Supplementary Document:
Support-Free Volume Printing by Multi-Axis Motion

CHENGKAI DAI, Delft University of Technology, The Netherlands
CHARLIE C. L. WANG*, Delft University of Technology, The Netherlands
CHENMING WU, Tsinghua University, China
SYLVAIN LEFEBVRE, INRIA, France
GUOXIN FANG, Delft University of Technology, The Netherlands
YONG-JIN LIU, Tsinghua University, China

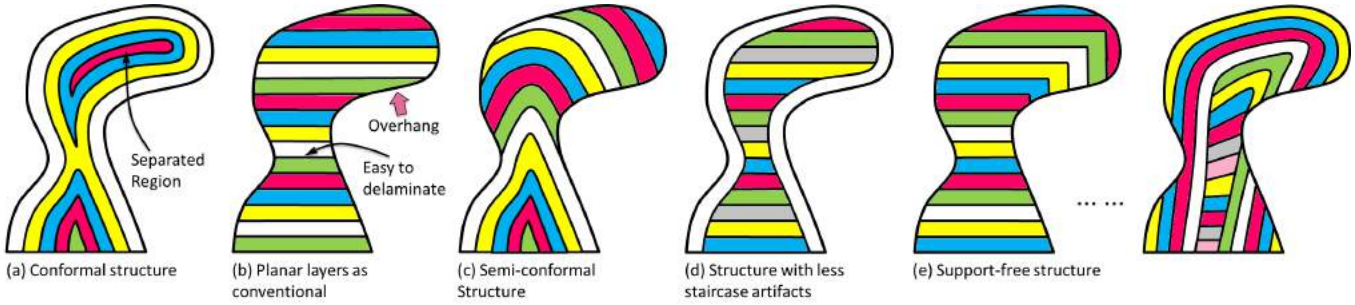

Fig. 1. There are many choices for decomposing a volume into curved layers with nearly uniform thickness for tool-path generation – here different layers are shown in different colors. Among all these listed decompositions, the 'micro'-structure of (a) is conformal to the boundary surface of a given solid. However, separated regions in (a) are impossible to be fabricated even when a multi-axis AM device is used. Alternatively, the process planned in (c) can be physically realized. When printing a model by using the layers in (b), supporting structures need to be added below overhangs. A better appearance with less staircase artifacts might be obtained from the decomposition as shown in (d). With the help of multi-axis motion, supporting structures can be avoided by using the curved layers shown in (e). The purpose of our work presented in this paper is to compute such a feasible solution to fabricate a given solid model by minimal supporting structures.

1 HARDWARE SYSTEM

Physical fabrications using the tool-paths generated by our approach have been taken on a filament-based 3D printing setup – i.e., *Fused Deposition Modeling* (FDM) equipped with a 6DOF UR5 robotic arm for multi-axis motion. As shown in Fig.2, we fix the nozzle of FDM printer head so that good material adhesion can be obtained comparing to the multi-axis AM with a moving nozzle (e.g., [Huang et al. 2016; Wu et al. 2016]). After installing the printer head and the UR5 robotic arm, the relative pose between them is calibrated.

In our hardware system, both the printer head and the UR5 robotic arm are controlled by the *robot operating system* (ROS) so that the communication between them can be synchronized. During the fabrication, the required volume of material is determined by the length of a tool-path, and the speed of material extrusion at the nozzle is determined by the time needed for traveling a given path. All are synchronized by ROS.

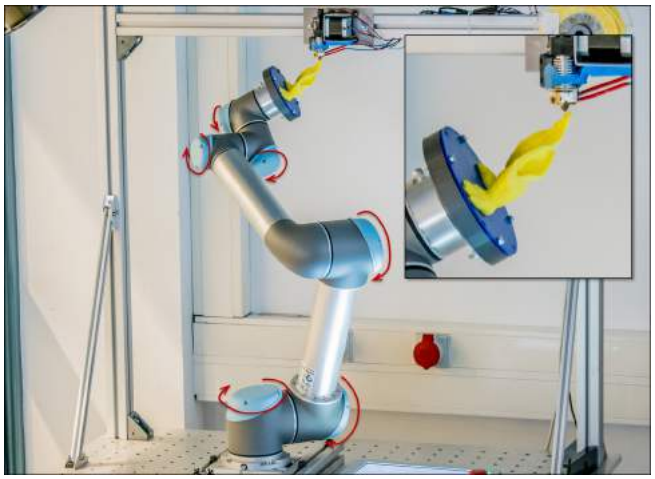

Fig. 2. The hardware of our multi-axis 3D printing system using a 6DOF robotic arm, where each red arrow indicates a motor to provide one rotational DOF. The printer head is fixed in our system so that the orientation change for material accumulation is realized by moving the end effect of the robot arm inversely.

2 PRIMARY GREEDY SCHEME

The pseudo-code for the greedy convex-front advancing as a primary scheme is given in **Algorithm** *GreedySchemeCFA*.

*Corresponding author: c.c.wang@tudelft.nl (Charlie C. L. Wang)

2018. 0730-0301/2018/7-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ALGORITHM 1: GreedyGrowingCFA

Input: Voxel representation of a solid model, $\bar{\mathcal{H}} = \{v_{i,j,k}\}$
Output: A growing field $G(\cdot)$ with value defined on every voxel of $\bar{\mathcal{H}}$

Adding all voxels adjacent to the platform $\mathcal{T}$ to the first layer, $\mathcal{L}_1$, as a set of voxels;
Set $\mathcal{L}_1$ as the current working layer $\mathcal{L}_c$ and $C_{prev} = \emptyset$;
Set the layer index $\tau = 1$;
while $\mathcal{L}_c \neq \emptyset$ **do**
  Add all voxels of $\mathcal{L}_c$ into the already processed set, $\mathcal{V}$;
  Compute the new convex-front by the convex hull of C_{prev}, $\mathcal{L}_c$ and $\mathcal{T}$ as $C_c = C(C_{prev} \cup \mathcal{T} \cup \mathcal{L}_c)$;
  Set $\mathcal{L}_{next} = \emptyset$ and $\tau = \tau + 1$;
  foreach $v_{i,j,k} \in \mathcal{L}_c$ **do**
    foreach $v_{r,s,t} \in \mathcal{N}(v_{i,j,k})$ **do**
      if $v_{r,s,t}$ *NOT inside* C_c **then**
        if $v_{r,s,t} \notin \mathcal{V}$ *AND* $v_{r,s,t} \notin \mathcal{L}_{next}$ **then**
          Add $v_{r,s,t}$ into $\mathcal{L}_{next}$;
        end
      end
    end
  end
  foreach $v_{r,s,t} \in \mathcal{L}_{next}$ **do**
    Assign the field-value as $G(\mathbf{c}(v_{r,s,t})) = \tau$;
  end
  Set $\mathcal{L}_c = \mathcal{L}_{next}$ and $C_{prev} = C_c$;
end

3 GROWING SCHEME WITH SHADOW PREVENTION

The incremental scheme of preserving accessibility can be summarized as the pseudo-code in **Algorithm** *IncrementalShadowPrevention*. The recursive refinement based shadow preservation that has a better efficiency is presented as the pseudo-code in **Algorithm** *AdaptiveRefinementShadowPrevention*.

4 CONVEX-FRONT PEELING

The procedure of peeling is formed by iteratively removing voxels that are ϵ-located on the boundary of convex-hulls containing unprocessed voxels. In this way, the voxels are always removed from the surface of a convex-front – therefore, collision-free is ensured. Again, a greedy strategy is employed here to remove as many as possible voxels from the current convex-front every time. Figure 3(a) illustrates the result of a convex-front peeling. The pseudo-code is given in **Algorithm** *PeelingFieldGeneration*. Note that, greedily peeling all the voxels on the current convex-front may easily lead to a result with isolated components, which are disconnected to the platform of fabrication and will collapse by gravity. However, as the shadow-prevention is conducted in the process of convex-front advancing, the disconnected region can be automatically avoided when computing the growing field (see Fig.3(b)).

5 PLANNING FOR POSE-CONTINUITY

The minimization problem of motion planning in Eq.(1) below is solved on a directed graph $\mathcal{G}_{mo}$ (see Fig.4 for an illustration).

$$\sum_j \|\hat{\mathbf{a}}_j \hat{\mathbf{a}}_{j+1}\|_1 \quad (\exists \hat{\mathbf{a}}_j \in \{\mathbf{a}_{j,k}\}), \tag{1}$$

ALGORITHM 2: IncrementalShadowPrevention

Input: The voxel set of an input model $\bar{\mathcal{H}}$, the set of processed voxels $\mathcal{V}$, the next layer $\mathcal{L}_{next}$ and the current set of shadow region $\mathcal{S}_c$
Output: An reduced set of $\mathcal{L}_{next}$

Set $\mathcal{S}_p = \emptyset$, and $\tilde{\mathcal{L}} = \emptyset$;
Compute $C_p = C(C_c \cup \mathcal{T} \cup \mathcal{L}_{next})$;
$\forall v \in (\bar{\mathcal{H}} \setminus \mathcal{V})$, add v into $\mathcal{S}_p$ if it is *inside* C_p;
if $\mathcal{S}_p = \mathcal{S}_c$ **then**
  Return $\mathcal{L}_{next}$;
end
Determine a heuristic sequence Q of voxels in $\mathcal{L}_{next}$ by a flooding algorithm;
while $Q \neq \emptyset$ **do**
  Remove a voxel v from the head of Q;
  Compute the set of shadowed voxels $\mathcal{S}_t$ according to $C_t = C(C_c \cup \mathcal{T} \cup \tilde{\mathcal{L}} \cup v)$;
  Add v into $\tilde{\mathcal{L}}$ if $\mathcal{S}_t = \mathcal{S}_c$;
end
if $\tilde{\mathcal{L}} \neq \emptyset$ **then**
  Set $\mathcal{L}_{next} = \tilde{\mathcal{L}}$;
else
  Set $\mathcal{S}_c = \mathcal{S}_p$; `// update the set of shadowed voxels`
end
return $\mathcal{L}_{next}$;

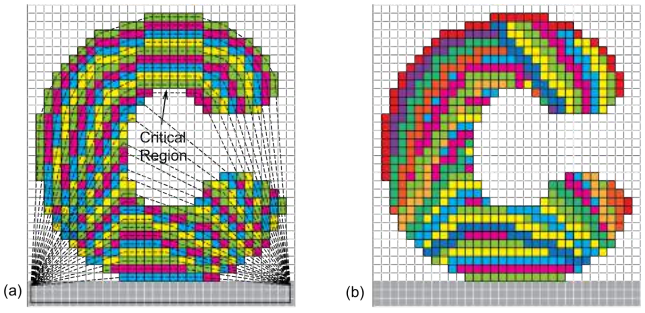

Fig. 3. An illustration of convex-front peeling and the peeling-governed convex-front advancing. (a) Isolated components will be generated when the voxels in the critical region are removed together with other voxels in the same layer. However, such problem on the peeling field $F(\cdot)$ will be automatically avoided when shadow-region preserved convex-front advancing is conducted – see (b) for a result.

Specifically, a node is defined in $\mathcal{G}_{mo}$ for each pose $\mathbf{a}_{j,k}$. For two neighboring sample points, a directed edge pointing from $\mathbf{a}_{j,k}$ to $\mathbf{a}_{j+1,l}$ $(\forall k, l)$ is constructed with the weight $\|\mathbf{a}_{j,k}\mathbf{a}_{j+1,l}\|_1$ when $\|\mathbf{a}_{j,k}\mathbf{a}_{j+1,l}\|_\infty/\Delta t$ is less than ξ – a hardware-dependent threshold (i.e., $\xi = 1$ rad./sec. is required by our hardware). Here the time step is $\Delta t = \|\mathbf{c}_j\mathbf{c}_{j+1}\|/\bar{v}$ with $\bar{v}$ being a setting linear speed of robot movement in the Euclidean space. Using the multi-source Dijkstra's algorithm (using all nodes of $\mathbf{a}_{1,k}$s as sources), the solution of the problem in Eq.(1) can be determined by computing the shortest path linked to one of these sources. $\{\hat{\mathbf{a}}_j\}$ are the nodes of this shortest path, which provides a smooth motion of robotic arm. Note that, it is possible that the graph $\mathcal{G}_{mo}$ is disconnected between $\mathbf{a}_{j,k}$ $(\forall k)$

ALGORITHM 3: AdaptiveRefinementShadowPrevention

Input: The voxel set of an input model $\bar{\mathcal{H}}$, the set of processed voxels $\mathcal{V}$, the next layer $\mathcal{L}_{next}$ and the current set of shadow region $\mathcal{S}_c$
Output: An reduced set of $\mathcal{L}_{next}$

Set $\mathcal{S}_p = \emptyset$, and $\tilde{\mathcal{L}} = \emptyset$;
Compute $C_p = C(C_c \cup \mathcal{T} \cup \mathcal{L}_{next})$;
$\forall v \in (\bar{\mathcal{H}} \setminus \mathcal{V})$, add v into $\mathcal{S}_p$ if it is *inside* C_p;
if $\mathcal{S}_p = \mathcal{S}_c$ **then**
  Return $\mathcal{L}_{next}$;
end
Call *Refinement*($\tilde{\mathcal{L}}$,$\mathcal{L}_{next}$);
if $\tilde{\mathcal{L}} \neq \emptyset$ **then**
  Set $\mathcal{L}_{next} = \tilde{\mathcal{L}}$;
else
  Set $\mathcal{S}_c = \mathcal{S}_p$; `// update the set of shadow points`
end
return $\mathcal{L}_{next}$;

`/* The recursive function for adaptive refinement */`

Procedure `Refinement(`$\tilde{\mathcal{L}}$,$\mathcal{L}_{sub}$`)`
  Compute the set of shadow points $\mathcal{S}_t$ according to $C_t = C(C_c \cup \mathcal{T} \cup \tilde{\mathcal{L}} \cup \mathcal{L}_{sub})$;
  if $\mathcal{S}_t \neq \mathcal{S}_c$ **then**
    if $|\mathcal{L}_{sub}| > 1$ `// Set` $\mathcal{L}_{sub}$ `has more than one voxel` **then**
      Divide $\mathcal{L}_{sub}$ into two subset $\mathcal{L}^L_{sub}$ and $\mathcal{L}^R_{sub}$ by PCA;
      Call *Refinement*($\tilde{\mathcal{L}}$,$\mathcal{L}^L_{sub}$);
      Call *Refinement*($\tilde{\mathcal{L}}$,$\mathcal{L}^R_{sub}$);
    end
  end
  Add $\mathcal{L}_{sub}$ into $\tilde{\mathcal{L}}$;
  return;

ALGORITHM 4: PeelingFieldGeneration

Input: Voxel representation of a solid model, $\bar{\mathcal{H}} = \{v_{i,j,k}\}$
Output: An indication-field $\bar{F}(\cdot)$ as the inverse of peeling with value defined on every voxel of $\bar{\mathcal{H}}$

Assign all voxels of $\bar{\mathcal{H}}$ into a set of unprocessed voxels, $\mathcal{U}$;
Compute the convex-hull of $\mathcal{U}$ and $\mathcal{T}$ as $C_c = C(\mathcal{U} \cup \mathcal{T})$;
Set the peeling index $\tau = 1$;
while $\mathcal{U} \neq \emptyset$ **do**
  Initialize $\mathcal{P}_c = \emptyset$;
  Assign all voxels that are ϵ-located on C_c into $\mathcal{P}_c$;
  Update $\mathcal{U}$ as $\mathcal{U} = \mathcal{U} \setminus \mathcal{P}_c$;
  For each $v_{r,s,t} \in \mathcal{P}_c$, assign its field value as $F(\mathbf{c}(v_{r,s,t})) = \tau$;
  Update the convex-front as $C_c = C(\mathcal{U} \cup \mathcal{T})$;
  $\tau = \tau + 1$.
end
foreach $v_{i,j,k} \in \bar{\mathcal{H}}$ **do**
  `/* Note that:` τ `holds the value of` $\max_{\forall \mathbf{q} \in \tilde{\mathcal{H}}}(F(\mathbf{q}))$ `now. */`
  $\bar{F}(\mathbf{c}(v_{r,s,t})) = 1 + (\tau - F(\mathbf{c}(v_{r,s,t})))$; `; // By Eq.(1) in the paper`
end

and $\mathbf{a}_{j+1,l}$ $(\forall l)$ (see Fig.4 for an example). In such a case, a smooth motion stops at a pose of $\mathbf{a}_{j,k}$ and thereafter start a new motion from $\mathbf{a}_{j+1,l}$. This disconnection can also be determined by the Dijkstra's algorithm automatically.

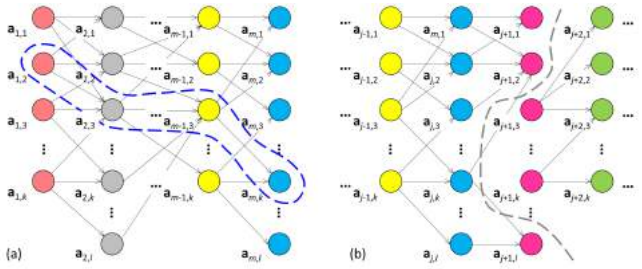

Fig. 4. Pose-continuity based motion planning is computed on a directed graph: (a) a smooth motion is determined on the graph by the Dijkstra's algorithm of shortest path (circled by the blue dash line) and (b) the disconnection (specified by the gray dash line) between the nodes of two neighboring samples can be found by the same algorithm.

REFERENCES

Yijiang Huang, Juyong Zhang, Xin Hu, Guoxian Song, Zhongyuan Liu, Lei Yu, and Ligang Liu. 2016. FrameFab: Robotic Fabrication of Frame Shapes. *ACM Trans. Graph.* 35, 6, Article 224 (2016), 224:1–224:11 pages. https://doi.org/10.1145/2980179.2982401

Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. 2016. Printing Arbitrary Meshes with a 5DOF Wireframe Printer. *ACM Trans. Graph.* 35, 4, Article 101 (2016), 101:1–101:9 pages. https://doi.org/10.1145/2897824.2925966