Construction and Fabrication of Reversible Shape Transforms

ANONYMOUS AUTHOR(S)

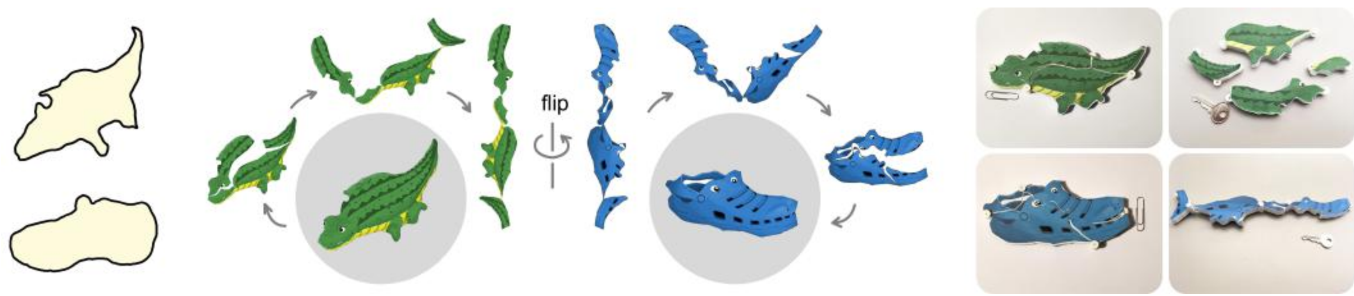


Fig. 1. We introduce a *fully automatic* algorithm to construct *reversible hinged dissections*: the crocodile and the Crocs shoe can be inverted *inside-out* and transformed into each other, bearing slight boundary deformation. The complete solution shown was computed from the input (left) without user assistance. We physically realize the transform through 3D printing (right) so that the pieces can be played as an assembly puzzle.

We study a new and elegant instance of geometric dissection of 2D shapes: *reversible hinged* dissection, which corresponds to a *dual transform* between two shapes where one of them can be dissected in its interior and then inverted *inside-out*, with hinges on the shape boundary, to reproduce the other shape, and vice versa. We call such a transform *reversible inside-out transform* or RIOT. Since it is rare for two shapes to possess even a rough RIOT, let alone an exact one, we develop both a RIOT construction algorithm and a quick filtering mechanism to pick, from a shape collection, potential shape pairs that are likely to possess the transform. Our construction algorithm is *fully automatic*. It computes an *approximate* RIOT between two given input 2D shapes, whose boundaries can undergo slight deformations, while the filtering scheme picks good inputs for the construction. Furthermore, we add properly designed hinges and connectors to the shape pieces and fabricate them using a 3D printer so that they can be played as an assembly puzzle. With many interesting and fun RIOT pairs constructed from shapes found online, we demonstrate that our method significantly expands the range of shapes to be considered for RIOT, a seemingly impossible shape transform, and offers a practical way to construct and physically realize these transforms.

Additional Key Words and Phrases: Hinged geometry dissection, reversible inside-out shape transform, fabrication

ACM Reference Format:

Anonymous Author(s). 2018. Construction and Fabrication of Reversible Shape Transforms. *ACM Trans. Graph.* 1, 1 (June 2018), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Geometric dissection problems have had a long history in recreational mathematics, arts, and puzzle making [Dudeney 1902; Frederickson 1997]. In computer graphics, a variety of geometric puzzles [Li et al. 2011; Löffler et al. 2014; Sun and Zheng 2015; Xin et al. 2011; Zou et al. 2016], including those involving dissections [Duncan et al. 2017], have also drawn interests, not only for their recreational value, but also owing to the geometric beauty and computational challenge the problems present. In the early 1800's, Wallace [1831] asked whether a polygon can always be dissected into pieces and then put together to reproduce another polygon of equal area. The positive answer has been known as the Wallace-Bolyai-Gerwien theorem [Gardner 1985]. A common *hinged* dissection [Frederickson 2002] between two equal-area polygons adds the extra constraint that the polygon pieces do not have complete freedom during assembly — they must be hinged at some of the polygon vertices. Hinged dissections have potential applications in reconfigurable robotics, programmable self-assembly, and nano-scale manufacturing.

A new and elegant special case of common hinged dissections for 2D shapes are *reversible* hinged dissections [Akiyama and Matsunaga 2015]. The added constraint over general hinged dissections between two polygons P and Q is that the boundary of P goes entirely into the interior of Q and *vice versa*. In other words, the transformation

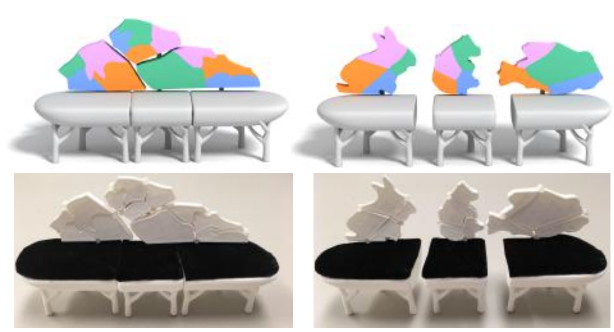


Fig. 2. Applying reversible shape transforms to a real sofa design. The three back pieces of the Borghese sofa can be transformed into different animals: bunny, bear, and fish. Top shows virtual models and bottom shows fabricated prototypes using a 3D printer.

2018. 0730-0301/2018/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

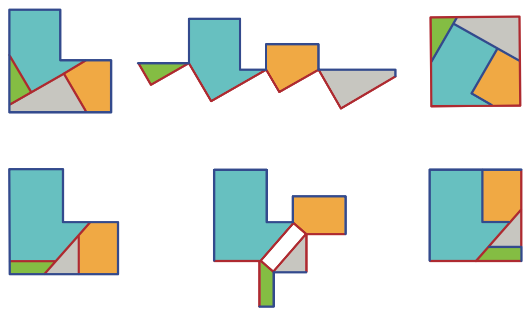


Fig. 3. Contrasting a reversible hinged dissection (top), i.e., a RIOT, and a non-reversible one (bottom) between an L shape and a square. The top example was introduced by Kelland, but with hinges applied to make it a hinged dissection, and the bottom example was by Hanegraaf. Both examples are based on figures from [Frederickson 2002].

from P to Q reverses P inside-out; we call this transform a *reversible inside-out transform*, or *RIOT*, for short. Figure 1 shows the first interesting example of RIOT and Figure 2 highlights how such a transform may add some fun to an elegant, real sofa design. For a simpler illustration of RIOT and to contrast it with other types of hinged dissections, please refer to Figure 3.

To the best of our knowledge, there are no known RIOT construction schemes between general shapes. Only a handful of results of exact RIOTs between non-trivial shapes have been shown [Akiyama and Matsunaga 2015] and it is unclear whether a RIOT always exists between two shapes of equal areas. In this paper, however, we are less interested in computing an exact transform between two given, fixed shapes. From a design and modeling perspective, users typically demand more degrees of freedom and control. A user may marvel at the ability to select input shapes to make the shape reversal fun, e.g., to transform a crocodile into a Crocs shoe (Figure 1). In another scenario, a user may already have one input shape in mind and wants to search for the most entertaining counterpart to have a RIOT.

To allow more freedom in reversible shape transforms, we relax exact RIOT construction into an *approximate* version, where the input shapes are allowed to deform slightly. In addition, we develop a tool to enable the exploration of many real-world shapes to quickly discover shape pairs which are likely to admit a RIOT that leads to small boundary deformations. We solve the approximation construction problem on candidate pairs and realize the solutions through physical fabrication. To make the experience even more fun and rewarding, we add properly designed hinges to the fabricated pieces so that they could be played as an assembly puzzle; see Figure 1.

Several key challenges must be addressed when developing our desired tool for RIOT construction, exploration, and fabrication. First, while the exact construction problem is already difficult and counter-intuitive in its own right [Akiyama and Matsunaga 2015], even for simple input shapes, combining boundary deformation and RIOT search offers an even greater computational challenge since the search space is significantly enlarged. Second, we want to avoid solutions with many small pieces. Our goal is to find a hinged dissection with a small number of pieces to reduce assembly cost and ensure that

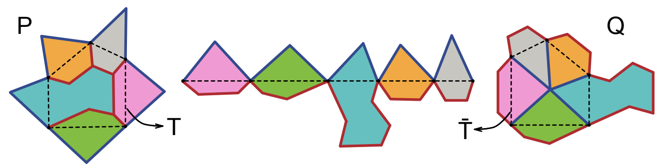


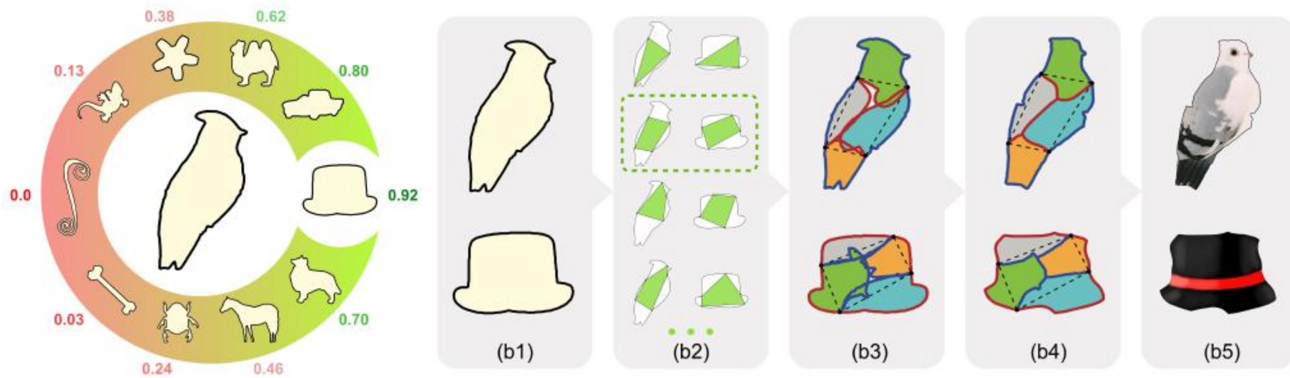
Fig. 4. Trunks and conjugate polygons. Shape P (left) has a trunk T (dashed line) whose exterior pieces can be rotated inward to form a polygon $\bar{T}$, which is a trunk for shape Q (right). T and $\bar{T}$ share the same edges (in inverse order); they are conjugate trunks of P and Q, respectively, implying a RIOT between them (see middle).

the pieces are large enough for 3D printing and to hold operational hinges. Third, the discovery of candidate shape pairs in a large shape collection calls for a *quick* scoring mechanism for the likelihood of a reversible transform and the scores must be obtained without constructing any transforms explicitly. And finally, physical realization of the hinged assembly must account for possible *collision* between the pieces when they are rotated about the hinges.

Given two 2D shapes P and Q scaled to unit areas, we formulate the approximate RIOT construction problem as seeking small boundary deformations to P and Q so that the deformed shapes $\tilde{P}$ and $\tilde{Q}$ would admit an exact RIOT. To compute reversible transforms between $\tilde{P}$ and $\tilde{Q}$, we rely on the notion of *trunks* for a 2D shape [Akiyama and Matsunaga 2015]. A trunk T of shape P is a convex polygon, inscribed in P, which can be opened up and reversed so that the exterior pieces would make up the interior of another convex polygon $\bar{T}$, without gaps or overlaps. The polygon $\bar{T}$ is not necessarily congruent to T, but they share the same set of edges in reverse order; these two polygons are said to be *conjugate* to each other. Two shapes $\tilde{P}$ and $\tilde{Q}$ have a RIOT, if they possess a pair of conjugate trunks; see Figure 4. Please note that this condition is not necessary; see Figure 3 (top).

Our construction scheme consists of two phases, as illustrated in Figure 5b. In the first phase, we perform *intra-shape reversibility* analysis on each input shape independently to identify *candidate trunk polygons* which have a low edge count and are convex and *approximately reversible*. The second phase constitutes *inter-shape* or *cross-reversibility* analysis, where we identify the *most conjugate* pair of candidate trunks T_P and T_Q from input shapes P and Q, respectively. We make T_P and T_Q conjugate to each other and deform the boundaries of P and Q to eliminate gaps and overlaps when applying an approximate RIOT between P and Q based on T_P and T_Q. Our approximate RIOT construction algorithm is *fully automatic*, while the boundary deformation step could benefit from light user assistance to perfect issues related to shape semantics.

To discover shape pairs, from a large shape collection, that are likely to possess a RIOT, we first filter out shapes based on a reversibility score computed for individual shapes. This score indicates how likely a shape possesses good trunks. Then among shapes with high reversibility scores, we identify pairs of them likely to possess *conjugate* trunks or in other words, RIOTs; see Figure 5a. To this end, we define a cross-reversibility score for shape pairs, which does not require explicit RIOT construction. The key is to enable quick computations of the reversibility and cross-reversibility scores.



(a) Reversibility scoring

(b) Candidate trunk pair selection and boundary deformation

Fig. 5. Overview of our work on reversible hinged dissections. Given a shape collection, we compute scores to quickly assess how likely two shapes possess a reversible transform. (a) Scores of different shapes with respect to the bird. Given a promising pair of shapes, e.g., the bird and the hat in (b1), we construct an approximate reversible inside-out transform through several steps: candidate trunk selection (b2), trunk pair selection (b2), and slight boundary deformation (b3)-(b4) to perfect the transform. The shapes can finally be textured (b5) and fabricated.

We demonstrate that our fully automatic RIOT construction algorithm operates effectively and efficiently over a variety of natural shapes — some fun RIOT pairs can be found in Figures 1 and 14. For evaluation, we compare our results to manual designs of reversible shape transforms. As well, we show that our quick reversibility and cross-reversibility scores can facilitate filtering of shapes and shape pairs from large shape datasets to discover shape pairs with high potential of producing reversible transforms.

With a constructed RIOT between two shapes, we can 3D print the pieces which constitute the transform. Each piece has sufficient thickness to allow embedding *hingeable connectors* so that the pieces can be linked physically to reproduce the transform. To address the collision problem, we alter the hinges so that they are *telescopic*. Such a hinge would allow a piece to be pushed into an offset plane, rotated in that plane without collision, and then pushed back to the base plane after rotation; see Figure 11.

2 RELATED WORK

Our problem is generally related to shape decomposition and dissection, which are well studied geometry problems with an extensive literature. This section only covers works we deem the most relevant.

Decompose-and-assemble. Most works on shape segmentation decompose a *single* shape into desirable parts [Shamir 2008]. Some works combine decomposition with assembly to produce another shape or volume. In Dapper [Chen et al. 2015], a mesh is decomposed into few parts and packed into the printing volume of a 3D printer for efficient fabrication. Song et al. [2017] construct reconfigurable furniture pieces made up using a common set of parts to assemble them into various forms. Unlike these works which involve 3D modeling, our problem analyzes 2D shapes and it is defined by an entirely different set of goals and constraints. Specifically, RIOT is a special instance of hinged geometric dissection.

Geometric dissection. While the Wallace-Bolyai-Gerwien theorem provides an existence proof, exact geometric dissections are difficult to construct. Zhou et al. [2012] discretize the input shapes over a quadrilateral or triangular lattice and resort to an exhaustive hierarchical search to merge lattice cells to find the minimum number of pieces that are necessary to construct both shapes. Recently, Duncan et al. [2017] pose and solve the approximate dissection problem which computes a common set of pieces that can be rearranged to reproduce two input shapes closely, but not necessarily exactly. To produce these pieces, they rely on a combinatorial search to prune the search space of solutions that are later refined and selected by users to deliver satisfying results. Our problem also approximates an exact geometric dissection problem, but it imposes two additional constraints as opposed to the dissection problem addressed by [Duncan et al. 2017]: hinged dissection *and* inside-out reversibility. As a result, we have taken a completely different approach based on finding conjugate trunks of two given shapes.

Hinged dissection. Exact hinged dissections have been examined in special cases, e.g., for transforming between squares and alphabet shapes [Demaine et al. 2005]. Abbott et al. [2012] gave an existence proof that two equal-area polygons must possess a hinged dissection. However, the status of reversible hinged dissection is not known to date. The problem we pose and solve in this paper is a novel one: *approximate* reversible hinged dissections.

To the best of our knowledge, there are two pieces of works in computer graphics which come somewhat close to a RIOT, both tackling intriguing and challenging 3D geometry problems.

In Boxelization, Zhou et.al. [2014] decompose a 3D model into *voxel*-like pieces which are joined by reflective and twisty connectors so that the resulting hinged structure can be re-assembled into a box, possibly still leaving some visible gaps in the assembled structure. The main technical challenges in Boxelization are posed by connector type assignment and computation of the structure transform, not by the decomposition, which is a voxelization process. Inspired

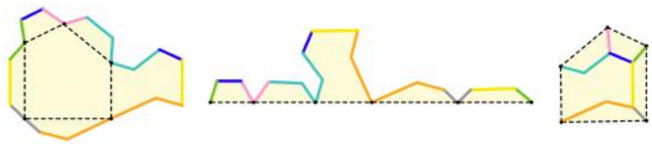


Fig. 6. The boundary of a reversible shape can be divided into congruent segment pairs. Two congruent segments are in the same color.

by Rubik's cubes, the work of Sun and Zheng [2015] introduces computational design of twisty joints and puzzles. Given a user-supplied 3D model and a small subset of cuts and rotation axes, their method automatically adjusts the given cuts and rotation axes and adds others to construct a "non-blocking" twisty joint structure in the shape of the input model. The resulting pieces can be directly 3D printed, assembled into an interlocking puzzle, and rotated against each other in a collision-free manner.

With the twisty hinges in these works, some voxels or rotating parts can certainly be turned inside-out. However, the type of pieces sought by the decomposition, the decomposition and assembly criteria, as well as the roles the hinges play in the construction are all quite different between these works and our problem. Decomposition is the main challenge for RIOT construction. The result dictates *where* hinges are to be placed, while all hinges rotate in the plane.

Reversible hinged dissection. Akiyama and Nakamura were the first to study the RIOT problem extensively and developed a construction method for specific convex polygons [Akiyama and Nakamura 2000]. Akiyama et al. [2015] extended this work later to process more complex shapes and proved a sufficient condition for two shapes to be reversible: they possess conjugate trunks. In this paper, we base our computation of approximate RIOTs on discovering conjugate trunks. With a distinctive goal of *approximate* reversible hinged dissections, our construction algorithm is completely different from that of [Akiyama et al. 2015] and it also involves boundary deformation in the final stage. In addition, we incorporate additional fabrication constraints into the construction and develop a quick filtering mechanism to select potential RIOT shape pairs.

3 NOTATION AND METHOD OVERVIEW

In this section, we first provide the background and notations that we use throughout the paper. We then present an overview of our methods to select potential RIOT pairs from a large database, and find a RIOT between two given shapes. For more illustrative explanations and examples, please refer to the accompanied video.

3.1 Notation

A pair of shapes P and Q is said to be a RIOT if the following conditions are satisfied [Akiyama and Matsunaga 2015] (Figure 4):

- There exists a dissection of P into pieces that can be hinged at vertices on the boundary of P and form a chain;
- When rotating pieces in clock-wise (CW) or counter clock-wise (CCW) directions with one end-piece of the chain fixed, P or Q is respectively generated;
- The boundary of P falls inside Q and becomes its dissection curves, and the same is true for the boundary of Q.

Regarding the existence and construction of such a transformation, Akiyama and Matsunaga [2015] have shown that if P is a shape with trunk T and conjugate trunk $\bar{T}$, and Q has trunk $\bar{T}$ and conjugate trunk T, then P and Q are reversible (Figure 4).

To define trunk and its conjugate trunk, we first need to define conjugate polygons. If T is a k-sided polygon with edges labeled in CW direction, $\bar{T}$ is its *conjugate polygon* if it has the same edges but in CCW order. Clearly, the mirrored version of any polygon is one of its conjugate polygons. However, there are an infinite number of conjugate polygons for a polygon with more than three edges. This is due to the fact that the conjugate polygon has the same edges as the polygon, but they do not necessarily form the same angle configuration. A *trunk* T of P is an inscribed convex polygon of P. Assume T is opened up and rotated to form its conjugate polygon $\bar{T}$, $\bar{T}$ is called the *conjugate trunk* of P, if the pieces of P falling out of T can be packed inside $\bar{T}$ without overlaps and gaps (Figure 4). This way, the boundary of a reversible shape is composed of congruent segment pairs that might be located at adjacent or non-adjacent exterior pieces (Figure 6). This property is called *boundary congruency*.

3.2 Method Overview

Here, we provide a brief overview of our method, illustrated in Figure 5. Since only a few known RIOT shapes existed prior to this work, to make RIOT pairs, we efficiently search through large databases of shapes to find the ones likely to be a RIOT through our *RIOT pair selection* process. Having a pair of shapes with high possibility of being RIOT, we perform its *RIOT construction* by finding a set of candidate trunks for the shapes and determining the best match for the pair. The boundary of shapes are then deformed to eliminate potential gaps and overlaps and a perfect RIOT is obtained. In the following, each of these steps are discussed in more details.

RIOT pair selection. Since most available shape pairs are not readily reversible, we develop a reversibility test to quickly filter out thousands of pairs and identify potential reversible pairs. This is a crucial step as it helps us avoid time consuming processes such as finding trunks for pairs that are certainly not reversible. Each input shape is represented by a set of contour points and the area of the discrete contour is normalized to one to ensure that all input shapes are of the same size. To perform reversibility test, we first compute a reversibility score that measures the probability of an individual shape to be reversible. We then test the cross-reversibility of two shapes of a pair to identify the pairs that are potentially reversible (Figure 5a). Since the reversibility scoring derives observations from the following RIOT construction, we describe it in Section 5, after discussing the RIOT construction, although it is executed first.

RIOT construction. Given a potential reversible pair (P, Q), our objective is to compute the candidate conjugate trunks T_P and T_Q

(Figure 5b). We consider the best candidate conjugate trunk as the one with minimal boundary deformation and consisting few pieces. One option to discover candidate trunks is to generate numerous polygons from all boundary points of each shape and then evaluate polygon pairs of two shapes under all possible edge correspondences. However, following this approach, the space of polygon pairs would be too large, especially when the number of edges in the trunks and their locations are unknown. Therefore, we first perform an *intra-shape reversibility assessment*, where we find an upper bound for the number of edges in trunks and also limit the location of trunks' vertices to sparsely sampled points that include shapes' features. We then generate a set of potential trunk vertices that forms a space for candidate trunks. The candidate trunks consist polygons with different number of edges starting from three (for triangles) to the upper bound. Finally, we perform a *cross-reversibility assessment* to select the best trunk pair (see Section 4), whose number of edges determines the number of dissection pieces.

To make a perfect RIOT, trunks are slightly modified to be conjugate and the boundaries of shapes are adjusted to contain new trunk vertices (Figure 5(b3)). Trunks are then fixed and shape P is deformed to eliminate overlaps and gaps inside T_Q as well as regions outside T_Q. The same process is performed for Q. For deformation, we use the 2D Laplacian editing method [Sorkine et al. 2004], which tends to preserve structural geometric details. The results can be then refined by users via an interactive interface to satisfy human perception (Section 4.3). To have aesthetically pleasing results, we either adopt available textures of the input shapes (see supplementary material) or manually texture deformed shapes when textures are not available. This way, we produce textured reversible shapes $\tilde{P}$ and $\tilde{Q}$ with trunks T_P and T_Q and their reversible inside-out transformation defined based on the boundary curves of the shapes.

Finally, to have a playable puzzle, we fabricate our results adding thickness to 2D pieces to make them 3D and printable. Special hinges are also added to deliver the possibility of rotating pieces in CW or CCW directions. To avoid collision between pieces, a telescopic structure is fabricated if two pieces collide during rotation along hinges (see Section 4.4). These telescopic structures take a colliding piece up to an offset plane, where it can be rotated freely. The piece can then be moved back to its base plane (Figures 11 and 12).

CRS and QCRS. Cross-reversibility analysis and cross-reversibility scores (CRS) are encountered in different contexts in our method. During RIOT construction, we define CRS between candidate trunks. Then the CRS between two input shapes is given by the maximum CRS between candidate trunks. Later, in our quick filtering mechanism, we define a *quick* CRS or QCRS to rank shape pairs, which can be considered as the simplified version of CRS. While the role of QCRS is to help us select promising shape pairs for RIOT construction, the CRS between two shapes provides a more accurate assessment of how likely the shapes would possess a reversible transform. This latter score, if small, may prevent us from performing the (relatively) expensive boundary deformation step.

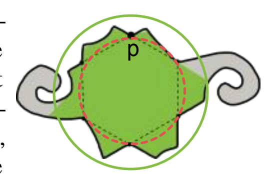
4 RIOT CONSTRUCTION

To construct a RIOT for a given pair of shapes that are not necessarily reversible to each other, we first need to search for a pair of conjugate trunks. To do so, we assess each shape individually and find a set of points, called *candidate vertices*, capable of being the vertices of candidate trunks. This set is further examined to provide a set of candidate trunks for each shape. We then assess the trunk pairs between the pair through a cross-reversibility score (CRS) and find a pair of trunks that are approximately conjugate. To make a perfect RIOT, trunks are first adjusted to be conjugate and then shapes are deformed to remove gaps and overlaps without an extreme deterioration of features. Finally, the resulting RIOT is fabricated to make a playable puzzle. In the following, each step is discussed in detail.

4.1 Candidate trunks per shape

Shapes are initially assessed for selecting candidate vertices. To do so, we first consider all the sampled points on the boundary of the shapes, and then exclude a large number of points with a *binary* score based on trunk convexity, area compatibility, and boundary congruency criteria. We then define a *congruency score* for the remaining points and only select the ones with high congruency scores.

To define the binary score, we start by considering the convexity of polygons at vertices. Since trunks must be convex, if point p is a trunk vertex, other vertices must lie in the visible region of p, defined as $VR(p)$ (Figure 7a,b)). As a result, invisible regions, (each one is denoted as $IVR_i(p)$), all belong to exterior pieces of a trunk with vertex p. We can define an area relationship between these regions that helps us include or exclude a point in the set of candidate vertices.



Consider a circle with the same perimeter as polygon T, called a T-Circle (the red circle in dashed lines in the right figure). Based on the isoperimetric inequality [Burago and Zalgaller 2013], the area of the T-Circle is larger than the area of T and its conjugate trunk $\bar{T}$. When T is a trunk, the total area of its exterior pieces is equal to the area of its conjugate trunk $\bar{T}$, which is smaller than the area of the T-Circle. Therefore, we can define an inequality relationship for regions of a shape as: $\sum_i$ Area($IVR_i(p)$) < Area(exterior pieces) < Area(T-Circle) < Area($VR(p)$-Circle). The green, solid circle in the right inset figure is the $VR(p)$-Circle which has the same perimeter as polygon $VR(p)$.

Moreover, when the perimeter of one of the boundary segments in invisible regions, defined by$L(IVR_i(p))$, is larger than half the perimeter of the entire shape ($L/2$), then there are not enough congruent boundary segments from the remaining exterior pieces to match to this perimeter. For example, in Figure 7a, the perimeter of the largest $IVR_i(p)$ is clearly longer than $L/2$, and there are not enough boundary segments in other pieces of $IVR_i(p)$ to match. As a result, this point should be excluded from the set of candidate vertices. These lead us

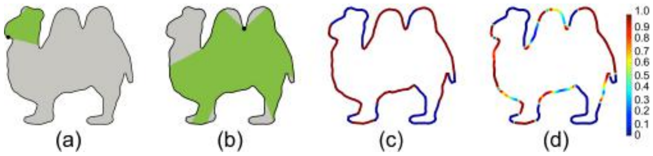


Fig. 7. The visible region (green) and invisible regions (grey) of a point in a narrow protrusion (a) and a regular point (b); the binary score to exclude (blue) and include (red) candidate vertices (c); the congruency score for included points (d) .

to define a binary score S_b to exclude invalid points (Figure 7c):

$$S_b(p) = \begin{cases} 0, & \text{if } \sum_i \text{Area}(IVR_i(p)) \geq \text{Area}(VR(p)\text{-Circle}), \\ 0, & \text{if } L(IVR_i(p)) \geq L/2, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

For further evaluating the remaining points for which $S_b(p) = 1$, we compute a point-level congruency score S_c (Figure 7d):

$$S_c(p) = \begin{cases} 0, & \text{if } L(C_l^p) + L(C_r^p) \leq 0.03L, \\ exp\left(-\frac{d_c^2(C_l^p, C_r^p)}{2\sigma_c^2}\right), & \text{otherwise,} \end{cases} \tag{2}$$

where C_l^p and C_r^p are two supposedly congruent segments meeting at p. The congruency score is zero for small segments. For any other point, it attains a value between zero and one based on the discrete Fréchet distance $d_c(C_l^p, C_r^p)$ between its two congruent segments. Note that the Fréchet distance is commonly used to measure the similarity of two curves [Eiter and Mannila 1994]. The parameter σ_c is set to $0.1D^c$, where D^c is the diameter of the unit area circle. We only consider adjacent segment pairs meeting at trunk vertices since such pairs are usually congruent in a RIOT. However, one could use the same technique and analyze all possible segment pairs resulting in a potentially more accurate but time consuming analysis.

Note that computing $L(C_l^p)$ and $L(C_r^p)$ is not a trivial task. One can progressively grow two equal-length segments from the left and right of p and stop when the segments are too dissimilar. However, this is inefficient as we have to run this process for all boundary points. To resolve this problem, we only keep important feature points of the boundary by simplifying shape P to $\hat{P}$ using Douglas-Peucker line simplification algorithm [Douglas and Peucker 1973] with distance tolerance $\tau_s = 0.1$. We then compute the length of congruent segments $\{C_l^{\hat{P}}, C_r^{\hat{P}}\}$ on $\hat{P}$ instead of P. Further details can be found in the supplementary material.

We consider points with a congruency score S_c larger than $\tau_c = 0.3$ as candidate trunk vertices, and generate a set of candidate trunks for each shape, in which trunks range from a triangle to a K-gon. The upper bound K is equal to the number of convex points of the simplified shape. For a reversible shape, for each edge of a trunk, its exterior piece must have at least one convex boundary point (Figure 8a). Based on this observation, the number of edges in a trunk cannot be larger than the number of convex boundary points. Despite having this constraint, there might still exist many convex points in complex shapes that do not affect the overall shape and can be removed. Thus, we only consider the convex points of the simplified shape $\hat{P}$ and denote them as $\{p_{c_1}, ..., p_{c_K}\}$. To diversify trunks, we only evaluate a sparse set of boundary points obtained by sampling. We use a method similar to the one we use for extracting points of input shapes, described in Section 6, but with $d_{space} = \frac{L^c}{15}$. With the upper bound K, we generate trunks T satisfying three conditions.

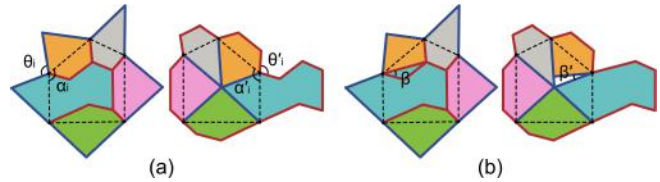


Fig. 8. For reversible shapes (a), we have angle relationships $2\pi - \theta_i - \alpha_i = \alpha_i'$, $2\pi - \theta_i' - \alpha_i' = \alpha_i$ at the i-th pair of corresponding trunk vertices. The shape pair in (b) approximates the reversible shape pair in (a) and $2\pi - \theta_i - \alpha_i = \alpha_i' - \beta'$, $2\pi - \theta_i' - \alpha_i' = \alpha_i + \beta$.

- T is inscribed and convex.
- There is at least one convex point from $\{p_{c_1}, ..., p_{c_K}\}$ on each exterior piece.
- The area of each exterior piece is larger than 0.01 and the boundary segment on each exterior piece is shorter than $L/2$ based on boundary congruency.

We then accept trunks with edges that are 90% inside the shape and exclude trunks with segments that are too incongruent at adjacent vertices, having large overlaps $\frac{L^c}{10}$ (L^c is the perimeter of the unit area circle). Typically, the number of constructed trunks are initially about 16,000, while the selected candidate trunks are about 900.

4.2 Trunk pair selection

For a pair of shapes (P, Q), we define a cross-reversibility score (CRS) for trunk pairs and select the best trunk pair. We first form each possible trunk pair (T, T') where T and T' respectively belong to the set of candidate trunks of P and Q and possess the same number of edges. The CRS is computed based on three criteria: edge conjugacy, area reversibility, and angle reversibility, discussed as follows.

Edge conjugacy. Suppose that we are given any trunk pair (T, T') for two shapes, where T and T' are n-gons with edges $\{e_0, e_1, ..., e_{n-1}\}$ and $\{e_0', e_1', ..., e_{n-1}'\}$ labeled in opposite directions. We define a score to measure their conjugacy under edge correspondence $\phi_i = \{0 \rightarrow i, 1 \rightarrow i+1 \pmod{n}, ..., n-1 \rightarrow i+n-1 \pmod{n}\}$:

$$S_E(T, T', \phi_i) = exp\left(-\frac{d_E^2(T, T', \phi_i)}{2\sigma_E^2}\right), \tag{3}$$

where $d_E(T, T', \phi_i) = \frac{\sum_{j=0}^{n-1} ||e_j| - |e'_{\phi_i(j)}||}{n}$ and $\sigma_E = 0.1D^c$.

Angle reversibility. For reversible shapes, we have the following angle relationships at two corresponding trunk vertices (Figure 8a):

$$2\pi - \theta_i - \alpha_i = \alpha_i', 2\pi - \theta_i' - \alpha_i' = \alpha_i,$$

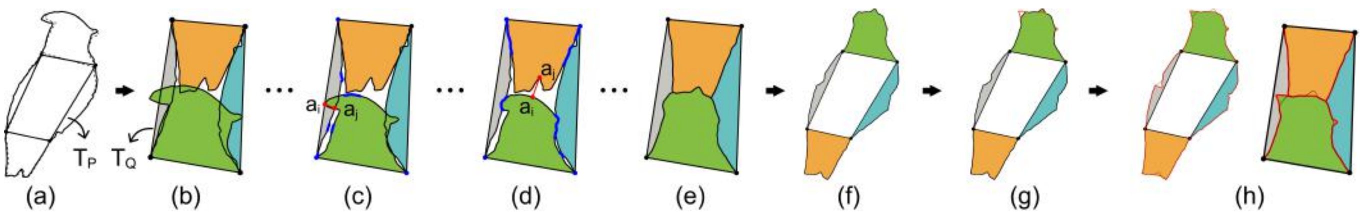


Fig. 9. When deforming shape P, we fix its candidate trunk T_P (a) and conjugate trunk T_Q (enlarged) (b). The goal is to eliminate regions outside T_Q (b to c), the overlaps (c to d), and gaps (d to e) inside T_Q. The user is allowed to directly draw new segments (red segments in g) on the deformed shape (f); The deformed shape and dissection curves inside T_Q are updated (h).

where θ_i and θ'_i are boundary angles, and α_i and α'_i are trunk angles. We call this observation *angle reversibility* and define its score as:

$$S_\angle(T, T', \phi_i) = exp\left(-\frac{d_\angle^2(T, T', \phi_i)}{2\sigma_\angle^2}\right), \tag{4}$$

where

$$d_\angle(T, T', \phi_i) = \frac{\sum_{j=0}^{n-1} |2\pi - \Omega_j| + |2\pi - \acute{\Omega}_j|}{2n},$$

and $\sigma_\angle$ is $\frac{\pi}{6}$. We defined parameters $\Omega_j = \theta_j + \alpha_j + \alpha'_{\phi_i(j)}$ and $\acute{\Omega}_j = \theta'_{\phi_i(j)} + \alpha'_{\phi_i(j)} + \alpha_j$ to shorten the equation. Note that both $S_\angle$ and S_E attain one for conjugate trunks under an accurate correspondence. In practice, replacing θ_j by the rotation angle of two congruent segments of j-th vertex in a trunk results in better robustness.

Area reversibility. In a perfect RIOT, the boundary of one shape fits inside the trunk of the other shape without any overlaps or gaps or pieces falling out of the trunk. This leads us to define a score as:

$$S_A(T, T', \phi_i) = exp\left(-\frac{d_A^2(T, T', \phi_i)}{2\sigma_A^2}\right), \tag{5}$$

where $d_A(T, T', \phi_i) = area(gaps) + area(overlaps) + area(outside)$ under edge correspondences ϕ_i and $\sigma_A = 0.3$. An efficient computation of these areas is discussed in the supplementary material.

We define the minimum of these scores as the cross-reversibility score of T and T' for the edge correspondence ϕ_i:

$$CRS_i(T, T') = \min\{S_E, S_\angle, S_A\}. \tag{6}$$

We can then define the CRS for (T, T') as:

$$CRS(T, T') = \max_{i=0,\ldots,n-1} CRS_i(T, T'). \tag{7}$$

The trunk pair (T, T') with highest CRS is selected for (P, Q) to perform deformations and obtain a perfect RIOT. This score can be used to define a cross-reversibility score of two input shapes as:

$$CRS(P, Q) = \max_{\{(T, T')\}} CRS(T, T'), \tag{8}$$

to filter out irreversible shapes and avoid the deformation step.

4.3 Boundary deformation

Trunks T and T' attaining the highest CRS score for shapes P and Q are not necessarily conjugate, therefore, they are initially adjusted to become conjugate and (T_P, T_Q) is obtained (Figure 9a); please refer to supplementary material for technical details.

If T_P has n vertices, it divides P into n curves along the boundary whose endpoints are two vertices of T_P. These curves must fit in T_Q and dissect it without any overlap and gap. Curves are initially rotated and translated into T_Q according to the edge correspondence of T_P and T_Q (Figure 9b). These transformed curves $\{C_1, ..., C_n\}$ are deformed using 2D Laplacian editing [Sorkine et al. 2004] to eliminate gaps and overlaps in T_Q while preserving the overall shape of P. Note that when deformed curves are transformed back to T_P, the final shape $\tilde{P}$ that is an approximation of P is obtained (Figure 9f). The process of deforming Q is the same. The two deformation processes are independent since T_P and T_Q are fixed.

For deformation, we first automatically remove overlaps and gaps and then offer a user interface to fine-tune the results. Small regions outside T_Q are initially eliminated by scaling curves while keeping the endpoints stationary (Figure 9b to c). Overlaps between any two curves C_i and C_j are then found. A set of vectors V_i connecting points a_i to a_j are defined, where $a_i \in C_i$ falls in C_j, $a_j \in C_j$ falls inside C_i, and a_j is the closest point to a_i among all points in C_j. dir_i is one of the vectors in V_i that attains the longest length and $dir_j = -dir_i$. Then, C_i and C_j are deformed iteratively along dir_i and dir_j until no overlaps exist. dir_i attaining the greatest magnitude is used to speed up deformation. We use 0.003 as the step size for iterations.

During the deformation, the endpoints of curves are stationary to fix T_Q and T_P. In addition, points that already meet along two curves and do not lie in any overlap are fixed to maintain the shape and avoid producing further gaps or overlaps. In Figure 9c, fixed points are highlighted in blue. A similar process is performed to eliminate gaps (more details in supplementary material); see Figure 9d to e.

In the automatic deformation process, some minor features with important semantics might be removed, such as the beak and crest of the bird in Figure 9f. To recover such features, we have provided a simple user interface illustrating both the input shape (dashed line in Figure 9a) and the deformed shape (Figure 9f) to users. Users can directly draw new segments to edit desired features (red segments in Figure 9g). The result of edits is interactively updated (Figure 9h).

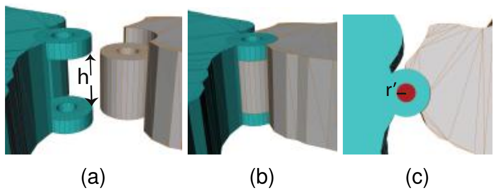


Fig. 10. Female (left) and male (right) hinges in open (a) and closed (b) configurations. Pivot inserted to hold the pieces together (c).

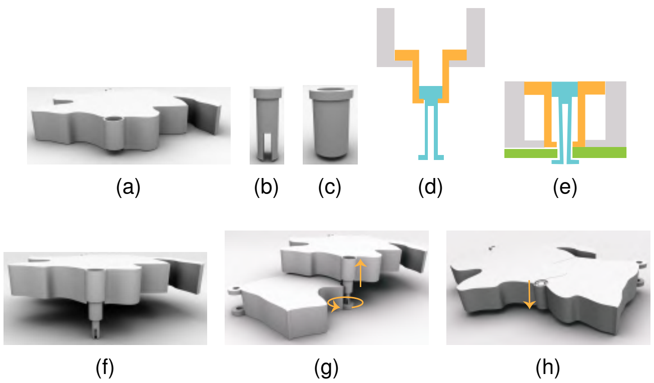
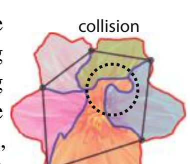


Fig. 11. Telescopic structure attached to a piece (a). Telescopic pieces that fit together (b), (c). Cross view of telescopic structure in extended and collapsed position (d), (e). Extended telescopic structure (f). Telescopic structure and its axis of movement and rotation (g), (h).

4.4 Fabrication

We finally fabricate the model to make an assembly puzzle. The fabricated model should resemble the RIOT by supporting rotation along hinges. We fabricate the (T_P, T_Q)-chain in which pieces of both shapes are attached along a straight line (Figure 4). We 3D print the two connected pieces of shapes P and Q along each edge of the (T_P, T_Q)-chain as a single piece that is thickened, and connect the different pieces with fabricated hinges.

To fabricate hinges supporting rotation, we have designed a set of female and male connectors. A male connector is a cylinder with height h attached to a piece and has a cylindrical hole with radius $\acute{r}$ (Figure 10). A female connector is composed of two cylinders with same radius and height difference h to hold the male piece (Figure 10b). A cylindrical pivot (Figure 10c) with radius $\acute{r}$ is inserted in the holes of female and male connectors to keep the pieces together.



In case of a collision, such as the right figure taken from our gallery (Figure 14), colliding pieces are also allowed to move vertically along the axis of rotation at each hinge. This way, one piece can be moved up to an offset plane, rotated, and moved down back to its place. To support both rotation and vertical movement, we used a cylindrical telescopic structure consisting of three cylinders with different sizes, where the largest cylinder must hold and contain two smaller cylinders (see Figure 11e) [Yu et al. 2017]. The largest cylinder (Figure 11a) is attached to a piece and only has a ledge to hold the smaller cylinders. The smaller cylinders are designed as T shapes so that they can be held in a chain. The smallest cylinder of the telescopic structure is pliable and has a knob to be inserted into the connector of the neighboring piece (green piece in Figure 11e). These structures are created by simple addition and subtraction operations of solid models.

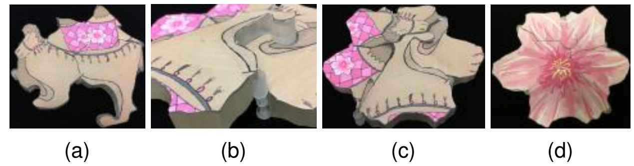


Fig. 12. A textured model (a) has a rotation collision, thus, a telescopic structure (b) can be used to lift a piece up, rotate it, and place it back to its base plane (c). The other side of the model (d).

It is also desired to attach textures to beautify fabricated objects. To do so, we use printable stickers on which properly scaled textures are printed. Textures are then cut and pasted on top of the fabricated pieces. Figure 12 shows a textured model with telescopic structures; more fabrication results can be found in accompanying video and supplementary material.

5 RIOT PAIR SELECTION

Finding reversible pairs is not a trivial task, therefore, we develop a method to identify potential pairs. Here, we discuss how we select a collection of pairs that are likely to be reversible from a large collection of shapes. To do so, we have defined two scores to measure the reversibility of individual shapes and the reversibility of a pair of shapes (cross-reversibility) in a quick fashion.

5.1 Reversibility Score of Shape

The reversibility score of an individual shape is used to filter out many shapes that are less likely to be reversible regardless of their pair. We have observed that shapes with very complex boundaries are less likely to hold boundary congruency and be reversible. Furthermore, thin shapes are not usually reversible as it is hard to pack the exterior pieces of other shapes into their narrow inscribed trunk (Figure 13a). Considering these observations, we define a reversibility score as:

$$S_r^1 = exp\left(-\frac{\tilde{r}_{PA}^2}{2\sigma_{PA}^2}\right), \tag{9}$$

where $\tilde{r}_{PA} = \frac{r_{PA}}{r_{PA}^c} - 1$, with r_{PA} and r_{PA}^c being the perimeter-area ratios of the shape and unit area circle, respectively, and $\sigma_{PA} = 1$. In this way, $S_r^1 = 1$ for the unit area circle, while shapes with thinner or more complex boundaries attain lower scores.

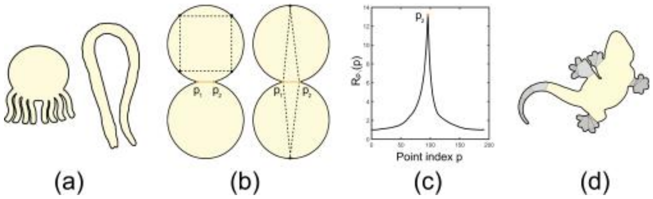


Fig. 13. Complex shapes and thin shapes are less likely to be reversible (a) The waist (orange lines) is a special type of neck for the concave point p_1, as it bisects the boundary. A candidate trunk (dashed line) on one side of the waist (left), and a candidate trunk (dashed line) going through the waist (right). The ratio function $R_{p_1}(p)$ and maximum point p_2 (c). Regions separated by narrow necks are in grey. The boundary points on these regions are excluded (d).

Furthermore, shapes with central *necks* (waists) are less likely to be reversible (Figure 13b). Geometrically, *necks* can be defined as two points that are close in the Euclidean domain but far geodesically. We define a neck as a line with p_1 and p_2 at the boundary satisfying the following conditions (Figure 13 (b,c)): (i) p_1 is a concave boundary point; (ii) p_2 is a local maximum of neck-ratio defined as $R_{p_1}(p) = \frac{d_{geo}(p_1,p)}{d(p_1,p)}$, where d_{geo} is the geodesic distance along the boundary and d is the Euclidean distance. Necks are narrower when the neck-ratio is larger; (iii) line $\mathcal{L}(p_1, p_2)$ is inscribed.

To find necks, we slightly simplify boundaries and obtain concave points. We then compute necks for each concave point. A central neck, that we call a *waist* approximately divides the whole shape boundary in half (orange lines in Figure 13b). Formally, waists are defined as necks $\mathcal{L}(p_1, p_2)$ satisfying $0.8 \le \frac{d_{geo}(p_1,p_2)}{L - d_{geo}(p_1,p_2)} \le 1.2$. If the shape has a waist, its trunks are either entirely located in one side of the waist, or pass through the waist (Figure 13b). In the first case, an exterior piece with a long boundary is created, which can not fit in its conjugate trunk (similar to the discussion in Section 4.1). In the second case, a narrow trunk that is incapable of encompassing the exterior pieces of its pair is likely to get produced. As a result, shapes with narrow waists should receive a low reversibility score, which we define as:

$$S_r^2 = \begin{cases} exp\left(-\frac{\tilde{r}_W^2}{2 * \sigma_W^2}\right), & \text{if waists exist,} \\ 1, & \text{otherwise.} \end{cases} \tag{10}$$

where $\tilde{r}_W = r_W - 1$, r_W is the largest ratio $R_{p_1}(p_2)$ among all waists, and $\sigma_W = 4$. Then, the reversibility score of an individual shape is:

$$S_r = min(S_r^1, S_r^2). \tag{11}$$

5.2 Cross-reversibility Score of Shape Pair

After filtering many irreversible shapes, we should quickly identify potential reversible shape pairs. We define a quick cross-reversibility score (QCRS) to perform this task. While it is possible to use the CRS defined in Section 4.2, the QCRS is computed faster and thus is better for dealing with a large collection of shapes.

Similar to the CRS, the QCRS is defined based on three terms: edge conjugacy, angle reversibility, and area reversibility. However, to reduce computations, we replace $d_A(T, T', \phi_i)$ by a less computationally expensive term $\tilde{d}_A(T, T') = |area(T) - (1 - area(T'))| + |area(T') - (1 - area(T))|$ with $\sigma_A = 0.1$. The idea behind this term is that the area of exterior pieces of trunk T should be equal to the area of the conjugate trunk T'. In contrast to the CRS, with this term we do not need to transform the boundary of one shape into the trunk of the other and compute the area of pieces falling out of the trunk, gaps, or overlaps under various edge correspondences. For further acceleration, boundary points isolated by necks $\mathcal{L}(p_1, p_2)$ with $d(p_1, p_2) < 0.2D^c$ are excluded (Figure 13d), following the same motivation for the binary score $S_b(p)$. Moreover, we instead use local boundary angles for θ_j in $d_\angle(T, T', \phi_i)$ and generate trunks for each shape without considering congruent segments.

6 RESULTS AND EVALUATION

We work with a large collection of 2D shapes to test and evaluate our work. The shape collection combines two public silhouette image datasets, i.e., the MPEG-7 database of [Latecki et al. 2000] and the Animal database from [Bai et al. 2009], resulting in 81 shape classes, and a total of 3,400 shapes. We also consider other shapes found online, possibly with textures, as potential test inputs to our method where they may lead to interesting reversible pairs.

For each silhouette image, we first fill any interior holes [Otsu 1979], if they exist, extract a single closed contour to define the shape, and normalize the shape to unit area through uniform scaling. Each shape boundary is adaptively sampled, starting from main feature points. Then we recursively insert midpoints along the boundary until the distance between any two consecutive boundary points is smaller than $\frac{L^c}{100}$, where L^c is the perimeter of the circle with unit area.

Visual results. The gallery in Figure 14 shows a sampler of RIOT results generated fully automatically by our construction algorithm. The shapes vary in their types (organic, man-made, or artistic) and geometric characteristics (rounded, elongated, or shapes with strong protrusions). All the shape pairs shown had passed the filtering test for pair selection. In the gallery, we deliberately selected high-ranking pairs in which the shapes possess certain contrasting or related semantics, e.g., a dog and his bowl, a bunny and carrot, etc., as they exhibit interesting and fun examples of reversible shape transforms. The selection of interesting RIOT pairs and their texture design are manual. Additional results can be found in the supplementary material. Note that we flip the textures to attach them on the two sides of fabricated results. When no texture is available such as the example in Figure 2, we do not need to flip the shapes.

For input shapes that are silhouette images, we had an artist manually design textures for the results. When the input shapes were textured to start with, we automatically transferred the original textures to the deformed output shapes, as explained in the supplementary material. Unlike the work of Sarhangi et al. [2008], in which a single texture is changed from one pleasing pattern to another after transformation, we have designed two separate textures for two sides of a shape.

Fig. 14. A gallery of reversible shape transforms computed *fully automatically* by our algorithm. For each pair, we show the input shapes in silhouette images and the resulting, possibly deformed, shapes which induce a RIOT in texture. Hinged dissections are shown in a circular sequence.

Parameters. There are five tunable parameters in our method and unless otherwise mentioned, all the results shown in the paper were obtained under the default parameter setting: sampling distance for candidate vertices $d_{space} = \frac{L^c}{15}$, distance tolerance for boundary simplification $\tau_s = 0.1$, threshold for congruency score $\tau_c = 0.3$, variances for reversibility score $\sigma_{PA} = 1$ and $\sigma_W = 4$.

Statistics and timing. We implemented our algorithms entirely in MATLAB and tested them on a 4 GHz desktop. When applying the filtering over our large shape collection, the average time to compute a reversibility score per shape and a cross-reversibility score per pair are 0.12 seconds and 1.99 seconds, respectively. The number of sample points along a shape boundary ranges from 128 to 282, with an average of 191. The number of sparse sample points for diverse polygons ranges from 22 to 54, with an average of 33.

The average time for intra-shape reversibility assessment (candidate trunks per shape) and cross-reversibility assessment (trunk pair selection) for input shape pairs which passed the filtering are 10.36 seconds and 11.90 seconds, respectively. The most time-consuming component of our RIOT construction is boundary deformation, requiring about 2.19 minutes on average for shape pairs with cross-reversibility scores greater than 0.5. With a C/C++ implementation, a significant speedup should be expected [Andrews 2012]. For a more concrete picture of statistics and timing, a table for the shape pairs in Figure 14 is provided in the supplementary material.

User study on human capability. Even for pairs of simple shapes, deciding whether a RIOT exists and if so, constructing the reversible transform, still appear to be highly challenging tasks for a human. We conducted a small user study to assess human capabilities in carrying out the first decision task. Clearly, the second task involving constructions is considerably more demanding.

In the study, each human participant is first shown what an exact RIOT is and then what an approximate RIOT is, along with visual examples. Then we show the participant 16 pairs of shapes. Eight of them were from the gallery (Figure 14), whose reversible transforms incur the least amount of boundary deformations; these eight pairs are considered as positive instances. The other eight shape pairs are from our large shape collection and they would require significant boundary deformations to attain an approximate reversible transform; these pairs are considered as the negative instances. We ask the participants to provide a yes/no answer relating to whether an approximate reversible transform, like the ones he/she had seen, exists for each of the 16 shape pairs. Note that we do not impose a time limit on the participants when they make their judgments.

We invited 30 participants who are graduate students with computer science or mathematics background. In the end, among a total of $30 \times 16 = 480$ responses, the percentage of correct answers, based on our designation of positive and negative instances in the 16 shape pairs, is only 41%. All the shape pairs and user study material can be found in the supplementary material.

Comparisons with manual designs. Before our work, the only available reversible transforms we could find were manually designed by Jin Akiyama; there were nine of them. In Figure 15, we show three such pairs with the manual designs and contrast them with fully automatic RIOT solutions found by our algorithm. Additional comparisons can be found in the supplementary material. In Figure 16, we show two designs which our current construction cannot handle since the boundary of the shapes are too complex and contain too many concave and protrusive features.

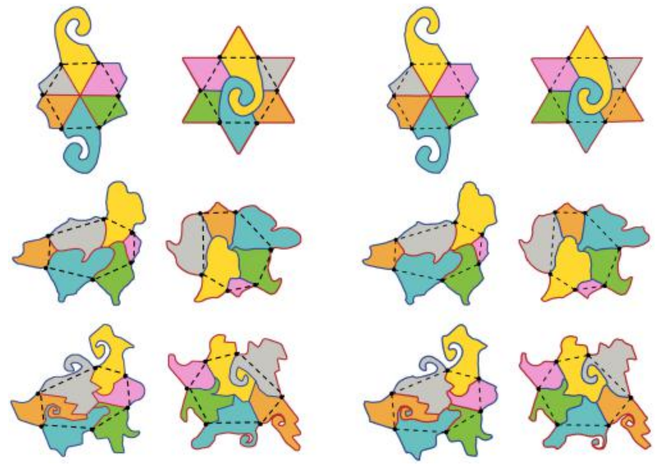


Fig. 15. Reversible shape transforms manually designed by Akiyama (left) vs. those computed by our automatic algorithm on the same input (right). The results are almost identical.

Fig. 16. Two manually designed RIOTs by Akiyama that our current algorithm cannot handle due to excessive boundary complexity.

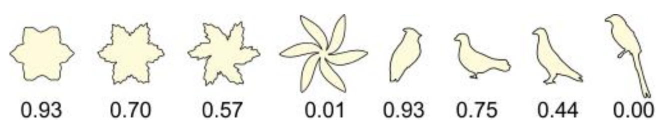


Fig. 17. A sampler of shapes (four from the 'device1' class and four from the 'bird' class) and their reversibility scores.

Aside from the two complex examples in Figure 16, our automatic algorithm is able to obtain nearly identical RIOT solutions as Akiyama's manual designs, bearing some barely noticeable variations arising from discrepancies in boundary discretization. Note that all the manual designs are exact RIOTs while our algorithm seeks an approximation transform. That said, we needed to adjust one parameter for two of seven test pairs, shown in the first two rows of Figure 15. Specifically, we relaxed the distance tolerance for boundary simplification τ_s from 0.1 to 0.07. All other parameters were set as defaults and no adjustment is needed for the remaining test pairs.

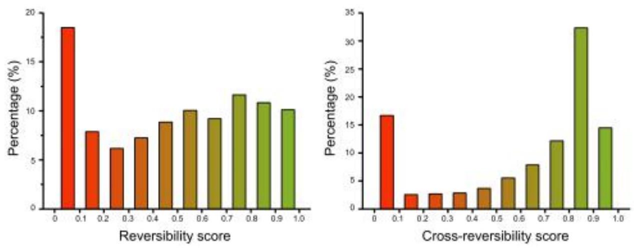


Fig. 18. Reversibility score distributions. Left: for scores of 3,400 individual shapes in our shape collection. Right: for quick cross-reversibility scores of 3,240 selected shape pairs.

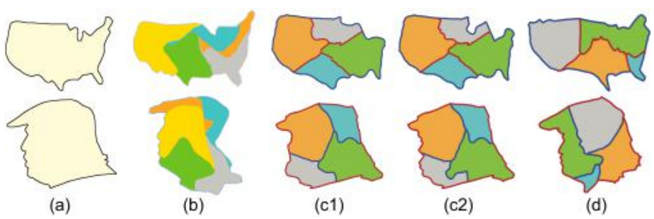


Fig. 19. User assistance in recognizing shape semantics helps improve results. The input pair (a) was from [Duncan et al. 2017] and (b) shows their approximate (non-hinged) dissection result. (c1): fully automatic result from our algorithm. (c2): result with user assistance during boundary deformation to better preserve the facial features. To obtain the best result in (d), the user selected a different trunk pair, the one ranked right after the trunk pair in (c). The new trunk pair does not involve a split of the face part of the shape.

Reversibility scores. We explore our large shape collection to discover potential RIOT pairs by computing reversibility scores for all shapes (see a few examples in Figure 17) and selecting high-score shapes from each class. Then quick cross-reversibility scores (QCRS) between selected shapes from different classes are computed. In Figure 18, we show the distribution of reversibility scores of individual shapes and the QCRS distribution for selected shape pairs.

In the supplementary material, we show reversible transforms computed by our algorithm for the top 100 shape pairs following the ranking given by the QCRS. This score is meant to enable a quick way to identify promising shape pairs as inputs for RIOT construction. On the other hand, the most costly CRS, given in Equation (8), is computed *during* RIOT construction and provides a more accurate assessment of whether two shapes possess a reversible transform. To evaluate QCRS, we test how consistent it is, with respect to the CRS, in rating cross-reversibility of shape pairs. We randomly sampled 1,000 pairs of shape pairs from our shape collection. For each shape pair $\{P_1, P_2\}$, we compute its CRS and QCRS, and each score provides an ordering of P_1 and P_2. We would like to examine how consistent these orderings are. In the end, among the 1,000 pairs of shape pairs, QCRS is consistent with CRS in 77.4% of the time.

User assistance. Our current fully automatic construction algorithm is *not* aware of shape semantics. It is not designed to recognize or preserve small-scale but semantically important shape features, e.g., the bird's beak in Figure 5 and the facial features in Figure 19.

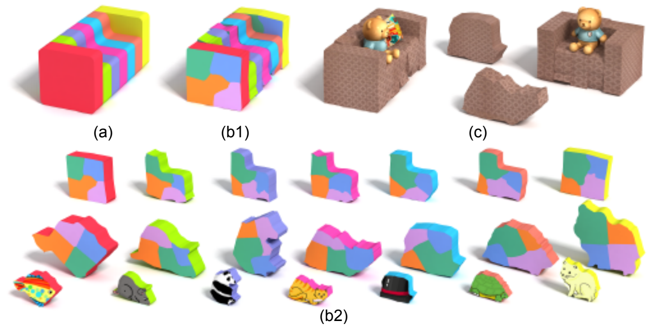


Fig. 20. A 3D sofa is partitioned into thick parallel slices and each slice can undergo a reversible transform. (a) Input sofa with two square slices and five L-shaped slices. (b1) Output sofa after computing RIOTs, resulting in slight deformations of the slice shapes. (b2) The RIOT pairs for all the slices. Small textured shapes are provided to hint what they are. (c) Two possible sofa configurations: we could remove two slices from the double sofa to obtain a loveseat.

As shown in Figure 19, with user assistance in recognizing shape semantics and using that knowledge during trunk pair selection and boundary deformation, we can obtain more meaningful results.

Application. Aside from puzzle making, one may also explore applications of reversible shape transforms to furniture or other artistic designs. When the design is for planar pieces, such as the sofa backs in Figure 2, the applicability is straightforward. One way to make RIOTs work for a 3D shape is to partition the shape into thick slices and compute a transform for each slice, as shown in Figure 20.

7 DISCUSSION AND FUTURE WORK

On first sight, reversible inside-out shape transform is a fascinating, but seemingly next-to-impossible, phenomenon. It is hard to imagine that there are much more than a handful of examples to support such transforms; they are difficult to visualize, let alone construct. In this paper, we show that by relaxing the problem, we can open a whole new set of possibilities for this new and elegant instance of hinged geometric dissections. Specifically, we pose the approximate reversible inside-out transform problem, where the input shapes can be slightly deformed, and present a construction algorithm that works effectively and efficiently on 2D shapes of many varieties. This is complemented by a quick mechanism to extract promising transformable pairs, allowing us to explore reversible hinged dissections over a large shape collection.

Limitations. Our construction algorithm is solely based on finding conjugate trunks, which only provide a sufficient condition for the existence of reversible hinged dissections. Therefore, even if our algorithm is unable to find a pair of conjugate trunks, it does not imply that a reversible transform does not exist. In some of the manual designs of Akiyama, the trunks are not convex and may contain curved edges, while our method assumes that all trunks are convex polygons. Moreover, our current construction is unable to handle input shapes with excessive boundary complexity such as the examples shown in Figure 16. Although both examples in the figure are reversible, our algorithm assigns low reversibility scores to them. Finally, our current boundary deformation scheme still leaves much room for improvement in terms of feature preservation and consideration of shape semantics.

Future work. Aside from addressing the technical limitations, we shall port our implementation from MATLAB to C/C++ which should result in a significant performance boost. We would also like to put together the various components of our method and develop an integrated tool for the design and fabrication of hinged dissections. A difficult but worthwhile technical problem to look into is how the interior dissections may be constrained to respect part boundaries; this may necessitate more aggressive boundary deformations. It is also natural to think about what may be a feasible extension of reversible hinged dissections to 3D shapes.

Compared to common dissection puzzles, reversibility and hinging should add some new twists and dynamics into the player experience. While the linear hinge topology and relatively fewer dissection pieces may make the puzzle rather simple for a smart adult, young children should still find it fun and challenging. From a puzzle design standpoint, there are simple ways to make such puzzles a lot more difficult. For example, we can further dissect the pieces resulting from our method. We can also mix pieces from different shape pairs together. Textures do not need to be already attached to the pieces; we can let the players paint or attach them over their solutions afterwards. For example, we can mix the dissection pieces (without texture) of the set of sofa back slices in Figure 2 to obtain a hard puzzle. All these possibilities can be further explored and added to upgrade the difficulty of the puzzles based on the consumer demands.

REFERENCES

Timothy G. Abbott, Zachary Abel, David Charlton, Erik D. Demaine, Martin L. Demaine, and Scott D. Kominers. 2012. Hinged dissections exist. *Discrete & Computational Geometry* 47, 1 (2012), 150–186.

Jin Akiyama, Stefan Langerman, and Kiyoko Matsunaga. 2015. Reversible nets of polyhedra. In *Japanese Conference on Discrete and Computational Geometry and Graphs*. Springer, 13–23.

Jin Akiyama and Kiyoko Matsunaga. 2015. *Treks into Intuitive Geometry*. Springer.

Jin Akiyama and Gisaku Nakamura. 2000. Dudeney Dissections of Polygons and Polyhedrons–A Survey–. In *Japanese Conference on Discrete and Computational Geometry*. Springer, 1–30.

Tyler Andrews. 2012. Computation time comparison between matlab and C++ using launch windows. *Research report submitted to American Institute of Aeronautics and Astronautics, California Polytechnic State University San Luis Obispo* (2012).

Xiang Bai, Wenyu Liu, and Zhuowen Tu. 2009. Integrating contour and skeleton for shape classification. In *Proc. ICCV*. IEEE, 360–367.

Yurii D. Burago and Viktor A. Zalgaller. 2013. *Geometric inequalities*. Vol. 285. Springer Science & Business Media.

Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qixing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. 2015. Dapper: Decompose-and-Pack for 3D Printing. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 34, 6 (2015), 213:1–213:12.

Erik D. Demaine, Martin L. Demaine, David Eppstein, Greg N. Frederickson, and Erich Friedman. 2005. Hinged Dissection of Polyominoes and Polyforms. *Computational Geometry* 31, 3 (2005), 237–262.

David H. Douglas and Thomas K. Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10, 2 (1973), 112–122.

Henry E. Dudeney. 1902. Puzzles and prizes. *Column in the Weekly Dispatch, April 19, 1896–March 27, 1904* (1902).

Noah Duncan, Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. 2017. Approximate Dissections. *ACM Trans. on Graph.* 36, 6, Article 182 (Nov. 2017), 13 pages.

Thomas Eiter and Heikki Mannila. 1994. *Computing discrete Fréchet distance*. Technical Report. Tech. Report CD-TR 94/64, Information Systems Department, Technical University of Vienna.

Greg N. Frederickson. 1997. *Dissections: Plane and Fancy*. Cambridge University Press.

Greg N. Frederickson. 2002. *Hinged Dissections: Swinging and Twisting*. Cambridge University Press.

RJ Gardner. 1985. A problem of Sallee on equidecomposable convex bodies. *Proc. Amer. Math. Soc.* 94, 2 (1985), 329–332.

Longin Jan Latecki, Rolf Lakamper, and T Eckhardt. 2000. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE CVPR*, Vol. 1. IEEE, 424–429.

Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. 2011. A geometric study of v-style pop-ups: theories and algorithms. In *ACM Trans. on Graph.*, Vol. 30. ACM, 98.

Maarten Löffler, Mira Kaiser, Tim van Kapel, Gerwin Klappe, Marc van Kreveld, and Frank Staals. 2014. The Connect-The-Dots Family of Puzzles: Design and Automatic Generation. *ACM Trans. on Graph.* 33, 4, Article 72 (2014), 72:1–72:10 pages.

Nobuyuki Otsu. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* 9, 1 (1979), 62–66.

Reza Sarhangi. 2008. Making patterns on the surfaces of swing-hinged dissections. *Bridges Leeuwarden Proceedings* (2008), 251–258.

Ariel Shamir. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum* 27, 6 (2008), 1539–1556.

Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. 2017. Reconfigurable Interlocking Furniture. *ACM Trans. on Graph.* 36, 6, Article 174 (Nov. 2017), 14 pages.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 175–184.

Timothy Sun and Changxi Zheng. 2015. Computational Design of Twisty Joints and Puzzles. *ACM Trans. on Graph.* 34, 4, Article 101 (2015), 101:1–101:11 pages.

William Wallace. 1831. *Elements of Geometry* (8th ed.). Bell & Bradfute.

Shiqing Xin, Chi-Fai Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. 2011. Making Burr Puzzles from 3D Models. *ACM Trans. on Graph.* 30, 4, Article 97 (2011), 97:1–97:8 pages.

Christopher Yu, Keenan Crane, and Stelian Coros. 2017. Computational Design of Telescoping Structures. *ACM Trans. on Graph.* 36, 4, Article 83 (July 2017), 9 pages.

Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. 2014. Boxelization: folding 3D objects into boxes. *ACM Trans. on Graph.* 33, 4 (2014), 71.

Yahan Zhou and Rui Wang. 2012. An Algorithm for Creating Geometric Dissection Puzzles. In *Proc. of Bridges Conf.* 49–58.

Changqing Zou, Junjie Cao, Warunika Ranaweera, Ibraheem Alhashim, Ping Tan, Alla Sheffer, and Hao Zhang. 2016. Legible Compact Calligrams. *ACM Trans. on Graph.* 35, 4, Article 122 (July 2016), 122:1–122:12 pages.