allow the extrudate to stay in deposit locations [29]. By imposing a side constraint on the arc, the void on the desired side can also be preserved [6]. However, these point-by-point modifications are not compatible with polylines of dense turning points whose segment lengths are less than the minimal curvature radius R_0. We introduce an iterative optimization algorithm that smooths a polyline curve with side and curvature

constraints (Figure 9). The NURBS toolpath curve is preprocessed as a polyline curve with equal or similar length segments by subdivision. The standard polyline smoothing algorithm updates a point P_i as $P_i \leftarrow P_i + f_i$ using a displacement vector

$$f_i = s_0 \left(P_{i-1} - 2P_i + P_{i+1}\right)/4 \tag{1}$$

Where $0 < s_0 \leq 1$ is the strength parameter and P_{i-1} and P_{i+1} are its neighboring vertices. To prevent the vertices from entering the locked side, we compute the permitted side normal n_i and apply f_i only if $n_i \cdot f_i \geq 0$. If not, to create the fillet, f_i is flipped and distributed to the two neighbors of P_i, called adjusted displacements $\hat{f}_{i-1}, \hat{f}_{i+1}$ (Figure 10). Since the adjusted displacement grows slower than the raw displacement, if $n_i \cdot f_i \geq 0$, f_i is penalized as $s_1 f_i$ (with $s_1 = 0.05$ implemented as a rule of thumb). As the polyline is smoothed, the length of the segments would change. Small segments are collapsed for acceleration. The complete algorithm is sketched in Algorithm 1.

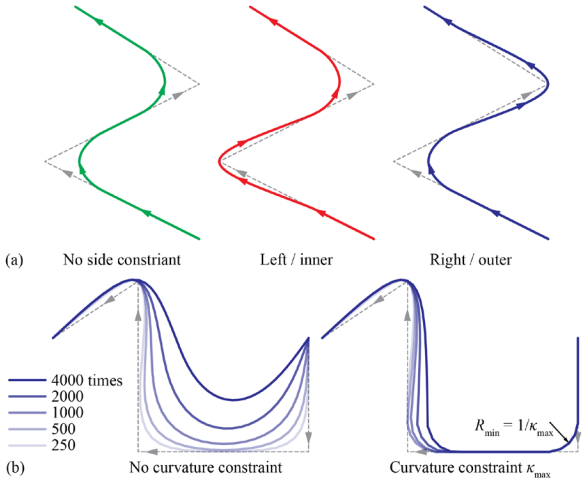

Figure 9: Smoothing of a polyline (gray) with (a) side constraint and (b) curvature constraint $\kappa_{\max}$ (right side) using the proposed algorithm.

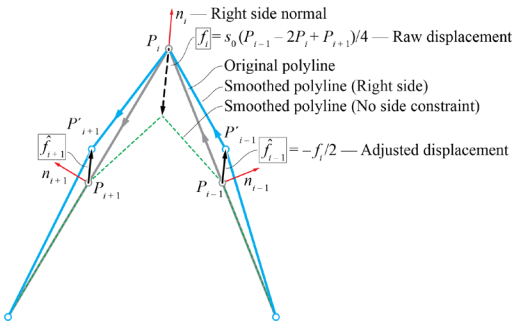

Figure 10: Adjusted displacement of points for smoothing with right side constraint (single smoothing step).

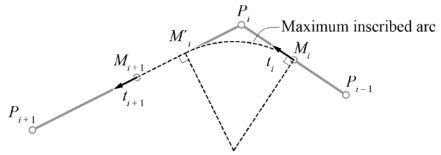

Figure 11: Polyline curvature estimation.

Algorithm 1: PolylineSmoothing(P, n_{smooth}, s_0, s_1, sgn_{side}, $\kappa_{\max}$, $\ell_{\min}$)

Input:
$P = \{P_0, P_1, ..., P_{n-1}\}$ — polyline vertices;
n_{smooth} — number of smoothing iterations;
s_0, s_1 — smoothing and adjustment strengths;
sgn_{side} — sign of the permitted side;
$\kappa_{\max}$ — maximal curvature;
$\ell_{\min}$ — minimal segment length;
Output:
P — smoothed polyline curve;

for $i_{\text{smooth}} \leftarrow 0$ **to** $n_{\text{smooth}} - 1$:
  for all P_i **in** P: // Initialization
    $n_i \leftarrow P_i$.PermittedNormal(sgn_{side})
    $\kappa_i \leftarrow P_i$.Curvature()
    $\hat{f}_i \leftarrow \mathbf{0}$
  for all P_i **in** P: // Calculate displacement
    $f_i \leftarrow s_0 \left(P_{i-1} - 2P_i + P_{i+1}\right)/4$
    if $f_i \cdot n_i < 0$: // Locked side
      $\hat{f}_{i-1} \leftarrow \hat{f}_{i-1} - f_i/2$
      $\hat{f}_{i+1} \leftarrow \hat{f}_{i+1} - f_i/2$
    else: // Permitted side
      $f_i \leftarrow s_1 f_i$
  for all P_i **in** P: // Apply displacement
    $f_i \leftarrow f_i + \hat{f}_i$
    if $f_i \cdot n_i \geq 0$ **and** ($\hat{f}_i \neq \mathbf{0}$ **or** $\kappa_i > \kappa_{\max}$):
      $P_i \leftarrow P_i + f_i$
  P.CollpaseShortSegments($\ell_{\min}$)
return P

To impose a curvature constraint, we can displace a point only if it exceeds the designated curvature $\kappa_{\max}$. For a point P_i, we can treat its neighborhood $M_i - P_i - M_{i+1}$ as a curve (M_i is the midpoint of segment $P_{i-1}P_i$, Figure 11). Denote the tangent at M_i as t_i, by definition, the curvature can be approximated as

$$\kappa_i = \frac{\mathrm{d}\theta}{\mathrm{d}x} = \frac{\mathrm{d}t}{\mathrm{d}x} \approx \frac{|t_{i+1} - t_i|}{|\overrightarrow{M_iP_i}| + |\overrightarrow{P_iM_{i+1}}|}. \tag{2}$$

However, as the polyline is smoothed, the segments have inconsistent lengths, and this κ_i becomes much smaller than the curvature of the maximum inscribed arc in this neighborhood (dashed in Figure 11, $M_i - P_i - M'_i$ when $|P_{i-1}P_i| \leq |P_iP_{i+1}|$), causing the smoothing to stop in advance. A better estimation is thus

$$\kappa_i \approx \frac{|t_{i+1} - t_i|}{|\overrightarrow{M_iP_i}| + |\overrightarrow{P_iM'_i}|} = \frac{|t_{i+1} - t_i|}{\min\left(|\overrightarrow{P_{i-1}P_i}|, |\overrightarrow{P_iP_{i+1}}|\right)}. \tag{3}$$

The smoothing of a vertex stops when it reaches the desired curvature (Figure 9b), preserving the input shape. Thus, the smoothing algorithm fulfills the objective of filleting curves with dense turning points.

4.2. Overfill optimization

In addition to sharp turning points, overfill occurs when a part of the curve gets too close to another part, or a second curve. This happens frequently at branching/merging points of a porous geometry when sliced. While this can be alleviated by Booleaning the main toolpath with auxiliary patches [6], the method requires refined manual modelling. We propose an automated optimization method on such overfills using an iterative algorithm. Similar to curve smoothing, the algorithm runs on polyline curves with equal or similar length segments. Let $d_{\min}$ denote the desired minimal distance between two separate parts, which can be the same as or a little smaller than the extrusion width. In each step, as illustrated in Figure 12, a point P_i is updated as $P_i \leftarrow P_i + g_i$ using a displacement vector

$$g_i = s_2 \cdot \frac{1}{|N_i|} \sum_{P_j \in N_i} \underbrace{\frac{\overrightarrow{P_jP_i}}{|\overrightarrow{P_jP_i}|}\left(d_{\min} - |\overrightarrow{P_jP_i}|\right)}_{g_{j,i}} \tag{4}$$

where $0 < s_2 \leq 1$ is the strength parameter and $N_i = \left\{P_j, i \neq j \;\middle|\; |\overrightarrow{P_jP_i}| < d_{\min}\right\}$ is the set of in the same layer that is within$d_{\min}$ distance to P_i. (4) is designed after the observation that when $|N_i| \leq 1$ holds for all points, we can use $s_2 = 0.5$ to get new polylines where the closest pairs become exactly $d_{\min}$ apart. However, as the displacement vector g_i is created additively on discrete sample points, the displacement magnitude becomes unpredictable, and its direction is contingent upon the distribution of the sample points. Therefore, we suggest using a small s_2 and repeat the optimization step n_{overfill} times. The results are compared in Figure 12. We use the RTree data structure [30] to efficiently solve N_i. For a moderate $d_{\min}$, the time complexity per step is $O(n \log n)$ with n being the number of sample points of that layer. Optimizations of the layers can be performed in parallel. Overfill optimization and smoothing with constraints help create overfill-free, dimensionally accurate tool-

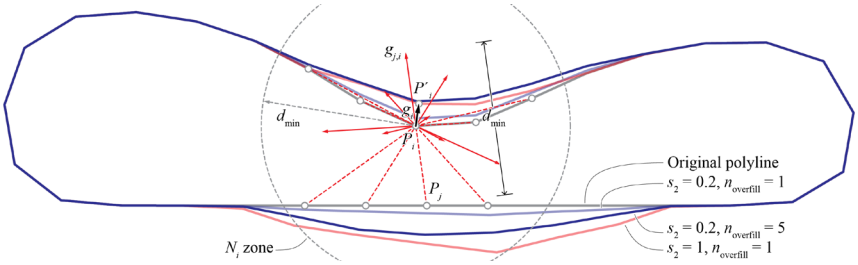

Figure 12: Overfill optimization at P_i. Using s_2 strength after n_{overfill} steps. The single-step results (gray, red) are twisted and stiff, while the low-strength, multi-step result (blue) is smoother and more symmetric.

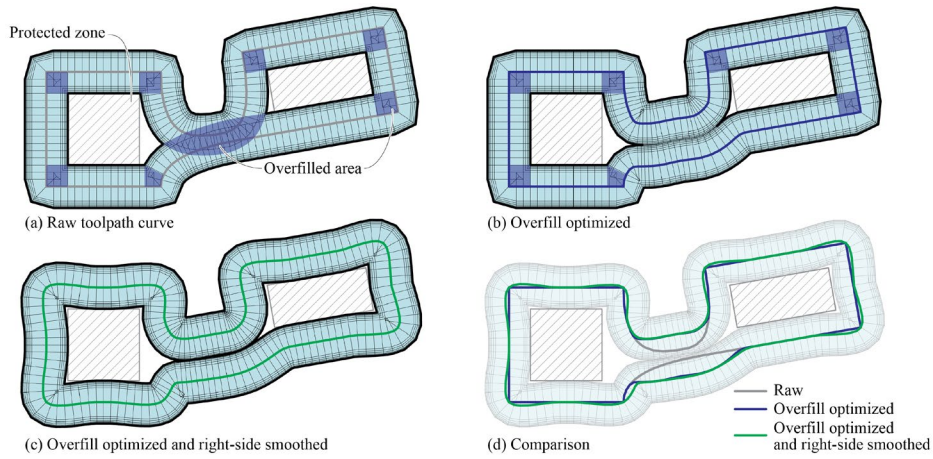

Figure 13: Optimization of a toolpath curve that embeds two protected reinforcement zones.

paths, thus improving the component's interface with external reinforcement or other components (some of these scenarios: [31], [32], [33]). Figure 13 illustrates a complete optimization example of a toolpath layer that embeds two rectangular reinforcements. The reinforcement areas are protected while overfills are reduced for optimized surface quality.

5. Bidirectional approach between design and fabrication

This section showcases the result of combining the synchronized surface-toolpath twins with the proposed optimization methods on the two parts. Three comprehensive examples are included (Figure 14, 15, 17, and 18), with workflow minimaps based on Figure 1. A scaled fabrication model (SFM) is a scale model 3D-printed using the same toolpath as in construction-scale 3D concrete printing to verify a toolpath design with minimal cost [34]. In this paper, accompanying these examples, 1:12.5 scaled fabrication models of 0.8 mm layer thickness are printed using an off-the-shelf desktop printer (a Creality CR-10 printer with a 1.5 mm nozzle) and PLA filaments.

5.1. Conventional design-to-production process with optimization

Figure 14 illustrates the design-to-production process of a discrete arch structure designed using polyhedral graphic statics (1). As shown in the minimap, the process follows the conventional forward direction. The springer component S, as a branching subdivision surface (2), is globally optimized to reduce its overhang angles (3). The toolpath curves TP are formed through rotary slicing (4) and are locally optimized to reduce extreme curvatures and overfilled areas (5). The fabrication result is visualized (6).

5.2. Inverse fabrication-aware design

Following Section 5.1, Figure 15 incorporates the inverse direction of fabrication-aware design where connections, functional details, etc., can be precisely integrated. Plates for anchoring the springer brick are modeled as curves (7) and wrapped as a toolpath (8), which is then unioned with the existing toolpath (9). The connection-embedded toolpath is then smoothed for fabrication. The surface M' representing the extruded volume is reconstructed (10) using the introduced method. M' serves as the input for FEA that reflects both the bolt connection's boundary condition and the internal void of the component (11). After FEA verifies the 3D printing scheme, the component is ready for printing and visualized (12). The SFM shows the effectiveness of the bolt connections (Figure 16).

5.3. Inverse alternative design and fabrication

Surface-toolpath twins offer a new design method that starts from customized curves. Figure 17 illustrates an example where the surface-reconstruction converts sparse sectional curves (1) wrapped as toolpath (2) into a continuous surface (3), which can be optimized (4) and then resliced (5) and printed (6) as desired. This method allows rapid modeling of continuous shell surfaces using minimal curve inputs

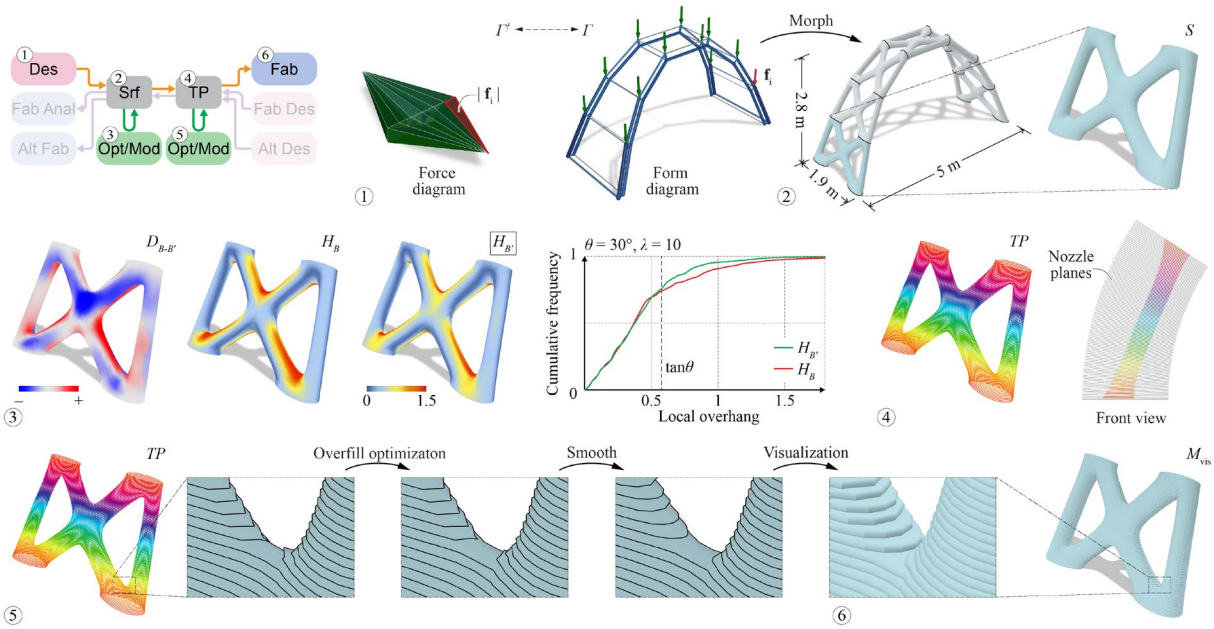

Figure 14: Example of a discrete arch, forward direction: (1) form-finding using polyhedral graphic statics [35]; (2) input surface modeled by morphing a designed pattern; (3) overhang optimization on the input surface mesh; (4) sliced toolpath; (5) optimization of toolpath curves; and (6) visualization of the 3D printing fabrication.

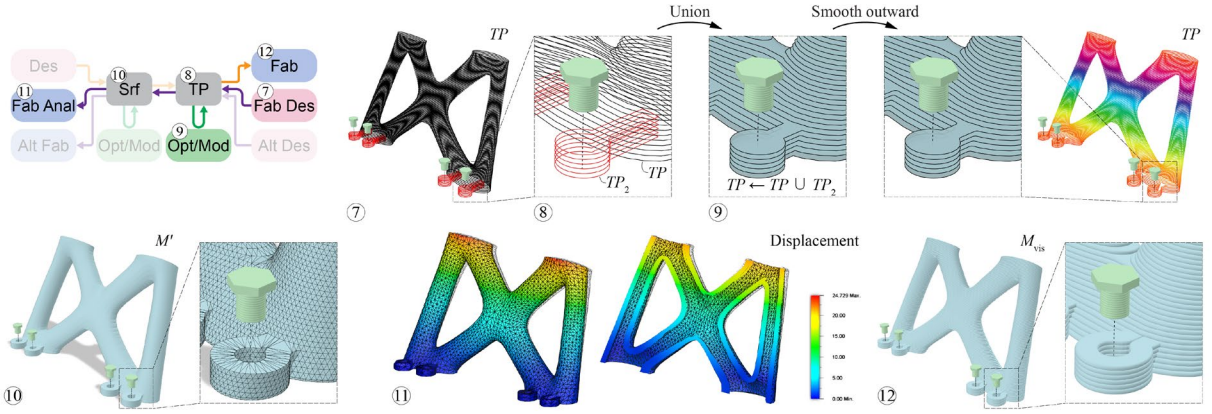

Figure 15: Example of a discrete arch, backward direction: (7) fabrication-oriented design: additional connection to anchor bolts; (8) additional toolpath; (9) toolpath Boolean union (see [6]) and side-and-curvature-constrained smoothing to get a joint toolpath; (10) toolpath mesh of the extruded volume; (11) fabrication-aware analysis: FEA of the anchored component using Fusion 360; and (12) visualization of the 3D printing fabrication.

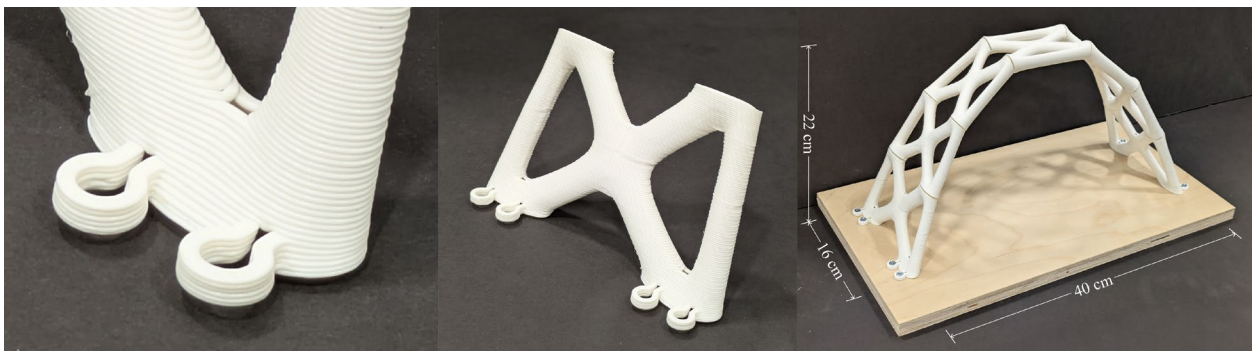

Figure 16: Scaled fabrication model of the discrete arch with 7 components. Each component takes around 1 h 30 mins to print.

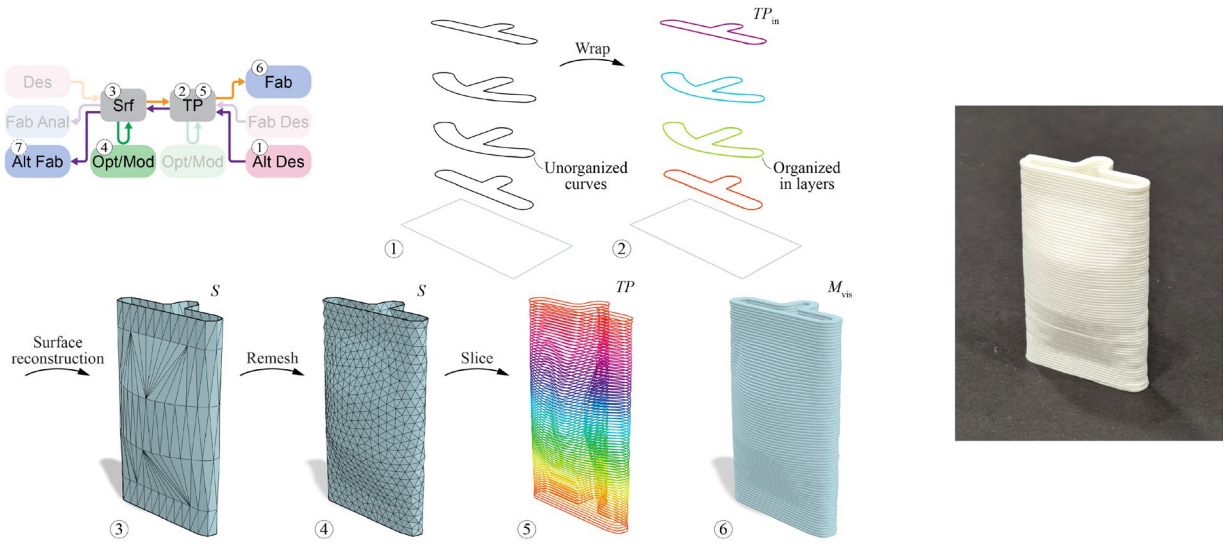

Figure 17: Example of a shell component designed in the inverse direction using sparse sectional curves. Left: (1) 4 unorganized sectional curves as input; (2) wrapped toolpath with layer information; (3) reconstructed surface without caps; (4) remeshed surface; (5) sliced toolpath with dense layers; and (6) visualization of the 3D printing fabrication. Right: scaled fabrication model.

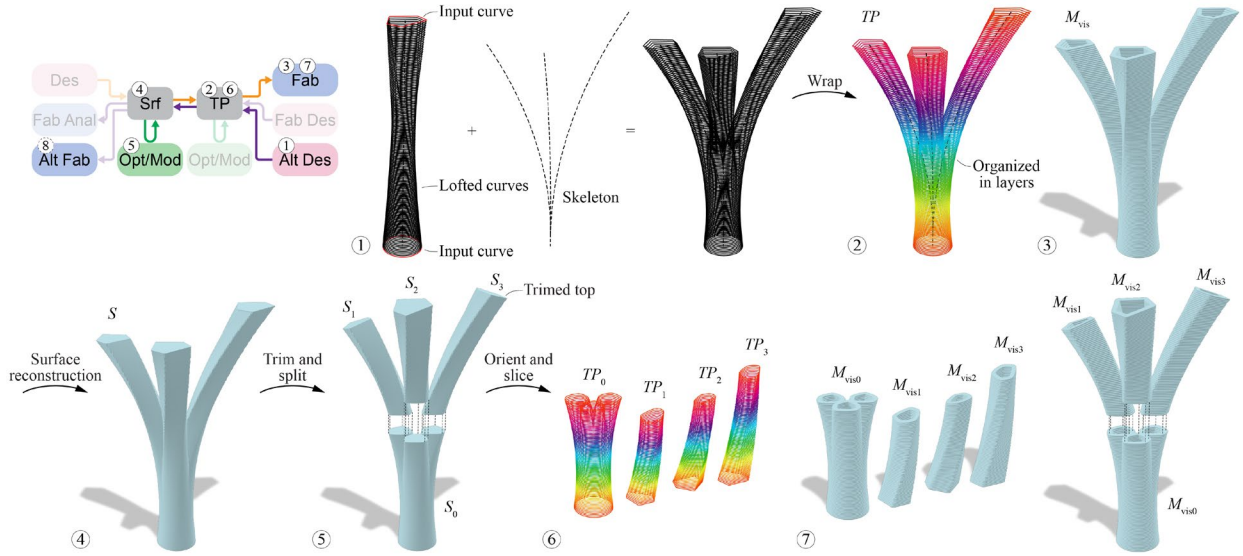

Figure 18: Example of a branching node designed in the inverse direction: (1) input curves created by mapping lofted curves on a skeleton and Boolean unioning each layer; (2) wrapped toolpath; (3) visualization of the 3D printing fabrication as a whole; (4) reconstructed surface; (5) modified surfaces as a discrete assembly; (6) oriented and sliced toolpaths; and (7) visualization of the 3D printing fabrication and assembly.

Figure 19: Scaled fabrication models of the branching node in two configurations.

from form-finding or manual design. In addition to 3D printing, the surface can also be realized through alternative fabrication methods such as casting, sheet metal forming, subtractive manufacturing, etc (7 in the minimap). Unlike in conventional curve-to-surface modeling techniques such as lofting or sweeping, the reconstructed surface is independent of the parameterization of the input curves, relying solely on the local proximity characteristics. The method can also reconstruct branching/porous surfaces from input layers with an inconsistent number of curves. (All curves are automatically organized in layers to form the input toolpath TP_{in}, where some layers can have more curves.)

Another example shows how surface reconstruction serves as an intermediate between steps of the inverse design-to-fabrication workflow. Branching nodes are often seen in porous shells and lattice or frame structures, but the manifold surface of a branching node can be hard to model due to its complex topology. Figure 18 shows how by organizing sectional curves (1), a branching node can be modeled as a toolpath (2). Two fabrication options are presented: printing as a whole (3); or reconstructing the surface (4), trimming and splitting it (5), and slicing (6) and printing (7) separately as a discrete assembly. The discrete assembly design is made possible by the reconstructed manifold surface, which can be altered freely. As an example, the branches are trimmed to have top faces perpendicular to the skeleton curves. The two fabrication schemes produce visually different results (Figure 19).

6. Software implementation

Methods and algorithms introduced in this paper have been implemented as a plug-in software named Ovenbird 2 [36] within Grasshopper® for Rhino® (Figure 20). Different from Ovenbird 1 [6], which was written using a combination of Grasshopper visual programming and Python, Ovenbird 2 is developed purely in C# and is thus faster and more robust. It uses MIConvexHull [37] for Delaunay triangulation and Clipper2 [38] for polygon offsetting. Proposed optimization methods in Ovenbird 2 rarely take more than one second to complete, with surface reconstruction being the most time-consuming (see Figure 6).

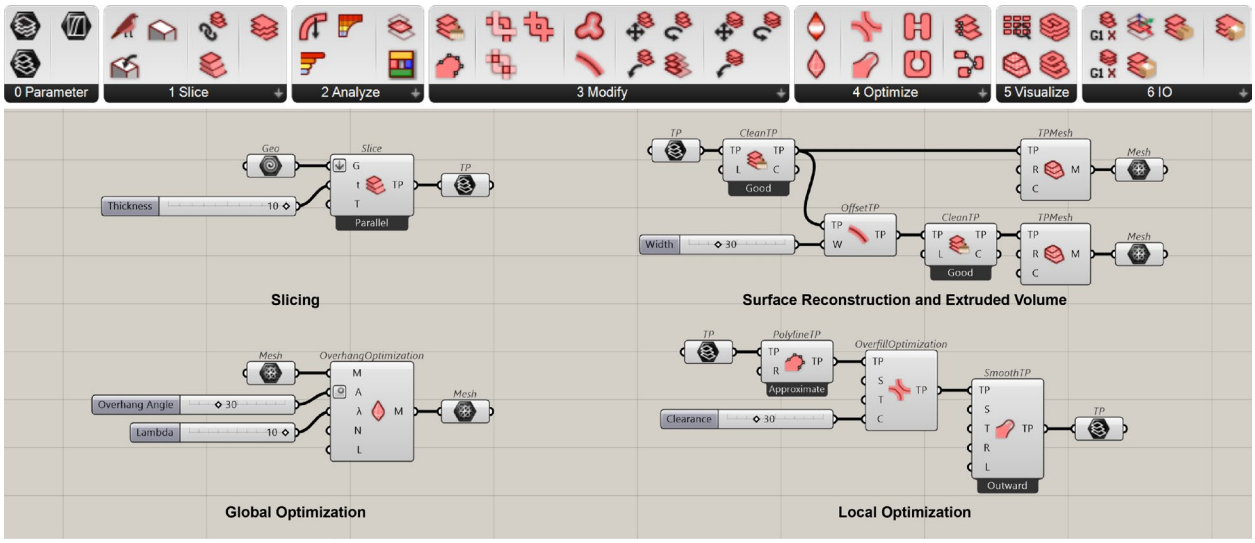

Figure 20: Top: the component panel of Ovenbird 2 for Grasshopper®. Bottom: concise usage examples of the methods introduced in this paper.

7. Conclusion

The proposed design framework based on surface-toolpath twins has successfully automated the toolpath design of some 3D concrete printing projects (Figure 21). In these design research studies, porous shells with extreme geometric features (bridges, large-overhang edges, etc.) are efficiently modeled and optimized for fabrication. Fabrication-aware optimization methods enhance the *resolution* of the printing and accommodate intricate forms, while the automated workflow extends the *design scale* previously limited by manual toolpath rationalization.

Figure 21: Top: a 3D-printed concrete wall with 6 components whose bridging points were reinforced by custom curves [6]. Bottom: a post-tensioned 3D-printed concrete canopy with embedded anticlastic geometry whose 9 components were optimized to reduce overfills while the conduits for post-tensioning were preserved [5], [39].

The study contributes to the design workflow of 3D-printed discrete concrete structures. The efficient buildability and surface quality optimization methods allow designers to plug and play different form-finding results, assess, optimize, and visualize their fabrication schemes in seconds. More importantly, with the bidirectional design-to-production framework, in an inverse way, one can leverage knowledge in fabrication design, model toolpath curves as sections, reconstruct surfaces, and merge them into form-finding and segmentation. In conclusion, the proposed framework broadens the design and fabrication possibilities for spatial shell structures through concrete 3D printing.

Future work. The study can be extended by a review of fabrication-oriented toolpath design in 3D concrete printing constructions. To further consolidate the method, a comparative experimental study with large-scale constructions is desired.

Acknowledgements

The authors gratefully acknowledge the support provided by the Advanced Research Projects Agency Energy (ARPA-E) Grant of the U.S. Department of Energy (DE-AR0001631) and the National Science Foundation Future Eco Manufacturing Research Grant (NSF, FMRG-CMMI2037097) awarded to Dr. Masoud Akbarzadeh. The authors also thank Mostafa Akbari for providing digital models of shellular structures as test examples, Yao Lu for help in Grasshopper plug-in development, and Teng Teng for the fabrication platform.

References

[1] A. Perrot and D. Rangeard, “3D Printing with Concrete: Impact and Designs of Structures,” in *3D Printing of Concrete*, John Wiley & Sons, Ltd, 2019, pp. 125–144. doi: 10.1002/9781119610755.ch5.

[2] R. A. Buswell, R. Soar, A. Gibb, and A. Thorpe, "The Potential of Freeform Construction Processes," in *2005 International Solid Freeform Fabrication Symposium*, The University of Texas at Austin, 2005, pp. 502–512. doi: 10.26153/TSW/7107.

[3] A. Dell'Endice *et al.*, "The Phoenix bridge: Improving circularity of 3D-concrete-printed unreinforced masonry structures," in *FABRICATE 2024*, , Ed., Copenhagen: UCL Press, 2024, pp. 90–97.

[4] A. Anton *et al.*, "Tor Alva: A 3D Concrete Printed Tower," in *FABRICATE 2024: Creating Resourceful Futures*, 2024, pp. 252–259.

[5] M. Akbarzadeh *et al.*, "Diamanti: 3D-Printed, Post-tensioned Concrete Canopy," in *FABRICATE 2024: Creating Resourceful Futures*, 2024, pp. 292–301.

[6] Y. Zhi, H. Chai, T. Teng, and M. Akbarzadeh, "Automated toolpath design of 3D concrete printing structural components," *Additive Manufacturing*, p. 104662, 2025, doi: 10.1016/j.addma.2025.104662.

[7] K. Linkwitz, "Force density method: Design of a timber shell," in *Shell Structures for Architecture*, 1st ed., Routledge, 2014, p. 12.

[8] P. Block, L. Lachauer, and M. Rippmann, "Thrust network analysis: Design of a cut-stone masonry vault," in *Shell Structures for Architecture*, 1st ed., Routledge, 2014, p. 18.

[9] M. Akbari and M. Akbarzadeh, "General translation of cellular to shellular polyhedral structures using reciprocity," *Structures*, vol. 63, 2024, doi: 10.1016/j.istruc.2024.106414.

[10] M. Akbarzadeh *et al.*, "Broader Applications of Polyhedral Graphic Statics," in *Polyhedral Graphical Statics: For Funicular Structural Form Finding*, Cambridge University Press, 2025, pp. 445–530.

[11] X. Huang and Y. Xie, "Evolutionary Topology Optimization of Continuum Structures," *Evolutionary Topology Optimization of Continuum Structures: Methods and Applications*, p. , 2010, doi: 10.1002/9780470689486.

[12] M. Bernhard, M. Hansmeyer, and B. Dillenburger, "Volumetric modelling for 3D printed architecture," in *Advances in Architectural Geometry 2018*, Göteborg, Sweden, 2018, pp. 392–415.

[13] S. Mozaffari, M. Hablicsek, M. Akbarzadeh, and T. Vogel, "Developing a polyhedral graphic statics formulation for tetrahedral truss systems," in *Proceedings of the IASS Annual Symposium 2020/21*, Surrey, UK, 2021.

[14] Y. Zhi, H. Chai, T. Teng, and M. Akbarzadeh, "Local optimization of self-supporting shell structures in 3D printing: a skeleton method," in *Proceedings of IASS 2023 symposium Integration of Design and Fabrication*, Melbourne, Australia, Jul. 2023.

[15] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Springer, 1997. doi: 10.1007/978-3-642-59223-2.

[16] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. A K Peters/CRC Press, 2010. doi: 10.1201/b10688.

[17] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, 1998, doi: 10.1145/280814.280826.

[18] M. Zou, M. Holloway, N. Carr, and T. Ju, "Topology-constrained surface reconstruction from cross-sections," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015, doi: 10.1145/2766976.

[19] J.-D. Boissonnat, "Shape reconstruction from planar cross sections," *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 1–29, 1988, doi: 10.1016/S0734-189X(88)80028-8.

[20] E. Hauer, *Erwin Hauer: continua, architectural screens and walls*. New York: Princeton Architectural Press, 2004.

[21] K. Q. Brown, "Voronoi diagrams from convex hulls," *Information Processing Letters*, vol. 9, no. 5, pp. 223–228, 1979, doi: 10.1016/0020-0190(79)90074-7.

[22] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, 1996, doi: 10.1145/235815.235821.

[23] M. Langelaar, "Topology optimization of 3D self-supporting structures for additive manufacturing," *Additive Manufacturing*, vol. 12, 2016, doi: 10.1016/j.addma.2016.06.010.

[24] J. Liu *et al.*, "Current and future trends in topology optimization for additive manufacturing," *Structural and Multidisciplinary Optimization*, vol. 57, no. 6, pp. 2457–2483, Jun. 2018, doi: 10.1007/s00158-018-1994-3.

[25] M. Mogra, O. Asaf, A. Sprecher, and O. Amir, "Design optimization of 3D printed concrete elements considering buildability," *Engineering Structures*, vol. 294, 2023, doi: 10.1016/j.engstruct.2023.116735.

[26] W. Yang, L. Wang, G. Ma, and P. Feng, "An integrated method of topological optimization and path design for 3D concrete printing," *Engineering Structures*, vol. 291, 2023, doi: 10.1016/j.engstruct.2023.116435.

[27] H.-T. D. Liu and A. Jacobson, "Normal-Driven Spherical Shape Analogies," 2021, [Online]. Available: http://arxiv.org/abs/2104.11993v2

[28] S. Sherugar, M. Birkett, and M. Blacklock, "Characterisation of print path deviation in material extrusion," *Progress in Additive Manufacturing*, Sep. 2023, doi: 10.1007/s40964-023-00502-y.

[29] L. Xia *et al.*, "Globally continuous hybrid path for extrusion-based additive manufacturing," *Automation in Construction*, vol. 137, p. 104175, 2022, doi: 10.1016/j.autcon.2022.104175.

[30] A. Guttman, "R-trees," *ACM SIGMOD Record*, vol. 14, no. 2, 1984, doi: 10.1145/971697.602266.

[31] J. Ribeiro, A. Morais, J. M. Silva, F. J. S. Brandão, B. Figueiredo, and P. J. S. Cruz, "Robotic 3DCP fabrication of custom-fit slabs for irregular pontoons," *Architectural Intelligence*, vol. 3, no. 1, 2024, doi: 10.1007/s44223-024-00056-1.

[32] J. Dobrzanski *et al.*, "From Digital Crafting to Digital Manufacturing: automated production using hybrid 3D concrete printing," *Journal of Building Engineering*, 2025, doi: 10.1016/j.jobe.2025.112640.

[33] M. Meibodi, Y. Lin, and H. Chen, "Hybrid Approaches Towards 3D Concrete Printing for Lightweight Reinforced Concrete Structures," *Digital Concrete 2024 - Supplementary Proceedings*, 2024, doi: 10.24355/DBBS.084-202408140649-0.

[34] Y. Zhi, T. Teng, and M. Akbarzadeh, "Designing 3D-printed concrete structures with scaled fabrication models," *Architectural Intelligence*, vol. 3, no. 1, 2024, doi: 10.1007/s44223-024-00070-3.

[35] Y. Lu, M. Hablicsek, and M. Akbarzadeh, "Algebraic 3D Graphic Statics with Edge and Vertex Constraints: A Comprehensive Approach to Extend the Solution Space for Polyhedral Form-Finding," *Computer-Aided Design*, vol. 166, p. 103620, Jan. 2024, doi: 10.1016/j.cad.2023.103620.

[36] Y. Zhi, "Ovenbird." [Online]. Available: https://www.food4rhino.com/en/app/ovenbird

[37] M. Campbell, "MIConvexHull - A .Net fast convex hull library for 2, 3, and higher dimensions." [Online]. Available: https://github.com/DesignEngrLab/MIConvexHull

[38] A. Johnson, "Clipper2 - Polygon Clipping and Offsetting Library." [Online]. Available: https://github.com/AngusJohnson/Clipper2

[39] M. E. Ororbia *et al.*, "Experimental study of a funicular concrete beam prototype," in *Proceedings of the IASS 2024 Symposium: Redefining the Art of Structural Design*, P. Block, G. Boller, C. DeWolf, J. Pauli, and W. Kaufmann, Eds., Zurich, Switzerland, Aug. 2024.