FrameFab: Robotic Fabrication of Frame Shapes

Yijiang Huang[1] Juyong Zhang[1*] Xin Hu[1] Guoxian Song[1] Zhongyuan Liu[2] Lei Yu[3] Ligang Liu[1*]

[1]University of Science and Technology of China [2]Luxun Academy of Fine Arts [3]Tsinghua University

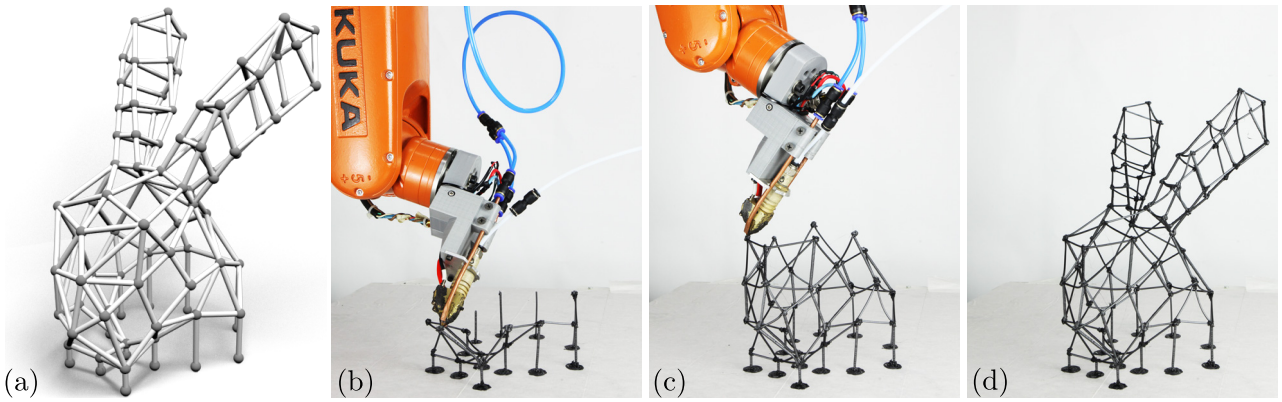


Figure 1: *Given a frame shape (a), we propose an algorithm for generating a feasible fabrication sequence of struts which guarantees that the already-printed part is in a stable equilibrium state and that the extrusion head avoids collision with the printed part at all fabrication stages. This is verified by a built prototype robotic fabrication system consisting of a 6-axis KUKA robotic arm with a customized extrusion head: (b) and (c) are intermediate fabrication states and (d) is the final fabrication object for the given frame shape (a).*

Abstract

Frame shapes, which are made of struts, have been widely used in many fields, such as art, sculpture, architecture, and geometric modeling, etc. An interest in robotic fabrication of frame shapes via spatial thermoplastic extrusion has been increasingly growing in recent years. In this paper, we present a novel algorithm to generate a feasible fabrication sequence for general frame shapes. To solve this non-trivial combinatorial problem, we develop a divide-and-conquer strategy that first decomposes the input frame shape into stable layers via a constrained sparse optimization model. Then we search a feasible sequence for each layer via a local optimization method together with a backtracking strategy. The generated sequence guarantees that the already-printed part is in a stable equilibrium state at all stages of fabrication, and that the 3D printing extrusion head does not collide with the printed part during the fabrication. Our algorithm has been validated by a built prototype robotic fabrication system made by a 6-axis KUKA robotic arm with a customized extrusion head. Experimental results demonstrate the feasibility and applicability of our algorithm.

Keywords: Frame shapes, frame structure, robotic fabrication, spatial thermoplastic extrusion, path planning

Concepts: •**Computing methodologies** → *Shape modeling;*

*Corresponding author: juyong@ustc.edu.cn, lgliu@ustc.edu.cn

SA '16 Technical Papers,, December 05-08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982401

1 Introduction

Frame shapes, i.e., wireframe structures consisting of struts, have been used to make everything from delicate jewelry to home decorative pieces to large structural objects, which makes them indispensable for artists, sculptors, and architects (Figure 2). In addition, frame shapes have gradually increased in popularity among designers in geometric modeling. For example, they are used as infill support in 3D objects for the purpose of saving material costs [Wang et al. 2013] and are printed as low-fidelity previews for designing 3D objects [Mueller et al. 2014].

Figure 2: *Frame shapes commonly appear in the fields of art, sculpture, architecture, and geometric modeling, etc.*

The fabrication of frame shapes using 3D printers, which prints layer-by-layer, results in slow fabrication and low-quality products. 3Doodler [3Doodler 2013] demonstrates the possibility of spatial thermoplastic extrusion and allows users to sketch into 3D space. WirePrint [Mueller et al. 2014] runs on standard FDM 3D printers to fabricate each strut in a single stroke by extruding the filament directly in 3D space, which can fabricate regular frame shapes using a particular contour-plus-zigzag approach. The very recent work of

the On-the-Fly Print system [Peng et al. 2016] introduces a 5DOF wireframe printer, by modifying a delta FDM 3D printer and adding two rotation axes on the printing platform. Concurrently Wu et al. [Wu et al. 2016] develops a method to generate printing order of struts, which guarantees no collisions, in printing wireframes by this 5DOF printer.

In recent years, various 6DOF robotic fabrication systems, such as Freeform Printing [Oxman et al. 2013; Helm et al. 2015], MeshMould [Hack and Lauer 2014], Anti-gravity Printing [Mataerial 2015], and Formwork Printing of Branch Technology [Branch 2015], which make use of an industrial 6-axis robotic arm equipped with a customized extrusion head, have been built for generating struts freely in 3D space. These systems are employed to fabricate frame shapes and have already shown great potential for fabricating geometrically complex frame shapes.

However, existing robotic fabrication systems require frame shapes with specific patterns, such as regular lattices as input and represent the shapes with layers of zigzag patterns. They then fabricate each layer by moving the extrusion head up and down repeatedly [Mueller et al. 2014; Branch 2015]. It is still challenging to fabricate general frame shapes with complicated structures like the bunny wireframe shown in Figure 1(a). The fabrication sequence of the struts, as a part of the installation design of the fabrication system, is nontrivial twofold. First, the already-fabricated parts should be in a static and stable equilibrium state. Each of the printed nodes should be within a small range of its input position so that the extrusion head can locate it in the following steps. Second, the moving part of the robotic system should avoid colliding with the fabricated parts as it moves at any time. The two issues are indeed coupled, i.e., collision detection between the moving part of the robotic system and the printed part is dynamically sequence dependent, which makes it highly computationally complicated.

Our Work In this work, we present a novel algorithm to find a fabrication sequence of struts for general frame shapes. The input of our algorithm is a physically self-supporting frame shape, which is itself in a stable equilibrium configuration. Our goal is to generate a feasible sequence of struts so that the printed part is in a stable equilibrium state and the moving part of the robotic fabrication system does not collide with the already-printed part during all stages of fabrication.

The search problem is actually a combinatorial optimization problem, which is substantially challenging because the solution space of the fabrication sequence is exponential in the number of struts. It suffers more with two constraints conducted in the fabrication:

- **Stability constraint.** The already-printed part should be in a stable equilibrium state during the entire fabrication process. This means that the printed part should be self-supporting, and each of the printed nodes should be within a small range of its input spatial position so that the extrusion head can locate them in the following steps;
- **Collision constraint.** The moving part of the robotic system such as the extrusion head should avoid colliding with the printed part as it moves. The collision detection between the printed part and the extrusion head is sequence dependent and is highly computationally complicated.

To make the non-trivial combinatorial problem computationally tractable, we propose a divide-and-conquer strategy. The input frame shape is first decomposed into *stable layers*. We formulate it as a sparse optimization problem with nonlinear constraints. Given the layer decomposition, we search a feasible fabrication sequence in an exponential space using an efficient heuristic scheme, which is based on a local optimization together with a backtracking strategy. The search is faster than the brute force one due to favoring the exploration of feasible parts of the solution space. Although the moving part of the robotic system consists of the robotic arm and the the extrusion head, we only consider to avoid collision between the extrusion head with the printed part to simplify the computation in our implementation, as the robotic arm is generally far away from the printed part and thus has less chance to collide with the printed part as it moves. We built up a prototype robotic 3D printing system, which is based on a 6-axis KUKA robotic arm and a customized extrusion head, to verify our algorithm (Figure 1 (b-d)). Various experimental results has demonstrated the feasibility and applicability of our algorithm.

Contributions Our contributions are summarized as follows.

- We propose a constrained sparse optimization method to decompose a large frame shape into several smaller stable layers, where each layer is guaranteed to be in a stable equilibrium state over its preceding layers;
- We propose an efficient scheme to search a feasible fabrication sequence for each layer, while taking both structural stability and collision detection into consideration.

To the best of our knowledge, our work is the first to offer an efficient computational way to find feasible fabrication sequences for creating frame shapes using 6DOF robotic fabrication systems. This provides a fundamental enabling algorithm and exemplifies a new area of research in the field of robotic fabrication.

2 Related Work

Construction Sequence of Spatial Structures The physical construction sequences for 3D objects with spatial structures have historically been studied for practical purposes [Fallacara and D'Amato 2012]. Various geometric and physical constraints have to be considered when constructing the subparts and pieces. Constructions of puzzles [Lo et al. 2009; Song et al. 2012], 3D assembly instructions [Agrawala et al. 2003], and planar interlocking pieces [Hildebrand et al. 2012; Schwartzburg and Pauly 2013], have been addressed with geometric constraints, which ensure that no piece is obstructed by the existing structure during assembly. Modern freeform shells are constructed with wood panels cut according to section curves [Wendland 2009]. 3D printing support struts generated by constructing frame structure around the object for FDM printing is studied in [Dumas et al. 2014], which also considers the stability at all stages of the print process. Recently, Deuss et al. [Deuss et al. 2014] have studied the physical construction of self-supporting structures with brick and stone blocks. Their solution leverages the internal force distribution of the partially assembled structure by keeping the structure in stable equilibrium at all stages of the assembly. Similarly, our problem is also a difficult combinatorial problem and solved by a computationally tractable divide-and-conquer strategy. However, we have to consider more complicated dynamic constraints, such as avoiding collision between the movement of the extrusion head and the already-fabricated parts.

Fabrication of Frame Shapes After 3Doodler [3Doodler 2013] demonstrates the spatial thermoplastic extrusion, spatial 3D printing of frame shapes has drawn much attention of people. WirePrint [Mueller et al. 2014] adopts standard FDM 3D printers to fabricate low-fidelity wireframe previews for 3D shapes. It converts a 3D object into a frame shape by slicing the 3D model into horizontal slices and filling the space between slices with a zigzag pattern. Then WirePrint constructs the layers by moving the extrusion

head up and down repeatedly. Protopiper [Agrawal et al. 2015], a computer aided hand-held fabrication device, is introduced to allow users to sketch room-sized frames at actual scale.

Recently, the On-the-Fly Print System [Peng et al. 2016], a 5DOF frame printer, is built based on a standard 3DOF delta 3D printer with a 2DOF rotation printing platform. The concurrent work of Wu et al. [2016] presents a collision avoidance algorithm for searching a feasible printing sequence for fabricating general frame shapes with the On-the-Fly Print System. In addition to collision avoidance, our algorithm also considers the the stability of the printed part during the whole fabrication process.

Robotic Fabrication Systems In recent years, remarkable work has been done by roboticists, architects, and designers in designing robotic fabrication systems, which are based on 6DOF industrial robotic arms amounted with customized extrusion heads, and employing them to fabricate frame shapes. The exploration of experiments on robotic printing, such as Freeform Printing [Oxman et al. 2013; Helm et al. 2015], Mesh-Mould [Hack and Lauer 2014], Anti-gravity Printing [Mataerial 2015], and Formwork Printing [Branch 2015], etc., have been carried out in architectural research facilities around the world, with custom extrusion system and industrial robots [Gramazio 2014]. These works have put substantial effort into the mechanical design of the fabrication systems, such as the custom-designed thermoplastic extruders and nozzles, melting temperature control, and air-cooling units, etc.

Our work, on the other hand, focuses on the valid fabrication sequence for general frame shapes instead of ones with regular patterns, which supplements these works and provides a fundamental enabling algorithm for these installation designs. And the proposed algorithm has been well validated by the prototype robotic 3D printing system [Yu et al. 2016].

Structural Analysis of Frame Shapes Structural analysis, which is based on the finite element method, is adopted for analyzing the stability of 3D models for physical fabrication [Stava et al. 2012; Zhou et al. 2013; Panetta et al. 2015]. The stiffness equation of frame shapes is derived based on beam theory. In this way, frame struts are assumed to behave like simple beams under linear deformation [Hughes 1987], so it is used to compute the stable equilibrium state of frame shapes [Wang et al. 2013]. In contrast to previous works, which focus on the stable equilibrium at only one state, our algorithm has to consider the stable equilibrium of fabricated frame shapes at all stages of fabrication.

3 Problem

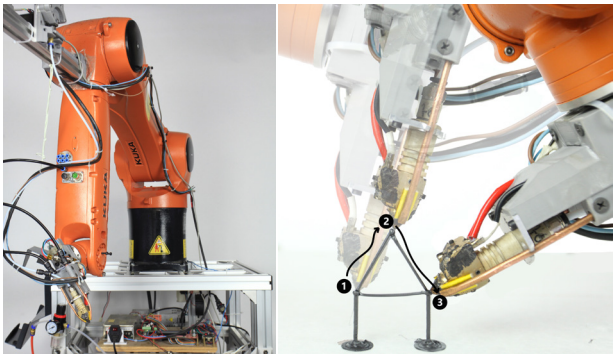
Frame Shapes A *frame shape* $G = \{\mathcal{N}, \mathcal{S}\}$ is composed of a set of frame nodes $\mathcal{N} = \{n_i, i = 1, 2, \ldots, |\mathcal{N}|\}$ and a set of frame struts $\mathcal{S} = \{s_i, i = 1, 2, \ldots, |\mathcal{S}|\}$, where each strut $s_i = [n_{i_1}, n_{i_2}]$ connects two adjacent nodes n_{i_1} and n_{i_2}. Each strut, as the extruded filament, is a cylindrical shape with a constant radius r.

A fabricated frame shape is called to be in a *stable* equilibrium state if each of its nodes is within a certain range of spatial positions, i.e., within a small distance of ε from its input position, when it is in a static equilibrium state. In other words, a stable fabricated frame shape is self-supporting with little deformation. A fabricated shape has to be stable so that the extrusion head can locate it in the following steps during the entire fabrication process.

Problem Given an input frame shape G, which is itself a stable structure, our goal is to generate a sequence of struts such that the fabricated frame structure should be in stable equilibrium states and the extrusion head should not collide with them during all stages of fabrication. The problem turns to find a feasible (not necessarily unique) permutation $\{i_1, \cdots, i_{|\mathcal{S}|}\}$ of $\{1, \cdots, |\mathcal{S}|\}$ such that the printed part $\mathcal{S}_k^{\mathrm{p}} = \{s_{i_1}, \cdots, s_{i_k}\}$ is in a stable equilibrium at the k-th state and the extrusion head does not collide with $\mathcal{S}_k^{\mathrm{p}}$ when printing the next strut $s_{i_{k+1}}$, for all $k = 1, 2, \cdots, |\mathcal{S}| - 1$. We denote $\mathcal{S}_k^{\mathrm{u}} = \mathcal{S} \setminus \mathcal{S}_k^{\mathrm{p}}$ as the unprinted struts.

3.1 Validation System

Experimental Robotic Fabrication System To verify our algorithm, we built up an experimental robotic printing system [Yu et al. 2016], as shown in Figure 3. Our setup consists of a 6-axis robotic arm (KUKA KR 10 R1100) equipped with a custom-built extrusion head, with the component buildup resembling standard filament extrusion technology. The robotic arm can move and rotate freely in space and can locate spatial points within a tolerance of ρ (in our system $\rho = 0.1\mathrm{mm}$). More details about the hardware setup of the robotic system can be found in the supplementary material.



(a) (b)

Figure 3: *Our experimental robotic fabrication system. (a) The robotic fabrication setup consists of a 6-axis KUKA robotic arm equipped with a custom-built extrusion head. The robotic arm can move and rotate freely in space. (b) The extrusion head continuously extrudes filament when it moves from node ① to ② to ③, and thus two struts are produced in space. The robotic arms at node ① and ② are rendered transparently.*

Extrusion Head The extrusion head H is modelled as a cone with an axis h and a cone angle α (Figure 4(a)). For simplicity, the orientation of H is fixed when it is printing a strut s in our system. The orientation of H is represented as the spherical coordinates (θ, ϕ) ($\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\phi \in [0, 2\pi]$) of h in a local coordinate frame T at the starting node of s, where the unit vector of s is set as z-axis of T and the unit vector of the cross product between s and z-axis of the world coordinate frame as y-axis of T (Figure 4(b)). Note that if s is parallel to z-axis, we switch z and y in the above construction of the frame. To avoid collisions between H and $\mathcal{S}_k^{\mathrm{p}}$, our algorithm finds an appropriate angle pair (θ, ϕ) as the orientation of H to print s.

3.2 Fabrication Constraints

Stiffness Equation When a frame shape G is in a static equilibrium configuration, each node of G suffers certain deflections (deformation) [Kassimali 2011] (see Figure 5(a)). The stiffness equation of G is described as [Hughes 1987; Wang et al. 2013]:

$$\mathbf{K}(\mathbf{N}, r) \cdot \mathbf{d} = \mathbf{f}(r), \tag{1}$$

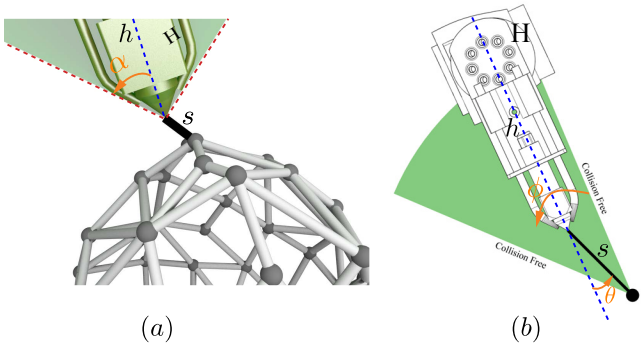


Figure 4: *Illustration of the extrusion head H while printing the strut s. (a) H is modelled as a 3D cone shape with the middle axis h and the cone angle α; (b) The orientation of H is determined by the angle pair (θ, ϕ). The green region is the valid region of feasible orientation of H which is collision-free with the printed part.*

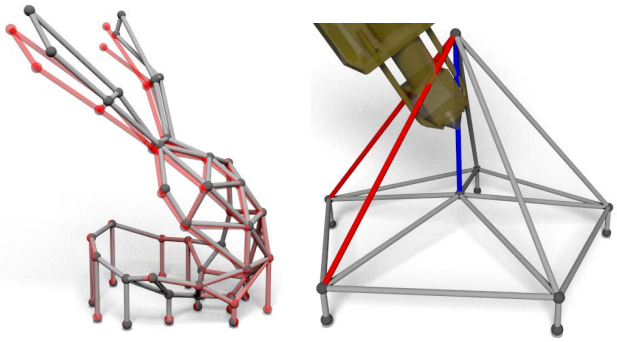


Figure 5: *(a) A frame shape is unstable if the fabricated shape has large deformation (in red) which deviates much from the input (in gray); (b) The extrusion head collides with two printed struts in red when it is printing the blue strut.*

where $\mathbf{K}(\mathbf{N}, r)$ is the stiffness matrix depending on node positions $\mathbf{N}$, the strut radius r, and the material properties including the density d, the Young's modulus γ_y, the shear modulus γ_s, and the Poisson ratio γ_p. $\mathbf{f}(r) = [\mathbf{g}_1, \mathbf{b}_1, \ldots, \mathbf{g}_{|\mathcal{N}|}, \mathbf{b}_{|\mathcal{N}|}]^T$ is the external loads (the gravity force and associated bending moment) acting on the nodes. $\mathbf{d} = [\mathbf{d}_1^t, \mathbf{d}_1^r, \ldots, \mathbf{d}_{|\mathcal{N}|}^t, \mathbf{d}_{|\mathcal{N}|}^r]^T$ is the deformation of the nodes caused by $\mathbf{f}(r)$. Each node n_i is with a 6D deformation vector $\mathbf{d}_i = (\mathbf{d}_i^t, \mathbf{d}_i^r)$ where $\mathbf{d}_i^t$ and $\mathbf{d}_i^r$ are respectively the 3D translation vector and the 3D rotation vector at node n_i. Thus, $\mathbf{d}$ is a $6|\mathcal{N}| \times 1$ column vector, and $\mathbf{K}$ is a $6|\mathcal{N}| \times 6|\mathcal{N}|$ matrix.

Stability Constraints At any fabrication stage, the fabricated frame shape should be in a stable equilibrium state (Figure 5(a)). That is, the deformation $\mathbf{d}_i^t$ of each node should be small enough

$$\|\mathbf{d}_i^t\|_2 < \varepsilon, \quad i = 1, 2, \ldots, |\mathcal{N}|. \tag{2}$$

The tolerance ε is set as $\varepsilon \leq r - \rho$ to guarantee that the extrusion head H can locate the fabricated node.

Collision Constraints At any fabrication stage, the extrusion head H should not collide with $\mathcal{S}_k^{\mathrm{p}}$ while it is printing the current strut s (Figure 5(b)). For a strut s_i in $\mathcal{S}_k^{\mathrm{p}}$, we compute a valid region of angle pairs (θ, ϕ) for s according to s_i, denoted as $\Omega(s, s_i)$, such that the movement of H with some fixed orientation $(\theta, \phi) \in \Omega(s, s_i)$ along s does not collide with s_i. Thus, the valid region of (θ, ϕ) for s according to $\mathcal{S}_k^{\mathrm{p}}$ is computed as:

$$\Omega(s, \mathcal{S}_k^{\mathrm{p}}) = \bigcap_{s_i \in \mathcal{S}_k^{\mathrm{p}}} \Omega(s, s_i). \tag{3}$$

If $\Omega(s, \mathcal{S}_k^{\mathrm{p}}) = \emptyset$, the strut s cannot be printed. Otherwise, our algorithm chooses one appropriate $(\theta, \phi) \in \Omega(s, s_i)$ and sets it as the orientation of H for fabricating s.

4 Method

4.1 Problem Formulation

We first consider a simpler problem of how to choose the next fabrication strut at each stage. At the k-stage, we have already fabricated the frame shape $\mathcal{S}_k^{\mathrm{p}}$. The goal is to find an optimal strut $s \in \mathcal{S}_k^{\mathrm{u}}$ with minimal cost (see below) to be fabricated in the following stage.

Candidates We construct a candidate set of eligible struts $\mathcal{C}_k \subset \mathcal{S}_k^{\mathrm{u}}$ where each eligible strut $s \in \mathcal{C}_k$ satisfies:

- s is connected to $\mathcal{S}_k^{\mathrm{p}}$;
- The frame shape $\mathcal{S}_k^{\mathrm{p}} \cup \{s\}$ is stable;
- $\Omega(s, \mathcal{S}_k^{\mathrm{p}})$ is not empty.

Denote $\bar{\mathcal{S}}_k^{\mathrm{p}} = \mathcal{S}_k^{\mathrm{p}} \cup \{s\}$ and $\bar{\mathcal{S}}_k^{\mathrm{u}} = \mathcal{S}_k^{\mathrm{u}} \setminus \{s\}$. For each strut $s \in \mathcal{C}_k$, we calculate a cost $E_{\mathrm{s}}(s, \mathcal{S}_k^{\mathrm{p}})$ (Equation (7)) for s, which consists of three terms as follows.

Stability Cost To make the new frame shape $\bar{\mathcal{S}}_k^{\mathrm{p}}$ as stable as possible, we are likely to choose the strut s with the minimal value of the following stability cost:

$$E_{\mathrm{s}}(s, \mathcal{S}_k^{\mathrm{p}}) = \frac{1}{\varepsilon} \max_{n_i \in \bar{\mathcal{S}}_k^{\mathrm{p}}} \|\mathbf{d}_i^t(\bar{\mathcal{S}}_k^{\mathrm{p}})\|_2. \tag{4}$$

where $\frac{1}{\varepsilon}$ is applied for normalization.

Collision Cost Although s is printable, it might cause some remaining unprinted struts in $\bar{\mathcal{S}}_k^{\mathrm{u}}$ to have no feasible orientation for H in the following stage. Thus, we prefer to choose the strut s with the minimal value of the following collision cost:

$$E_{\mathrm{c}}(s, \mathcal{S}_k^{\mathrm{p}}) = \frac{1}{|\bar{\mathcal{S}}_k^{\mathrm{u}}|} \sum_{\bar{s} \in \bar{\mathcal{S}}_k^{\mathrm{u}}} \exp(-B^2(\Omega(\bar{s}, \bar{\mathcal{S}}_k^{\mathrm{p}}))), \tag{5}$$

where $B(\Omega) = \frac{A(\Omega)}{2\pi}$ and $A(\Omega)$ is the area of Ω on the unit sphere. A small value of $E_{\mathrm{c}}(s, \mathcal{S}_k^{\mathrm{p}})$ indicates that the remaining unprinted struts still have a wide range of orientations after s is printed.

Proximity Cost A strut s connecting to the last strut s_{i_k} in $\mathcal{S}_k^{\mathrm{p}}$ is preferred as the extrusion head H continuously moves to print s after it completes printing s_{i_k} for time saving purpose. The proximity cost is defined as:

$$E_{\mathrm{p}}(s, \mathcal{S}_k^{\mathrm{p}}) = \begin{cases} 0, & s \text{ shares node with } s_{i_k}, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Overall Cost Thus we have a cost function as

$$\min_{s\in\mathcal{C}} E(s,\mathcal{S}_k^{\mathrm{p}}) = \omega_{\mathrm{s}}E_{\mathrm{s}}(s,\mathcal{S}_k^{\mathrm{p}}) + \omega_{\mathrm{c}}E_{\mathrm{c}}(s,\mathcal{S}_k^{\mathrm{p}}) + \omega_{\mathrm{p}}E_{\mathrm{p}}(s,\mathcal{S}_k^{\mathrm{p}}), \tag{7}$$

where $\omega_{\mathrm{s}}, \omega_{\mathrm{c}}$, and ω_{p} are weights for the terms.

Global Optimization Our problem is essentially a strut fabrication sequence design problem with stability and collision free hard constraints. The problem can be formulated in the following way. Given the frame shape and its strut set $\mathcal{S} = \{s_i, i = 1, 2, \ldots, |\mathcal{S}|\}$, the goal is to find a permutation of strut set $\mathcal{S}$, such that the fabrication in the order of the optimized permutation is stable and collision free. This problem is formulated as:

$$\min_{\{i_1,\cdots,i_{|\mathcal{S}|}\}} \sum_{k=1}^{|\mathcal{S}|-1} E(s,\mathcal{S}_k^{\mathrm{p}}), \tag{8}$$

where $\{i_1, \cdots, i_{|\mathcal{S}|}\}$ is a permutation of $\{1, \cdots, |\mathcal{S}|\}$. The cost function $E(s, \mathcal{S}_k^{\mathrm{p}})$ is defined in Equation (7).

4.2 Sequence Searching

Challenges The optimization in (8) is substantially difficult in two aspects. First, the objective energy function is sequence-dependent and thus can hardly have the explicit formulation. Second, the stability constraints and collision constraints are coupled together, making the sequence searching problem even more complex. The complexity of the brute-force search method for (8) is in exponential time, and thus it is impossible to directly solve it especially when the number of struts is large. We propose an efficient local optimization method for searching a valid fabrication sequence with a backtracking strategy.

Local Optimization If $\mathcal{C}_k \neq \emptyset$, the strut $s \in \mathcal{C}_k$ with the smallest value of $E(s, \mathcal{S}_k^{\mathrm{p}})$ in (7) is then set as $s_{i_{k+1}} = s$ and added to the fabrication sequence as $\mathcal{S}_{k+1}^{\mathrm{p}}$. We update the set $\mathcal{C}_k = \mathcal{C}_k \setminus \{s\}$.

If $\mathcal{C}_k = \emptyset$, this indicates that there is no feasible strut in $\mathcal{S}_k^{\mathrm{u}}$ that can be printed, which means that the current fabrication sequence is not reasonable. Therefore, we remove s_{i_k} from $\mathcal{S}_k^{\mathrm{p}}$, and backtrack to the previous stage $\mathcal{S}_{k-1}^{\mathrm{p}}$ searching another eligible strut in $\mathcal{C}_{k-1}$. These steps are repeated until we find a feasible fabrication sequence or it terminates without any solution. The pseudo code for the sequence-searching algorithm is shown in Algorithm 1.

Algorithm 1: SequenceSearching(k, s_{i_k}, G)

```
S_k^p = S_{k-1}^p ∪ s_{i_k}, S_k^u = S_{k-1}^u \ s_{i_k};
Create candidate strut set C_k
while C_k ≠ ∅ do
    Set the strut s with minimum cost E(s, S_k^p) as s_{i_{k+1}};
    if SequenceDesign(k + 1, s_{i_{k+1}}, G) = True then
        Print strut s_{i_{k+1}};
        Return True;
    else
        C_k = C_k \ s;
if C_k ≠ ∅ then
    Return True;
else
    Return False;
```

The first strut of the fabrication sequence s_{i_1} is selected from the struts which touch the floor or connect with the last already fabricated strut in last layer by optimizing (7), and then the fabrication can be generated by recursively calling the same function. Thanks to the backtracking strategy, Algorithm 1 can search all possible fabrication sequences, therefore, the proposed algorithm could return one valid fabrication sequence if the input frame shape has valid fabrication sequences. The efficiency of Algorithm 1 is guaranteed as it always selects the "best" strut from the candidate set and add it into the fabrication sequence based on the fabrication cost defined in (7). The algorithm will return "False" if the input frame shape has no valid fabrication sequence.

4.3 Layer Decomposition

Algorithm 1 can efficiently search for a feasible fabrication sequence for a given frame shape, which is much faster than the brute-force searching scheme. However, it may still take quite a long time to find a feasible fabrication sequence for frame shape with large numbers of struts.

Sub-layers To further improve the efficiency, we develop a divide-and-conquer approach to decompose the input frame shape G into a few sub-parts, such that each sub-part does not have many struts and thus a feasible sequence can be easily searched for it using Algorithm 1. Specifically, the strut set $\mathcal{S}$ is decomposed into two layers $\mathcal{S} = \mathcal{B} \cup \mathcal{T}$ each time, where $\mathcal{T}$ is the top layer and $\mathcal{B}$ is the bottom layer ($\mathcal{B}$ is fabricated before $\mathcal{T}$), which satisfies

- The number of struts of $\mathcal{T}$ is small enough so that Algorithm 1 can be efficiently applied on it;
- $\mathcal{B}$ is in a stable equilibrium state so that it can be physically fabricated;
- The collision constraint from $\mathcal{B}$ to $\mathcal{T}$ should be small while the collision constraint from $\mathcal{T}$ to $\mathcal{B}$ could be large, so that the set $\mathcal{B}$ should be fabricated before $\mathcal{T}$.

Then the decomposition scheme is recursively applied on the bottom part $\mathcal{B}$ until the termination criteria is satisfied.

Cost of Cut The decomposition problem can be cast into finding a cut on $\mathcal{S}$. Considering two struts s_i and s_j in $\mathcal{S}$. If s_i has little collision constraint over s_j while s_j has large collision constraint over s_i, it is more reasonable to fabricate s_i before s_j. Otherwise, chance is less to fabricate s_i because of the large collision constraint from s_j. Therefore, s_i is more likely to be classified into $\mathcal{B}$ while s_j is more likely to be classified into $\mathcal{T}$. Based on this observation, we define a cost between a pair of struts s_i and s_j as:

$$w(s_i,s_j) = \begin{cases} c(s_i,s_j)h(s_i,s_j), & \text{if } s_i \text{ and } s_j \text{ share a node,} \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $c(s_i, s_j)$ and $h(s_i, s_j)$ are defined as: $c(s_i, s_j) = \exp(-\lambda_1(\Omega(s_i, s_j) - \Omega(s_j, s_i))_+^2)$, where the function $(x)_+$ is defined as $(x)_+ = \max(0, x)$, which is commonly used as the rectified linear unit activation function in deep learning community; $h(s_i, s_j) = \exp(-\lambda_2 \bar{h}^2(n_{i,j}))$, where $n_{i,j}$ is the common node which s_i and s_j share, $\bar{h}(n_{i,j})$ is the normalized height of node $n_{i,j}$, and λ_1, λ_2 are weights.

Accordingly, the cut cost between $\mathcal{B}$ and $\mathcal{T}$ is defined as:

$$w(\mathcal{B},\mathcal{T}) = \sum_{s_i\in\mathcal{B}, s_j\in\mathcal{T}} w(s_i,s_j). \tag{10}$$

Optimization Model Therefore, the decomposition problem is formulated as:

$$\begin{aligned} \min_{\mathcal{B},\mathcal{T}} \ & w(\mathcal{B},\mathcal{T}) \\ \text{s.t. } & \mathbf{K}(\mathcal{B})\mathbf{d}(\mathcal{B}) = \mathbf{f}(B), \\ & \|\mathbf{d}_i^t(\mathcal{B})\|_2 < \varepsilon, \\ & S^t \subset \mathcal{T}, \\ & S^b \subset \mathcal{B}, \end{aligned} \tag{11}$$

where $\mathcal{S}^t$ and $\mathcal{S}^b$ are respectively the top and bottom boundary struts of $\mathcal{S}$.

Relaxed The optimization (11) is actually a constrained discrete minimum cut problem with $\mathcal{S}^t$ and $\mathcal{S}^b$ as the boundary constraints. The classic max-flow min-cut algorithm for network flow like graph cut methods [Boykov et al. 2001; Kolmogorov and Zabih 2004] cannot be applied on it due to the nonlinear constraints. To solve this problem, we first relax the binary optimization problem (11) into a continuous optimization problem as:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{d}} & \sum w(s_i,s_j)(x_i - x_j)_+ \\ \text{s.t. } & \mathbf{K}(\mathbf{x})\mathbf{d} = \mathbf{f}(\mathbf{x}), \\ & \|\mathbf{d}_i^t(\mathcal{B})\|_2 < \varepsilon, \\ & x_i = 0, \qquad \text{if } s_i \in \mathcal{S}^t, \\ & x_i = 1, \qquad \text{if } s_i \in \mathcal{S}^b, \\ & 0 \le x_i \le 1, \end{aligned} \tag{12}$$

where $\mathbf{x} = [x_1, x_2, \cdots, x_{|\mathcal{S}|}]$ with x_i as the label of strut s_i. Note that $w(s_i,s_j) \neq w(s_j,s_i)$. The label variable $\mathbf{x}$ is plugged into the stiffness matrix as a weighting parameter for each element of the stiffness matrix. This eliminates the influence of nodes that are not in the bottom set. After solving the minimization problem, strut s_i with $x_i \geq 0.5$ is classified into $\mathcal{B}$, otherwise, it is classified into $\mathcal{T}$.

Reweighting Scheme The objective energy of Equation (12) is non-differentiable and the first constraint is nonlinear. To solve the constrained sparse optimization problem (12), we adopt the iterative reweighting method [Lai et al. 2013] and each reweighting sub-problem can be expressed as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{d}} & \sum \frac{1}{\gamma_{ij}^{iter}} w^2(s_i,s_j)(x_i - x_j)_+^2 \\ \text{s.t. } & \mathbf{K}(\mathbf{x})\mathbf{d} = \mathbf{f}(\mathbf{x}), \\ & \|\mathbf{d}_i^t(\mathcal{B})\|_2 < \varepsilon, \\ & x_i = 0, \qquad \text{if } s_i \in \mathcal{S}^t, \\ & x_i = 1, \qquad \text{if } s_i \in \mathcal{S}^b, \\ & 0 \le x_i \le 1, \end{aligned} \tag{13}$$

where γ_{ij}^{iter} is updated in each reweighting iteration according to the $\mathbf{x}$ value of the previous iteration and it is set as

$$\gamma_{ij}^{iter} = \tau + w(s_i,s_j)(x_i^{iter} - x_j^{iter})_+, \tag{14}$$

where τ is set to be a small positive value to avoid division by zero. Figure 6 shows an example of how the reweighting scheme works for (12). Given the frame shape, the weight setting, and the boundary conditions, the $\mathbf{x}$ values equally distribute in the whole range $[0,1]$ at the first reweighting iterations. As the iteration number increases, the $\mathbf{x}$ values concentrate more closely to 0 and 1. Besides, the objective energy monotonously decreases with more iterations. With the convergence of the reweighting scheme, the first decomposition result is given in the right in Figure 6.

Algorithm 2: LayerDecomposition(G)

if $|\mathcal{S}| > \delta$ **then**
  Define the weight w_{ij} according to Equation (9);
  Construct stiffness matrix $\mathbf{K}$;
  Set $\mathbf{x}$'s value for boundary set $\mathcal{S}^t, \mathcal{S}^b$;
  $iter = 0$;
  while *Reweighting termination criteria = False* **do**
    $k = 0$;
    while *ADMM termination criteria = False* **do**
      $\mathbf{x}^{k+1} = \text{argmin}_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}^k, \mathbf{d}^k, \boldsymbol{\lambda}_1^k, \boldsymbol{\lambda}_2^k)$;
      $\mathbf{y}^{k+1} = \text{argmin}_{\mathbf{y}} L(\mathbf{x}^{k+1}, \mathbf{y}, \mathbf{d}^k, \boldsymbol{\lambda}_1^k, \boldsymbol{\lambda}_2^k)$;
      $\mathbf{d}^{k+1} = \text{argmin}_{\mathbf{d}} L(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{d}, \boldsymbol{\lambda}_1^k, \boldsymbol{\lambda}_2^k)$;
      $\boldsymbol{\lambda}_1^{k+1} = \boldsymbol{\lambda}_1^k + \mu(\mathbf{y}^{k+1} - \mathbf{E}\mathbf{x}^{k+1})$;
      $\boldsymbol{\lambda}_2^{k+1} = \boldsymbol{\lambda}_2^k + \mu(\mathbf{K}(\mathbf{x}^{k+1})\mathbf{d}^{k+1} - \mathbf{f}(\mathbf{x}^{k+1}))$;
      $k = k + 1$;
    Update γ_{ij}^{iter} according to Eq (14);
    $iter = iter + 1$;
  if *Solution* $\mathbf{x}$ *is feasible* **then**
    Cut $\mathcal{S}$ into set $\mathcal{B}$ and $\mathcal{T}$ according to $\mathbf{x}$ value;
  else
    Return False;

Numerical Algorithm To solve (13), we introduce a new variable y_{ij} to replace $x_i - x_j$ in the objective energy function as $y_{ij} = x_i - x_j$. $\mathbf{y} = [\cdots, y_{ij}, \cdots]$ is the vector representation of y_{ij}. By introducing the strut incidence matrix $\mathbf{E}$ as:

$$e_{pq} = \begin{cases} +1, & \text{if the first index of } p\text{-th element of } \mathbf{y} \text{ is } q, \\ -1, & \text{if the second index of } p\text{-th element of } \mathbf{y} \text{ is } q, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

We apply the augmented Lagrangian method to (13) and get:

$$\begin{aligned} & L(\mathbf{x},\mathbf{y},\mathbf{d},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2) \\ & = \mathbf{y}_+^T\mathbf{D}\mathbf{y}_+ + \boldsymbol{\lambda}_1^T(\mathbf{y} - \mathbf{E}\mathbf{x}) + \boldsymbol{\lambda}_2^T(\mathbf{K}(\mathbf{x})\mathbf{d} - \mathbf{f}(\mathbf{x})) \\ & + \frac{\mu}{2}(\|\mathbf{y} - \mathbf{E}\mathbf{x}\|_2^2 + \|\mathbf{K}(\mathbf{x})\mathbf{d} - \mathbf{f}(\mathbf{x})\|_2^2), \end{aligned}$$

where $\mathbf{D}$ is a diagonal matrix with its diagonal element as

$$d_{kk} = \frac{w^2(s_i,s_j)}{\gamma_{ij}^{iter}}. \tag{16}$$

Therefore, (13) is reformulated as the following form:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y},\mathbf{d},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2} & L(\mathbf{x},\mathbf{y},\mathbf{d},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2) \\ \text{s.t. } & \|\mathbf{d}_i^t(\mathcal{B})\|_2 < \varepsilon, \\ & x_i = 0, \qquad s_i \in \mathcal{S}^t, \\ & x_i = 1, \qquad s_i \in \mathcal{S}^b, \\ & 0 \le x_i \le 1. \end{aligned} \tag{17}$$

We adopt the Alternating Direction Method of Multipliers (ADMM) [Boyd et al. 2011] to solve this problem by decomposing it into several sub-problems, as shown in Algorithm 2. The $\mathbf{x}$-subproblem and $\mathbf{d}$-subproblem are solved via the conic solver in the

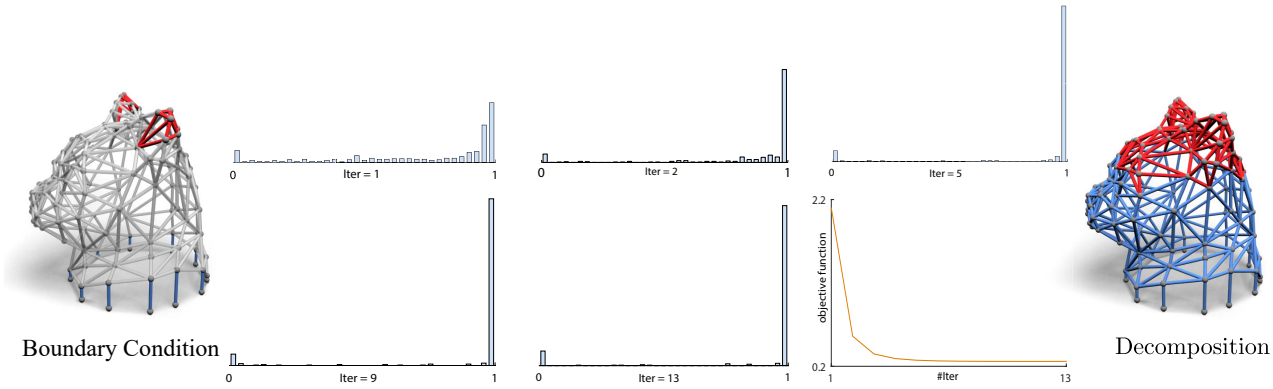


Figure 6: *The left shows the input frame shape with top and bottom boundary struts indicated with different colors. The middle part shows how the distributions of* $\mathbf{x}$ *values and the cut energy values change relative to the iteration number of the reweighting scheme* (13)*. The figure on the right shows the decomposition result of model* (11)*.*

MOSEK optimization library [Mosek 2015]. The $\mathbf{y}$-subproblem is separable, and each one has a closed form solution. If the algorithm finds a valid decomposition such that $\mathcal{B}$ and $\mathcal{T}$ satisfy the hard constraints in (11), it returns true. Otherwise, it means that there is no valid decomposition, and then returns false.

4.4 Algorithm Pipeline

We propose a divide-and-conquer strategy and Algorithm 3 shows the psudeo-code. The first strut s_{i_1} of each layer G_l is chosen from the struts that touch the floor (the bottom layer) or connect with the last fabricated strut in layer G_{l-1} by optimizing (7). Given a frame shape, it is first decomposed into several layers according to Algorithm 2. These parts are fabricated one by one from bottom to top. For each layer, the fabrication sequence is designed by Algorithm 1. The algorithm will return "False" if the sequence design algorithm fails to find a valid fabrication sequence for some layer, which means that our algorithm cannot find a feasible fabrication sequence for the input frame shape. During the whole fabrication process, the deformation for each node is guaranteed to be smaller than the given tolerance. Therefore, the fabricated part is always in a stable equilibrium state, and the extrusion head can locate the node in the following steps.

Algorithm 3: Fabrication sequence design for G

```
G is decomposed into layers of G_1, ··· , G_L by iteratively
  applying Algorithm 2;
for each layer G_l do
    if SequenceDesign(1, s_{i_1}, G_l) = False then
        Return False;
Return True;
```

5 Results

We implemented our algorithm in C++ and tested it on various frame shapes. All experiments were performed on a desktop PC with a quad-core Intel CPU i7 and 4GB RAM. The C++ source code of our implementation is available online.[1]

[1] https://github.com/Juyong/FrameFab

Parameters The material used in our experimental robotic system is modified thermoplastic polymer ABS mixed with certain amount of carbon powder with the density $d = 1210\mathrm{kg/m}^3$, the Young's modulus $\gamma_y = 3457\mathrm{MPa}$ (Mega Pascals), the shear modulus $\gamma_s = 1294\mathrm{MPa}$, and the Poisson ratio $\gamma_p = 0.335$. The location precision of the robotic arm is $\rho = 0.1\mathrm{mm}$. The cone angle of the extrusion head is $\alpha = \pi/4$. The radius of the extruded filament is $r = 0.75\mathrm{mm}$.

The parameters used in the numerical optimization are $\varepsilon = 0.65\mathrm{mm}$ (Equation (2)), $\lambda_1 = 0.5, \lambda_2 = 3$ (Equation (9)), $\tau = 10^{-5}$ (Equation (14)), $\mu = 100$ (Equation (17)), $\delta = 20$ (Algorithm 2), and $\omega_\mathrm{s} = 1, \omega_\mathrm{c} = 5, \omega_\mathrm{p} = 1$ (Equation (7)).

5.1 Results and Performance

Figure 7 shows an example of bunny frame shape (a). The shape is decomposed into 8 layers by applying Algorithm 2 as shown in (b). The simulated fabricated results of 3 intermediate states are shown in (c-e), respectively, with node deformation color-coded. The real fabrication results with snapped photographs are shown in Figure 1 (b-d) where it is seen that the intermediate fabricated parts are all in stable equilibrium states. More snapped photographs of the fabrication process are shown in Figure 8 and the live fabrication process can be seen in the accompanying video.

Performance of Our Algorithm From our experiments, Algorithm 1 can be directly applied to frame shapes in acceptable computation time for two cases. First, there are only a small number of struts in the input frame. However, for large size frame shapes with about 200 struts, such as the Bunny (Figure 1), the C-shape (Figure 11), and the Beetles (Figure 12, right), it takes more than 7 hours to compute the sequences by applying Algorithm 1 only, as shown in Table 1. With our Algorithm 3, it takes only a few seconds to compute the layer decomposition (with 8-9 layers) by applying Algorithm 2 and the computation of all fabrication sequences can be completed in 1 minute by applying Algorithm 1 (Table 1). For larger frame shape like the Fertility (Figure 9, left), it takes more than one day to compute the sequence by applying Algorithm 1, which is not practically acceptable, while it takes only 16 minutes to compute the sequence by applying Algorithm 3. The decomposed layers are labeled with different colors, as shown in Figure 9 (left) and the intermediate sequence searching results are shown in Figure 9 (right) with strut order color-coded.

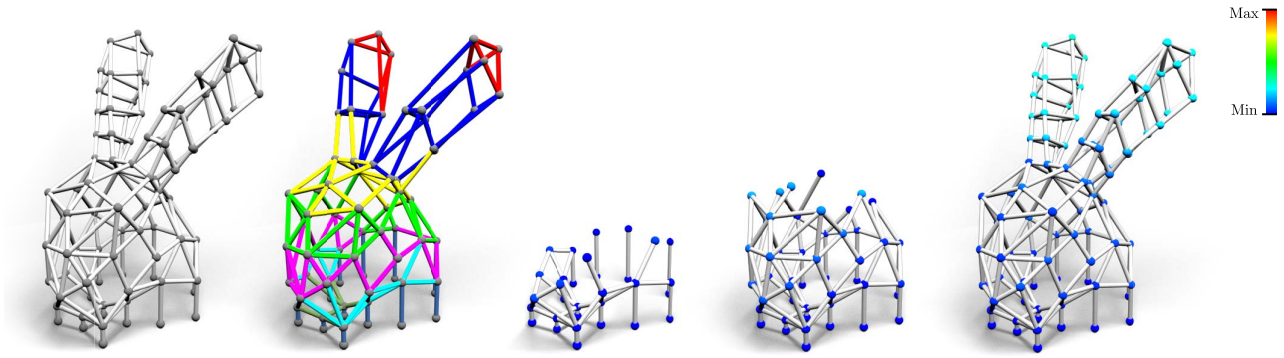


Figure 7: *Illustration of our algorithm. An input frame shape (a) is decomposed into 8 layers by applying Algorithm 2. Then the layers are fabricated from the bottom to the top by applying Algorithm 1. The simulated three intermediate fabricated states are shown in (c), (d) and (e) respectively, with node deformation color-coded. The real fabrication results with snapped photos are shown in Figure 1.*

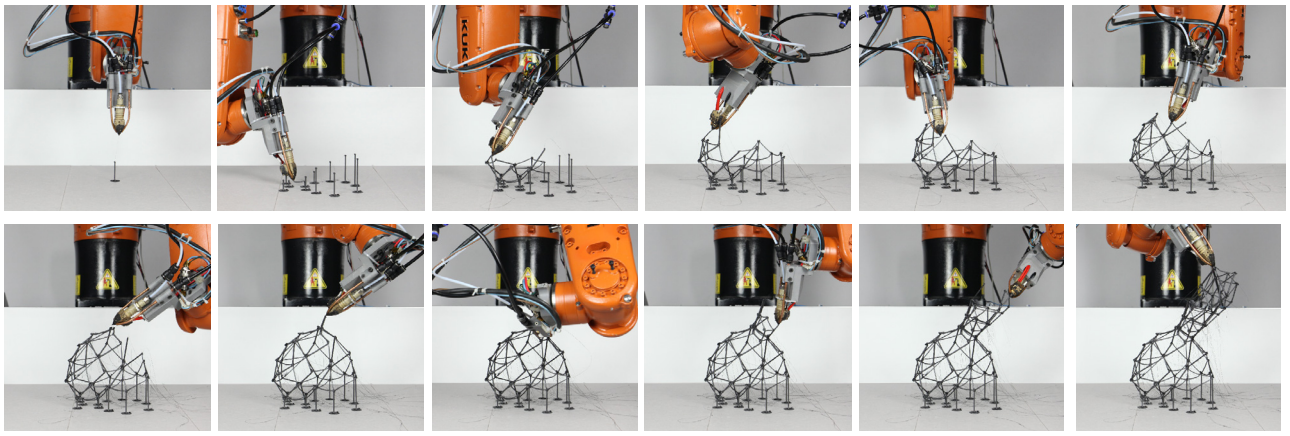


Figure 8: *The bunny frame shape is fabricated using the optimized fabrication sequence produced by our algorithm. The photos show the intermediate state of the whole fabrication process.*

Second, the input frame has simple structure with regular patterns like lattice. For example, the Bottle (Figure 12, left) has a very regular lattice pattern. It takes only one and half minutes to find a reasonable sequence by using Algorithm 1. This is because the backtracking step seldom happens in the searching of the sequence in this case.

Number of Decomposition Layers It is a trade-off to choose the number of layers in the decomposition phase. On one hand, the sequence searching phase (Algorithm 1) costs much more than the layer decomposition phase (Algorithm 2) and it is expected to decompose the input frame shape into more smaller layers. On the other hand, the layer decomposition will reduce the space of feasible fabrication sequence in general, which might result no feasible sequence solution for some layers. The parameter δ in Algorithm 2 controls the size of the bottom layer because the bottom layer will not be decomposed if its number of struts is smaller than δ. By experience we set $\delta = 20$ used in Algorithm 2 in our experiments.

5.2 Comparisons

Comparison to the Brute Force Searching Algorithm 1 is actually a heuristic searching scheme with quick pruning, favoring the exploration of good parts of the solution space. It is indeed much more efficient than the brute force searching scheme. Take the frame shape in Figure 10 (a) as an example, Algorithm 1 finds a feasible fabrication sequence, which is color-coded in Figure 10 (b), in only 27 seconds. However, it takes over 12 hours for the brute force scheme to find a feasible fabrication sequence. The reason is that our algorithm takes advantage of the order constraints that the three red struts must be fabricated before the four green struts shown in Figure 10 (a). As Algorithm 1 is heuristic, it may take exponential time to find a valid solution in the worst case. However, we do not encounter such cases in all our experiments.

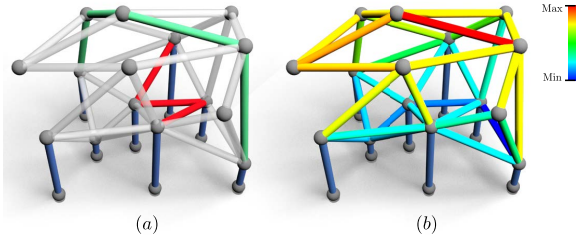


Figure 10: *Algorithm 1 is much more efficient than the brute force searching scheme. (a) To avoid the collision constraint between the extrusion head and the printed part, the red struts must be fabricated before the green struts. (b) The valid fabrication sequence generated by our method.*

Model	$\|\mathcal{N}\|$	$\|\mathcal{S}\|$	Time (Alg. 1 only)	#Layers	Time (Alg. 2)	Time (Alg. 1)	Fab. Time	Size
Bunny (Figure 1)	91	179	>7 h	8	3.02 s	35.22 s	91 min	371 mm
C-shape (Figure 11)	77	199	>7 h	9	3.71 s	36.42 s	102 min	277 mm
Bottle (Figure 12)	90	230	84.94 s	9	4.45 s	71.06 s	120 min	405 mm
Beetles (Figure 12)	134	276	>7 h	8	6.35 s	60.46 s	185 min	620 mm
Fertility (Figure 9)	316	911	>24 h	10	140.09 s	829.11 s	-	-

Table 1: *The statistics of the experimental results. $|\mathcal{N}|$ and $|\mathcal{S}|$ denote the number of nodes and struts respectively. The computation time by directly applying Algorithm 1 is shown in the 4-th column. The statistics by applying our Algorithm 3 are shown in the 5-th to 7-th columns: number of layers (5-th column), computation time of applying Algorithm 2 (6-th column), and computation time of applying Algorithm 1 (7-th column). Column 8 shows the fabrication time and column 9 shows the size, measured by the diagonal length of printed object's bounding box. "-" means that the model is not fabricated.*

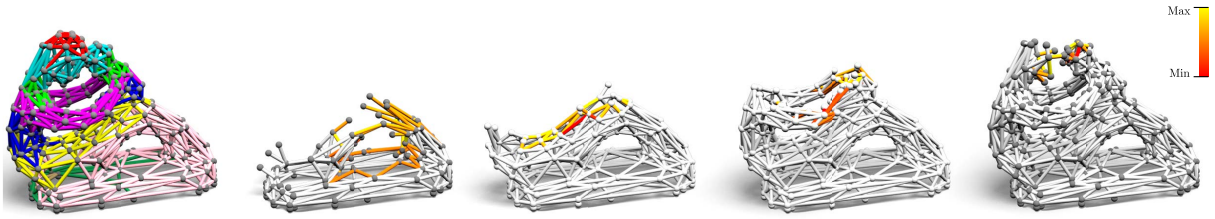


Figure 9: *The simulation process of a complex frame shape Fertility. The left shows the decomposition result, and the right four figures show the intermediate sequence searching stages.*

Comparison to the Naïve Sweeping Decomposition Method A straightforward idea on decomposing the input shape into smaller scales of layers is to continuously move a sweeping plane from bottom to top and intersect with the shape. The struts between two adjacent planes are classified into one layer. As this method does not consider the stability of the layers, the obtained layers might be deformed too much which makes the fabrication fail. Let us see an example of a C-shape model shown in Figure 11. The result of using this naive sweeping decomposition method is shown in the upper row: the layers are shown in different colors in the left and the simulated fabricated shape is shown in red in the right (the original shape is shown in gray). It is seen that the deformation is too much to be printed. The decomposed layers obtained by our algorithm is shown in the lower row: the layers are shown in different colors in the left and the photo of the real fabricated object is shown in the right.

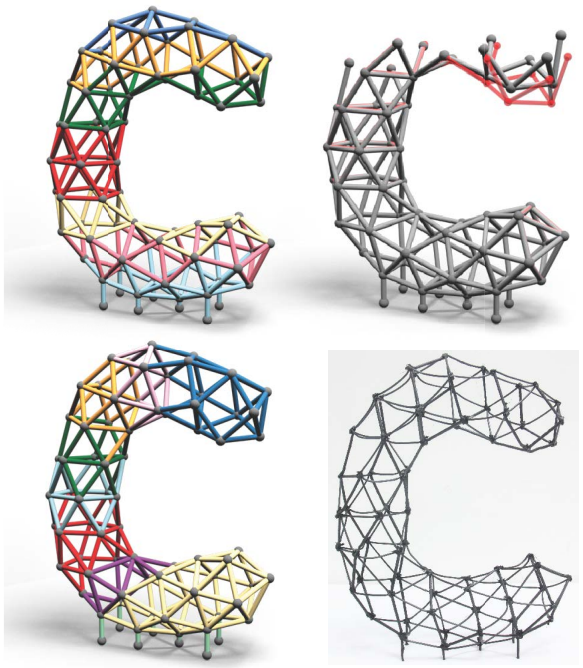


Figure 11: *Comparison to the naïve sweeping decomposition method. The first row shows the decomposition layers in colors by the naïve method (left) and the simulated fabricated shape in red (right) (original shape in gray). The second row shows the decomposition result by our algorithm and the real fabrication result.*

Comparison to WirePrint WirePrint [Mueller et al. 2014] employs standard FDM 3D printer to fabricate wireframes. However, the input wireframe is generated with a zigzag pattern and WirePrint constructs it by moving the extrusion head up and down repeatedly. For frame shapes with regular pattern, our Algorithm 1 generates similar fabrication sequences to WirePrint, as shown in the accompanying video. See Figure 12 (left) for the fabricated bottle frame shape with regular lattice pattern produced by our system. However, for frame shapes with irregular pattern as shown in Figure 12 (right) and the other examples shown in the paper, WirePrint cannot handle with them while our algorithm can generate feasible sequences and fabricate them. It is worthwhile mentioning that much of the benefit gained by our system is credited to the extra freedom of motion of the robotic arm.

Comparison to [Wu et al. 2016] The concurrent work of Wu et al. [2016] presents a collision avoidance algorithm to generate fabrication sequences for general frame shapes as well. Our work differs with this work in both hardware setup and algorithm design. First, the 5DOF frame printer, called the On-the-Fly Print System [Peng et al. 2016], used in [Wu et al. 2016] is based on a standard delta 3D printer by adding two rotation axes in the printing platform. Thus the printed object has to be rotated during the printing process. Instead we use a 6DOF fabrication system with a 6 axis robotic arm and a customized extrusion head [Yu et al. 2016]. The robotic arm draws the spatial strokes freely and the printed object is kept still during the printing process, which is more practically applicable when creating large scale frame shapes such as architec-

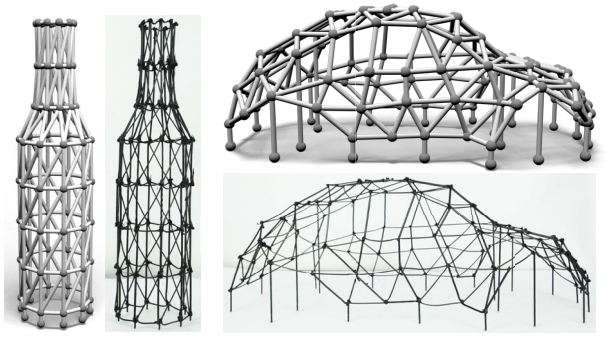


Figure 12: *Fabrication results using our system. Left: the bottle model has regular lattice pattern. Our algorithm generates similar zig-zag fabrication path as WirePrint does; Right: the beetle model has irregular pattern and thus WirePrint cannot handle with it.*

tural frames over the ground. Second, the algorithm of [Wu et al. 2016] considers only collision avoidance problem in generating the fabrication sequence and formalizes it using a directed graph. Our algorithm considers both collision avoidance and the stability of the printed part during the whole fabrication process by using a local optimization together with a backtracking strategy to search the fabrication sequence. A divide-and-conquer scheme is proposed to decompose the input frame into stable layers so that the fabrication sequence can be quickly found for each layer.

5.3 Discussions and Limitations

Stability Constraints It is worthwhile mentioning that the stability constraints in Equation (2) has to be considered in our algorithm. Unlike [Wu et al. 2016] where the struts are extruded vertically, our robotic fabrication system may extrude struts at any direction and thus accurate location of the extrusion head with spatial points is extremely important in the algorithm design. Removing stability constraints in the algorithm will result in collapsed struts and thus fail to fabricate the objects.

Global Collision The moving part of the robotic fabrication system consists of two parts: the robotic arm and the extrusion head. In our implementation, we model the extrusion head as a cone shape and detect the collision between the cone shape and the printed shape. However, the robotic arm is composed of a series of arm segments, which are connected and rotate at joints, and is modeled as a dynamic piecewise rigid object when it moves. As the configurations (positions and orientations) of arm segments are implicitly determined by the position and orientation of the extrusion head via an inverse kinematic solution, it is extremely difficult to consider the collision detection between the robotic arm and the printed shape in our algorithm. As the robotic arm is generally far away from the printed shape, it has much less chance to collide with the printed shape than the extrusion head does. Therefore, we ignore the collision between the robotic arm and the printed shape to simplify the computation in our implementation. Fortunately, we do not observe the collision between the robotic arm and the printed shape in all experiments shown in our paper. We will consider the global collision as future work.

Solution Guarantee Algorithm 1 is guaranteed to find a feasible fabrication sequence if there exists one. Though there does exist some frame whose feasible solution exists but our algorithm with layer decomposition does not find it, our algorithm can fabricate a wide range of frame shapes that previous methods cannot.

Fabrication Quality We built a prototype robotic fabrication system, as shown in Figure 3, to validate our algorithm. All the real examples shown in the paper were fabricated by our system. Actually it is substantially nontrivial to build up such a robotic fabrication system, which involves lots of careful control issues on extrusion heating temperature, filament extrusion speed, cooling air pressure, and robotic arm moving speed, etc. Moreover, many parameters have to be tuned to figure out a stable extrusion of filament. See more detail about in the supplementary material.

Although we have put much effort on making a stable robotic fabrication system, we can see some appearance artifacts in the fabricated objects. First, there are some minor thread on the printed objects as the filament disconnection process is hard to perform when the extrusion head is extruding filament. This is not a big issue as these thread can be easily removed manually afterwards. Second, many fabricated struts present serious sagging due to gravity and the hard control of the cooling temperature and time, like all previous fabrication system [Mueller et al. 2014; Wu et al. 2016]. This will be likely to affect the mechanical properties of the printed object and thus will make the stability analysis and planning inaccurate. However, it is extremely hard to test the material properties of a printed strut and take it into account in the optimization. Fortunately, our experiments have shown that all test frame shapes were successfully fabricated by our system even when there are some sagging struts in the results, which means that the approximated computation is also practically applicable for a wide range of input frame shapes.

6 Conclusion

In this paper, we propose an approach for creating a feasible fabrication sequence for general frame shapes. To solve this challenging combinatorial optimization problem, we develop a divide-and-conquer strategy that first decomposes the input frame shape into stable sub layers. This is followed by an efficient method to search a feasible fabrication sequence for each layer. Our algorithm guarantees that the printed frame shape is in a state of stable equilibrium during the whole fabrication process. To validate our algorithm, we built a robotic fabrication system, which is based on a 6-axis KUKA robotic arm with a customized extrusion head. The system has produced various real frame shapes by applying our algorithm, which has verified our algorithm successfully. We believe that our algorithm will be widely used for fabricating lightweight frame shapes after robot based fabrication systems become commercially available in the future.

Future Work Our research opens many directions for future studies. First, the input frame shape may not be self-supported and/or no feasible fabrication sequence exists at all. It would be interesting to study how to transform the unstable frame shapes into stable ones. One possible idea is to optimize the node positions and/or refine the connectivity of the frame mesh so that the modified shapes are stable. An alternative way is to add extra supporting structures inside or around the input shape.

Second, the robotic arm may self-lock due to the singularity problem and the mechanical constraints, and thus cause the whole process to halt. This always happens when the robotic arm moves in large ranges. To reduce the possibility of self-lock, one idea is to put the fabricated object in a rotation platform along the z-axis so that the fabrication of the struts far from the robot can be easily made by rotating the platform.

Third, improving the hardware setup, and thus making better qual-

ity of fabricated frames, is a promising direction for future work. We will try to build a more powerful robotic system, enabling better collaboration over extrusion speed, printing temperature, and robotic arm moving speed, etc. We also hope that professional roboticists and material scientists will find more suitable printing material and resolve the robotic control issues in the near future.

Acknowledgements

We thank Zishun Liu, Shizhe Zhou, and Zhipei Yan for their insightful discussions and the anonymous reviewers for their valuable comments. We thank third parties to grant the permission of using their pictures: Continuum (Shoes in Figure 2), Dominik Raskin (Lamborghini and Sculpture in Figure 2). This work was supported by the National Key R&D Program of China (No. 2016YFC0800501), the National Natural Science Foundation of China (Nos. 61672481, 61672482, 61303148, 11526212), and the One Hundred Talent Project of the Chinese Academy of Sciences.

References

3DOODLER, 2013. 3Doodler pen. http://the3doodler.com.

AGRAWAL, H., UMAPATHI, U., KOVACS, R., FROHNHOFEN, J., CHEN, H., MÜLLER, S., AND BAUDISCH, P. 2015. Protopiper: Physically sketching room-sized objects at actual scale. In *UIST*, 427–436.

AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. *ACM Trans. Graph. 22*, 3, 828–837.

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning 3*, 1, 1–122.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11, 1222–1239.

BRANCH, 2015. Branch technology. http://www.branch.technology.

DEUSS, M., PANOZZO, D., WHITING, E., LIU, Y., BLOCK, P., SORKINE-HORNUNG, O., AND PAULY, M. 2014. Assembling self-supporting structures. *ACM Trans. Graph. 33*, 6.

DUMAS, J., HERGEL, J., AND LEFEBVRE, S. 2014. Bridging the gap: automated steady scaffoldings for 3D printing. *ACM Trans. Graph. 33*, 4, 98:1–98:10.

FALLACARA, G., AND D'AMATO, C. 2012. *Stereotomy: Stone Architecture and New Research*. Presses des Ponts.

GRAMAZIO, F. 2014. *The Robotic touch: How robots change architecture*. Park Books 2014.

HACK, N., AND LAUER, W. V. 2014. Mesh-mould: Robotically fabricated spatial meshes as reinforced concrete formwork. *Architectural Design 84*, 3, 44–53.

HELM, V., WILLMANN, J., THOMA, A., PIŠKOREC, L., HACK, N., GRAMAZIO, F., AND KOHLER, M. 2015. Iridescence print: Robotically printed lightweight mesh structures. *3D Printing and Additive Manufacturing 2*, 3, 117–122.

HILDEBRAND, K., BICKEL, B., AND ALEXA, M. 2012. crdbrd: Shape fabrication by sliding planar slices. *Computer Graphics Forum 31*, 2pt3, 583–592.

HUGHES, T. J. R. 1987. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall.

KASSIMALI, A. 2011. *Matrix Analysis of Structures SI Version*. Cengage Learning.

KOLMOGOROV, V., AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 2, 147–159.

LAI, M., XU, Y., AND YIN, W. 2013. Improved iteratively reweighted least squares for unconstrained smoothed ℓ_q minimization. *SIAM J. Numerical Analysis 51*, 2, 927–957.

LO, K.-Y., FU, C.-W., AND LI, H. 2009. 3D polyomino puzzle. *ACM Trans. Graph. 28*, 5, 157.

MATAERIAL, 2015. Mataerial. website. http://www.mataerial.com.

MOSEK, A. 2015. The mosek optimization software. *Online at http://www. mosek. com*.

MUELLER, S., IM, S., GUREVICH, S., TEIBRICH, A., PFISTERER, L., GUIMBRETIÈRE, F., AND BAUDISCH, P. 2014. Wireprint: 3D printed previews for fast prototyping. In *UIST*, 273–280.

OXMAN, N., LAUCKS, J., KAYSER, M., TSAI, E., AND FIRSTENBERG, M. 2013. Freeform 3D printing: Towards a sustainable approach to additive manufacturing. *Green Design, Materials and Manufacturing Processes*, 479.

PANETTA, J., ZHOU, Q., MALOMO, L., PIETRONI, N., CIGNONI, P., AND ZORIN, D. 2015. Elastic textures for additive fabrication. *ACM Trans. Graph. 34*, 4, 135.

PENG, H., WU, R., MARSCHNER, S., AND GUIMBRETIÈRE, F. 2016. On-the-fly print: Incremental printing while modelling. In *CHI*, 887–896.

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum 32*, 2, 317–326.

SONG, P., FU, C.-W., AND COHEN-OR, D. 2012. Recursive interlocking puzzles. *ACM Trans. Graph. 31*, 6, 128.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: improving structural strength of 3D printable objects. *ACM Trans. Graph. 31*, 4, 48.

WANG, W., WANG, T. Y., YANG, Z., LIU, L., TONG, X., TONG, W., DENG, J., CHEN, F., AND LIU, X. 2013. Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph. 32*, 6, 177.

WENDLAND, D. 2009. Experimental construction of a free-form shell structure in masonry. *International Journal of Space Structures 24*, 1, 1–11.

WU, R., PENG, H., GUIMBRETIÈRE, F., AND MARSCHNER, S. 2016. Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Trans. Graph. 35*, 4, 101.

YU, L., HUANG, Y., LIU, Z., XIAO, S., LIU, L., SONG, G., AND WANG, Y. 2016. Highly informed robotic 3d printed polygon mesh - a novel strategy of 3D spatial printing. In *The Association for Computer Aided Design in Architecture (ACADIA)*.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Trans. Graph. 32*, 4, 137.