EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

Computational Design of Steady 3D Dissection Puzzles

Keke Tang[1,2] Peng Song[3†] Xiaofei Wang[4] Bailin Deng[5] Chi-Wing Fu[6] Ligang Liu[4]

[1] Cyberspace Institute of Advanced Technology, Guangzhou University [2] University of Hong Kong [3] EPFL
[4] University of Science and Technology of China [5] Cardiff University [6] The Chinese University of Hong Kong

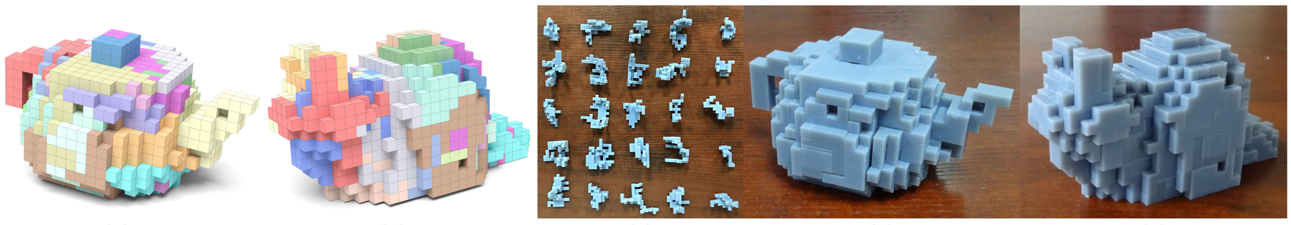


Figure 1: *(a&b)* TEAPOT - SNAIL *dissection puzzle designed by our method; (c) 25 3D-printed pieces; and (d&e) assembled puzzles.*

Abstract

Dissection puzzles require assembling a common set of pieces into multiple distinct forms. Existing works focus on creating 2D dissection puzzles that form primitive or naturalistic shapes. Unlike 2D dissection puzzles that could be supported on a tabletop surface, 3D dissection puzzles are preferable to be steady by themselves for each assembly form. In this work, we aim at computationally designing steady 3D dissection puzzles. *We address this challenging problem with three key contributions. First, we take two voxelized shapes as inputs and dissect them into a common set of puzzle pieces, during which we allow slightly modifying the input shapes, preferably on their internal volume, to preserve the external appearance. Second, we formulate a formal model of generalized interlocking for connecting pieces into a steady assembly using both their geometric arrangements and friction. Third, we modify the geometry of each dissected puzzle piece based on the formal model such that each assembly form is steady accordingly. We demonstrate the effectiveness of our approach on a wide variety of shapes, compare it with the state-of-the-art on 2D and 3D examples, and fabricate some of our designed puzzles to validate their steadiness.*

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Shape modeling;*** • ***Applied computing*** $\rightarrow$ ***Computer-aided manufacturing;***

1. Introduction

Dissection puzzles require assembling a common set of pieces in different ways to produce two or more distinct shapes (i.e., forms). A typical example is a dissection of a triangle to a square in only four pieces, also known as the Haberdasher's problem [Dud07]. Another example is a visual proof of the Pythagorean theorem with Perigal's dissection [Per72], where a large square is dissected into two smaller squares. Different from conventional puzzles that have a single form when assembled, dissection puzzles have multiple (typically two) forms and can be reconfigured among these forms by transforming and reassembling the pieces, making them more intriguing for playing.

Designing dissection puzzles (i.e., the dissection problem) is an interesting problem studied in recreational math and computational geometry [LF72]. In its basic form, geometric dissection partitions a 2D figure into a finite number of pieces that can be rearranged into a new figure of equal area. In practice, it is usually required that the dissection uses just a few pieces, such that the puzzle is manageable for playing. Early research works [Coh75, KKU00, AAC*12] focus on analytic approaches to dissect 2D primitive shapes such as triangles, squares and other regular polygons into congruent polygonal pieces. Until recently, a few computational methods have been developed to dissect general 2D shapes represented as discrete squares [ZW12] or dissect naturalistic 2D shapes approximately [DYYT17].

The dissection problem becomes considerably more challenging when moving from two dimensions to three dimensions. For 3D dissection, we aim at finding a common set of pieces that can be assembled into two different 3D shapes of equal volume. It is more challenging due to representing more complex 3D shapes with a few common pieces and requiring higher degrees of freedom (DOF) to reconfigure the pieces into different assembly forms. Therefore, very few 3D dissection results have been reported [The18, Fre09, HN06, Fre97], which are elegant primitive 3D shapes, such as tetrahedrons, cubes and prisms, and were created by solving intricate math problems with analytic approaches;

† The corresponding author, email: songpenghit@gmail.com

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

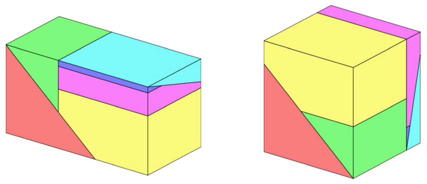


Figure 2: CUBOID-CUBE *dissection puzzle with six pieces, designed by Anton Hanegraaf in 1989. [Copyright figure: Hungerbühler and Nüsken [HN06]]*

see Figure 2 for an example. Unlike 2D dissection puzzles that are simply placed on a tabletop surface, 3D dissection puzzles are preferable to be steady for each assembly form; e.g., every puzzle piece is supported by its neighboring pieces and no puzzle piece should fall apart due to gravity.

In this work, we aim at computational design of *steady 3D dissection puzzles* from user-specified 3D shapes, motivated by the use of 3D printing for making personalized toys [BCMP18]; see Figure 1 for an example result. We explore steady structures based on mechanical interlocking [SFCO12] by immobilizing component parts through their geometric arrangements. Both dissection and interlocking are very strong constraints on the geometry of the puzzle pieces: dissection requires the pieces to be reconfigurable into different target shapes, while interlocking requires the pieces to connect with one another in a steady assembly form. Finding pieces that satisfy both constraints is an extremely challenging task. To make the problem tractable, we take a pair of voxelized shapes as inputs, and allow small modifications on them, preferably on the internal volume, to preserve their external appearance.

To meet the above challenges, we make the following contributions:

- First, we dissect the input shapes into a common set of puzzle pieces, during which the piece orientation in each form is computed automatically. Small modifications on the input shapes are allowed to facilitate the search of valid dissection solutions.
- Second, we present a *generalized interlocking* formal model that relaxes the geometric constraints in conventional interlocking methods [SFCO12] by allowing connecting puzzle pieces into a steady assembly using not just their geometric arrangements but also the friction among them.
- Third, we make steady each assembly form of the dissection puzzle by modifying the geometry of each dissected puzzle piece according to the generalized interlocking formal model, without violating the dissection requirement that is already satisfied.

We demonstrate the effectiveness of our approach on a variety of 3D shapes, and fabricate some of our designed puzzles to validate their steadiness. We compare our dissection method with a state-of-the-art method [ZW12] on both 2D and 3D examples, and show that our 3D puzzles are more steady due to mechanical interlocking employed to connect the pieces. We also compare our generalized interlocking model with the recursive interlocking model [SFCO12] to demonstrate its flexibility for designing interlocking assemblies.

2. Related Work

Geometric Dissections. In the 1830s, Bolyai [Bol32] and Gerwien [Ger33] already proved that whenever two 2D polygons have the same area, one of them can be dissected using a finite number of pieces to form the other. This is known as the Bolyai-Gerwien Theorem. Nonetheless, the number of required pieces by the theorem generally far exceeds the minimum number of pieces needed to form a dissection. Hence, a number of analytic approaches have been developed to find geometric dissections with the fewest number of pieces. However, they mainly focus on 2D primitive shapes such as triangles and squares [Coh75, KKU00, CKU07].

On the other hand, geometric dissections have been studied subject to various special constraints, such as hinged dissections that connect all the pieces into a chain at "hinged" points and convert a figure to another by swinging the chain continuously. A famous example was introduced by Dudeney [Dud07] which dissects a square into an equilateral triangle. Frederickson studied other types of "hinges" such as twisting-hinges and piano-hinges [Fre02, Fre07b]. Abbott et al. [AAC*12] proved that a hinged dissection always exists when two polygons have equal area.

Only very recently, computational tools were developed to create dissection puzzles. Zhou et al. [ZW12] proposed a stochastic search approach to create dissection puzzles on general 2D shapes represented on a discrete grid, and extended it to create a few 3D dissection puzzles. Duncan et al. [DYYT17] created approximate dissections between 2D naturalistic shapes with minimized modifications while Li et al. [LMaH*18] further constructed a special hinged dissection that requires a reversible inside-out transform to switch between two 2D shapes. The 3D dissection puzzles designed by [ZW12] use primitive shapes. In contrast, we design 3D dissection results with significantly higher complexity at the cost of introducing slight modifications on the inputs. Moreover, our work stabilizes each assembly form with interlocking.

Transformable Objects. Recently, there is an emerging interest in the computer graphics community on fabricating 3D assemblies that can be transformed into different forms, such as articulated models [BBJP12, CCA*12], foldable furniture [LHAZ15, Fre07a, Fre08], twisty puzzles [SZ15], transformables [YZC18, HCLC16], and reconfigurables [GJG16]. In particular, Zhou et al. [ZSMS14] designed transformable objects that can be folded into a box by addressing a hinged 3D dissection problem. In contrast, our work addresses a different 3D dissection problem, in which the target object models can have arbitrary shapes and the component pieces are disjoint before the assembly. Very recently, Song et al. [SFJ*17] developed a computational approach for designing reconfigurable interlocking furniture by co-decomposing cage-based input models into a common set of parts and co-constructing compatible joints among the parts. Compared with this work, we focus on general voxelized 3D shapes rather than furniture, and allow modifying the interior of the input shapes to relax the constraints in creating dissected puzzle pieces.

Interlocking Assembly. Mechanical interlocking is an intriguing method for assembling component parts, while avoiding the use of additional connectors among them. Conventional interlocking methods immobilize every single piece, as well as every subset of pieces, except for *a single key* which is the only movable piece in the entire assembly. A few computational methods have been developed to construct interlocking assemblies with this single-key property, including puzzles [XLF*11, SFCO12], 3D-printed object assem-

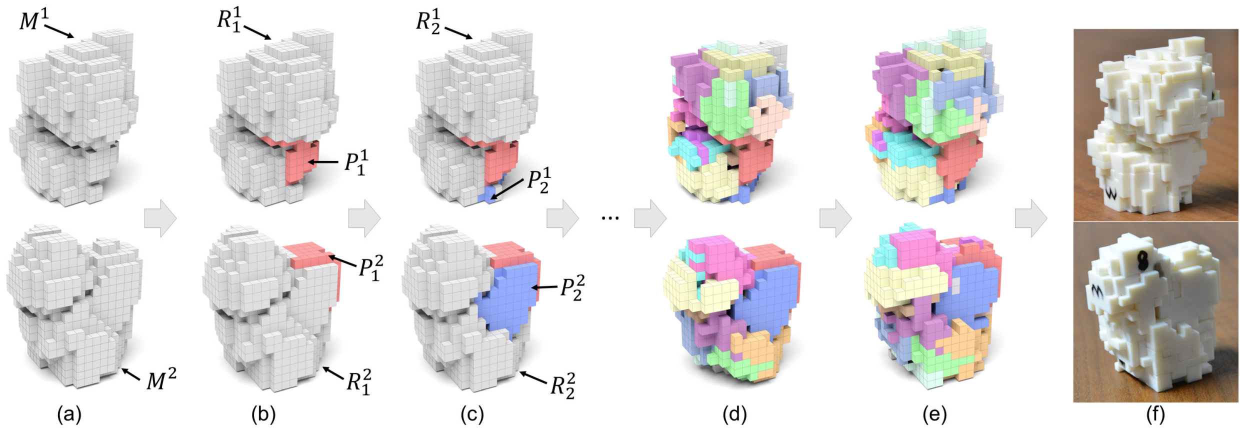


Figure 3: *Overview: (a) two voxelized models* KITTEN *and* SQUIRREL *as inputs; iteratively construct (b) the first piece P_1, (c) the second piece P_2, and (d) the other pieces; (e) post-process all puzzle pieces to preserve the models' appearance; and (f) 3D-printed dissection puzzle.*

blies [SFLF15, YCXW17], furniture assemblies [FSY*15], laser-cut polyhedrons [SDW*16], and frame/plate structures [WSP18].

Later, this single-key interlocking is relaxed to multi-key interlocking. Zhang and Balkcom [ZB16] presented a set of voxel-like interlocking parts whose instances can be connected layer-by-layer to form various interlocking voxelized shapes with multiple keys. Song et al. [SFJ*17] explicitly formulated the multi-key interlocking model for connecting parts in reconfigurable furniture assemblies.

All the above works require immobilizing every single piece and every subset of pieces, except for the key(s), inducing very strong geometric constraints on the pieces and making it overly challenging to construct pieces of 3D dissection puzzles. This work presents a new *generalized interlocking model* that immobilizes the pieces with the assistance of friction. Although the resulting assemblies can be less steady than conventional interlocking assemblies, our new model provides more flexibility for constructing interlocking assemblies that are sufficiently steady for practical usage.

3. Overview

Given two 3D shapes represented as mesh models, we first voxelize them such that the number of voxels in the two models, denoted as $\mathbf{M}^1$ and $\mathbf{M}^2$, are roughly the same; see Figure 3(a). This is achieved by scaling the mesh models such that they have exactly the same volume and then voxelizing the mesh models using the same voxel size. Taking $\mathbf{M}^1$, $\mathbf{M}^2$ and a user-desired number of pieces m ($m \geq 2$) as inputs, our goal is to generate a common set of puzzle pieces P_1, ..., P_m that satisfy the following requirements:

- *Dissection.* The common set of pieces can form each target shape via different arrangements.
- *Assemblablity.* The pieces can be physically assembled into each target shape.
- *Steadiness.* Each assembled shape should form a steady assembly, where no piece would fall off easily due to gravity and/or external forces.
- *Aesthetics.* Modifications on the input shapes' appearance should be as small as possible.
- *Avoiding tiny pieces.* Pieces consisting of a small number of voxels should be avoided since such pieces are difficult to identify and handle for physical assembly.

Besides the geometry of the common set of puzzle pieces, our problem also requires computing a rigid transformation T_i (translation and rotation) for reconfiguring each puzzle piece P_i across the two forms, i.e., from P_i^1 in $\mathbf{M}^1$ to P_i^2 in $\mathbf{M}^2$, where P_i^1 and P_i^2 are instances of P_i in the two models.

Overview of Our Approach. We construct 3D dissection puzzle pieces by iteratively co-extracting the pieces from $\mathbf{M}^1$ and $\mathbf{M}^2$ one by one, until reaching the desired number of pieces m. We denote the extracted puzzle pieces as P_1^k, P_2^k, ..., P_i^k, with R_i^k being the remaining volume in $\mathbf{M}^k$, where $1 \leq i \leq m-1$ and $k \in \{1,2\}$. Here P_i^1 and P_i^2 ($1 \leq i \leq m-1$), as well as R_{m-1}^1 and R_{m-1}^2, should have exactly the same shape to ensure a successful dissection. To facilitate the understanding, we use a consistent color scheme on the following puzzle pieces: red for P_1^k, blue for P_2^k, and gray for R_i^k; see Figure 3.

Directly generating puzzle pieces that satisfy all the above requirements simultaneously is challenging. Thus, we first construct each puzzle piece subject to the *dissection* and *assemblability* requirements, while relaxing the *aesthetics* requirement, and then modify it to satisfy the *steadiness* requirement based on our generalized interlocking formal model. After generating all puzzle pieces, we post-process them to improve the *aesthetics* of final assemblies as much as possible. During all the above procedures, we ensure that each puzzle piece has at least a certain amount of voxels to *avoid tiny pieces*. The key steps of our approach are detailed as follows:

1. *Dissect P_i.* To construct an initial P_i, we identify similar geometric features between the two models and make them corresponding to form a common piece P_i under a certain transform T_i. This is achieved by a co-seeding procedure to find corresponding seed voxels, a co-expanding procedure to grow P_i^k from the seeds, and a co-cleaning procedure to include fragmental voxels in P_i^k; see Figure 3(b-d) and Subsection 5.2.
2. *Stabilize P_i with generalized interlocking.* To make P_i steady, we propose a new generalized interlocking model that immobilizes parts using their geometric arrangements and friction (see Section 4). Our formal model ensures generalized interlocking of the assembly by enforcing local constraints when constructing each P_i. Guided by this model, we modify the geometry of P_i such that

$[P_1^k, P_2^k, ..., P_i^k, R_i^k]$ form a generalized interlocking intermediate assembly for each k (i.e., form); see Subsection 5.3.

3. *Post-process all puzzle pieces.* The generated puzzle's appearance could be different from that of the input models (i.e., introducing holes) since we allow deleting voxels to facilitate the puzzle piece construction. We alleviate modifications on the input models' appearance by adding back some of the deleted exterior voxels to the puzzle pieces while maintaining the dissection and interlocking requirements; see Figure 3(e) and Subsection 5.5.

4. Generalized Interlocking Formal Model

Interlocking adopted by existing works [XLF*11, SFCO12, SFLF15, FSY*15, SDW*16, ZB16, SFJ*17, YCXW17, WSP18] for designing steady assemblies is defined as follows:

> *An assembly of pieces (with at least three pieces) is said to be* interlocking *if every single piece as well as every subset of the pieces are* immobilized relative to one another, *except a single or a few movable key(s).*

Conventional interlocking is very restrictive since it connects pieces with pure geometric arrangements. As a result, it can only be achieved for input shapes that satisfy certain requirements; e.g., [SFCO12, SFLF15] require the input object to have large internal volume while [FSY*15, SFJ*17] require the input furniture model to have sufficient cyclic substructures.

In our daily life, many assemblies in which parts are connected by integral joints can be steady without conventional interlocking, e.g., 3D printed assemblies connected with male and female connectors [LBRM12], laser-cut assemblies connected with halved joints [CPMS14], and furniture assemblies connected with woodworking joints [Rog02]. This is because friction is employed to prevent the part(s) from moving along *the single* direction that is not restricted by the joint; see Figure 4 for an illustration. Note that the connection strength by friction depends on the tightness of the joint.

Inspired by this observation, we propose a new concept called *generalized interlocking*:

> *An assembly of pieces (with at least three pieces) is said to be* generalized interlocking *if every single piece as well as every subset of the pieces are* either immobilized or movable along a single axial direction relative to one another, *except a single or a few movable key(s).*

According to this definition, conventional interlocking is a subclass

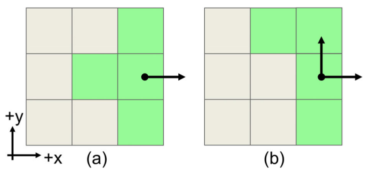


Figure 4: *(a) A joint connection where the green piece is movable along a single axial direction (i.e., $+x$); this DOF can be removed by considering friction. (b) A joint connection where the green piece is movable along more than one axial direction (i.e., $+x$, $+y$); friction cannot stabilize such a connection due to insufficient pressure.*

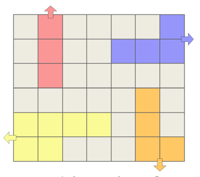
of generalized interlocking. Yet, generalized interlocking significantly relaxes the geometric constraints of conventional interlocking by allowing each piece and each subset of pieces to be movable along at most one axial direction, while retaining the steady assembly property by preventing each piece (and each group of pieces) from moving along this single axial direction using friction. In this work, we do not explicitly model the friction but simply assume that sufficient friction can be obtained by adjusting the fabrication tolerance among the pieces during their manufacture.

In the following, we introduce three formal models that guarantee generalized interlocking by enforcing geometric constraints locally when constructing each individual piece. We first introduce a naive formal model (Subsection 4.1) and a basic formal model (Subsection 4.2) to facilitate the understanding, and then our formal model (Subsection 4.3) that will be used in our puzzle design approach.

4.1. Naive Formal Model

Given a voxelized model $\mathbf{M}$, we iteratively extract each individual piece from it one by one, forming a sequence of extracted pieces P_1, P_2, ..., P_n, with R_n, the remaining part of $\mathbf{M}$, as the last piece. Let $\mathbf{D}$ be the set of six axial directions (i.e., $+x, -x, +y, -y, +z, -z$) aligned with the voxelization. We denote the extraction direction of each piece P_i as $d_i \in \mathbf{D}$.

To construct pieces from $\mathbf{M}$ that can form a generalized interlocking assembly, a straightforward way is to ensure that each piece P_i is immobilized by the last piece R_n such that P_i is only movable along d_i relative to R_n. The inset figure shows an example, where each piece is movable along a single direction (see arrows) relative to the gray piece. However, although this naive formal model can guarantee generalized interlocking, the resulting R_n usually has very irregular shape and is much larger than the other pieces since it needs to immobilize them. Hence, this naive formal model is not suitable for our 3D dissection problem.



4.2. Basic Formal Model

Rather than relying on the last piece (i.e., R_n) to immobilize all the other pieces, we can immobilize each piece P_i using R_i together with P_{i-1} when constructing P_i, thus relaxing the geometric constraint on R_n. This leads to the following requirements when constructing P_i, which are proved to guarantee the resulting pieces to be generalized interlocking (see the supplementary material for the proof).

Requirements for P_1. When constructing P_1, it should be immobilized by R_1 such that it is only movable along d_1.

Requirements for P_i $(2 \leq i \leq n)$. When constructing P_i, it should satisfy the following requirements, where $\mathbf{S}_i$ denotes the set of all neighboring pieces of P_i that have been extracted before P_i:

1. P_i should be immobilized by P_{i-1} and R_i such that it is movable only along d_i.
2. R_i should block P_i from moving along d_{i-1}, if $d_{i-1} \neq d_i$.
3. For each $P_j \in \mathbf{S}_i$ and each direction $d' \in \mathbf{D} \setminus \{d_i, d_j\}$, if R_i does not block P_j while P_i blocks P_j from moving along d', then R_i should block P_i from moving along d'.

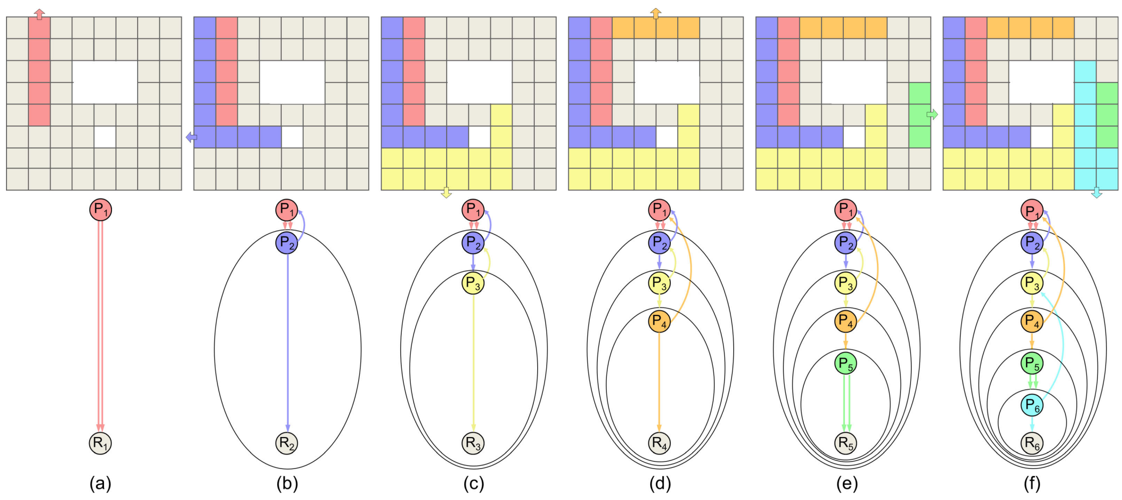


Figure 5: *Top row: constructing pieces following (a-c) the basic formal model and (d-f) our formal model, in which the extraction direction of each piece is highlighted with a small arrow. Bottom row: the corresponding dependency graphs.*

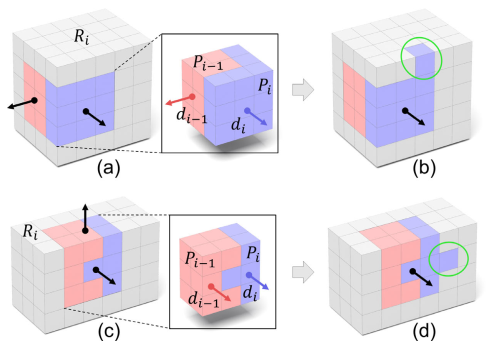


Figure 6: *(a&c) $\{P_{i-1}, P_i\}$ are movable together along more than one axial direction, violating the definition of generalized interlocking. (b&d) By modifying P_i (see green circles) following the basic formal model, $\{P_{i-1}, P_i, R_i\}$ form a generalized interlocking assembly. The movable direction(s) of P_{i-1}, P_i, and $\{P_{i-1}, P_i\}$ are indicated with red, blue, and black arrows respectively.*

Note that when constructing P_i, simply applying the first requirement is not sufficient since P_i and $P_j \in \mathbf{S}_i$ might be movable together along more than one axial direction. For example, when $d_{i-1} \neq d_i$, $\{P_{i-1}, P_i\}$ could be movable along both d_{i-1} and d_i; see Figure 6(a). For this case, we enforce the second requirement on P_i and R_i to avoid this; see Figure 6(b). Also, $\{P_j, P_i\}$ could be movable along the contacting direction(s) between P_j and P_i; see Figure 6(c). For this case, we enforce the third requirement on P_i and R_i to prevent the undesired movement; see Figure 6(d).

Dependency Graph. By iteratively extracting a new piece from R_i following the above requirements, we can draw a dependency graph for each intermediate step, where each node represents a piece and each directed edge represents the "immobilization dependency" between associated nodes; see Figure 5(a-c). In other words, if P_i links to both P_{i-1} and R_i with a directed edge, it means P_{i-1} and R_i are selected to immobilize P_i when constructing P_i. Sometimes, it is sufficient to immobilize P_i by R_i only (see Figure 5(a)), and we show two directed edges from P_i to R_i. During the extraction process, the immobilization dependency between P_i and R_i can be inherited in the successive extraction steps even if R_i has been partitioned into multiple pieces. To highlight this, we draw large circles in the graphs to indicate previous R_i's in the current step; see again Figure 5. As we assume all the part movements are relative to R_i, there is no directed edge from R_i to any other node in the graphs.

4.3. Our Formal Model

In the basic formal model, we always choose P_{i-1} and R_i to immobilize P_i; see again Figure 5(a-c). We generalize this constraint by choosing an arbitrary $P_t \in \mathbf{S}_i$ $(1 \leq t \leq i-1)$ and R_i to immobilize P_i; see Figure 5(d&f). In this formal model, each piece P_i $(2 \leq i \leq n)$ should be constructed with the following requirements.

1. P_i should be immobilized by P_t and R_i such that it is movable only along d_i.
2. For each $P_j \in \{P_1, ..., P_{i-1}\} \setminus \{P_t\}$, R_i should block P_i from moving along d_j, if R_i does not block P_j or P_t from moving along d_i and $d_i \neq d_j$.
3. For each $P_j \in \mathbf{S}_i$ and each direction $d' \in \mathbf{D} \setminus \{d_i, d_j\}$, if R_i does not block P_j while P_i blocks P_j from moving along d', then R_i should block P_i from moving along d'.

In case we could not find any P_t that satisfies the above requirements, we allow P_i to be immobilized by R_i only such that it is movable only along d_i; see Figure 5(e) for an example. It can be proved that the pieces constructed following these requirements are guaranteed to be generalized interlocking (see the supplementary material).

5. Our Approach

Given two voxelized shapes $\mathbf{M}^1$ and $\mathbf{M}^2$, and a user-desired number of pieces m, we first compute several heuristic measures on the shapes to guide the piece construction process (Subsection 5.1). Next, we dissect a piece from the two shapes (Subsection 5.2), and modify its geometry to satisfy the requirements of our formal model for stabilizing it (Subsection 5.3). We iterate this processes for each piece until the number of dissected pieces reaches m (Subsection 5.4). Lastly, we post-process all the generated puzzle pieces to improve appearance of the final assemblies (Subsection 5.5).

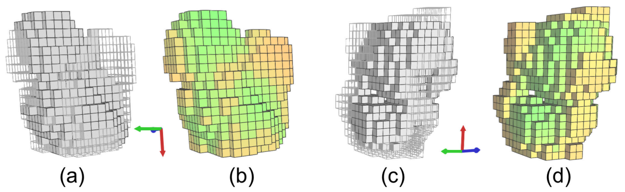


Figure 7: *Identify overlapping volume of (a)* KITTEN *and (c)* SQUIRREL*, where overlapping (non-overlapping) voxels are rendered in solid (wireframe). The coordinate axes beside each model indicate the rigid transform to align the models. (b&d) Visualize compatibility of each voxel in the two models, where orange (green) indicates low (high) compatibility.*

5.1. Solution Metrics

We compute four measures for the input models: compatibility between the two models, compactness, accessibility and noticeability for each model. The measure of noticeability is computed only once for each model, while the other measures need to be recomputed for the remaining volume R_i after extracting each piece P_i.

Compatibility. We observe that the more similar two input models are, the easier it is to perform dissection between them. In the extreme case, two models with exactly the same shape can be trivially dissected by partitioning them in the same way. Therefore, we first identify the largest overlapping volume between the two models by aligning their centroids and principal directions followed by small axis-aligned translations. Based on this overlapping volume, we introduce a measure of compatibility for each voxel v:

$$O(v) = \frac{1}{1+\exp(\mathrm{SDF}(v))}, \tag{1}$$

where SDF is the signed distance function to the boundary of the overlapping volume, with positive values for voxels outside the volume (see Figure 7). When dissecting a piece, we prefer to extract voxels with low compatibility first while leaving voxels with high compatibility in the remaining volume, to facilitate dissection of the successive pieces.

Compactness. When constructing a new piece, we prefer to select a seed voxel that is near the previously extracted pieces, to make the remaining volume as compact as possible (see Figure 8). We introduce a compactness measure C for each boundary voxel v on the remaining volume, which has a larger value for a voxel closer to the previous pieces:

$$C(v) = \delta - \min_{v_b \in \{v_b\}} \mathrm{dist}(v, v_b), \tag{2}$$

where δ is the diagonal length of the model's bounding box, $\{v_b\}$ denotes the set of voxels on the boundary of the remaining volume

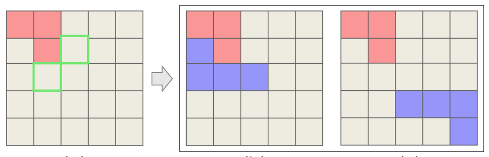


Figure 8: *The remaining volume (in gray) in (b) is more compact than that in (c) after extracting a new piece (in blue) since the piece in (b) is closer to $\{v_b\}$ (voxels with a green outline) in (a).*

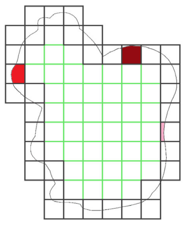
but not on the boundary of the model, and $\mathrm{dist}(v, v_b)$ is the Euclidean distance between the centroids of voxels v and v_b.

Accessibility. After extracting a puzzle piece, the remaining volume R_i has to be connected. Hence, we compute an accessibility value $A_j(v)$ for each voxel v in R_i, and use it later as a heuristic to alleviate fragmentation. Following [SFCO12], $A_j(v)$ is computed by recursively counting the (weighted) number of voxel neighbors:

$$A_j(v) = \begin{cases} \text{number of neighbors of } v, & \text{for } j = 0 \\ A_{j-1}(v) + \alpha^j \sum_i A_{j-1}(y_i(v)) & \text{for } j > 0\,, \end{cases}$$

where $\{y_i(v)\}$ are neighboring voxels of v in the remaining volume, α is set to 0.1, and j is set to 3 in our implementation. In the following, we abbreviate $A_3(v)$ as $A(v)$. Since voxels with low accessibility are likely to be fragmented, we prioritize to include them when constructing a puzzle piece.

Noticeability. As we allow modifications on the input models, we prefer to delete voxels that cannot be easily noticed rather than adding extra (exterior) voxels, since our inputs are voxelized models with solid interior. A voxel completely inside the input model (internal voxel; see green-bounded voxels in the inset) can be deleted without affecting the external appearance (i.e., unnoticeable). On the contrary, deleting voxels that intersect with the original model surface (boundary voxels; see black-bounded voxels in the inset) might remove important shape features. Thus, we measure the noticeability of each voxel v as:



$$N(v) = \begin{cases} 0, & \text{if v is an internal voxel} \\ a(v) & \text{otherwise} \end{cases}$$

where $a(v)$ is the volume of original model shape contained in the boundary voxel v normalized by the total volume of the voxel. The inset figure visualizes the noticeability of a few voxels, where dark (light) red colors indicate high (low) noticeability.

5.2. Dissect a Piece

In this subsection, we dissect a puzzle piece P_i ($1 \le i \le n$) from R^1_{i-1} and R^2_{i-1}, with $R^k_0 = \mathbf{M}^k$ ($k = 1, 2$) for consistency. We first find a pair of corresponding seed voxels, along with the transform T_i that reconfigures P_i from $\mathbf{M}^1$ to $\mathbf{M}^2$. Next, we apply a co-expanding process to grow P_i simultaneously in the two models from each seed voxel. Lastly, we perform a co-cleaning process to let P_i include fragmental voxels that are near P_i in both models, to prevent the remaining volumes from being disconnected.

Co-seeding. This procedure has the following three steps:

1. *Select Seed Candidates.* As voxels with low compatibility (e.g., non-overlapping voxels in Figure 7) are the major challenge for 3D dissection, we prefer to construct P_i from such voxels first, to improve compatibility of the remaining volumes R^k_i ($k = 1, 2$). We also prefer to start from voxels with low accessibility to avoid fragmental voxels in the remaining volumes. Thus we evaluate the suitability of each boundary voxel v of R^k_{i-1} as a seed via:

$$S(v) = \frac{1}{O(v)A(v)^{\gamma_A}} \tag{3}$$

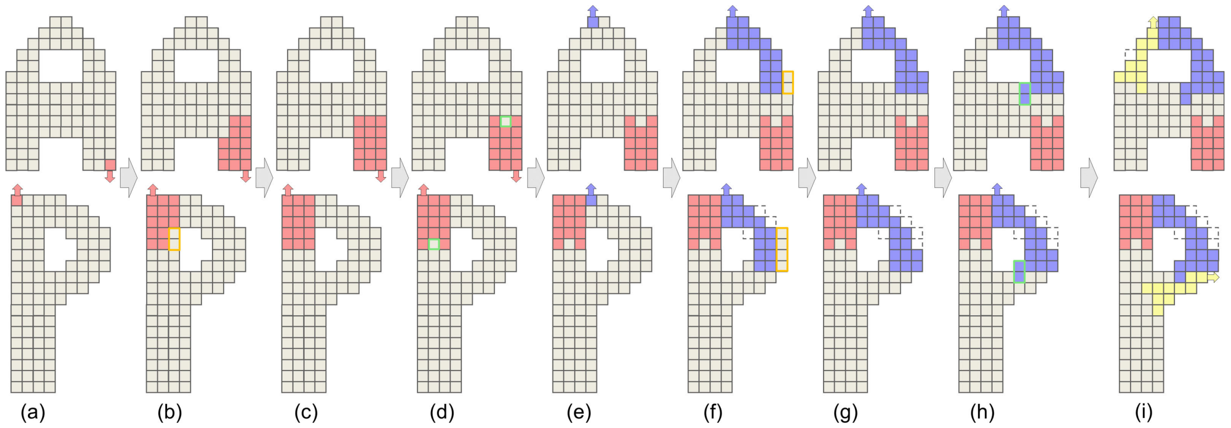


Figure 9: *Constructing a few pieces on a 2D example. (a) Co-seed, (b) co-expand, (c) co-clean, and (d) stabilize the first piece; (e) co-seed, (f) co-expand, (g) co-clean, and (h) stabilize the second piece; (i) construct the third piece. Fragmental voxels that are avoided by the co-cleaning process are highlighted in yellow in (b&f), and the co-blocking structures are highlighted in green in (d&h).*

where γ_A is a weighting factor for accessibility ($\gamma_A = 3.0$ in our experiments). We choose L ($L = 15$ in our experiments) voxels with the highest $S(v)$ as candidates of our seed voxels.

2. *Generate Corresponding Seeds.* To generate corresponding seeds from the candidates, we need to consider their compactness (i.e., $C(v)$) since we want the seeds (and thus P_i) to be closer to the previously extracted pieces (i.e., P_1, ..., P_{i-1}) such that the remaining volumes can be compact. Moreover, we prefer a pair of seeds with similar local shapes around them to facilitate the construction of P_i. To achieve this, we first identify the local shape around each seed candidate and extract the centroids of these voxels. Next, we perform eigendecomposition on the covariance matrix of these centroids and obtain the three principal axes and the eigenvalues $\lambda_1, \lambda_2, \lambda_3$. For each pair of candidates v^1, v^2 from $\mathbf{M}^1, \mathbf{M}^2$, we evaluate
$$S(v^1, v^2) = \frac{C(v^1)C(v^2)}{\text{dist}(F^1, F^2)} \tag{4}$$
where $F^k = [\lambda_1^k, \lambda_2^k, \lambda_3^k]$. We then select K ($K = 10$ in all our experiments) pairs with the highest $S(v^1, v^2)$ as the final set of corresponding seeds. For each pair, we randomly select one axial direction that $v^k (k = 1,2)$ can be directly taken out, as the extraction direction of P_i^k in $\mathbf{M}^k$, denoted as d_i^k; see Figure 9 (a&e) for two examples.
3. *Determine Piece Transform.* For each pair of corresponding seeds, we obtain T_i by computing the transform that aligns the three principal axes associated with the seeds, and snapping it to the closest axis-to-axis transform. Due to ambiguity of the principal axis directions obtained from the eigendecomposition, we may have four possible T_i.

Co-expanding. Given a pair of corresponding seed voxels (v^1, v^2), their associated extraction directions (d_i^1, d_i^2), and the piece transform matrix T_i, we iteratively augment each seed voxel with more voxels to increase the size of P_i^1 and P_i^2, while ensuring they have exactly the same shape.

1. *Identify Candidate Voxel Pairs.* We first identify all neighboring voxels of P_i^1, and transform them into $\mathbf{M}^2$ using T_i. Afterwards, we only keep the neighboring voxels in $\mathbf{M}^1$ that have corresponding voxels in $\mathbf{M}^2$, and consider them as the candidate voxel pairs for co-expanding P_i.
2. *Evaluate Candidate Voxel Pairs.* For each pair of candidate voxels (u^1, u^2), we identify all voxels above u^k ($k = 1,2$) along the extraction direction d_i^k, and denote the set of these voxels together with u^k as U^k. Note that all voxels in U^k will eventually be assigned to P_i^k to ensure its mobility along d_i^k. Specifically, the furthest voxel along direction d_i^k is denoted as $u^{k'}$ (i.e., a boundary voxel of R_{i-1}^k). A good candidate pair should facilitate 3D dissection (e.g., lower accessibility and lower compatibility), and at the same time preserve appearance of input models. Thus we evaluate each candidate pair using:
$$E(u^1, u^2) = \sum_{k=1,2} \sum_{u \in U^k} A(u)O(u) + \omega(\|U^2\| - \|U^1\|)N(u^{2'}), \tag{5}$$
where the second term represents the amount of modification on the input models' appearance, and ω is a weight ($\omega = 0.5$ in all our experiments). Here, we assume U^2 has a larger number of voxels than U^1, and thus we need to delete $\|U^2\| - \|U^1\|$ voxels from P_i^2 (including $u^{2'}$) to ensure P_i^1 and P_i^2 have exactly the same shape. Moreover, since $u^{2'}$ will be deleted while $u^{1'}$ will be kept, only the appearance of $\mathbf{M}_2$ could be changed (see Figure 10(b) bottom). We penalize this by multiplying $N(u^{2'})$ in the second term. In particular, the second term is equal to zero if $u^{2'}$ is not a boundary voxel of $\mathbf{M}^2$ since $N(u^{2'}) = 0$.

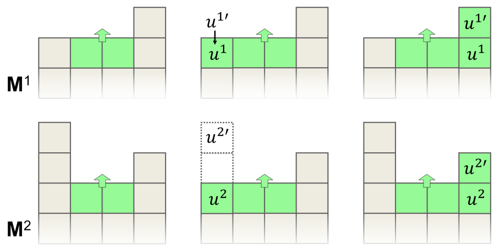


Figure 10: *(a) A green piece with an upward extraction direction. (b&c) Two possible solutions to co-expand the piece, where (b) is a worse choice since it needs to delete two voxels (i.e., $u^{2'}$ and the one below it) in* $\mathbf{M}^2$.

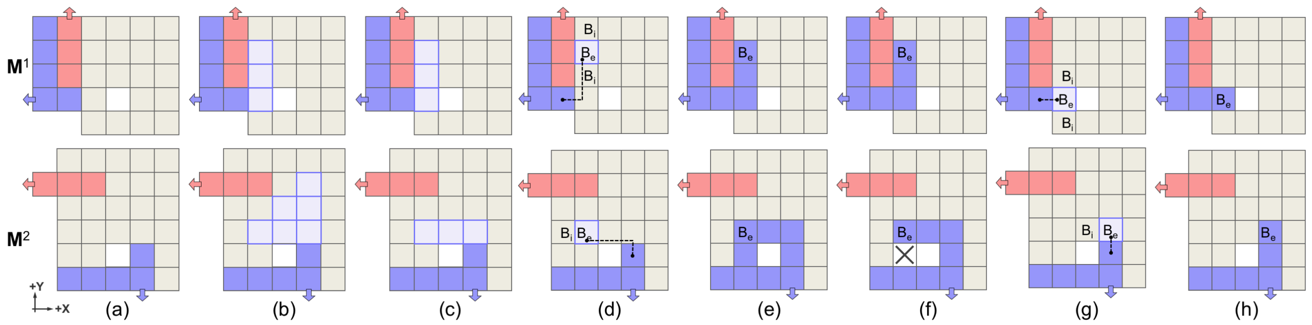


Figure 11: *Stabilize a dissected piece in the two models. (a) The dissected blue piece; (b) identify blockee voxel candidates separately (in light blue); (c) identify blockee voxel pairs (in light blue); (d) select a blockee voxel (B_e) pair as well as associated blocking voxels (B_i's), and identify the shortest path from B_e to the blue piece; (e) connect B_e to the blue piece and ensure its mobility; (f) one voxel (marked with a cross) in $\mathbf{M}^2$ needs to be deleted to ensure compatibility of the piece; (g) another better choice of the blockee voxel pair and (h) the resulting piece.*

3. *Iterate Co-expanding Process.* We keep randomly selecting a pair of voxels from the candidates with probability inversely proportional to $E(u^1,u^2)$, and include the voxels to P_i^1 and P_i^2 respectively. We iterate the process of identifying candidates, evaluating candidates, and adding voxels to expand P_i, until the number of voxels in P_i is around $0.6M$, where M is the desired number of voxels for each piece; see Figure 9 (b&f).

Co-cleaning. The co-expanding process may leave the remaining volumes with fragmented voxels, which need to be included into P_i such that the remaining volumes are still connected. In particular, all voxels above the fragmented voxels along extraction direction d_i^k should also be included into P_i^k to ensure its movement along d_i^k. In addition, for each identified voxel in one model, its corresponding voxel in the other model should also be included in P_i; compare Figure 9 (b&c) for an example. For each fragmented voxel without a corresponding voxel in the other model, we delete it from the remaining volume; compare Figure 9 (f&g) for an example.

The co-cleaning process may fail due to two reasons. First, too many voxels may need to be included in P_i to resolve the fragmented voxels, resulting in a too large piece. In our experiments, P_i with more than $1.4M$ voxels will be discarded. Second, a number of model boundary voxels could be deleted by the co-cleaning process. We discard P_i if the total noticeability of all the deleted voxels is larger than a threshold (set as $0.05M$ in our experiments).

This dissection process outputs a number of (less than K) candidates of P_i that have exactly the same shape in $\mathbf{M}^1$ and $\mathbf{M}^2$. For each candidate of P_i, we know its extraction direction in each model, as well as the transform T_i to reconfigure it across the two models.

5.3. Stabilize a Piece

The dissection process simply ensures that each P_i^k $(k=1,2)$ is movable along its extraction direction d_i^k. However, it is possible that P_i^k is movable along more than one axial directions, e.g., the blue piece in $\mathbf{M}^1$ in Figure 11(a) is movable along $-x$ and $-y$. It is also possible that P_i^k together with some previously extracted piece(s) are movable along more than one axial directions, e.g., the red and blue pieces in $\mathbf{M}^1$ in Figure 11(a) are movable together along $-x$, $-y$, and $+y$. To stabilize P_i (as well as groups that contain P_i) in both models, we modify the geometry of P_i according to our formal model in Section 4.3, with the following steps:

Check Satisfaction of Our Formal Model. For the dissected P_i^k in $\mathbf{M}^k$, we identify the set of axial directions $\mathbf{D}_i^k$ along which R_i^k blocks P_i^k. If both $\mathbf{D}_i^1$ and $\mathbf{D}_i^2$ contain five directions (there are at most five directions in $\mathbf{D}_i^k$, since P_i^k is movable along d_i^k), then P_i^1 and P_i^2 are already steady according to our formal model (see again Subsection 4.3), and the following steps can be skipped.

Otherwise, recall that our formal model allows a piece P_i^k to be stabilized by R_i^k together with a piece that has been previously extracted. Thus, we identify all the previously extracted pieces P_j^k $(1 \le j < i)$ that are neighbours of P_i^k, and check for each of them whether the three requirements in the formal model are satisfied. If we can find such a piece for P_i^1 and P_i^2 respectively, then P_i is also steady according to our formal model and we can skip the following steps.

Identify Missing Blocking Directions. If none of the neighboring pieces P_j^k $(1 \le j < i)$ of P_i^k satisfy all the requirements, we identify for each P_j^k the missing directions along which R_i^k should block P_i^k to satisfy the requirements. We first compute the required directions $\widehat{\mathbf{D}}_i^k$ along which R_i^k should block P_i^k according to the three requirements of the formal model. Then the set of missing blocking directions of P_i^k is $\widehat{\mathbf{D}}_i^k \setminus \mathbf{D}_i^k$. Consider the blue piece (i.e., P_2^1) in $\mathbf{M}^1$ in Figure 11(a) as an example. The set $\widehat{\mathbf{D}}_2^1$ is computed as follows (we remove the superscript for simplicity as all the computations are done in $\mathbf{M}^1$):

- P_2 should be immobilized by P_1 and R_2 and only movable along d_2 (i.e., $-x$). As P_1 blocks P_2 in $\{+x,+y\}$, R_2 should block P_1 in $\{+x,+y,-y\} \setminus \{+x,+y\} = \{-y\}$.
- R_2 should block P_2 from moving along d_1, if $d_1 \neq d_2$. In the current example, $d_1 = +y$, $d_2 = -x$. Thus R_2 should block P_2 in $\{+y\}$.
- For each direction $d' \in \{-x,+x,-y,+y\} \setminus \{d_1,d_2\}$ (i.e., $d' \in \{+x,-y\}$), if P_2 blocks P_1 from moving along d' but R_2 does not, then R_2 should block P_2 from moving along d'. As P_2 blocks P_1 in $\{-x,-y\}$ while R_2 does not, R_2 should block P_2 in $\{+x,-y\} \cap \{-x,-y\} = \{-y\}$.

To summarize, we have $\widehat{\mathbf{D}}_2^1 = \{-y,+y\}$. As $\mathbf{D}_2^1 = \{+x\}$, the missing blocking directions are $\{-y,+y\}$.

For each P_j^k, we compute the missing blocking directions of P_i^k following the above procedure. We then choose a P_j^k with the smallest number of missing blocking directions, and employ these directions to guide the following steps.

Find Blockee Voxels Separately. Given the identified missing blocking directions of P_i^k ($k = 1,2$), we next modify its geometry so that it can be blocked by R_i^k along these directions. Inspired by [SFCO12], we find additional blockee and blocking voxels and restrict the movement of P_i^k by assigning blockee voxel (B_e^k) to P_i^k and leaving blocking voxel(s) (B_i^k's) in R_i^k, and vice versa; see Figure 9(d&h) for two examples.

In detail, we do breadth-first traversal for the nearby voxels of P_i^k that belong to R_i^k, and choose those voxels that have neighboring voxels along each of the missing blocking directions as blockee voxel candidates; see Figure 11(b) for an example. When the set of missing blocking directions of P_i^k is empty, we consider all the nearby voxel of P_i^k that belong to R_i^k as the blockee voxel candidates.

Identify Blockee Voxel Pairs. Among all the blockee voxel candidates identified separately in $\mathbf{M}^1$ and $\mathbf{M}^2$, we consider the corresponding blockee voxel candidates across the two models as "blockee voxel pairs"; see Figure 11(c). We select N ($N = 10$ in all our experiments) blockee voxel pairs as the candidates and rank them according to their distance to P_i.

Construct Co-blocking Structure. For each blockee voxel pair candidate B_e^k ($k = 1,2$), we try to connect it with P_i^k using a co-blocking structure while keeping the blocking voxels B_i^k's in R_i^k; see Figure 11(d). Specifically, we identify a set of shortest path candidates from P_i^k to B_e^k within R_i^k, without crossing the related blocking voxels and the voxels below them along d_i^k. Since we need to guarantee that the modified piece P_i^k is still movable along d_i^k, not only voxels on a shortest path but also voxels above the path along d_i^k should be assigned from R_i^k to P_i^k; see Figure 11(e). For these identified voxels in one model, we need to delete some of them from the model if we cannot find a corresponding voxel in the other model; see Figure 11(f). Finally, we choose the co-blocking structure candidate that has the least total value of $E(u^1,u^2)$, and assign all voxels in the structure from R_i^k to P_i^k for each model; see Figure 11(h).

The output of this stabilization process is a number of (less than K) candidates of P_i that satisfy the requirements of both 3D dissection and generalized interlocking.

5.4. Iterate Piece Construction

Our system iteratively constructs pieces from P_1 to P_n, where $n = m-1$, following the procedures of dissecting and stabilizing a piece as described above. When constructing a piece P_i, we maintain at most K candidates. In this way, our approach keeps expanding a construction tree, where the root represents the input models (Figure 3(a)), the leaves represent different 3D dissection results (Figure 3(d)), and the other nodes represent the intermediate piece construction configurations (Figure 3(b&c)). In case no valid candidate of P_i can be found, a backtracking to siblings of the parent node in the tree is required. Since most chains in the tree stop before reaching a depth of n, our algorithm requires a manageable amount of memory.

A candidate of P_i is considered valid if it satisfies all of the following requirements:

- The number of voxels in P_i is within $[0.6M, 1.4M]$;
- The total noticeability of voxels in $D(P_i^1)$ and in $D(P_i^2)$ is lower than a threshold α, where $D(P_i^k)$ denotes the set of deleted model boundary voxels when constructing P_i^k;
- The set of blocking voxels of each P_j^k ($1 \leq j \leq i-1$) along each blocking direction should not become empty after deleting voxels in $D(P_i^k)$, to maintain the interlocking property.

We evaluate all the valid candidates using the following measure, and select the one with the smallest measure as P_i:

$$F = \omega_1 F_1 + \omega_2 F_2 - \omega_3 F_3 \quad (6)$$

Here $F_1 = \sum_{k=1,2} \sum_{u \in D(P_i^k)} N(u)$ avoids excessive modification on the input models' appearance by penalizing the total noticeability of all the deleted model boundary voxels; F_2 and F_3 facilitate further dissection of the two remaining volumes R_i^1 and R_i^2: $F_2 = \left| \|R_i^1\| - \|R_i^2\| \right|$ penalizes the difference between the numbers of their voxels, while $F_3 = \sum_{k=1,2} \sum_{v \in R_i^k} A(v)O(v)/\|R_i^k\|$ encourages them to be less fragmented and more compatible; the weights ω_1, ω_2 and ω_3 are set to 1.0, 0.5 and 2.5 in all our experiments.

After constructing a valid P_n, we have two remaining pieces R_n^1 and R_n^2 that are unlikely to have exactly the same shape by default. Hence, we identify all corresponding voxels in R_n^1 and R_n^2 to form the last piece R_n, and delete the remaining voxels. We check the validity of R_n using the same requirements as P_i, and terminate the dissection if R_n is valid.

5.5. Post-process Puzzle Pieces

As the piece construction process allows deleting voxels, the constructed puzzle may have quite different appearance compared to the input models. To alleviate this issue, we post-process the pieces and add back some deleted model boundary voxels, while still satisfying the dissection and interlocking requirements. Our approach is based on the observation that deleting internal voxels or unnoticeable boundary voxels results in fewer modifications on the appearance than deleting noticeable boundary voxels (see again Section 5.1). It consists of the following steps.

Rank the Deleted Voxels. We only add back deleted voxels on the model boundary to restore the local appearance. We evaluate the priority of such a deleted voxel according to its largest distance to the generated puzzle along the six axial directions in both forms. A voxel with a larger distance indicates a deeper and/or wider hole on the puzzle surface, and is given higher priority. Candidates with the same distance are ranked according to their noticeability.

Add back the Deleted Voxels. For each ranked voxel v_s^1 in $\mathbf{M}_1$, we first identify its neighboring puzzle pieces and assign v_s^1 to a randomly selected neighboring piece P_j^1. Afterwards, the corresponding voxel v_s^2 on $\mathbf{M}_2$ is processed as follows:

- v_s^2 is directly added to P_j^2 if the voxel has been deleted (see Figure 12(b));
- If v_s^2 is occupied by another piece $P_{j'}^2$, then we delete this voxel from $P_{j'}^2$ and add it to P_j^2 (see Figure 12(d)). To satisfy the dissection requirement, after deleting the voxel from $P_{j'}^2$, the corresponding voxel in $\mathbf{M}_1$ should be deleted from $P_{j'}^1$ (see Figure 12(e)). If

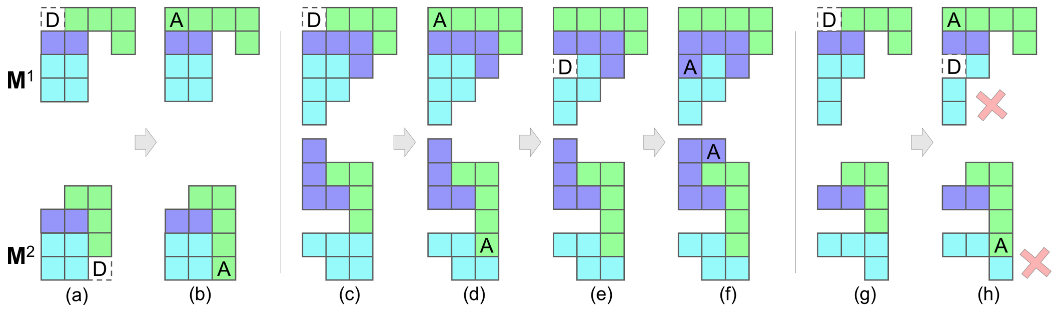


Figure 12: *(a&b) Add back deleted model boundary voxels (marked as "D"), with the added voxels marked as "A". (c-f) Perform voxel adding back twice to fix a deleted boundary voxel in (c). (g&h) A failure case where the cyan piece becomes disconnected.*

this deleted voxel is still a model boundary voxel, we need to add it back using the same strategy on v_s^1 (see Figure 12(f)).

Validate the Results. After adding back a voxel, we check validity of the resulting pieces (see Figure 12(h)) and the puzzle piece mobility, as well as test the interlocking property of the whole puzzle based on our formal model. If any of the above checks fails, we undo the operation. For each ranked voxel, we add back deleted voxels and validate the result iteratively until finding a valid solution or the number of iterations reaches 10.

We iterate the above three steps and count the number of deleted model boundary voxels before and afterwards, denoted as N_{before} and N_{after}. We terminate the process when N_{after}/N_{before} is close to 1; i.e., when the appearance cannot be improved by post-processing. For further improvement, it is also possible to add back a deleted voxel in one model while introducing an extra voxel in the other model, as long as users do not feel the introduced voxel is distracting.

6. Experiment Results

Implementation. We implement our method in C++ and run it on a laptop computer with a 2.8GHz CPU and 16GB memory. We allow users to specify the resolution of the input voxelized models, usually within a range between $8 \times 8 \times 8$ and $35 \times 35 \times 35$. This is because lower resolution models may result in difficult-to-recognize shapes while higher resolution models require an excessive use of computing resources. We also allow users to specify the number (m) of pieces to be generated, usually between 10 and 30. A larger m could result in failure of interlocking since each piece has too few voxels while a smaller m could lead to failure of 3D dissection since two different shapes is hardly to be represented with a too small set of common pieces. In general, two input models with larger difference in their shapes require a larger m. Our results are also affected by other parameters. For example, a larger noticeability threshold makes it easier to find a valid solution, yet allows larger modifications on the input models' appearance.

Results and Statistics. Our method can create 3D dissection puzzles from 3D models of various shapes and topologies (see Figures 1, 3, 13), e.g., TEAPOT and SNAIL in Figure 1 with different topologies (genus one vs zero), ANGRY BIRD and SHUTTLE in Figure 13 with very different shape (spherical vs elongated), and BUNNY and CUBE in Figure 13 that dissects an organic model into a primitive shape. Figure 14 shows the sequence of re-assembling DUCK into TEAPOT. Please refer to the supplementary video for the animated results.

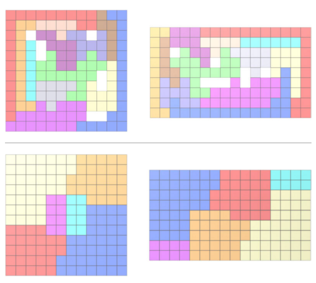
Table 1: *Statistics of the 3D dissection results shown in this paper.*

Fig.	Forms	Resolutions	# Pieces	T-o-C (min)	T-o-P (min)
1	Teapot - Snail	25x13x16 - 29x11x14	25	66.2	38.1
3	Kitten - Squirrel	14x20x12 - 11x20x12	17	28.5	11.3
13	Angry Bird - Shuttle	11x16x11 - 16x11x25	15	120.2	13.4
	Bunny - Cube	15x15x12 - 10x10x10	19	10.7	18.2
	Pear - Banana	15x15x25 - 35x23x9	24	180.3	39.2
	Bowl - Coke	18x9x18 - 8x8x26	21	160.8	34.9
	Wolf Head - Rhino Head	19x20x24 - 16x21x24	25	83.7	30.5
	Mario - Treasure Chest	19x30x16 - 20x13x17	29	94.5	49.2
14	Duck - Teapot	27x13x17 - 16x17x19	22	112.2	33.7
17	Stool - Dinosaur	12x12x12 - 8x22x17	15	30.1	23.1
	Ring - Cube	13x15x13 - 8x8x8	14	19.2	14.3
	Sofa - Chair	10x9x8 - 8x8x12	16	7.3	4.8
	Barrel - Elephant	15x15x17 - 12x20x15	19	93.2	42.8

Table 1 presents the statistics of our results, including the resolution of input voxelized models, the number of pieces, the time of constructing the puzzle pieces (T-o-C) and the time of the post-processing (T-o-P). In practice, the timing is affected by multiple factors. It usually takes longer time to generate results from higher resolution models (e.g., MARIO - TREASURE CHEST in Figure 13) as well as input models with very different shapes (e.g., ANGRY BIRD - SHUTTLE in Figure 13).

Due to the strong constraints induced by 3D dissection and interlocking, we cannot always avoid noticeable holes in the results; see the CUBE and PEAR in Figure 13. This strategy of modifying shape appearance to find valid dissection results also has been explored in a recent work [DYYT17].

Comparison with [Zhou et al. 2012]. Our method can be easily applied to handle 2D input models. The inset figure compares our approach with [ZW12] on designing a 2D dissection puzzle SQUARE - RECTANGLE. Our result has a larger number of pieces (12 vs 6), and our pieces have more irregular shapes due to the interlocking requirement. As this is a 2D example, our approach only considers the outermost voxels as boundary voxels, and allows deleting some internal voxels, resulting in holes in the puzzle. These holes may make it difficult to assemble the puzzle pieces without a manual since the users cannot foresee where the holes are located in the puzzle.



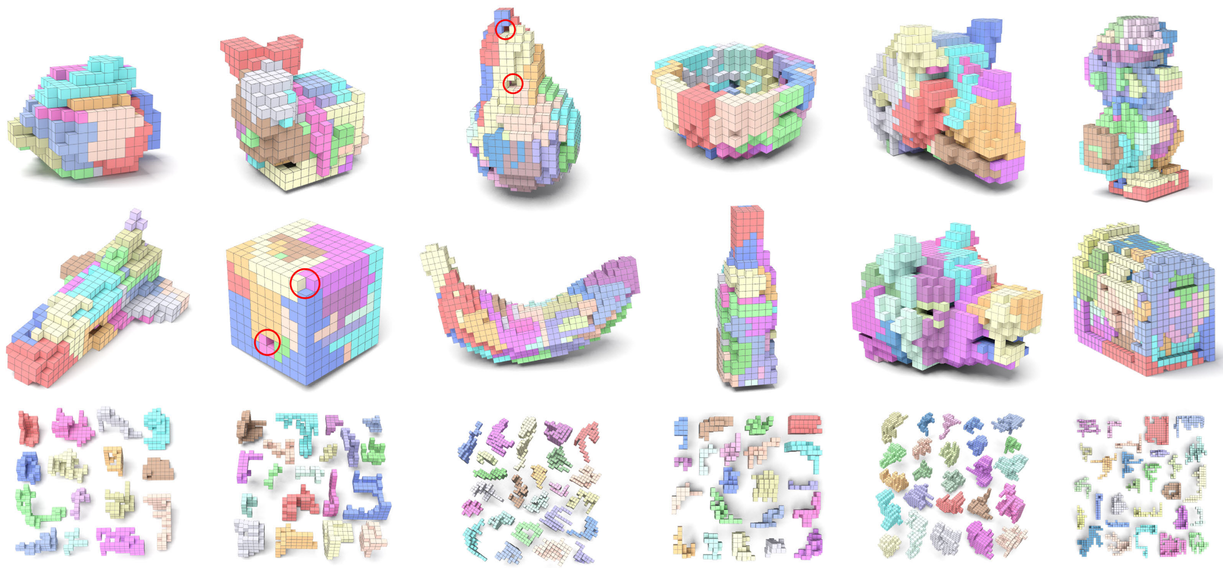


Figure 13: *Our generated dissection puzzles (top, middle) and the pieces (bottom). From left to right:* ANGRY BIRD - SHUTTLE, BUNNY - CUBE, PEAR - BANANA, BOWL - COKE, WOLF HEAD - RHINO HEAD, *and* MARIO - TREASURE CHEST. *Some undesired holes are highlighted with red circles.*

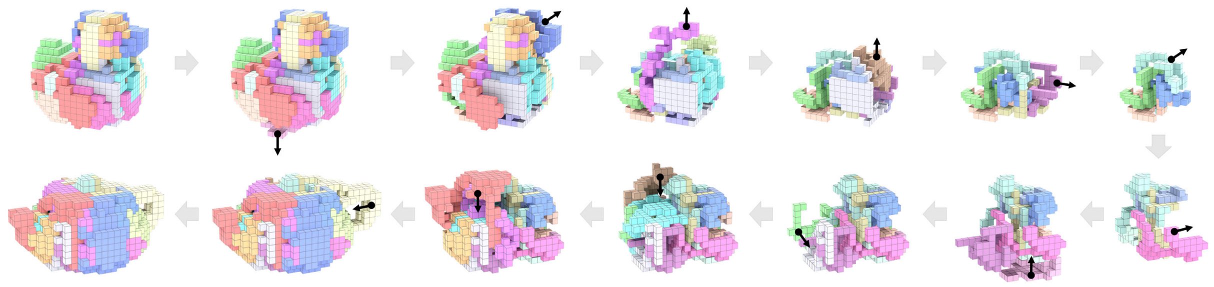


Figure 14: *Snapshots of reconfiguring* DUCK *into* TEAPOT*; the black arrow shows the moving direction of the associated part for (dis)assembly.*

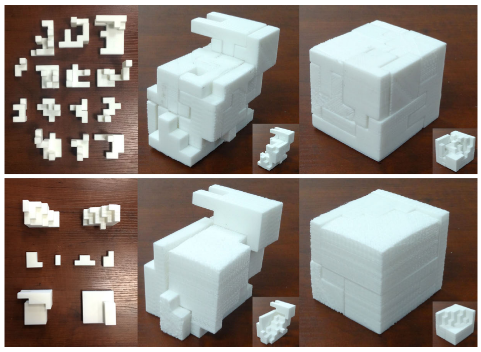


Figure 15: *3D dissection puzzles* BUNNY - CUBOID *designed by our approach (top) and by [ZW12] (bottom).*

We further compare our approach with [ZW12] on a 3D dissection puzzle BUNNY - CUBOID in Figure 15. Our result still has a larger number of pieces (14 vs 8), and our pieces have more similar sizes. More importantly, our puzzle can be manipulated in the hand without falling apart, thanks to the interlocking property, which is not the case for the puzzle by [ZW12].

Comparison with [Song et al. 2012]. We also compare our generalized interlocking model with the recursive interlocking model in [SFCO12]. Generally speaking, our formal model is less restrictive since it takes advantage of friction to immobilize pieces and does not require the puzzle to be interlocking for each (dis)assembly state. Although this makes the resulting puzzles less steady than those from [SFCO12], it provides more flexibility for designing interlocking assemblies.

To evaluate the flexibility of our formal model, we conduct a quantitative comparison on designing 4^3 interlocking CUBEs with m ($6 \leq m \leq 10$) pieces and require all the pieces to have exactly the same number of voxels or at most one voxel difference. If a result cannot be generated within 48 hours, we consider no result can be found for this case. Figure 16 shows the results from each formal model, where recursive interlocking CUBE can at most have 8 pieces while generalized interlocking CUBE can reach 10 pieces. Moreover, for puzzles with the same m, our formal model can find a result within a much shorter time.

Fabrication. We 3D print some of our dissection results, as shown in Figures 1, 3, 15, 17. Among them, BUNNY-CUBOID, RING-CUBE and SOFA-CHAIR are printed using an Ultimaker 2+ FDM printer with PLA plastic material, while all the others are printed using a Stratasys SLA printer. In practice, we select a small tolerance for 3D printing the puzzle pieces, to ensure sufficient friction for

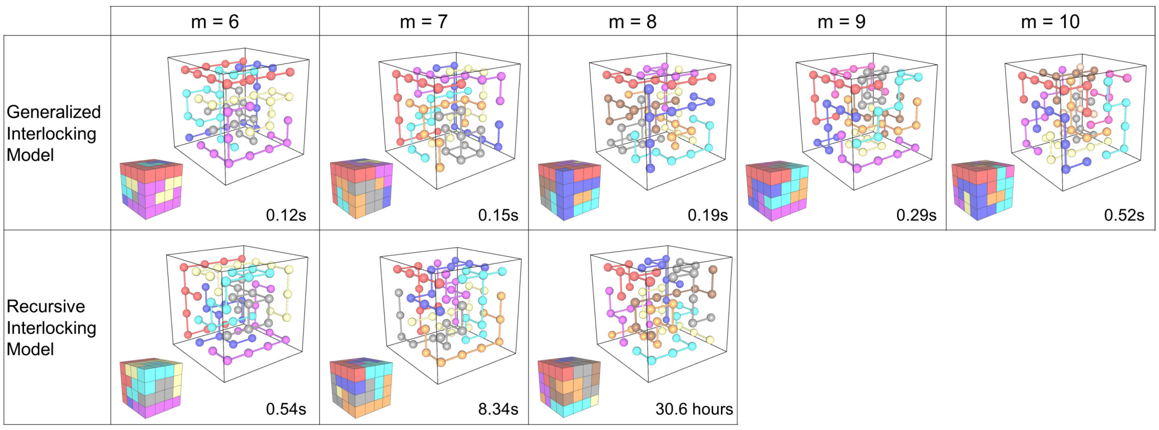


Figure 16: *Designing 4^3 interlocking* Cubes *with different m using our generalized interlocking model and recursive interlocking model [SFCO12]. The corresponding computation time for each result is shown at the lower right corner.*

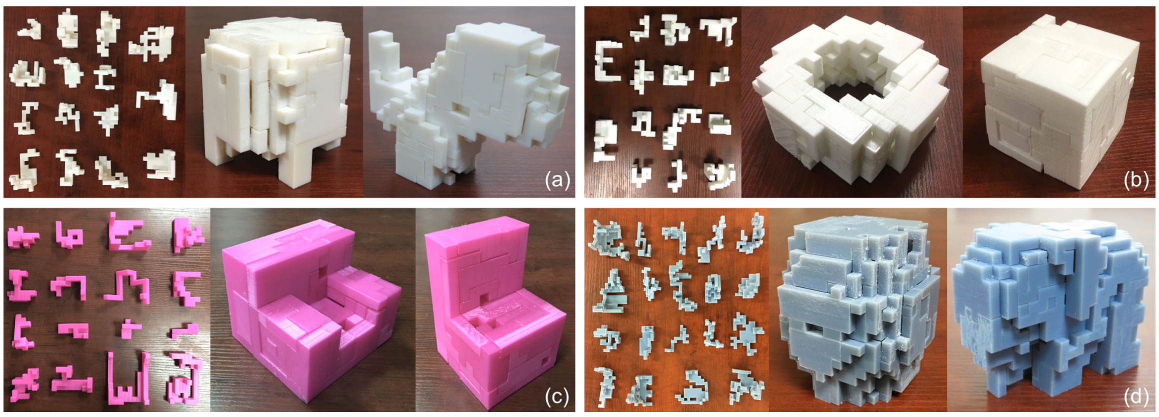


Figure 17: *3D-printed puzzle pieces and the two assembled forms: (a)* Stool - Dinosaur*, (b)* Ring - Cube*, (c)* Sofa - Chair*, (d)* Barrel - Elephant.

immobilizing the pieces. It typically took 10 to 30 mins for a novice user to assemble the puzzle with the guidance of a manual, for which the actual assembly time depends on the number of component pieces as well as their shapes.

To evaluate steadiness of our puzzles subject to external forces, we conducted two experiments. In the first one, we shake the assembled puzzles and find the puzzles remain steady upon shaking. The second experiment is to drop the puzzles from a height of 80cm. To avoid breaking the puzzle pieces, we drop the puzzle onto a blanket rather than a hard ground. We find that our puzzles remain well connected after the dropping test, validating the effectiveness of our formal model. Please refer to the supplementary video for more details.

7. Conclusion

This paper presents a computational method for designing steady 3D dissection puzzles. For this purpose, we develop a novel approach to dissect two voxelized models into a common set of puzzle pieces while allowing small modifications on the input models' appearance. We also propose a generalized interlocking formal model for connecting puzzle pieces with the help of friction. Guided by this formal model, our approach can automatically modify each puzzle piece such that the resulting puzzle is steady. We demonstrate the effectiveness of our approach on 3D models of various shapes and topologies, show advantages of our approach and formal model over existing works, and validate the steadiness of our designed dissection puzzles on a few 3D printed examples.

Limitations and Future Work. First, our approach takes voxelized models as inputs, and the results may be less appealing (or even unrecognizable) for low-resolution models. In the future, we may extend our approach to handle 3D models with smooth appearance [Fre97, Fre02]. Second, our approach needs to modify the input model appearance and/or its interior volume to find valid dissection results. Minimizing such modification, especially on salient shape features, would be an interesting problem to explore. Third, we require the number of puzzle pieces to be within a certain range. Finding the minimum number of pieces for 2D/3D dissection remains a challenging open problem [LF72, Fre17]. Fourth, our current approach only handles two input models. Extending it to support three or more input models [Fre09] would be an interesting future work. Fifth, we plan to conduct a formal stability analysis [YKGA17] on our resulting puzzles, and employ our interlocking model for constructing larger-scale steady assemblies such as furniture or timber structures. Lastly, we consider making 3D puzzles steady by using some alternative approaches (e.g., form-fitting boxes or magnets [Fre97]) as an exciting research topic.

Acknowledgments

This work was partially supported by China Grant (U1636215), the National Key R&D Program of China (2018YEB1004003), the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK 14203416 and 14201717), National Natural Science Foundation of China (61672482) and the One Hundred Talent Project of the Chinese Academy of Sciences.

References

[AAC*12] ABBOTT T. G., ABEL Z., CHARLTON D., DEMAINE E. D., DEMAINE M. L., KOMINERS S. D.: Hinged dissections exist. *Discrete & Comp. Geom. 47*, 1 (2012), 150–186. 1, 2

[BBJP12] BÄCHER M., BICKEL B., JAMES D. L., PFISTER H.: Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (2012), 47:1–47:9. 2

[BCMP18] BICKEL B., CIGNONI P., MALOMO L., PIETRONI N.: State of the art on stylized fabrication. *Comp. Graph. Forum 37* (2018). 2

[Bol32] BOLYAI F.: *Tentamen juventutem. Typis Collegii Refomatorum per Josephum et Simeonem Kali.* Maros Vásárhely, 1832. 2

[CCA*12] CALÌ J., CALIAN D. A., AMATI C., KLEINBERGER R., STEED A., KAUTZ J., WEYRICH T.: 3D-printing of non-assembly, articulated models. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (2012), 130:1–130:8. 2

[CKU07] CZYZOWICZ J., KRANAKIS E., URRUTIA J.: Rectilinear glass-cut dissections of rectangles to squares. *Applied Mathematical Sciences 1*, 52 (2007), 2593–2600. 2

[Coh75] COHN M. J.: Economical triangle-square dissection. *Geometriae Dedicata 3*, 4 (1975), 447–467. 1, 2

[CPMS14] CIGNONI P., PIETRONI N., MALOMO L., SCOPIGNO R.: Field-aligned mesh joinery. *ACM Trans. on Graph. 33*, 1 (2014), 11:1–11:12. 4

[Dud07] DUDENEY H. E.: *The Canterbury Puzzles, and Other Curious Problems*. Thomas Nelson and Sons, 1907. 1, 2

[DYYT17] DUNCAN N., YU L.-F., YEUNG S.-K., TERZOPOULOS D.: Approximate dissections. *ACM Trans. on Graph. (SIGGRAPH Asia) 36*, 6 (2017), 182:1–182:14. 1, 2, 10

[Fre97] FREDERICKSON G. N.: *Dissections: Plane and Fancy*. Cambridge University Press, 1997. 1, 12

[Fre02] FREDERICKSON G. N.: *Hinged Dissections: Swinging and Twisting*. Cambridge University Press, 2002. 2, 12

[Fre07a] FREDERICKSON G. N.: Symmetry and structure in twist-hinged dissections of polygonal rings and polygonal anti-rings. *Proc. Bridges Donostia: Mathematics, Music, Art, Architecture, Culture* (2007), 21–28. 2

[Fre07b] FREDERICKSON G. N.: Unexpected twists in geometric dissections. *Graphs and Combinatorics 23*, 1 (2007), 245–258. 2

[Fre08] FREDERICKSON G. N.: Designing a table both swinging and stable. *The College Mathematics Journal 39*, 4 (2008), 258–266. 2

[Fre09] FREDERICKSON G. N.: Casting light on cube dissections. *Mathematics Magazine 82*, 5 (2009), 323–331. 1, 12

[Fre17] FREDERICKSON G. N.: *Ernest Irving Freese's Geometric Transformations: The Man, The Manuscript, The Magnificent Dissections!* World Scientific, 2017. 12

[FSY*15] FU C.-W., SONG P., YAN X., YANG L. W., JAYARAMAN P. K., COHEN-OR D.: Computational interlocking furniture assembly. *ACM Trans. on Graph. (SIGGRAPH) 34*, 4 (2015), 91:1–91:11. 3, 4

[Ger33] GERWIEN P.: Zerschneidung jeder beliebigen anzahl von gleichen geradlinigen figuren in dieselben stücke. *Journal für die reine und angewandte Mathematik (Crelle's Journal) 10* (1833), 228–224. 2

[GJG16] GARG A., JACOBSON A., GRINSPUN E.: Computational design of reconfigurables. *ACM Trans. on Graph. (SIGGRAPH) 35*, 4 (2016), 90:1–90:14. 2

[HCLC16] HUANG Y.-J., CHAN S.-Y., LIN W.-C., CHUANG S.-Y.: Making and animating transformable 3D models. *Computers & Graphics (Proc. of CAD/Graphics) 54* (2016), 127–134. 2

[HN06] HUNGERBÜHLER N., NÜSKEN M.: Delian metamorphoses. *Elemente der Mathematik 61*, 1 (2006), 1–19. 1, 2

[KKU00] KRANAKIS E., KRIZANC D., URRUTIA J.: Efficient regular polygon dissections. *Geometriae Dedicata 80*, 1 (2000), 247–262. 1, 2

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3D-printable parts. *ACM Trans. on Graph. (SIGGRAPH Asia) 31*, 6 (2012), 129:1–129:9. 4

[LF72] LINDGREN H., FREDERICKSON G.: *Recreational Problems in Geometric Dissections and How to Solve Them*. Dover Publications, 1972. 1, 12

[LHAZ15] LI H., HU R., ALHASHIM I., ZHANG H.: Foldabilizing furniture. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (2015), 90:1–90:12. 2

[LMaH*18] LI S., MAHDAVI-AMIRI A., HU R., LIU H., ZOU C., KAICK O. V., LIU X., HUANG H., ZHANG H.: Construction and fabrication of reversible shape transforms. *ACM Trans. on Graph. (SIGGRAPH Asia) 37*, 6 (2018), 190:1–190:14. 2

[Per72] PERIGAL H.: On geometric dissections and transformations. *Messenger of Mathematics (second series) 1* (1872), 103–105. 1

[Rog02] ROGOWSKI G.: *The complete illustrated guide to joinery*. Taunton Press, 2002. 4

[SDW*16] SONG P., DENG B., WANG Z., DONG Z., LI W., FU C.-W., LIU L.: CofiFab: Coarse-to-fine fabrication of large 3D objects. *ACM Trans. on Graph. (SIGGRAPH) 35*, 4 (2016), 45:1–45:11. 3, 4

[SFCO12] SONG P., FU C.-W., COHEN-OR D.: Recursive interlocking puzzles. *ACM Trans. on Graph. (SIGGRAPH Asia) 31*, 6 (2012), 128:1–128:10. 2, 4, 6, 9, 11, 12

[SFJ*17] SONG P., FU C.-W., JIN Y., XU H., LIU L., HENG P.-A., COHEN-OR D.: Reconfigurable interlocking furniture. *ACM Trans. on Graph. (SIGGRAPH Asia) 36*, 6 (2017), 174:1–174:14. 2, 3, 4

[SFLF15] SONG P., FU Z., LIU L., FU C.-W.: Printing 3D objects with interlocking parts. *Comp. Aided Geom. Des. 35-36* (2015), 137–148. 3, 4

[SZ15] SUN T., ZHENG C.: Computational design of twisty joints and puzzles. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (2015), 101:1–101:11. 2

[The18] THEOBALD G.: Solid dissections, 2018. `http://gavin-theobald.uk/HTML/3D.html`. 1

[WSP18] WANG Z., SONG P., PAULY M.: DESIA: A general framework for designing interlocking assemblies. *ACM Trans. on Graph. (SIGGRAPH Asia) 37*, 6 (2018), 191:1–191:14. 3, 4

[XLF*11] XIN S.-Q., LAI C.-F., FU C.-W., WONG T.-T., HE Y., COHEN-OR D.: Making burr puzzles from 3D models. *ACM Trans. on Graph. (SIGGRAPH) 30*, 4 (2011), 97:1–97:8. 2, 4

[YCXW17] YAO M., CHEN Z., XU W., WANG H.: Modeling, evaluation and optimization of interlocking shell pieces. *Comp. Graph. Forum (Pacific Graphics) 36*, 7 (2017), 1–13. 3, 4

[YKGA17] YAO J., KAUFMAN D. M., GINGOLD Y., AGRAWALA M.: Interactive design and stability analysis of decorative joinery for furniture. *ACM Transactions on Graphics (TOG) 36*, 2 (2017), 20:1–20:16. 12

[YZC18] YUAN Y., ZHENG C., COROS S.: Computational design of transformables. *Computer Graphics Forum (SCA) 37*, 8 (2018). 2

[ZB16] ZHANG Y., BALKCOM D.: Interlocking structure assembly with voxels. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (2016), pp. 2173–2180. 3, 4

[ZSMS14] ZHOU Y., SUEDA S., MATUSIK W., SHAMIR A.: Boxelization: Folding 3D objects into boxes. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (2014), 71:1–71:8. 2

[ZW12] ZHOU Y., WANG R.: An algorithm for creating geometric dissection puzzles. In *Bridges Towson: Mathematics, Music, Art, Architecture, Culture* (2012), pp. 49–56. 1, 2, 10, 11