Computer-Aided Design 115 (2019) 135–146

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Automatic Support Removal for Additive Manufacturing Post Processing☆,☆☆

Saigopal Nelaturi, Morad Behandish *, Amir M. Mirzendehdel, Johan de Kleer

Palo Alto Research Center (PARC), 3333 Coyote Hill Road, Palo Alto, CA 94304, United States

ARTICLE INFO

Article history:
Received 27 April 2019
Accepted 3 May 2019

Keywords:
Support removal
Additive manufacturing
Post-processing
Accessibility analysis
Configuration space

ABSTRACT

An additive manufacturing (AM) process often produces a *near-net* shape that closely conforms to the intended design to be manufactured. It sometimes contains additional support structure (also called scaffolding), which has to be removed in post-processing. We describe an approach to automatically generate process plans for support removal using a multi-axis machining instrument. The goal is to fracture the contact regions between each support component and the part, and to do it in the most cost-effective order while avoiding collisions with evolving near-net shape, including the remaining support components. A recursive algorithm identifies a maximal collection of support components whose connection regions to the part are accessible as well as the orientations at which they can be removed at a given round. For every such region, the accessible orientations appear as a 'fiber' in the collision-free space of the evolving near-net shape and the tool assembly. To order the removal of accessible supports, the algorithm constructs a search graph whose edges are weighted by the Riemannian distance between the fibers. The least expensive process plan is obtained by solving a traveling salesman problem (TSP) over the search graph. The sequence of configurations obtained by solving TSP is used as the input to a motion planner that finds collision free paths to visit all accessible features. The resulting part without the support structure can then be finished using traditional machining to produce the intended design. The effectiveness of the method is demonstrated through benchmark examples in 3D.

1. Introduction

Many AM processes require generating support structures to sustain the manufactured part so that it does not collapse under its own weight (in polymer AM), burn due to intensive laser power, or warp due to thermal residual stresses (in metal AM) as the material is deposited throughout the process. For a given design, support structures directly add to AM cost and engineers aim to minimize required supports typically by optimizing the build direction. However, selecting the build direction is usually influenced or even dictated by other parameters such as material properties, feature size, accuracy, and so on. Thus, at least for the foreseeable future, support structures are inevitable for fabricating many industrial parts across numerous AM technologies.

1.1. Subtractive post-processing of near-net shapes

Producing quality parts using AM often requires post-processing operations on the *near-net* shape, including the AM part and the support structure. The post-processing is typically in the form of machining or subtractive manufacturing (SM) in general. Currently, post-process planning is performed manually and is labor-intensive, especially in metal AM. Moreover, with the advent of hybrid manufacturing technologies [1], the entire workflow can be potentially automated for interleaved AM and SM operations [2], which requires accurate and efficient process plans to generate as well as remove supports.

Engineers may be able to recommend a candidate set of support removal sequences by visual inspection. For example, they quickly decide if a near-net shape has support components that cannot be removed—a pathological example being when supports are placed in internal voids. However, cutting tool assemblies are complex constructions with indexable inserts, tool-holders, and optimized tool shanks. Beyond simple cases, it is hard to manually plan the support removal sequence, especially when there are many feasible solutions of marginally different cost—which can make a difference for mass production.

☆ This paper has been recommended for acceptance by Pierre Alliez, Yong-Jin Liu & Xin Li.

☆☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cad.2019.05.030.

* Corresponding author.
E-mail address: moradbeh@parc.com (M. Behandish).

https://doi.org/10.1016/j.cad.2019.05.030
0010-4485/

Fig. 1. A part made using metal AM. The scaffolding is shown (top-left) along with a close-up (top-right). Two separate setups for SM are used for manual support removal (bottom-left and -right).
Source: The images are taken from [4].

The combination of the part, support, and tool assembly geometries define a complex space of non-colliding tool *configurations*, i.e., combinations of translations and orientations. This free-space is a subset of the 6D configuration space of rigid motions [3]. At the beginning of the post-process, only a subset of support components is accessible through the free space, while others are occluded by the part and surrounding support components. Every time a support component is removed, more components become exposed and the free space constantly grows. Finding a feasible sequence for removing all supports one after another is a challenging spatial planning problem; because it requires navigating a *dynamic* free space against a near-net shape that is updated at every step. This problem does not have an automated solution as of today.

For the purposes of this paper, we are only interested in post-process planning for a *given* support structure, i.e., one that is generated by any method and given as input. Post-processing ranges from picking the supports by hand using a chisel-like tool, to programming collision free tool-paths for a CNC-mill to machine them off. In either case, the engineer often minimizes the effort by fracturing only the contact region between each support component and the additively manufactured part, so that at the end of the post-process the part and supports can be disengaged from the base-plate of the AM machine. This approach is efficient because it does not require machining the entire scaffold to expose the part and therefore saves time while extending tool life. Our strategy is to use this property to our advantage to navigate the gigantic search space.

Fig. 1 illustrates a typical example of a metal AM part with support structure.

1.2. Related work

The literature on support structures for AM is broad. Without attempting a comprehensive review, we present some background on support structure reduction, generation, and accessibility analysis. For a comprehensive review of the influence of support materials on design, planning, and post-processing for AM, see [5].

1.2.1. Reducing support structures

Support structures are sacrificial features that substantially add to manufacturing cost by increasing build time, material usage, and post-fabrication clean-up time. In metal AM, for instance, machine operation costs (energy, hardware, and labor) are about 63% of the total cost, material costs are about 18%, and post-processing costs are about 8%, on average [6]. Numerous researchers have investigated eliminating or reducing support materials for AM from different perspectives. Support structure layout is intimately coupled with the build direction. One of the most popular manufacturing planning strategies in AM is to choose a build direction that requires the least support material [7–13]. It is also possible to design the part itself in such a way that it requires the least amount of support while meeting other performance criteria. Multiple part design optimization approaches focus on eliminating support structures altogether by designing "self-supporting" shapes [14,15]. Others optimize the design by considering support minimization in a trade-off with competing objective functions as a practical alternative [16,17].

1.2.2. Generating support structures

Support structures must be designed to be accessible, heat-dissipating, lightweight and load-bearing [5]. Tree-like supports are a popular design for metal AM that can be manually generated through commercial packages such as Meshmixer™. An automated approach for creating such supports was proposed in [18]. For powder-bed metal AM, a contact-free approach was developed in [19] where the support structure is not attached to the build part but effectively dissipates heat in the surrounding powder. For some polymer AM processes, it is also possible to use water-soluble support materials that can alleviate issues regarding support components that are hard-to-access or connected to thin features [20]. There are also recent efforts towards generating dissolvable supports for metal AM that exploit chemical and electrochemical stability between different metals [21]. However most of the metal AM processes are based on powder-bed fusion that are predominantly single-material.

1.2.3. Accessibility analysis

Since it is not always possible to completely eliminate the support structures (or make them dissolvable, etc.), it must be ensured that all support components are accessible according to a specified post-process. Little attention has been paid to automating the accessibility analysis to remove supports, presumably because the interacting geometries are usually complex. While the ability to fabricate such complex shapes is one of AM's great strengths, it makes post-process planning more challenging than, for instance, SM planning for traditional feature-based CAD models. When the tool assembly can translate and rotate at the same time, it tremendously adds to the complexity. Before addressing these problems, we briefly overview some of the challenges in accessibility analysis.

Accessibility analysis can be conveniently formulated in terms of the absence of interference at key locations on the intermediate part. The shape and degrees of freedom (DOF) of the moving tool substantially contribute to the difficulty of accessibility analysis, which is why many approaches try to simplify one or both of these inputs. For example, Spitz et al. [22] analyze the accessibility of coordinate measuring machines (CMM) under translation in terms of the free space (computed via a Minkowski sum) of part and CMM instrument. CMMs often have elongated stems to probe the part and therefore local accessibility analysis can be simplified in terms of visibility. In general, visibility analysis is used commonly as an approximation of accessibility analysis when tools can be abstracted into one or more rays (ignoring the thickness and shape) [23–25]. These approaches can be quite

effective for a variety of problems, but generalize poorly when the interference is substantially influenced by tool geometry, as is the case with realistic tool assemblies.

Analysis of contact and interfering configurations is a fundamental problem in robot motion planning. Classical approaches focus on analytically describing the configuration space (C-space) obstacle as the solution to polynomial equations that capture all possible contact interactions (e.g., vertex–face, face–face, etc.) between a polyhedral part/obstacle and tool/robot [26]. When the interacting shapes are complex non-convex polyhedra, exact computation of the C-space obstacle may require intersecting thousands of primitive patches. For general spatial motions, C-space mapping using analytical representations is typically restricted to relatively simple geometries. For a comprehensive review of C-space mapping, see [27].

Formulating motion planning problems in the configuration space of rigid motions is a classical approach [3] and has been studied extensively. The key insight is to transform the dynamic problem of planning the motion of a moving robot to the static problem of finding a path for a point in the free space. while the formalism is elegant, computing 6D free spaces has been considered a barrier to its practical implementation. Modern motion planners rely on sampling-based approaches such as probabilistic roadmaps [28–31] to locally sample collision-free configurations and concatenate collision-free paths. This approach is quite successful, but requires explicitly specifying start and goal states for the motion planner. Identifying these start and goal states is itself a challenging spatial reasoning problem in support removal and other manufacturing planning applications.

To summarize, most approaches to accessibility analysis are based on simplifying either the interacting geometries, or the DOF available to the moving shape. In this paper, we make no such restrictions. We use the formalism of C-space modeling to formulate accessibility analysis for support removal, while using efficient computations of the free space by taking advantage of the known locations for finite and (presumably small) dislocation features. We show that the approach scales to practical situations involving complex part and tool geometries.

1.3. Contributions & outline

We introduce an automatic spatial planning approach to identify a feasible and cost-effective sequence of support removal operations and a path to execute them.

Our approach relies on an *explicit* computation of the sampled (dynamically evolving) free space. It applies to parts, support structures, and tool assemblies of arbitrary shape, and multi-axis machines that have simultaneous access to translating and rotating DOF.

The paper is organized as follows:

In Section 2, we formulate the accessibility analysis problem in the C-space and outlines a recursive algorithm to identify a sequence in which the supports can be removed from a near-net shape. A key idea to make the computations on an evolving C-space tractable is to compute a maximal collection of removable supports at every "round" of the recursive algorithm, i.e., the collection of all supports whose every connection to the part is accessible in at least one orientation (Section 2.3). Brief introductions to C-space formulation and computations that are required to identify the maximal collection are provided in Sections 2.1 and 2.2. In particular, we model the set of accessible orientations at the connections of each support component to the part can be viewed as a 'fiber' in configuration space obtained from a lifting operation.

In Section 3 we present algorithms to plan for the sequence in which the supports of a given maximal collection are removed to optimize a cost function. We formulate this task as a traveling sales problem (TSP) [32] on a graph whose nodes are represented by the fibers and edges are weighted by the cost of traveling between them. Once a candidate plan is generated, we invoke a standard motion planning algorithm to compute collision-free tool-paths between fracture points to remove support components for the current round. The recursive algorithm is repeated until no support components remain.

Section 4 demonstrates nontrivial support removal on a nontrivial 3D bracket and 3D tool that has access to all six DOFs. Results are provided to underscore the practical implementation of the proposed algorithms.

2. Identifying removable supports

Removing supports by cutting off their contact regions with the part is an efficient alternative to machining the entire support material using a traditional milling process. For the latter, in the same way that one would remove material from a full raw stock [25,33], one can clean out the support material by sweeping the tool within the support material over the *continuum* of accessible configurations. In our approach, we need only *finite* contact configurations to peel off the support components. This assumption is critical in our ability to efficiently compute the free space for an evolving near-net shape.

At the beginning, it is likely that only a subset of support components are removable in this manner with a given tool. Intuitively, we may imagine a forest of support components (e.g., columns) in a near-net shape, where the occluded columns are inaccessible until the columns along the periphery are removed.

Our approach uses known properties of the C-space of relative rigid transformations (detailed in Sections 2.1 and 2.2) to automatically identify the maximal collection of supports that are guaranteed to be removable from a near-net shape in an intermediate state. Briefly, the algorithm proceeds as follows: The C-space obstacle is computed explicitly and all contacting but non-colliding configurations of the tool are extracted. A support component is deemed removable iff all of its dislocation features are accessible through *contact*, meaning that there exists at least one configuration in the *boundary* of the C-space obstacle that corresponds to each dislocation feature. All removable support components are collected and designated for removal at the current "round" of the post-process plan. After removing this collection from the near-net shape, the C-obstacle reduces in size, more dislocation features become accessible, hence more support components become removable. The algorithm recursively identifies a sequence of removable supports that are peeled off by the cutting tool to finally converge to the desired part.

Note that this process is independent of the precise order and tool-path[1] in which the collected support components of a given round are removed in practice. A manufacturability test is also encoded in the algorithm, which can identify early on if the near-net shape includes support components that are inherently unreachable even after removing the outer layers.

2.1. Constructing configuration space obstacles

We formulate the support removal problem for d-dimensional shapes (typically, $d = 2$ or 3). Throughout the paper, we will use illustrations in 2D for building intuition, and show 3D results in Section 4.

[1] For the moment, we take the existence of such a path for granted, which is not always the case. We return to this issue in Section 3.2.

Queries about the interference of a moving body (e.g., the tool assembly) $T \subseteq \mathbb{R}^d$ against a stationary set of 3D obstacles (e.g., the near-net shape) $N \subseteq \mathbb{R}^d$ are conceptualized as evaluating membership predicted against the C-space obstacle $\mathcal{O}(N,T) \subseteq \mathsf{C}$ which partitions the C-space of relative rigid transformations $\mathsf{C} := \mathrm{SE(d)}$ into three disjoint regions representing free, contacting, and interfering configurations respectively. These regions correspond to the exterior, boundary, and interior of the C-space obstacle, respectively.

Constructing the obstacle $\mathcal{O}(N,T)$ is a critical step in our algorithm. Practical approaches use the fact that the particular group structure of $\mathrm{SE(d)} = \mathrm{SO(d)} \rtimes \mathbb{R}^d$ (i.e., the Lie group of rigid transformations) allows it to be factored [34] into two simpler and lower-dimensional subgroups; namely, SO(d) (for rotations) and $\mathbb{R}^d$ (for translations). Hence, we can represent each rigid transformation $\tau \in \mathrm{SE(d)}$ as a tuple $\tau \equiv (r, \mathbf{t})$ with $\mathbf{t} \in \mathbb{R}^d$ and $r \in \mathrm{SO(d)}$. In 2D, the former is a 2D vector while the latter is represented by a angle $\theta \in [0, 2\pi)$. In 3D, the former is a 3D vector while the latter can be represented in a number of different ways (e.g., orthogonal matrices, quaternions, axis-angle, Euler angles, etc.).

Let us denote a rotation of $T \subseteq \mathbb{R}^d$ by $r \in \mathrm{SO(d)}$ by $T_r \subseteq \mathbb{R}^d$. Assuming that we are only dealing with solids (i.e., compact-regular seminalaytic sets in $\mathbb{R}^d$) [35], the C-obstacle $\mathcal{O}(N,T)$ may be expressed as follows:

$$\mathcal{O}(P,T) = \bigcup_{r \in \mathrm{SO(d)}} \mathsf{i}(N \oplus (-T_r)), \tag{1}$$

where $\mathsf{i}(\cdot)$ represents the topological interior operator. The pointset $(N \oplus (-T_r))$ is the topological closure of the translational C-space obstacle of N with T_r, i.e., the C-obstacle for a *fixed* orientation $r \in \mathrm{SO(d)}$. It is well-known [36–38] that this pointset may be calculated in terms of a Minkowski sum.[2] The C-space for all rotations is thus obtained by unifying the different translational C-spaces, each forming a slice of the C-obstacle. In practice, $\mathcal{O}(N,T)$ may be approximated as a *stack* of a finite number of r-slices, indexed by rotations sampled in SO(d). This approach is commonly used to calculate C-space obstacles for planar motions [38].

Notice that $(N \oplus (-T_r))$ represents the closure of the translational C-space obstacle and its boundary, denoted by $\partial(N \oplus (-T_r))$ represents all the translations of the moving body (fir the fixed orientation) that cause surface contact but no volumetric interference with P. The contact configurations $\mathcal{C}(N,T) \subseteq \mathrm{SE(d)}$ are therefore obtained as:

$$\mathcal{C}(P,T) = \bigcup_{r \in \mathrm{SO(d)}} \partial(N \oplus (-T_r)). \tag{2}$$

The contact space is normally a lower-dimensional submanifold of SE(d), which makes its computation challenging. In practice, we approximate $\mathcal{C}(N,T)$ with a slightly "thickened" set, i.e., a narrow (but measurable) region surrounding the boundary of the C-space obstacle.

2.2. Querying contact configurations

Our approach uses the fact that the Minkowski sum $(A \oplus B) \subseteq \mathbb{R}^d$ of a pair of arbitrary solids $A, B \subseteq \mathbb{R}^d$ corresponds to the support (i.e., 0-superlevel set) of the *convolution* $(\mathbf{1}_A * \mathbf{1}_B)$ of the indicator functions $\mathbf{1}_A, \mathbf{1}_B : \mathbb{R}^d \to \{0,1\}$ of the two sets.[3] The advantage of this approach is that the convolution may be implemented in terms of Fourier transforms.[4] When the sets are sampled uniformly on a regular grid, the fast Fourier transform (FFT) provides an efficient and scalable implementation of each translational C-space obstacle (i.e, r-slice of $\mathcal{O}(P,T)$):

$$(r, \mathbf{t}) \in \mathcal{O}(N,T) \quad \text{iff} \quad (\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) > 0, \tag{3}$$

A configuration $\tau \equiv (r, \mathbf{t})$ is inside the obstacle if the convolution does not vanish for at least one orientation $r \in \mathrm{SO(d)}$. Note that the convolution of nonnegative functions is always nonnegative, hence the free space is defined implicitly by and equality test:

$$(r, \mathbf{t}) \notin \mathcal{O}(P,T) \quad \text{iff} \quad (\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) = 0. \tag{4}$$

The convolution provides a field of *overlap measure* values between P and T_r over the translational C-space. In other words, $(\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) = \mu^d[N \cap (T_r + \mathbf{t})]$ measures the volume of intersection (if any) between P and the moved body $T_\tau := (T_r + \mathbf{t}) = \{\mathbf{x} + \mathbf{t} \mid \mathbf{x} \in T_r\}$. Here, $\mu^d[\cdot]$ represents the Lebesgue d-measure (e.g., area for $d = 2$ and volume for $d = 3$) [39]. The queried configuration is in the C-space obstacle (resp. free space) iff this measure is greater than (resp. equal to) zero.

The contact space $\mathcal{C}(P,T)$ consists of configurations at which the overlap measure is *critically* zero, meaning that there exists an infinitesimal change to the configuration that can make it nonzero. To approximate $\mathcal{C}(N,T)$ in a computable fashion, we implicitly define an 'ϵ-contact' space using the following predicate:

$$(r, \mathbf{t}) \in \mathcal{C}_\epsilon(N,T) \quad \text{iff} \quad 0 < (\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) < \epsilon. \tag{5}$$

Using a small the threshold $\epsilon > 0$, one may assume that $\mathcal{C}_\epsilon(N,T)$ closely approximates $\mathcal{C}(N,T)$ from a measure-theoretic standpoint.

To summarize, if the r-slices are computed and stored beforehand (as a finite number of convolutions), it is possible to query the overlap measure in constant time at any sampled translation and rotation by interpolation against the stack of convolution fields. In particular, $\mathcal{C}(N,T)$ is numerically approximated as the set of all transformations $(r, \mathbf{t}) \in \mathrm{SE(d)}$ where $0 < (\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) < \epsilon$ for some tolerable interference volume $\epsilon > 0$.

Fig. 2 illustrates in 2D how the computed stack of convolutions for different r-slices for sampled rotations is used to classify a given tool configuration as free, contact, or colliding. In 3D, the same exact method applies, except that each convolution is a 3D field and the rotations are samples in 3D using any number of standard uniform or pseudo-uniform/random sampling methods [40].

Algorithm 1 describes a simple (and parallel) procedure to compute the ϵ-contact space for a finite sample $\{r_s\}_{1 \le s \le n_1} \subset \mathrm{SO(d)}$ of orientations. The convolution is faster to compute on a regular sample $\{\mathbf{t}_s\}_{1 \le s \le n_2} \subset \mathbb{R}^d$ of translations using uniform FFTs [36], which requires a regular axis-aligned sample (i.e., voxelization) of the part and rotated tool. The procedure takes $O(n_1 n_2 \log n_2)$ which is very close to linear time in the number of sampled configurations $n := n_1 n_2$. Note that this procedure is called only once per round of the recursive outer-loop (Algorithm 2). See the results in Fig. 8 of Section 4.

Note that storing the entire 6D set of C-space obstacle $\mathcal{O}(N,T)$ for 3D parts is not practical. However, the contact space $\mathcal{C}(N,T)$, approximated by $\mathcal{C}_\epsilon(N,T)$, is a sparse subset of the C-space that can be precomputed and queried later in constant-time.

[2] The Minkowski sum of two pointsets $A, B \subseteq \mathbb{R}^d$ is defined as $(A \oplus B) = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A \text{ and } \mathbf{b} \in B\}$.

[3] The indicator function of a pointset $\Omega \subseteq \mathbb{R}^d$ at a query point $\mathbf{x} \in \mathbb{R}^d$ is defined as $\mathbf{1}_\Omega(\mathbf{x}) = 1$ if $\mathbf{x} \in \Omega$ and $\mathbf{1}_\Omega(\mathbf{x}) = 0$ if $\mathbf{x} \notin \Omega$.

[4] The Fourier transform of convolution is the same as the product of Fourier transforms. Hence, the convolution can be computed by two forward transforms, one pointwise multiplication in frequency domain, and an inverse transform.

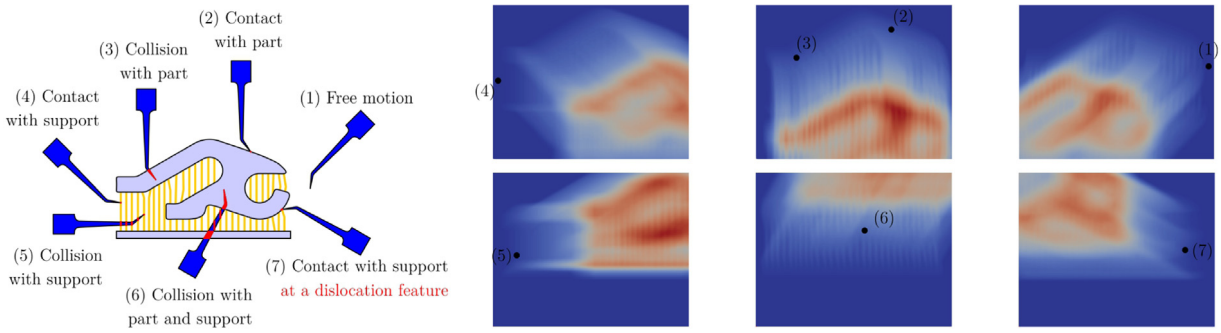


Fig. 2. At every iteration of support removal planning, different tool configuration modes (e.g., free, contact, and collision) are determined by point membership classification (PMC) against the C-space obstacle. In this 2D example, at every tool orientation $r \in \mathrm{SO}(2)$ in (a), the overlap measure (i.e., area of the interference regions highlighted in red) for all possible translations is given by a convolution field over the r-slice in (b). For a given translation (annotated points on the r-slices), the value of the field determines the overlap measure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 1 Computing $\epsilon-$contact configurations.

procedure CONTACT($N, T; n_1, n_2, \epsilon$)
  $\mathcal{C}_\epsilon(N,T) \leftarrow \emptyset$
  $\Theta \leftarrow$ SAMPLEROT(n_1) ▷ e.g., Hopf fibration [40].
  for all $r \in \Theta$ **do**
    $T_r \leftarrow$ ROTATE(T, r) ▷ Rotates and resamples.
    $\mu_r \leftarrow$ CONV(N, T_r) ▷ At once via FFTs [36].
    for all $\mathbf{t} \in$ Grid(n_2) **do**
      $\mathcal{C}_\epsilon(N,T) \leftarrow \mathcal{C}_\epsilon(N,T) \cup \{(\mathbf{t}, r) \mid 0 < \mu_r(\mathbf{t}) < \epsilon\}$
    end for
  end for
  return $\mathcal{C}_\epsilon(N,T)$
end procedure

2.3. Identifying removable support components

In this Section, we reason about the tool assembly's free space to automatically construct a maximal collection of removable support components. This collection is defined as the set of all support components that may be removed by the tool assembly without interfering with the near-net shape, except by the tool tip at the fracture points.

Let $P, S \subseteq \mathbb{R}^d$ denote the part and support structure, respectively, which only contact over their common boundary denoted by $F := (P \cap S) = (\partial P \cap \partial S)$. The initial near-net shape is obtained as $N := (P \cup S)$.

The support structure commonly consists of many connected components (called hereafter *support components*):

$$S = \bigcup_{1 \le i \le n_S} S_i, \quad (S_i \cap S_{i'}) = \emptyset \quad \text{if } i \neq i', \tag{6}$$

where $S_i \subseteq \mathbb{R}^d$ stands for the ith component ($1 \le i \le n_S$). We assume that a given support component is removable with a given cutting tool if the tip of the tool can access and fracture all contact regions between the support component and the part boundary in one or more collision-free configurations, i.e., positions and orientations of the tool at which the tool does not interfere with the part or other support components. The contact region $F = (P \cap S) = (\partial P \cap \partial S)$ is also decomposed into its connected components (hereafter called *dislocation features*):

$$F = \bigcup_{1 \le j \le n_F} F_j, \quad (F_j \cap F_{j'}) = \emptyset \quad \text{if } j \neq j'. \tag{7}$$

where $F_j \subseteq \mathbb{R}^d$ stands for the jth feature ($1 \le i \le n_F$). Note that the indexing scheme for (6) and (7) are different because every support component is connected to the part via two or more dislocation features hence $n_F \ge 2n_S$. The nomenclature is illustrated in Fig. 3 for a 2D example.

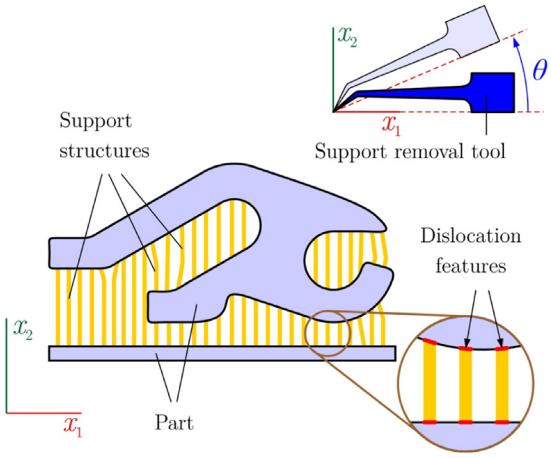
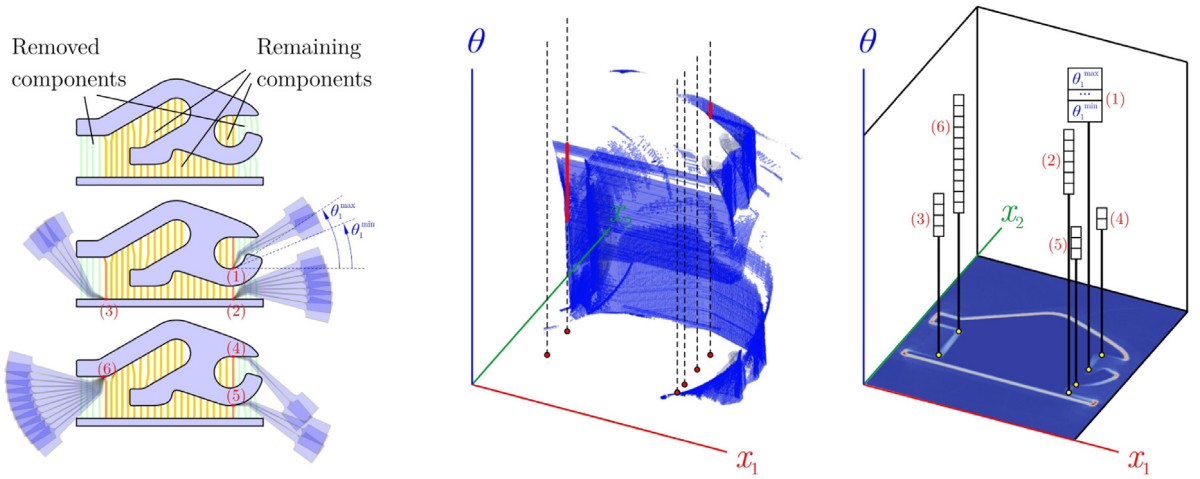


Fig. 3. A 2D near-net shape, including the part and support structure (i.e., scaffolding), and a support removal tool with three DOFs (two translations and one rotation). The C-space of rigid motions is SE(2) whose elements are parameterized by (x_1, x_2, θ).

Support structures are typically designed to have small contact regions with the part's surface to enable easier fracturing and removal and to minimize surface roughness after removal. Note that sometimes support components are constructed from one location on the part to another, but it is preferred that they are constructed connect the part's surface to the machine's base plate. Moreover, one support component can be connected to the part at multiple locations, e.g., when they are designed to branch out from a main trunk. To accommodate the most general condition, we do not make any simplifying assumption on the shape, connectivity, number, or layout of support components and their dislocation features.

Let us position the tool in its original configuration in such a way that the cutter tip (or some point on the cutting surface) coincides with the origin of the coordinate system used as a reference frame to quantify the rigid transformations of the tool assembly $T \subseteq \mathbb{R}^d$ with respect to the near-net shape $N \subseteq \mathbb{R}^d$ at a given round. This choice ensures that a transformation $\tau \equiv (r, \mathbf{t}) \in \mathrm{SE}(d)$ brings the tool's cutter (and not any other point on the tool) to the point $\mathbf{t} \in \mathbb{R}^3$ in the Euclidean d-space where the near-net shape resides. Although this choice is not necessary from



(a) Accessible orientations by example (b) The contact space as a fibration (c) Sparse representation of fibration

Fig. 4. At every recursion of support removal planning, the accessible dislocation features (exemplified in (a)) are characterized by their inclusion in the projection of the contact configurations. This implies the existence of a nonempty subset of orientations at which the feature's lifting by to the C-space (i.e., a 'fiber') intersects the C-space obstacle's boundary. The resulting fibration is represented by a list of lists, each containing sampled orientations at which the dislocation feature is accessible.

a theoretical standpoint, it simplifies the accessibility analysis by querying the pre-computed C-space map.

In a d-dimensional space, the tool can operate with all $\mathrm{d}(\mathrm{d}+1)/2$ degrees of freedom, $\mathrm{d}(\mathrm{d}-1)/2$ of which are for rotations and the remaining d are for translations. We can define *projections* from $\mathrm{SE(d)} = \mathrm{SO(d)} \rtimes \mathbb{R}^{\mathrm{d}}$ to SO(d) and $\mathbb{R}^{\mathrm{d}}$, respectively, that map every rigid transformation $\tau \equiv (r, \mathbf{t}) \in \mathrm{SE(d)}$ to $r \in \mathrm{SO(d)}$ and $\mathbf{t} \in \mathbb{R}^{\mathrm{d}}$:

$$\begin{array}{ccc} & \mathrm{SE(d)} & \\ {\scriptstyle \pi_1}\swarrow & & \searrow{\scriptstyle \pi_2} \\ \mathrm{SO(d)} & & \mathbb{R}^{\mathrm{d}} \end{array} \qquad \begin{aligned} \pi_1(\tau) &= \pi_1(r, \mathbf{t}) := r, \\ \pi_2(\tau) &= \pi_2(r, \mathbf{t}) := \mathbf{t}, \end{aligned} \tag{8}$$

Notice that the second projection depends on the earlier choice of origin. Clearly, these maps are not invertible as functions. Nonetheless, we can define *liftings* that take every $r \in \mathrm{SO(d)}$ and $\mathbf{t} \in \mathbb{R}^{\mathrm{d}}$ to a *subspace* of SE(d)[5]:

$$\begin{array}{ccc} & \mathcal{P}(\mathrm{SE(d)}) & \\ {\scriptstyle \pi_1^{-1}}\nearrow & & \nwarrow{\scriptstyle \pi_2^{-1}} \\ \mathrm{SO(d)} & & \mathbb{R}^{\mathrm{d}} \end{array} \qquad \begin{aligned} \pi_1^{-1}(r) &= \left\{(r, \mathbf{t}) \mid \mathbf{t} \in \mathbb{R}^{\mathrm{d}}\right\}, \\ \pi_2^{-1}(\mathbf{t}) &= \left\{(r, \mathbf{t}) \mid r \in \mathrm{SO(d)}\right\}, \end{aligned} \tag{9}$$

By extension, we can define set functions; for example, $\pi_2^{-1} : \mathcal{P}(\mathbb{R}^{\mathrm{d}}) \to \mathcal{P}(\mathrm{SE(d)})$ such that $\pi_2^{-1}(F)$ assigns a copy of the entire SO(d) to every point $\mathbf{t} \in F$:

$$\pi_2^{-1}(\Omega) := \{\pi_2^{-1}(\mathbf{t}) \mid \mathbf{t} \in F\} \cong (\mathrm{SO(d)} \times F). \tag{10}$$

Given a dislocation feature $F_j \subset P$, the set of all configurations at which it can be touched by the tool tip can be obtained in terms of the above projection/lifting maps:

$$\mathcal{f}_j := f(F_j, N, T) := \pi_2^{-1}(F_j) \cap \mathcal{C}(N, T). \tag{11}$$

In other words, every dislocation feature is lifted to the C-space by pairing it with all possible orientations. The pairing results in the set of all possible configurations that bring the tool tip to the dislocation feature (at different orientations). Among them, the ones that are contained within the contact space $\mathcal{C}(N, T)$ are collision-free, hence they represent the set of all accessible configurations. Hereafter, we refer to $\mathcal{f}_j \subset \mathrm{SE(d)}$ as the *contact fiber* for the dislocation feature. Once again, for computational purposes, we can approximate the contact fibers via ϵ-fibers, using the measurable ϵ-contact region:

$$\mathcal{f}_j \approx \mathcal{f}_\epsilon(F_j, N, T) := \pi_2^{-1}(F_j) \cap \mathcal{C}_\epsilon(N, T). \tag{12}$$

In practice, the dislocation features are small enough to assume that touching *any* point on their circumference is sufficient to fracture it. As a result, we can simply project the fiber from SE(d) to SO(d) to collect all accessible orientations at all positions within the feature:

$$f_j^* := f^*(F_j, N, T) = \pi_1\big(\pi_2^{-1}(F_j) \cap \mathcal{C}(N, T)\big). \tag{13}$$

whose ϵ-fiber approximations are given as expected by:

$$f_j^* \approx f_\epsilon^*(F_j, N, T) = \pi_1\big(\pi_2^{-1}(F_j) \cap \mathcal{C}_\epsilon(N, T)\big). \tag{14}$$

Fig. 4 illustrates the fibers for a 2D example in which the dislocation features are contracted to points. From an algorithmic perspective, each ϵ-fiber can be hashed into a list of sampled orientations anchored at a different dislocation features. *A dislocation feature is accessible iff its fiber is nonempty*. As a result, we observe that a given support component is removable iff every one of its dislocation features has a nonempty fiber.

Algorithm 2 describes the procedure for finding the contact fibers in a single round of recursive support removal planning. The procedure performs $O(n_F)$ queries on every one of the convolution fields precomputed and stored for n_1 sampled orientations, resulting in a total of $O(n_F n_1)$ queries—each convolution-read taking constant time. This analysis assumes that we need $O(1)$ queries per each dislocation feature, because each feature is sampled by one or at most a few point(s), which is reasonable. Every orientation that is accessible is pushed into a stack that represents the fiber for the queried feature.

Note that before performing the queries, we can shortlist the n_F features rapidly to accessible ones by projecting the ϵ-contact space to the Euclidean d-space (i.e., losing orientation information) and finding its common points with the dislocation features:

F_j is accessible iff $\pi_2(\mathcal{C}(N, T)) \cap F_j \neq \emptyset$. (15)

[5] $\mathcal{P}(\Omega) = \{\Omega' \mid \Omega' \subseteq \Omega\}$ denotes the powerset of a set Ω.

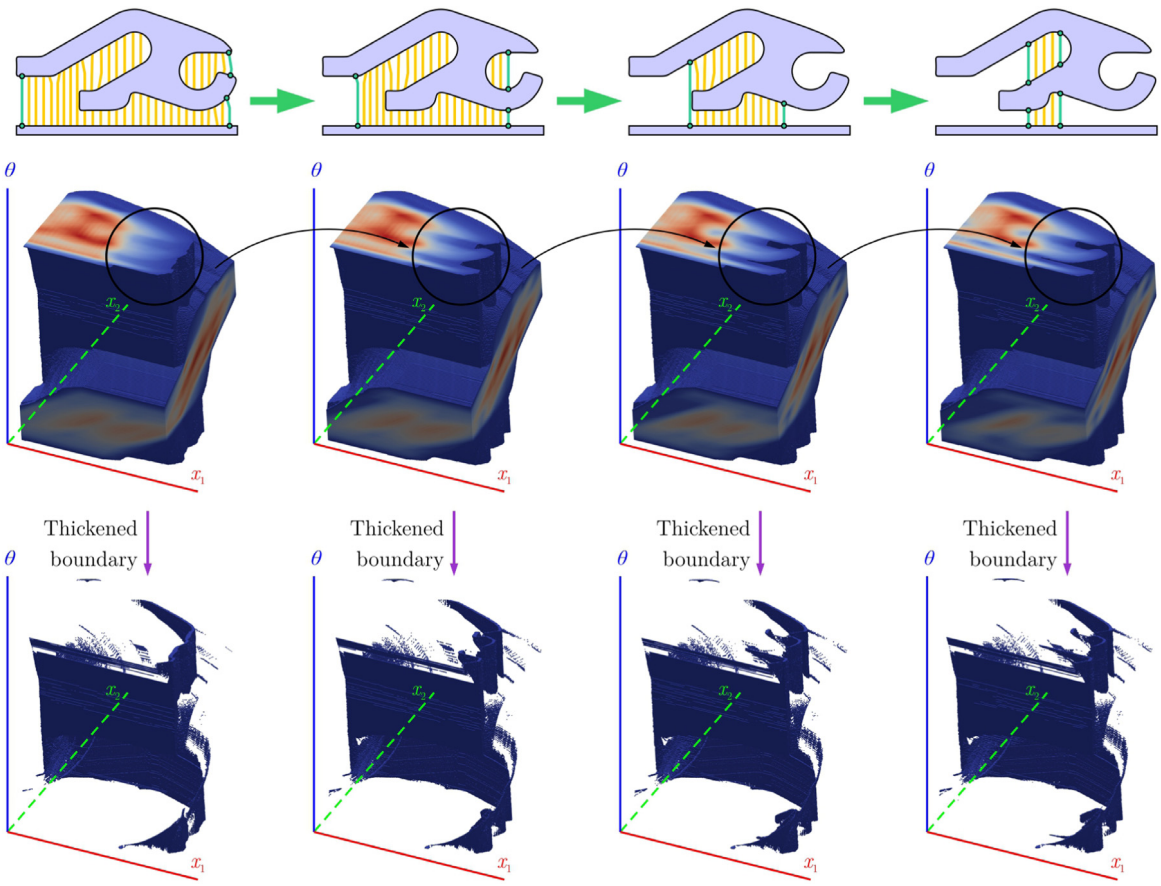


Fig. 5. As the near-net shape goes through Algorithm 3, the C-space obstacle keeps shrinking in size and more support structures become removable. At each recursion, the accessible dislocation features are identified by their fibers, exposed at the $\mathcal{C}_\epsilon(N, T)$.

Algorithm 2 Finding contact fibers.

procedure FIBRATION($P, S, T, n_1, n_2, \epsilon$)
  $N \leftarrow P \cup S$ ▷ Current near-net shape.
  $\{S_i\}_{1\le i\le n_S} \leftarrow$ CC(S) ▷ Support components.
  $\{F_j\}_{1\le j\le n_F} \leftarrow$ CC($P \cap S$) ▷ Dislocation features.
  for all $1 \le j \le n_F$ **do**
    $f_j^* \leftarrow \emptyset$ ▷ Initialize contact fibers.
    $\mathcal{C}_\epsilon(N, T) \leftarrow$ CONTACT(N, T, n_1, n_2, ϵ)
    if $\pi_2(\mathcal{C}_\epsilon(N, T)) \cap F_j \neq \emptyset$ **then**
      $f_j^* \leftarrow f_j^* \cup \pi_1(\pi_2^{-1}(F_j) \cap \mathcal{C}_\epsilon(N, T))$
    end if
  end for
  return $\{f_j^*\}_{1\le j\le n_F}$
end procedure

This means that there exists at least one point in the dislocation feature F_j that belongs to the projected contact space. The projection ensures that there exists at least one non-colliding orientation at which the said point is accessible. Note that the projected contact space can be obtained by unifying the translational contact spaces $\partial(N \oplus (-T_r))$ for different orientations:

$$\pi_2(\mathcal{C}(N, T)) = \bigcup_{r\in \mathrm{SO(d)}} \partial(N \oplus (-T_r)), \tag{16}$$

Once again, the contact space can be approximated by the ϵ-contact criterion. As such, the above *early test* for accessibility can be rapidly computed by summing up the indicator functions of the translational contact spaces, which, in turn, are obtained as the $(0, \epsilon)$-interval level set of the convolution fields for different r-slices. The resulting pointset is a single field over the d-space with a narrow support, containing all positions in the vicinity of the near-net shape's boundary that can be accessed by at least one orientation. For every point $\mathbf{t} \in \mathbb{R}^d$, we accumulate the convolution value over r-slices for which the condition $0 < (\mathbf{1}_N * \mathbf{1}_{-T_r})(\mathbf{t}) < \epsilon$ holds. This incurs no additional computation cost because as the convolutions are precomputed, their $(0, \epsilon)$-interval level sets can be extracted and accumulated on-the-fly. The result is a field with a narrow-band support over the d-space that quantifies non-colliding orientations of a given point. A 2D example of the accumulated and projected field of contact measures is illustrated in Fig. 4(c) (base field). A dislocation feature is accessible iff it has at least one point at which this function is nonzero. Note that this early test counts the accessible orientations; however, it does not tell us *which* orientations (i.e., the fiber).

2.4. Recursive support removal rounds

It is likely that multiple support components are removable from the near-net shape at a given intermediate state. Intuitively, every round of the recursive algorithm peels off all removable components to expose a new set of components. The process is repeated until either all supports are removed, or the part is deemed non-manufacturable because some supports are inaccessible. See the results in Fig. 10 of Section 4.

Algorithm 3 Identifying removable supports.

procedure REMOVABLE($P, S^{\mathtt{t}}, T, n_1, n_2, \epsilon$)
  if $S = \emptyset$ **then** ▷ Base case: all supports removed.
    return $\emptyset$ ▷ Final round: success.
  end if
  $\{f_j^*\}_{1\le j\le n_F} \leftarrow$ FIBRATION($P, S, T, n_1, n_2, \epsilon$)
    ▷ Obtain nonempty contact fibers.
  $\mathbf{I}^{\mathtt{t}} \leftarrow$ NONEMPTY($\{f_j^*\}_{1\le j\le n_F}$) ▷ Defined in (17).
  if $\mathbf{I}^{\mathtt{t}} = \emptyset$ **then**
    return $\perp$ ▷ Final round: failure.
  end if
  $S^{\mathtt{t}+1} = S^{\mathtt{t}} - \bigcup_{i\in\mathbf{I}^{\mathtt{t}}} S_i$ ▷ Remaining supports.
  return $\langle \mathbf{I}^{\mathtt{t}}$, REMOVABLE($P, S^{\mathtt{t}+1}, T, n_1, n_2, \epsilon$)$\rangle$
    ▷ Appending sequence.
end procedure

At round−$\mathtt{t}$ of the support removal algorithm,[6] the near-net shape is $N^{\mathtt{t}} := (P \cup S^{\mathtt{t}})$ with initial conditions $N^0 = N$ and $S^0 = S$. Both $N^{\mathtt{t}}$ and $S^{\mathtt{t}}$ are monotonically reduced (in terms of set containment) from one round to the next, i.e., $N^{\mathtt{t}+1} \subseteq N^{\mathtt{t}}$ and $S^{\mathtt{t}+1} \subseteq S^{\mathtt{t}}$.

Let $\mathbf{I}^{\mathtt{t}}$ represent the indices for the maximal collection of removal supports at a given round, i.e., for every $i \in \mathbf{I}^{\mathtt{t}}$, the support component $S_i \subseteq S^{\mathtt{t}}$ is removable at round−$\mathtt{t}$. If $\mathbf{J}(i)$ represents the indices of the dislocation features for the same support component, then $i \in \mathbf{I}^{\mathtt{t}}$ iff for every $j \in \mathbf{J}(i)$, the fiber f_j is nonempty, i.e.,

$$\mathbf{I}^{\mathtt{t}} = \left\{1 \le i \le n_S \mid \forall j \in \mathbf{J}(i) : f_j \neq \emptyset\right\}. \tag{17}$$

The remaining support for the next round is computed as: $S^{\mathtt{t}+1} = S^{\mathtt{t}} - \bigcup_{i\in\mathbf{I}^{\mathtt{t}}} S_i$. The algorithm continues until either of two termination criteria can happen:

1. $N^{\mathtt{t}} = P$, i.e., $S^{\mathtt{t}} = \emptyset$, which implies successful removal of the entire support; or
2. $N^{\mathtt{t}} = N^{\mathtt{t}+1}$ and $S^{\mathtt{t}} = S^{\mathtt{t}+1}$ which indicates that the remaining support components cannot be reached.

Fig. 5 illustrates a few rounds of recursive support removal for a 2D example. The C-space obstacle and contact space change as more supports are removed.

2.5. Necessary and sufficient conditions

Algorithm 3 illustrates an implementation of the recursive algorithm. All *potentially* accessible support components are identified by checking if $\pi_2(\mathcal{C}_\epsilon(N,T)) \cap F_j \neq \emptyset$ for every $j \in \mathbf{I}_i$ for every remaining support component $S_i \subseteq S^{\mathtt{t}}$. If at any given round, no supports are identified for removal, the algorithm returns a failure message indicating that the support structure is inherently not removable. In other words, this test provides a *necessary* condition for the AM post-process to be feasible. While the test is quite useful in finding the order in which the support components should be fractured, it does not provide a *sufficient* condition. For example, the test does account for the situations where there is a collision-free final configuration to touch the dislocation feature, but there may not exist a collision-free *path* in the C-space that brings the tool from its initial configuration to the final pose. This situation can happen if the free space $(\mathcal{O}(N,T))^c$ is not path-connected, and the initial and final configurations are located in different connected components. Fig. 6 illustrates an example of a false-positive. Such scenarios are rare if the support structure is designed properly.

[6] Note that superscripts here are used to indicate the rounds of the algorithm, and should not be confused with the subscripts used earlier to indicate connected components.

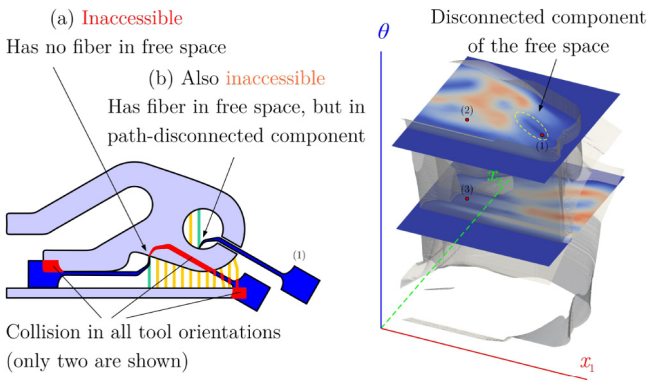


Fig. 6. The test provided by Algorithm 2 is a necessary but not sufficient condition for manufacturability because it does not consider the path-connectedness of the free configuration space. If the free space is not path-connected, it is possible (but rare) for dislocation features to have a collision-free orientation that is not accessible through a collision-free path from an initial configuration.

Although analyzing path-connectedness is important, SE(d) is high-dimensional for d $\geq$ 3 and the free space topology can be highly complex. As opposed to reasoning about the topology, we can use a sampling-based motion planner (e.g., based on probabilistic roadmaps [28–31]) to find a collision-free path from a reasonable starting configuration in $(\mathcal{O}(N,T))^c$ to at least one configuration in the fiber.

3. Support removal planning

In Section 2.3 we have shown how an ordered sequence $\langle \mathbf{I}^0, \mathbf{I}^1, \ldots, \mathbf{I}^{\mathtt{t}}, \ldots\rangle$ of the indices for maximal collections of removable support components may be computed. Each $\mathbf{I}^{\mathtt{t}}$ represents a collection of support components that can be removed *in any order* as long as all the support components preceding them in the sequence have been removed in earlier rounds of the algorithm.

In this section, we describe approaches to plan for the removal of the support components within each group $\mathbf{I}^{\mathtt{t}}$ to reduce a cost function. At the outset, we observe that any two collections $\mathbf{I}^{\mathtt{t}_1}$ and $\mathbf{I}^{\mathtt{t}_2}$ may be processed independently, because their C-space obstacles are defined with respect to distinct near-net shapes $N^{\mathtt{t}_1}, N^{\mathtt{t}_1}$. In other words, the order in which the support components are removed at one round does not affect the order for another round. As far as we assume an additive cost function, we may solve the planning problem for different rounds in parallel.

3.1. A traveling salesman formulation

Given a near-net shape $N^{\mathtt{t}} \subseteq \mathbb{R}^{\mathrm{d}}$, the tool assembly $T \subseteq \mathbb{R}^{\mathrm{d}}$, and a sub-collection of support components consisting of $\{S_i \mid i \in \mathbf{I}^{\mathtt{t}}\}$, removing all $|\mathbf{I}^{\mathtt{t}}|$ components amounts to finding a collision-free path in the C-space that moves the tool tip to all dislocation features that connect the said support components to the part. These dislocation features are given by $\{F_j \mid j \in \mathbf{J}_i \text{ and } i \in \mathbf{I}^{\mathtt{t}}\}$. The accessible configurations for every one of them is given by a fiber; hence, the planning problem can be formulated as finding a path in the C-space that visits all fibers one after another. It suffices for the path to intersect with every fiber at least once, meaning that

the tool touches the dislocation feature in at least one accessible orientation.

The cost function for a candidate path can be defined in several plausible ways. If we use machining time as the criteria and assume an almost constant speed in the C-space, the path length in the C-space may serve as a cost function. However, it is likely in practice to have different cost factors associated with translational and rotational machine DOF. Here are two extreme cases:

- If we have a 3-axis machine that needs re-fixturing for every change of orientation, moving in the rotation space is significantly costlier than moving in the translational space. In this case, the cost function is to minimize the total number of orientations needed to visit all fibers. Using the mental picture in Fig. 4, we may formulate this problem as finding the minimal number of r-slices to "cut" the given set of fibers, where each fiber is cut at least once.
- If for any reason changing orientations is cheaper, the problem is dramatically simplified. We can formulate the path planning conveniently in the Euclidean d-space, especially since every fiber is projected to a small region (i.e., dislocation feature) which can be approximated by a point. The translational distance between the points, obtained by measuring the curve length in the d-space serves as a cost function.

Consider a pair of configurations $\tau_1 \in \mathcal{f}_{j_1}$ and $\tau_2 \in \mathcal{f}_{j_2}$, each selected from a different fiber that needs to be visited at a given round, i.e., $j_1 \in \mathbf{J}_{i_1}$, $j_2 \in \mathbf{J}_{i_2}$, $i_1, i_2 \in \mathbf{I}^{\mathrm{t}}$, and $j_1 \neq j_2$. The curve-length of a geodesic in the C-space the connects these configurations is given by the Riemannian distance $\| \ln(\tau_1^{-1}\tau_2)\|_2$ in which $\ln(\cdot)$ stands for the standard logarithm function,[7] which can be computed directly on the matrix representation of the relative transformation $\tau_1^{-1}\tau_2 \in \mathrm{SE(d)}$, and $\|\cdot\|_2$ is the Frobenius (i.e., L_2) norm. The minimum distance between all pairs of configurations from the two fibers is a lower-bound to the length of any curve segment that connects the two fibers in the C-space:

$$\mathrm{d}(\mathcal{f}_{j_1}, \mathcal{f}_{j_2}) := \min_{\tau_1 \in \mathcal{f}_{j_1}} \min_{\tau_2 \in \mathcal{f}_{j_2}} \| \ln(\tau_1^{-1}\tau_2)\|_2. \tag{18}$$

The above distance function is well-defined for every pair of fibers (as a whole). We can construct an undirected complete graph that has $|\mathbf{I}^{\mathrm{t}}| + 1$ vertices, one for each fiber and an additional one for the reference configuration from which the tool assembly starts moving and returns to after removing supports. We can associate the above distance function with every edge, noting that it provides an admissible heuristic to approximate the actual cost function *before* motion planning. This is important for computational tractability of the algorithm.

Finding the shortest path to visit all fibers (i.e., vertices or "cities") at least once is thus formulated as a traveling salesman problem (TSP) [32]. Our goal is to create a Hamiltonian cycle (a tour) with minimum cost, obtained by adding the costs of individual edges on the graph, starting and ending at the reference vertex.

The TSP is NP-complete, but we may use an approximation algorithm [41] if the cost-function is a proper metric. Distance functions defined via minima of pairwise distances as in (18) do not normally qualify because they may not satisfy reflexivity and triangle inequality, even though the pairwise distance itself is a metric, as in the case of the Riemannian metric. The more appropriate formulation in this case would be to associate every configuration (as opposed to every fiber) with a graph vertex, use the pairwise Riemannian distance, which is a proper metric, as the cost associated with every edge, and solve a more complex generalization of TSP in which every *group* of cities (i.e., fibers) has to be visited at least once.

Fortunately, if the dislocation features are small and ignore the cost of moving along a single fiber – i.e., in-place rotations at a given dislocation feature before moving to the next feature – we can still formulate and solve the problem as a TSP. Using the approximation algorithm in [41], we may assert that the cost of the tour is no more than twice that of the minimum spanning tree's weight. The algorithm will find a sequence of graph vertices (configurations) that are visited exactly once. The vertex sequence then delineates the start and goal configurations for a sequence of motion planning problems that can be solved by a standard motion planner. Note that the contact fiber configurations will include minimal interference with (corresponding to cutting) the supports. The start and end configurations are locally perturbed to ensure a collision free path can be found.

3.2. Motion planning over the fibration

We use the Open Motion Planning Library (OMPL) [42] to compute collision-free paths to go from one fiber to the next in the sequence prescribed by the TSP solution. The start and goal configurations on every fiber are selected using a *greedy* policy as follows: For the first configuration, we start from a given initial pose of the tool. For every fiber that is visited next in the sequence, we select the configuration on the fiber that is the nearest (in terms of the Riemannian metric) to the preceding configuration selected on the previously visited fiber in the sequence.

As discussed earlier, there may be cases when the free space is not path-connected. In such cases, even after successful determination of removable support components with nonempty fibers (via Algorithm 3) and ordering them via TSP solution, the motion planner may not be able to find a collision-free path to a dislocation feature.

We note that having $\mathcal{O}(N, T)$ explicitly computed could help find collision-free paths by overloading the collision checker of the motion planner with a constant-time membership query against $\mathcal{O}(N, T)$. However, as pointed out in Section 2.1, it is impractical to store an explicit representation of the entire $\mathcal{O}(N, T)$. The stored contact space $\mathcal{C}(N, T)$ approximated by $\mathcal{C}_\epsilon(N, T)$ represents a boundary of the C-space obstacle under fairly general conditions, hence can be used as a substitute.

However, membership queries against boundary representations might be more time-consuming than highly-optimized collision detection algorithms. Moreover, once the start and goal configurations are known, motion planning via probabilistic roadmaps is hard to beat in terms of both efficiency and accuracy. In particular, the accuracy of membership queries against precomputed $\mathcal{O}(N, T)$ (even when it is possible, e.g., in 2D) is limited by the rasterization resolution. We can use a conservative policy, meaning that the discretized 3D models are chosen to strictly contain the "exact" CAD models to ensure that the approximation to $\mathcal{O}(N, T)$ obtained via discrete convolutions also contains the exact C-space obstacle. While this is safe policy as it overestimates collisions, it commonly results in false-positives, especially at contact configurations, e.g., detecting a collision when the tool is touching the dislocation feature while there is none.

The convolution field is a fast and effective approach to computing the collision measures for a large number of configurations *at once*—unlike collision detection on B-reps, which is better suited for a *single* configuration at-a-time. It is critical in our ability to sift through a large number of dislocation features and

[7] The logarithm function on a Lie group defines a local coordinate system over a neighborhood by mapping the group elements to the Lie algebra (i.e., the tangent vector space) [34].

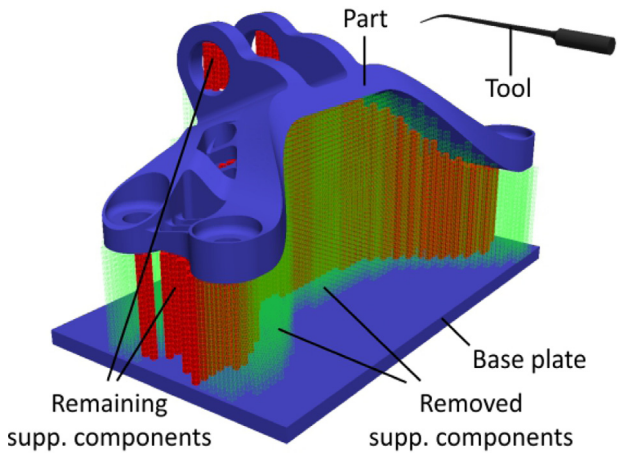


Fig. 7. A near-net shape highlighting the AM part (blue), support removal tool (red), support structure (white), and dislocation features (green). The support components must be peeled off to fully disengage the part without leaving printed material behind . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

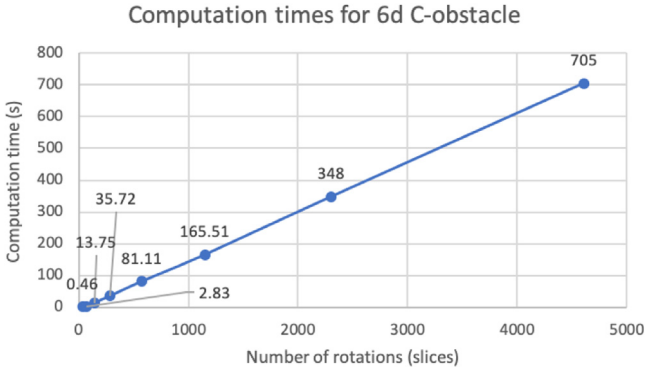


Fig. 8. Computation times for the 6DC-space obstacle by sampling rotations and computing convolutions at a resolution of 512^3 to represent each r-slice of the obstacle. The computations are performed using FFTs via ArrayFire on an NVIDIA GTX 1080 GPU (2,560 CUDA cores, 8 GB RAM) via OpenCL.

decide their accessibility quickly at every round of the recursion with evolving near-net shape. However, after the accessible configurations, motion planning can be done with collision detection between the master CAD models (e.g., B-reps) with a much higher precision than that of rasterized models.

4. Results

We demonstrate the approach using an illustrative 3D example. A near-net shape along with a cutting tool to remove supports at dislocation features is shown in Fig. 7.

We assume that the part is fixtured properly (not shown here, similarly to Fig. 1) such that the removal sequence of the support components does not impact its stability.

At every round, Algorithm 3 identifies the outer layer of supports that can be peeled off, using FFT-based convolution (Sections 2.1 and 2.2) as illustrated in Fig. 10. For each layer, solving the TSP on the graph of fibers (Section 3) gives a sequence of configurations to visit, while asserting all of them will contact a distinct dislocation feature without collision.

The OMPL motion planner is invoked to find a collision-free tool-path to fracture the dislocation features one after another (Section 3.2), as illustrated in Fig. 11.

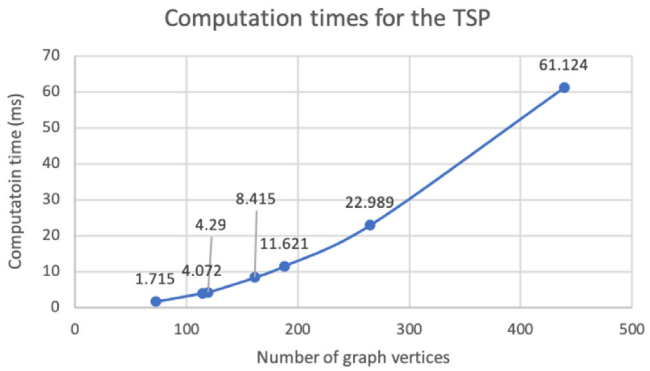


Fig. 9. Computation times for the TSP in configuration space using the Boost Graph Library.

Figs. 8 and 9 show running times for computing C-space obstacles and solving TSP, respectively.

5. Conclusion

Support removal is an important post-processing step for AM and is often a bottleneck for manufacturability. We have shown an automated solution to identifying and removing support materials in an AM part, by exploiting the properties of the C-space obstacle, free space, and contact space obtained as their shared boundary. The contact space is extracted as an interval level set of a sparse field that is precomputed and queried during a recursive algorithm to peel off removable supports. The approach does not make assumptions on part, tool, or support structure geometries, or on the DOF with which a tool can move.

Several improvements to this paper are possible. The computations are highly parallelizable and offer opportunities for more efficient implementation. Furthermore, uniform sampling of the rotation space for C-space computations can be improved by choosing rotations based on more specific manufacturing constraints.

As opposed to explicitly computing C-space obstacles and finding goal states for a motion planner, one may directly find collision-free paths between neighboring dislocation features by locally sampling configurations and invoking rapid collision-checkers. This is similar to the probabilistic roadmap approach to path planning [28–31]. Nonetheless, this paper demonstrates that explicit calculation of 6D C-space maps should not be considered as an impediment to solving practical mechanical design and manufacturing problems. We hope that it inspires researchers to apply these methods to other practical problems in which the specific properties of the application can be leveraged to make C-space computations tractable.

Last but not least, while this paper's focus has been kinematics and spatial planning aspects, there are many unsolved problems with regard to the physics of support removal (e.g., mechanics of the fracture process) that have been ignored and need to be considered in practice.

Acknowledgment

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA), United States. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or U.S. Government.

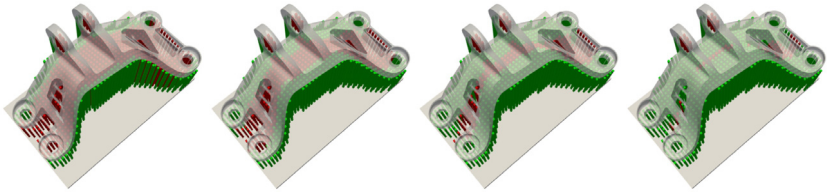


Fig. 10. Several rounds of support removal via Algorithm 3. At each round, a subset of the support components is connected to the part and the base plate by dislocation features that are accessible in at least one orientation.

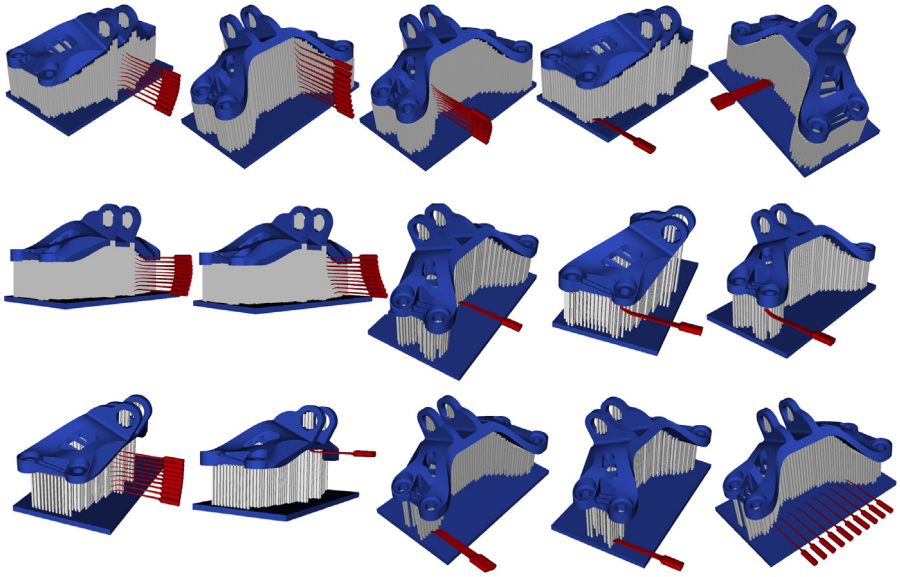


Fig. 11. Several intermediate collision-free paths for the cutting tool to contact the supports at dislocation features. Each row represents a single recursion of Algorithm 3. Observe that the motion planner ensures that the tool changes orientation if required when moving from one dislocation feature to the next. Each individual collision free path is computed using OMPL [42] in approximately 0.5 s.

References

[1] Yamazaki T. Development of a hybrid multi-tasking machine tool: integration of additive manufacturing technology with cnc machining. Procedia CIRP 2016;42:81–6.

[2] Behandish M, Nelaturi S, de Kleer J. Automated process planning for hybrid manufacturing. Comput Aided Des 2018;102:115–27.

[3] Lozano-Perez T. Spatial planning: a configuration space approach. IEEE Trans Comput 1983;C-32(2):108–20. http://dx.doi.org/10.1109/TC.1983.1676196.

[4] Hamilton K. Planning, preparing and producing: walking the tightrope between additive and subtractive manufacturing. In: Metal AM, vol. 2. Inovar Communications, Ltd.; 2016, p. 39–56.

[5] Jiang J, Xu X, Stringer J. Support structures for additive manufacturing: a review. J Manuf Mater Process 2018;2(4):64.

[6] Thomas DS, Gilbert SW. Costs and cost effectiveness of additive manufacturing. NIST SP 2014;1176:12.

[7] Nezhad AS, Barazandeh F, Rahimi AR, Vatani M. Pareto-based optimization of part orientation in stereolithography. Proc Inst Mech Eng B 2010;224(10):1591–8.

[8] Das P, Chandran R, Samant R, Anand S. Optimum part build orientation in additive manufacturing for minimizing part errors and support structures. Procedia Manuf 2015;1:343–54.

[9] Paul R, Anand S. Optimization of layered manufacturing process for reducing form errors with minimal support structures. J Manuf Syst 2015;36:231–43.

[10] Ezair B, Massarwi F, Elber G. Orientation analysis of 3d objects toward minimal support volume in 3d-printing. Comput Graph 2015;51:117–24.

[11] Morgan HD, Cherry JA, Jonnalagadda S, Ewing D, Sienz J. Part orientation optimisation for the additive layer manufacture of metal components. Int J Adv Manuf Technol 2016;86(5–8):1679–87.

[12] Coupek D, Friedrich J, Battran D, Riedel O. Reduction of support structures and building time by optimized path planning algorithms in multi-axis additive manufacturing. Procedia CIRP 2018;67:221–6.

[13] Hehr A, Wenning J, Terrani K, Babu SS, Norfolk M. Five-axis ultrasonic additive manufacturing for nuclear component manufacture. JOM 2017;69(3):485–90.

[14] Gaynor AT, Guest JK. Topology optimization considering overhang constraints: eliminating sacrificial support material in additive manufacturing through design. Struct Multidiscip Optim 2016;54(5):1157–72.

[15] Langelaar M. Topology optimization of 3d self-supporting structures for additive manufacturing. Addit Manuf 2016;12:60–70.

[16] Mirzendehdel AM, Suresh K. Support structure constrained topology optimization for additive manufacturing. Comput Aided Des 2016;81:1–13.

[17] Qian X. Undercut and overhang angle control in topology optimization: a density gradient based integral approach. Internat J Numer Methods Engrg 2017;111(3):247–72.

[18] Vanek J, Galicia JAG, Benes B. Clever support: efficient support structure generation for digital fabrication. Comput Graph Forum 2014;33(5):117–25.

[19] Cooper K, Steele P, Cheng B, Chou K. Contact-free support structures for part overhangs in powder-bed metal additive manufacturing. Inventions 2018;3(1):2.

[20] Ni F, Wang G, Zhao H. Fabrication of water-soluble poly (vinyl alcohol)-based composites with improved thermal behavior for potential three-dimensional printing application. J Appl Polym Sci 2017;134(24).

[21] Hildreth OJ, Nassar AR, Chasse KR, Simpson TW. Dissolvable metal supports for 3d direct metal printing. 3D Print Addit Manuf 2016;3(2):90–7.

[22] Spitz SN, Spyridi AJ, Requicha AAG. Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. IEEE Trans Robot Autom 1998;15:714–27.

[23] Limaiem A, ElMaraghy HA. Integrated accessibility analysis and measurement operations sequencing for cmms. J Manuf Syst 2000;19(2):83–93.

[24] Arbelaez D, Avila MC, Krishnamurthy A, Li W, Yasui Y, Dornfeld D, McMains S. Cleanability of mechanical components. Tech. rep., University of California, Berkeley; 2008.

[25] Nelaturi S, Burton G, Fritz C, Kurtoglu T. Automatic spatial planning for machining operations. In: Proceedings of the 2015 IEEE international conference on automation science and engineering (CASE'2015). 2015, p. 677–82. http://dx.doi.org/10.1109/CoASE.2015.7294158.

[26] Canny J. The complexity of robot motion planning. Acm doctoral dissertation awards, ACM; 1988.

[27] Wise KD, Bowyer A. A survey of global configuration-space mapping techniques for a single robot in a static environment. Int J Robot Res 2000;19(8):762–79.
[28] Kavraki L, Svestka P, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces, vol. 1994. IEEE; 1994.
[29] Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 1996;12(4):566–80.
[30] Hsu D, Kavraki L, Latombe JC, Motwani R, Sorkin S. On finding narrow passages with probabilistic roadmap planners. In: Proceedings of the international workshop on algorithmic foundations of robotics (WAFR), vol. 1998. ACM; 1998.
[31] Boor V, Overmars MH, van der Stappen AF. The gaussian sampling strategy for probabilistic roadmap planners. In: IEEE international conference on robotics and automation (ICRA'1999), vol. 2. IEEE; 1999, p. 1018–23.
[32] Lin S. Computer solutions of the traveling salesman problem. Bell Syst Tech J 1965;44(10):2245–69.
[33] Behandish M, Nelaturi S, Allard M. Automated process planning for turning: a feature-free approach. In: Advances in manufacturing techonology XXXII. Advances in transdisciplinary engineering series, vol. 8, 2018, p. 45–50, Proceedings of the International Conference on Manufacturing Researh (ICMR'2018).
[34] Selig JM. Geometrical fundamentals of robotics. Springer-Verlag New York; 2005.
[35] Requicha AAG. Representations for rigid solids: theory, methods, and systems. ACM Comput Surv 1980;12(4):437–64.
[36] Kavraki LE. Computation of configuration-space obstacles using the fast fourier transform. IEEE Trans Robot Autom 1995;11(3):408–13.
[37] Middleditch AE. Application of vector sum operator. Comput Aided Des 1988;20(4):183–8.
[38] Latombe JC. Robot motion planning, vol. 124. Springer Science & Business Media; 2012.
[39] Lysenko M, Nelaturi S, Shapiro V. Group morphology with convolution algebras. In: Proceedings of the 14th ACM symposium on solid and physical modeling (SMP'2010). ACM; 2010, p. 11–22.
[40] Yershova A, Jain S, LaValle SM, Mitchell JC. Generating uniform incremental grids on SO(3) using the hopf fibration. Int J Robot Res 2010;29(7):801.
[41] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT press; 2009.
[42] Şucan IA, Moll M, Kavraki LE. The open motion planning library. IEEE Robot Autom Mag 2012;19(4):72–82. http://dx.doi.org/10.1109/MRA.2012.2205651, http://ompl.kavrakilab.org.