Approximate Dissections

NOAH DUNCAN, University of California, Los Angeles
LAP-FAI YU, University of Massachusetts, Boston
SAI-KIT YEUNG, Singapore University of Technology and Design
DEMETRI TERZOPOULOS, University of California, Los Angeles

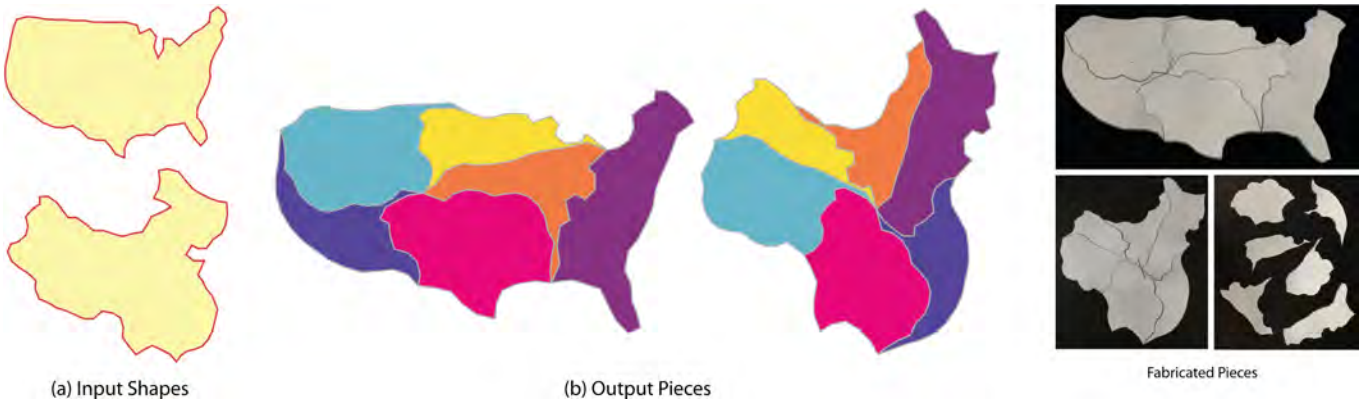


Fig. 1. Given two input shapes (a), our approach generates a small number of pieces that may be arranged to form close approximations of the shapes. Here, we generate six pieces that can form the outlines of either the continental United States or China, demonstrating that the countries have roughly equal area (at the same scale). Note that in (b) both shapes are composed of the same set of pieces.

A geometric dissection is a set of pieces which can be assembled in different ways to form distinct shapes. Dissections are used as recreational puzzles because it is striking when a single set of pieces can construct highly different forms. Existing techniques for creating dissections find pieces that reconstruct two input shapes exactly. Unfortunately, these methods only support simple, abstract shapes because an excessive number of pieces may be needed to reconstruct more complex, naturalistic shapes. We introduce a dissection design technique that supports such shapes by requiring that the pieces reconstruct the shapes only *approximately*. We find that, in most cases, a small number of pieces suffices to tightly approximate the input shapes. We frame the search for a viable dissection as a combinatorial optimization problem, where the goal is to search for the best approximation to the input shapes using a given number of pieces. We find a lower bound on the tightness of the approximation for a partial dissection solution, which allows us to prune the search space and makes the problem tractable. We demonstrate our approach on several challenging examples, showing that it can create dissections between shapes of significantly greater complexity than those supported by previous techniques.

CCS Concepts: • **Computing methodologies** → **Shape analysis**;

Additional Key Words and Phrases: Computational Geometry, Dissection

0730-0301/2017/11-ART182 $15.00
https://doi.org/10.1145/3130800.3130831

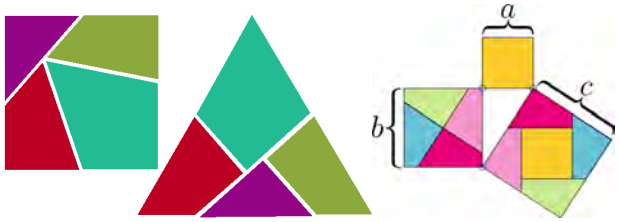
ACM Reference format:
Noah Duncan, Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. 2017. Approximate Dissections. *ACM Trans. Graph.* 36, 6, Article 182 (November 2017), 13 pages.
https://doi.org/10.1145/3130800.3130831

1 INTRODUCTION

Geometric dissections are a popular type of puzzle and mathematical tool. A geometric dissection between two shapes is a partition of one shape into pieces, such that the pieces can be rearranged through rigid motion to form the other shape. Dissections have been known since ancient times. For example, Plato described a dissection between two equally sized squares and one larger one [Frederickson 2002]. Perhaps their first known appearance is on a Babylonian tablet from 1800 BC, which shows the Pythagorean theorem for the special case of a right isosceles triangle. Fig. 2 shows a classic dissection and one that illustrates the classic theorem.[1]

Dissections fascinate us because of the counter-intuitive property that a suitable set of pieces can transform between two distinctive shapes. This striking property means that dissections are popular as recreational puzzles. The transformation property is most impressive when the number of pieces used in the puzzle is minimized. Therefore designers of dissection puzzles usually try to minimize the number of pieces.

[1] More recent mathematical results include the proof that any two polygons of equal area have a dissection between them (and that two polyhedra of equal volume do not, in general, have a dissection between them).



(a) (b)

Fig. 2. (a) A classic four piece dissection between a square and a triangle. (b) A dissection between a square and a pair of smaller squares, which illustrates the Pythagorean theorem.

In mathematics research there has been significant work in finding minimal dissections for analytical shapes such as circles, triangles and regular polygons.[2] The Wallace-Bolyai-Gerwien theorem [Gardner 1985] describes a procedure to create a dissection between any two polygons of equal area, but the number of pieces it uses may be much larger than the minimal number needed. The computational task of determining whether a K-piece dissection exists between two polygons has been shown to be NP-hard by Bosboom et al. [2015]. Manurangsi et al. [2016] showed that the task is similarly hard to solve approximately.

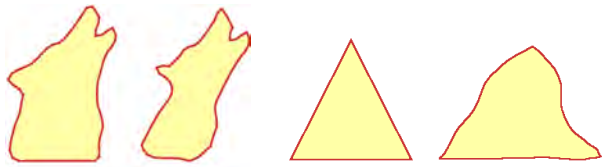
A practical tool for designing dissections between arbitrary shapes with a minimal number of pieces is desirable. However, the complexity arguments cited above make this an intractable task and even if such a tool did exist, the minimum number of pieces might still be extremely large for many shapes.

In this paper, we introduce a practical technique for dissection design that largely avoids these issues. Our technique is based on the observation that, for a large class of shapes, it is acceptable for a dissection to *approximate* rather than exactly construct the shapes. These are the shapes of complex real-world objects whose geometric specification is fuzzy, such as the dog's head in Fig. 3(a). Our technique is not intended for use on abstract shapes with an exact geometric specification, because human perception is sensitive to distortions in these shapes (see, e.g., Fig. 3(b)). Based on this observation, we propose a modified dissection problem in which the input shapes impose soft rather than hard constraints. This relaxation of the problem lets us develop an algorithm that generates dissections that differ qualitatively from traditional ones.

Problem Statement. Let S_1 and S_2 denote the input shapes, represented as polygons. Let $\mathbf{P}$ denote the set of dissection pieces. Let $A_1(\mathbf{P})$ and $A_2(\mathbf{P})$ denote two non-overlapping arrangements of the pieces $\mathbf{P}$. Fig. 4 illustrates these variables.

In the classic K-piece dissection problem, given the input shapes and an integer K, we find pieces and arrangements such that $|\mathbf{P}| = K$ and the constraints $A_1(\mathbf{P}) = S_1$ and $A_2(\mathbf{P}) = S_2$ are satisfied. In the approximate K-piece dissection problem, we instead minimize a shape difference measure between the input shapes and the arranged pieces: $||A_1(\mathbf{P}) - S_1|| + ||A_2(\mathbf{P}) - S_2||$.

[2]Cohn et al. [1975] investigated the minimum number of pieces needed for a triangle-square dissection. Kranakis et al. [2000] gave an asymptotic result on the minimum number of pieces needed for a dissection between a regular m-gon and an n-gon.



(a) (b)

Fig. 3. (a) The distorted dog head is still perceived as a dog head. (b) The distorted triangle is no longer perceived as a triangle.

(a) Input Shapes (b) Pieces (c) Arrangements

Fig. 4. The variables of the dissection problem.

Our core technical contribution in this work is the introduction of the approximate dissection problem and a practical technique for solving it. To our knowledge this is the first general technique for dissections between naturalistic shapes.

As an extension to our method, we develop a graphical user interface for refining dissections that suggests edits to the user and visualizes how altering one part of the dissection affects the remainder.

2 RELATED WORK

2D Shape-Guided Synthesis. Our work belongs to a family of graphics research in which a 2D shape guides the synthesis of some object, such as ASCII art [Xu et al. 2010], mazes [Xu and Kaplan 2007], Escher tiles [Kaplan and Salesin 2000], calligrams [Zou et al. 2016], and connect-the-dot puzzles [Löffler et al. 2014]. In our work, we are guided by two shapes and the object is a set of pieces that can approximate both shapes.

Manual Dissection Design. The last 50 years have seen a surge of recreational interest in finding minimal-piece dissections between certain abstract figures, which are often regular polygons. These results are only for specific instances of the dissection problem, but some heuristic techniques for finding solutions have been identified. Frederickson [2003] did the seminal work in this area. The shapes used in these dissections are much simpler than those featured in our work.

Computational Dissection Design. There has been a modest amount of research in computational techniques for solving the dissection problem and some similar problems. Zhou et al. [2012] introduced an algorithm for finding the minimum number of pieces for an *exact* dissection between shapes. Their method uses a voxel grid to represent the input shapes and a stochastic search strategy. In theory, a sufficient number of voxels could accurately capture the complex

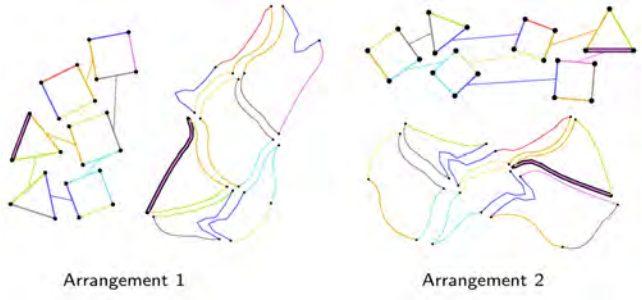
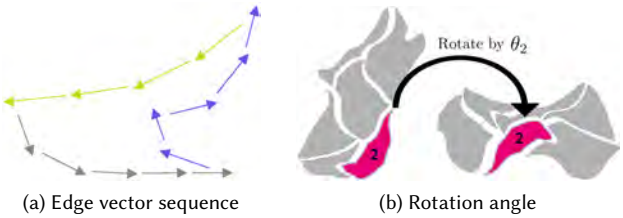


Fig. 5. A dissection between a dog head and a bone. The boundary intervals are colored according to which connection constraint they belong and are separated by black dots. In the first arrangement, the bolded pink boundary interval is *external* because it is not adjacent to another boundary interval. In the second arrangement, the same boundary interval is no longer external. We show the connectivity representations next to the corresponding geometry.



(a) Edge vector sequence (b) Rotation angle

Fig. 6. Visualizing the geometric parameters. (a) A dissection piece is represented by a sequence of edge vectors, colored according to their boundary interval. (b) The change in the orientation of each piece from the first arrangement to the second is represented by an angle for that piece.

organic shapes targeted by our approach, but the results shown in their paper are limited to coarse voxel grids and simple shapes. Our continuous solution representation allows for more complex shapes. Zhou et al. [2014] proposed an algorithm that partitions a 3D shape into approximately cubic pieces and connects them with hinges so that it can be folded into a cube. Their problem statement is related to the dissection problem in that they try to partition an object into pieces so that it can transform to another shape. Unlike our approach they make no effort to minimize the number of pieces, instead focusing on finding a viable hinge connectivity. Huang et al. [2016] proposed a method that, given two shapes, partitions the first into pieces that can be connected through hinges to transform into a shape that roughly approximates the second. Unlike the former work, they do not require physical feasibility. They determine the partition and hinge connectivity of the first shape through a user-provided skeleton, which limits the generality of their solutions. Their work targets significantly coarser approximations than ours. Kwan et al. [2016] recently proposed a novel shape descriptor that can be used to solve the 2D collage problem. In this problem, the goal is to tightly pack shapes from a given library so that they approximate a larger shape. Their problem is similar to ours in that they approximate a shape with a set of pieces, but different in that the pieces are fixed and only need to form a single shape. Concurrent research by Song et al. [2017] explored the design of furniture that can reconfigured to form a different type of furniture, which is highly related to the dissection problem.

3 REPRESENTATION

Our solution to the approximate dissection problem stems from a novel way of representing dissection solutions that supports the notion of a *lower bound* on the objective value of a partial dissection solution. The lower bound allows us to prune the search space by telling us when a partial solution cannot lead to high quality solutions.

A solution in our representation consists of three components: (1) the connectivity representation (Fig. 5) describes how the dissection pieces connect, (2) the geometric parameters (Fig. 6) specify the actual dissection geometry under the constraint of the given connectivity, and (3) the reconstruction mapping (Fig. 9) describes which regions of the input shapes are reconstructed by which regions of the dissection pieces.

Connectivity Representation. The connectivity representation of a dissection solution specifies how the dissection pieces fit together in each arrangement.

The boundary of each dissection piece is partitioned into *boundary intervals*. Fig. 5 shows the pieces for a complete dissection with the boundary intervals delimited. For each arrangement of the pieces, we note the adjacencies between boundary intervals; i.e., when two boundary intervals are touching each other. If a boundary interval is not adjacent to any others in a given arrangement, then it is termed *external* in that arrangement. The external boundary intervals reconstruct the input shape in each arrangement. The boundary intervals are not allowed to partially overlap; i.e., in a given arrangement, all boundary intervals are adjacent to at most one other boundary interval.

The connectivity representation formally consists of a sequence of boundary intervals for each dissection piece specifying their clockwise order and a record of the boundary interval adjacency for each arrangement.

Geometric Parameters. The connectivity representation does not specify the geometry of the dissection. We specify the geometry of each dissection piece as a sequence of edge displacement vectors in clockwise order around the piece. Each edge displacement vector forms a portion of one of the piece's boundary intervals.

To specify how the orientation of each dissection piece changes as it moves from one arrangement to the other, we use a rotation angle for each piece. Specifically, dissection piece k is rotated θ_k degrees in Arrangement 2 relative to its orientation in Arrangement 1. The geometric parameters are visualized in Fig. 6.

We determine the value of these geometric parameters for a given solution by optimizing how well they reconstruct the target shapes. This process is described in Section 4.

Connection Constraints. The connection constraints are geometric constraints placed on boundary intervals that ensure the dissection pieces can physically connect. They reduce the number of independent geometric parameters.

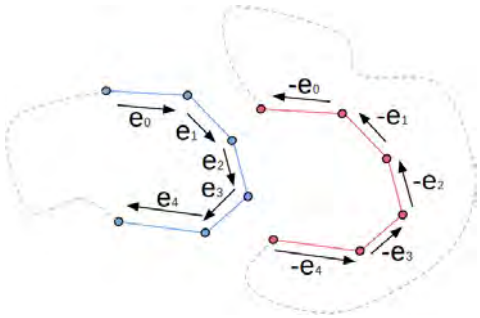


Fig. 7. Two boundary intervals that connect. The geometry of both boundary intervals is parameterized by 5 edge displacement vectors. The rest of the dissection pieces are in grey.

As mentioned previously, each boundary interval is specified as a sequence of edge displacement vectors in clockwise order along the dissection piece. If a pair of boundary intervals β and β' are adjacent, then for β' to connect flush with β, it must have an edge vector sequence equal to the reversal and negation of β's. Formally if $\beta = \{\vec{e}_1, \ldots, \vec{e}_N\}$ then $\beta' = \{-\vec{e}_N, \ldots, -\vec{e}_1\}$. Intuitively, if β is "male" then β' must be "female". Fig. 7 visualizes this constraint.

Connection constraints lock the geometry of adjacent boundary intervals into a common parameterization. In general, connection constraints parameterize the geometry of *sets* of boundary intervals, rather than pairs. This is because a boundary interval β may be adjacent to β' in Arrangement 1 and β'' in Arrangement 2.

Fig. 8 shows how a connection constraint controls the geometry of part of the dissection. Here, a single set of edge vectors parameterizes the geometry of four different boundary intervals. Fig. 8(b) shows how this parameterization propagates edits to the dissection.

We can form a *partition* of the boundary intervals by grouping them according to which connection constraint controls them. Fig. 5 visualizes this property. Boundary intervals of the same color belong to the same connection constraint.

Reconstruction Mapping. The reconstruction mapping (Fig. 9) describes which regions of the input shapes are reconstructed by which regions of the dissection pieces. The mapping consists of two bijections $\Gamma_1(I)$ and $\Gamma_2(I)$, where $\Gamma_\alpha(I)$ is a bijection which maps a boundary interval I on input shape α to an external boundary interval β on the dissection pieces in arrangement α.

4 EVALUATION

Given a candidate dissection solution, we evaluate how well it approximates the input shapes by optimizing over the dissection geometry parameters for the best approximation; i.e., we solve the following constrained optimization problem:

$$\begin{aligned} \underset{\mathbf{E},\boldsymbol{\theta}}{\text{minimize}} \quad & \max_{\alpha,i} \arccos(\hat{\mathbf{b}}_{\alpha,i} \cdot \hat{\mathbf{t}}_{\alpha,i}) + \lambda \sum_{\alpha} \sum_{i} ||\mathbf{b}_{\alpha,i} - \mathbf{t}_{\alpha,i}||^2 \\ \text{subject to} \quad & \sum_{l} \mathbf{p}_{k,l} = \mathbf{0}, \quad \forall k. \end{aligned} \tag{1}$$

The objective function measures how tightly the dissection pieces approximate the input shapes, according to the maximum angular

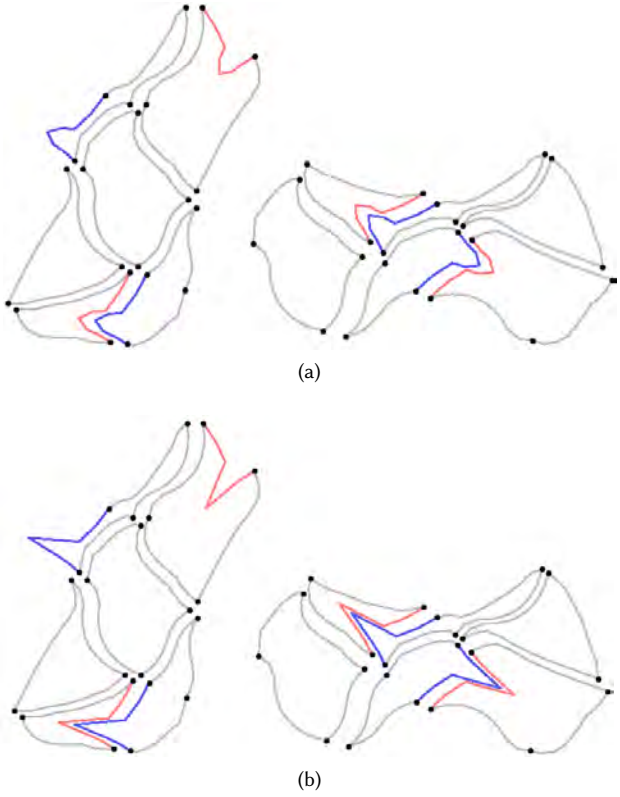


Fig. 8. (a) A connection constraint that controls the geometry of 4 different boundary intervals on the dissection pieces. Blue boundary intervals are "male" connectors, red ones are "female", and grey ones are not part of the connection constraint. (b) Changes to one boundary interval in the connection constraint affect the others. In this example, if the dog's mouth becomes deeper, the ear becomes longer.

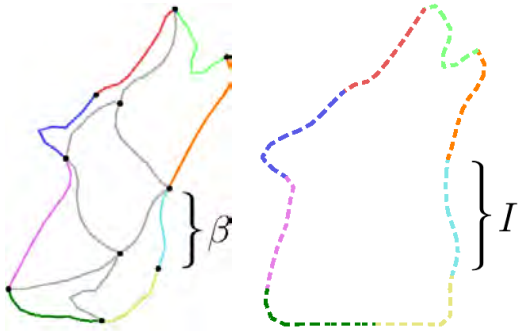
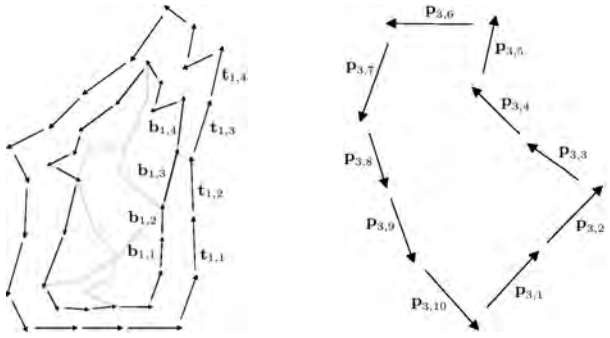


Fig. 9. Reconstruction mapping for the dog arrangement. The external boundary intervals on the dissection pieces (a) are colored according to which boundary interval on the input shape (b) they reconstruct. Internal boundary intervals (grey) are not involved in the reconstruction mapping. Notation is shown for a dissection piece boundary interval (β) and its corresponding input shape boundary interval (I). Since the dog is the first input shape, this figure visualizes Γ_1.



(a) Vectors for objective (b) Vectors in a constraint

Fig. 10. (a) Edge vectors involved in the objective function in (1). (b) Edge vectors involved in the constraint for a piece in (1).

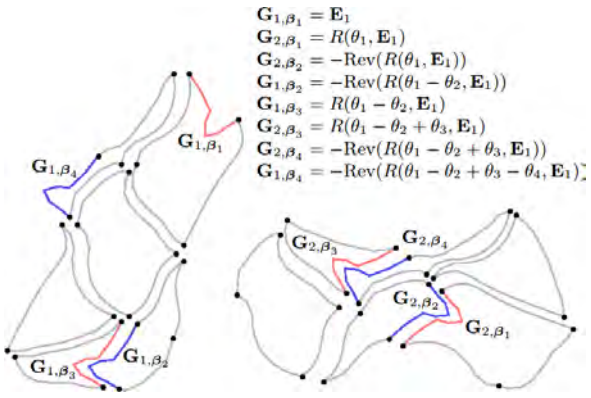


Fig. 11. Derivation of the formulas for the boundary interval geometry for a single connection constraint.

error between dissection pieces and input shape edge vectors. Integer i indexes the edges of the reconstructed shape in arrangement α. Vector $\mathbf{b}_{\alpha,i}$ is edge vector i in the reconstruction of input shape α, and $\mathbf{t}_{\alpha,i}$ is the corresponding original input shape edge vector (i.e., the target of $\mathbf{b}_{\alpha,i}$). The value of $\mathbf{t}_{\alpha,i}$ comes from the reconstruction mapping. We set the regularization parameter to $\lambda = 0.01$.

The constraints in (1) ensure that the edge vector sequence for each dissection piece forms a closed curve. Integer k indexes dissection pieces while l indexes edges within a dissection piece. Vector $\mathbf{p}_{k,l}$ is edge vector l of dissection piece k. Fig. 10 illustrates the vectors involved in the objective and constraints.

The optimization variables in (1) are $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_K\}$, the set of rotation angles for the K dissection pieces, and $\mathbf{E} = \{\mathbf{E}_1, \ldots, \mathbf{E}_M\}$, the set of edge vector sequences that parameterize the geometry for the M connection constraints. Connection constraint edge vector sequence $\mathbf{E}_i = \{\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,N_i}\}$ comprises N_i edge vectors. The vector $\mathbf{e}_{i,j}$ denotes edge vector j within connection constraint i.

We solve the optimization problem (1) using IPOPT [Wächter and Biegler 2006]. The optimal objective value determines the quality of the solution.

Geometry Functions. The edge vectors $\mathbf{b}_{\alpha,i}$ and $\mathbf{p}_{k,l}$ may be written as functions of the optimization variables $\mathbf{E}$ and $\boldsymbol{\theta}$. Each of these edge vectors belongs to a specific boundary interval β in an arrangement α. Therefore, we denote these functions as $\mathbf{g}_{\alpha,\beta,i}$, where $\mathbf{g}_{\alpha,\beta,i}(\mathbf{E}, \boldsymbol{\theta})$ is edge vector i within boundary interval β in arrangement α. For brevity, let $\mathbf{G}_{\alpha,\beta}$ denote the sequence of all the edge vectors in boundary interval β in arrangement α; i.e., $\mathbf{G}_{\alpha,\beta} = \{\mathbf{g}_{\alpha,\beta,1}, \ldots, \mathbf{g}_{\alpha,\beta,N}\}$. Fig. 11 visualizes this notation. The functions can be derived using two geometric properties:

(1) For a boundary interval β lying on dissection piece k, we have $\mathbf{G}_{2,\beta} = R(\theta_k, \mathbf{G}_{1,\beta})$ and $\mathbf{G}_{1,\beta} = R(-\theta_k, \mathbf{G}_{2,\beta})$, where $R(\theta, \mathbf{v})$ is an element-wise θ-degree rotation of the vector sequence $\mathbf{v}$.
(2) Let β' denote a boundary interval adjacent to β in arrangement α. Then $\mathbf{G}_{\alpha,\beta'} = -\text{Rev}(\mathbf{G}_{\alpha,\beta})$ where $-\text{Rev}(\mathbf{v})$ reverses the vector sequence $\mathbf{v}$ and applies element-wise negation.

From these properties we can write a formula for any boundary interval in terms of any other in the same connection constraint. To obtain formulas in terms of the optimization variables $\mathbf{E}$, we choose one boundary interval, denoted β_r, for each connection constraint i, and one arrangement denoted α_r, to be the "roots" and set $\mathbf{G}_{\alpha_r,\beta_r} = \mathbf{E}_i$. Formulas for the other boundary intervals in the connection constraint can easily be derived by applying the two rules. Fig. 11 shows an example of this process.

In general, the formula $\mathbf{G}_{\alpha,\beta}$ for a boundary interval in connection constraint i will have the form $R(\pm\theta_{k_1} \cdots \pm \theta_{k_W}, \mathbf{E}_i)$ or $-\text{Rev}(R(\pm\theta_{k_1} \cdots \pm \theta_{k_W}, \mathbf{E}_i))$; i.e., it consists of a series of rotations of the "root" edge vector sequence possibly followed by a reversal and negation.

5 SOLUTION SEARCH

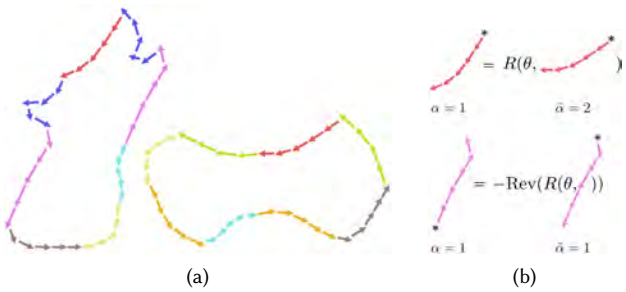
The solution search efficiently explores our solution space for high-quality dissections. The main idea is to enumerate partial dissection solutions in a tree structure (Fig. 13) and prune nodes of the tree when the partial solution they represent has a quality lower bound greater than a pruning threshold.

5.1 Initializing the Mapping and Constraints

The initialization step divides the boundary of each input shape into intervals. These intervals are the *domain* of the reconstruction mapping—the mapping between input shape boundary intervals and the dissection piece boundary intervals that reconstruct them. The initialization also separates the input shape intervals into corresponding pairs (Fig. 12). These correspondences restrict the search space to a promising region. Selecting them also initializes the connection constraints that will be used in the dissection. Each correspondence implies the existence of a single connection constraint, as we show next.

Existence of Correspondences. Recall that the reconstruction mapping is a pair of bijections $\Gamma_1(I)$ and $\Gamma_2(I)$, where $\Gamma_\alpha(I)$ maps a boundary interval I on input shape α to a boundary interval on a dissection piece in arrangement α.

We say that an input shape boundary interval I belongs to a connection constraint $\mathbf{C}$ if $\Gamma_\alpha(I) \in \mathbf{C}$. It turns out that the connection



(a) (b)

Fig. 12. (a) Correspondences for the dog-bone dissection. Edge vectors are colored according to the correspondence to which they belong. Note that correspondences can be between boundary intervals on the same or on different input shapes. (b) Examples of the relationship between edge vector sequences for corresponding boundary intervals. Asterisks mark the beginning of sequences. The indices α and $\tilde{\alpha}$ denote the shape on which each boundary interval lies.

constraints partition the input shape boundary intervals into corresponding pairs. Formally, each input shape boundary interval I belongs to some connection constraint $\mathbf{C}$, and there exists exactly one other input shape boundary interval $\tilde{I}$ that also belongs to $\mathbf{C}$.[3]

Any dissection solution will partition the input boundary intervals into corresponding pairs. We choose to specify these corresponding pairs in advance in order to restrict the search space.

Evaluating a Correspondence. Consider a pair of corresponding input shape boundary intervals $(I, \tilde{I})$ that lie on input shapes $(\alpha, \tilde{\alpha})$ and belong to connection constraint $\mathbf{C}$. The eventual dissection solution will map I and $\tilde{I}$ to dissection piece boundary intervals $(\Gamma_\alpha(I), \Gamma_{\tilde{\alpha}}(\tilde{I})) = (\beta, \tilde{\beta})$. Since β and $\tilde{\beta}$ both belong to $\mathbf{C}$ their geometry (in any eventual solution) must obey the constraint $\beta - Q(\tilde{\beta}) = 0$, where $Q(x) = R(\theta, x)$ when $\alpha \neq \tilde{\alpha}$ (intervals lie on different input shapes) and $Q(x) = -\mathrm{Rev}(R(\theta, x))$ when $\alpha = \tilde{\alpha}$ (intervals lie on the same input shape). The value of θ depends on the eventual solution. Fig. 12(b) visualizes these relations.

To reconstruct the input shapes accurately, the geometry of $(\beta, \tilde{\beta})$ should resemble that of $(I, \tilde{I})$. Thus, in order to allow for the best reconstruction under the connection constraint, the correspondence should minimize $\bar{Q}(I, \tilde{I}) = \min_\theta ||I - Q(\tilde{I})||$; i.e., minimize shape incompatibility given freedom of rotation.

Joint Selection of Intervals and Correspondences. Using this criterion, we jointly select the input shape boundary intervals and correspondences. The input shape boundaries are discretized into a large number of uniformly sized segments so that input shape boundary intervals can be defined discretely. Given user-specified minimum and maximum interval lengths, we generate a set of M candidate boundary intervals and $\binom{M}{2}$ candidate correspondences. We seek a set of correspondences $\{(I_1, \tilde{I}_1), \ldots, (I_N, \tilde{I}_N)\}$ which form a partition of the input shape boundaries and minimize the following objective:

$$\sum_{i=1}^{N} \bar{Q}(I_i, \tilde{I}_i), \tag{2}$$

which sums the shape incompatibility across the chosen correspondences. The shape incompatibility measure $\bar{Q}$ is defined in the previous subsection. We search for solutions using a simple enumerative approach, ignoring correspondences with poor similarity.

[3]To see this property, consider an input shape boundary interval I. Without loss of generality, assume I lies on the first input shape. In any eventual dissection solution, I will map to a dissection piece boundary interval β; i.e., $\Gamma_1(I) = \beta$, which belongs to some connection constraint $\mathbf{C}$. So I belongs to $\mathbf{C}$.

In Arrangement 2, β is either external or it connects to a different boundary interval β'. In the first case, there exists a boundary interval on the input shape for Arrangement 2, denoted $\tilde{I}$, for which $\Gamma_2(\tilde{I}) = \beta$. Since β' belongs to $\mathbf{C}$, $\tilde{I}$ belongs to $\mathbf{C}$. So both I and $\tilde{I}$ belong to $\mathbf{C}$. In the second case, the statement just made about β applies to β', except with Arrangement 1 instead of Arrangement 2. Since the number of boundary intervals is finite, we must eventually reach the first case.

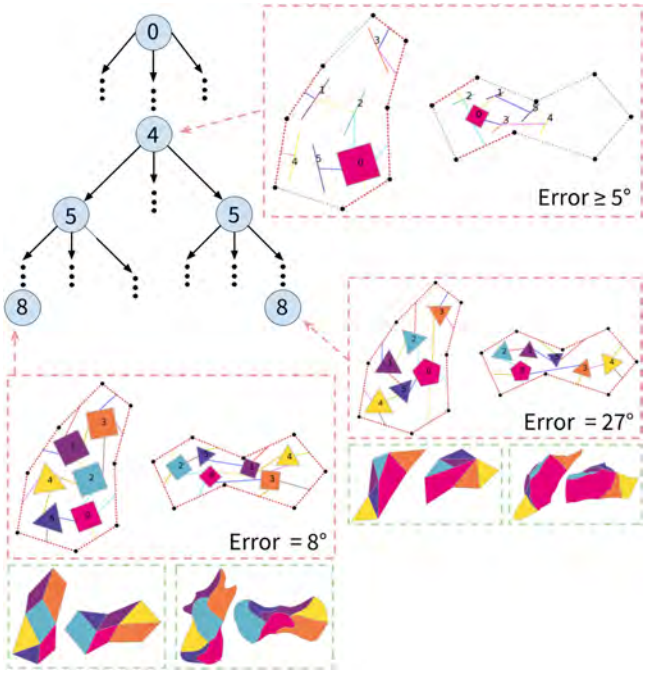
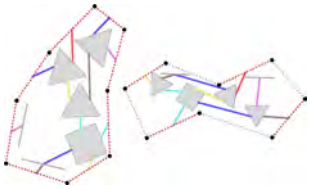


Fig. 13. Visualizing the tree search. Tree nodes (partial solutions) are labeled with the number of connection constraints added (equal to the search depth). Pink boxes show the connectivity representation at each state along with the associated error for complete solutions and the error bound for partial solutions. Input shapes (at low resolution) are drawn around the connectivity representations in dashed lines. Input shape boundary intervals that have their connection constraint assigned are red, others are grey. Green boxes show the dissection geometry at low (left) and high (right) resolutions. In the partial solution shown at Depth 4, pieces that have only one or two boundary intervals are drawn as line segments.

5.2 Tree Search

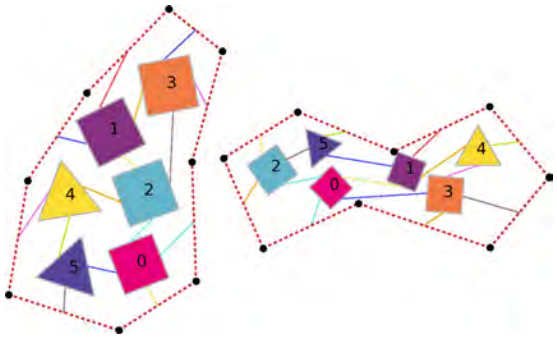
Once the input shape correspondences have been chosen, we enumerate the dissection solutions derived from them using a search tree. This step operates in the discrete connectivity space rather the continuous geometric parameter space.

Each node of the tree corresponds to a partial dissection solution with the root node being the empty solution. A child of a node is generated by extending the node's partial solution by inserting one or more boundary intervals into one or more dissection pieces.



(a) Connection constraint ordering (Constraints 0 - 5)

(b) Dissection piece ordering (Pieces 0 - 3)



(c) Complete solution

Fig. 14. Two different partial solutions (a), (b) which lead to the same complete solution (c). Ordering by connection constraint means that all the boundary intervals in the partial solution have connections. Ordering by dissection piece means that the connectivity of several boundary intervals (dashed lines) is unknown. Boundary interval connections in (a) are colored according to the connection constraint.

The newly created boundary intervals can be connected to other dissection piece boundary intervals or to a boundary interval on an input shape. For the second case, connecting a boundary interval β on a dissection piece to a boundary interval I on input shape α means we set $\Gamma_\alpha(I) = \beta$. In the leaf nodes (complete solutions), every boundary interval will have a connection in both arrangements. Fig. 13 shows how the tree search incrementally extends partial dissection solutions into complete solutions.

Search Ordering. The choice of the *order* in which to insert the boundary intervals affects the ease of pruning partial solutions. An obvious choice is to order by dissection piece; i.e., a node at depth i specifies the boundary intervals for dissection pieces $1, \ldots, i$ (Fig. 14(b)). However, this ordering has the problem that after we insert the boundary intervals for a given piece, we do not necessarily know which boundary intervals they will connect to, since they may connect to boundary intervals on pieces that we have not yet reached. This uncertainty makes it difficult to formulate a bound on the quality of partial solutions.

Therefore, we order the boundary intervals by connection constraint rather than by dissection piece (Fig. 14(a)). That is, a node at depth i specifies the boundary intervals that belong to connection constraints $1, \ldots, i$. To generate the children of a node at depth i, we consider all possible ways of generating the boundary intervals for the connection constraint $i + 1$. Each way creates an additional child.

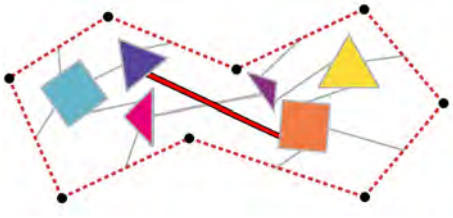


Fig. 15. An example of a topologically invalid connection (shown in red) between boundary intervals.

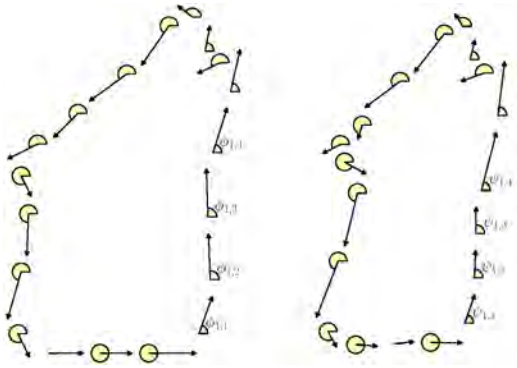


Fig. 16. Visualizing the angular optimization problem. Vector magnitudes are ignored and we optimize only over vector angles. The notation is shown for some of the angles (ϕ and ψ) used in the optimization problem.

This ordering avoids the problem with the dissection-piece-based ordering, because a boundary interval connects only to other boundary intervals in its connection constraint. In other words, for every boundary interval β in a partial solution, we know that the connectivity of β will not change in any subsequent solution. Fig. 14 compares the two ways of ordering the search.

6 PRUNING THE SEARCH SPACE

We now discuss the several ways in which we restrict the search space to make the problem tractable.

Boundary Interval Limit. To ensure that the search terminates, we limit the total number of boundary intervals that can be present in a solution. For the results shown herein, we set this limit to four times the number of connection constraints.

Connectivity. We restrict the search to topologically valid connectivities. We maintain a half-edge data structure throughout the search that detects when connecting a pair of boundary intervals is an invalid operation. Fig. 15 shows an example.

6.1 Orientation Based Pruning

In orientation based pruning, we solve a relaxed version of the optimization problem described in Section 4 that gives a lower bound on the solution quality. The main idea is to solve for the dissection geometry in terms of edge vector *orientations* while ignoring magnitudes. Fig. 16 visualizes this concept.

As in the full optimization problem, we want to minimize the maximum angular difference between the input shape edge vectors and their matching dissection piece edge vectors:

$$\underset{\boldsymbol{\theta}_E, \boldsymbol{\theta}, \mathbf{n}}{\text{minimize}} \quad \max_{\alpha, i} |\phi_{\alpha,i} - \psi_{\alpha,i}(\boldsymbol{\theta}_E, \boldsymbol{\theta}) + 2\pi n_{\alpha,i}|, \tag{3}$$

where α indexes over arrangements, i indexes over the input shape edge vectors, $\phi_{\alpha,i}$ and $\psi_{\alpha,i}$ are matching input shape and dissection piece edge vector orientations, $\boldsymbol{\theta}_E$ is the set of edge vector orientation sequences for the connection constraints, and $\boldsymbol{\theta}$ is the dissection piece rotation angles. The integer variables $\mathbf{n}$ make the angular differences modulo 2π. This objective is similar to that in (1), but expressed only in terms of vector angles. Thus, $\boldsymbol{\theta}_E$ in the angular problem corresponds to $\boldsymbol{E}$ in the full problem, while $\boldsymbol{\theta}$ has the same meaning in both problems.

The dissection piece edge vector orientations $\boldsymbol{\psi}$ can be written as linear functions of $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}$, by converting the functions for the edge vector geometry in Section 4 to angular terms. In general the formulas have the form $\pm\theta_{k_1} \cdots \pm \theta_{k_W} + \boldsymbol{\theta}_{E_i}$ or $\pm\theta_{k_1} \cdots \pm \theta_{k_W} + \pi + \text{Rev}(\boldsymbol{\theta}_{E_i})$. The second form corresponds to the form in the full problem where we reverse and negate the edge vector sequence, because negating a vector adds π to its orientation.

The optimization may be written as a mixed integer optimization, where the variables $\mathbf{n}$ are constrained to be integers, and $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}$ are continuous. We solve it using a commodity solver [Gurobi Optimization, Inc. 2016].

This optimization immediately generalizes to partial solutions. In (3), we maximize only over the edge vectors belonging to connection constraints that are specified in the partial solution. Due to the connection constraint-based ordering, the orientation formulas for these edge vectors will be defined and will not change in any subsequent solution. When a new connection constraint is added to the partial solution, the maximization is taken over additional terms.

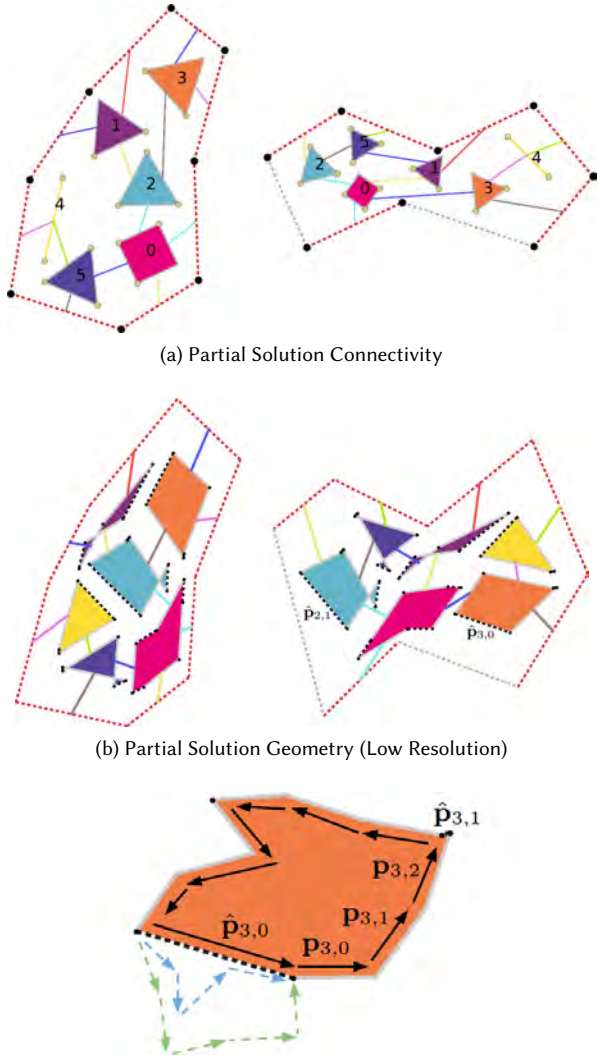
6.2 Full Geometry Pruning

The full geometry pruning generalizes the dissection geometry optimization in Section 4 to partial dissection solutions. The objective value given by this optimization is a lower bound, meaning that no complete solution derived from the partial solution will attain a better objective value. Fig. 17 visualizes this optimization. The main idea is to relax the discreteness of the problem by allowing for the insertion of "fractional" boundary intervals.

The structure of the objective function and constraints in this optimization are identical to that in (1). However, they are expressed differently, due to the incomplete information in a partial solution.

Objective for Partial Solutions. Recall that the objective function compares each input shape edge vector $\mathbf{t}_{\alpha,i}$ with its corresponding dissection piece edge vector $\mathbf{b}_{\alpha,i}$ (Fig. 10(a)). Each $\mathbf{t}_{\alpha,i}$ lies on an input shape boundary interval I that belongs to a connection constraint $\mathbf{C}$. If the partial solution has not yet added the boundary intervals for $\mathbf{C}$, then the vector $\mathbf{b}_{\alpha,i}$ is unknown. Next, we show how to express it in terms of other variables.

Suppose I lies on input shape α. We know that in any complete solution derived from the partial solution, I will be reconstructed by some boundary interval β. Formally, we know there will exist



(a) Partial Solution Connectivity

(b) Partial Solution Geometry (Low Resolution)

(c) Piece 3 Edge Vectors (High Resolution)

Fig. 17. Visualizing the geometry optimization for a partial solution. (a) The connectivity representation of a partial solution where the boundary intervals for the last connection constraint have not been inserted. Locations where new boundary intervals can be inserted are marked with gold dots. (b) The corresponding (low resolution) geometry after optimization. Potential boundary intervals are drawn as dotted lines. (c) The edge vectors for Piece 3 at higher resolution. The edge vectors for boundary intervals in two different complete solutions are shown in blue and green.

a β for which $\Gamma_\alpha(I) = \beta$. As discussed in Section 5.1, I has a corresponding boundary interval $\tilde{I}$ on input shape $\tilde{\alpha}$. Thus β will have a corresponding dissection piece boundary interval $\tilde{\beta} = \Gamma_{\tilde{\alpha}}(\tilde{I})$. The boundary intervals β and $\tilde{\beta}$ will belong to the same connection constraint, so they both can be expressed as functions of an edge vector sequence $\mathbf{E}_i$. Denoting these functions as $\bar{\mathbf{G}}_\beta(\mathbf{E}_i)$ and $\bar{\mathbf{G}}_{\tilde{\beta}}(\mathbf{E}_i)$, we define them as $\bar{\mathbf{G}}_b(\mathbf{E}_i) = \mathbf{E}_i$ and $\bar{\mathbf{G}}_{\tilde{\beta}}(\mathbf{E}_i) = -\text{Rev}(R(\hat{\theta}, \mathbf{E}_i))$ if $\alpha = \tilde{\alpha}$,

and $\bar{\mathbf{G}}_{\hat{\beta}}(\mathbf{E}_i) = R(\hat{\theta}, \mathbf{E}_i)$ otherwise. $\hat{\theta}$ is added to the optimization as a free variable; i.e., it is independent of the rotation angles for the dissection pieces. Note that these functions are like $\mathbf{G}_{\alpha,\beta}$ from Section 4, except that the sum of dissection piece rotation angles is replaced by the free variable $\hat{\theta}$. These functions provide the values for any unknown $\mathbf{b}_{\alpha,i}$.

Constraint for Partial Solutions. The constraint for dissection piece l ensures that the edge vectors around the piece $\mathbf{p}_{k,l}$ form a closed curve (Fig. 10(b)). This formulation does not work for a partial solution, because some of the pieces will have new boundary intervals inserted in any complete solution descended from the partial solution. Thus, it is incorrect to take the sum only around the existing $\mathbf{p}_{k,l}$.

We deal with this issue by inserting *potential* boundary intervals into each dissection piece at points on the piece's boundary where it would be legal to insert a new boundary interval (Fig. 17). Each potential boundary interval represents geometry that could potentially be present in an eventual complete solution. We represent the potential geometry for dissection piece k at insertion point v with a single edge vector, denoted $\hat{\mathbf{p}}_{k,v}$. These edge vectors are free variables in the optimization.

More specifically, $\hat{\mathbf{p}}_{k,v}$ represents the *net translation* of zero or more potentially inserted boundary intervals. By net translation, we mean the vector obtained by summing the edge vectors in the boundary interval. It suffices to consider only the net translations of the potential boundary intervals, because they are connected to nothing in this setting. Fig. 17(c) illustrates this principle. The single edge vector $\hat{\mathbf{p}}_{3,0}$ has the same net translation as the two edge vector sequences shown in blue and green.

The constraints from (1) are modified to take the potential geometry into account, as follows:

$$\sum_l \mathbf{p}_{k,l} + \sum_v \hat{\mathbf{p}}_{k,v} = \mathbf{0}, \quad \forall k. \tag{4}$$

We can tighten the bound by constraining the total length of the potential geometry. Suppose our search has an upper limit of S boundary intervals in the solution, and our partial solution has already inserted S' boundary intervals. Then, we can have at most $S - S'$ additional boundary intervals in any eventual complete solution. Denoting the *net length* of the edge vector sequence for connection constraint i as $L_i = ||\sum_j \mathbf{e}_{i,j}||$ (i.e., L_i is the magnitude of the net translation of the edge vector sequence for connection constraint i), and letting $L_{\max} = \max_i L_i$, we can add the constraint

$$\sum_k \sum_v ||\hat{\mathbf{p}}_{k,v}|| \leq (S - S')L_{\max}. \tag{5}$$

This constraint reflects the fact that we can add at most $S - S'$ boundary intervals and each boundary interval consumes at most $L_{\max}$ length.

The use of potential boundary intervals is conceptually similar to a continuous relaxation of an integer programming problem. The number of boundary intervals inserted into a piece is a discrete property, like the value of an integer variable. The potential boundary intervals relax the discreteness by allowing for the insertion of a "fraction" of a boundary interval, just like continuous relaxation allows for assigning fractional values to an integer variable.

6.3 Pruning Usage

We now describe how the pruning tests are incorporated into the tree search.

The two pruning tests described in the previous subsection require the solution of an optimization problem, which incurs a significant computational expense, especially for the full geometry pruning (Section 6.2). Additionally, at a sufficient depth in the search tree, a significant proportion of tree nodes (partial solutions) do not expand into *any* complete solutions, because it is impossible to form a solution with valid connectivity. Expanding a tree node (by inserting boundary intervals for a given connection constraint and validating their connectivity) is extremely cheap relative to the pruning tests. Therefore, for these nodes it is more efficient (on average) to check if the node expands to any complete solutions, and run the pruning tests only if it does. For even deeper depths of the search tree, it is most efficient to refrain from running the pruning tests at all.

Procedure Details. For all the results shown in the paper, we used the following search procedure. Let M equal the number of connection constraints we obtain from the correspondence search described in Section 5.1. First, all partial solutions at depth $M/2$ are generated and the pruning tests are applied to them. Next, the remaining partial solutions are assigned between P processes that run in parallel. Each process expands its partial solutions serially. At depth $3M/4$, we apply the pruning tests again, but only after verifying that the node contains complete solutions as described above. This simple procedure could likely be improved, but we found it performed acceptably in practice. We set the angular error threshold for pruning to $\min(15^\circ, 1.5 \cdot \tau_{\min})$, where $\tau_{\min}$ is the lowest error among the solutions found so far.

7 USER INTERACTION

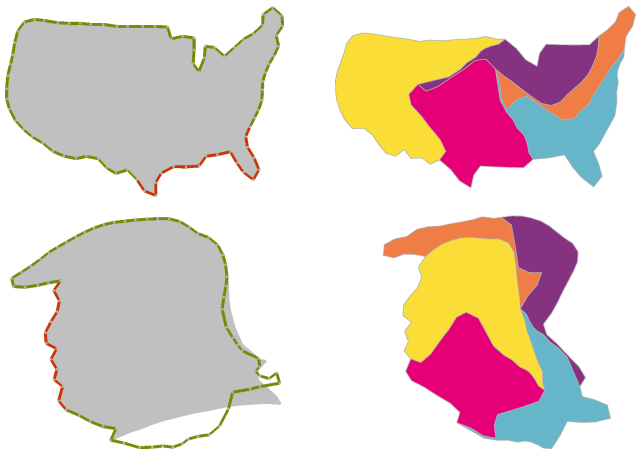
Our solution search uses a geometric criterion to evaluate the quality of a dissection. This criterion works well for weeding out poor solutions, but it has difficulty discriminating between high-quality solutions because it does not explicitly consider human perception. We resolve this shortcoming by allowing user guidance towards a final dissection solution, after the automatic approach described in Section 5 has identified a small set of promising candidate solutions.

The user selects one of the candidate solutions, which they can refine by adjusting the terms of the optimization in (1) through a graphical user interface. Fig. 18 shows an example of this process.

In this phase of our approach, the optimization objective uses a least squares term instead of the maximum angular error in order to control the tradeoff in deformation between different parts of the mesh. The new objective is $\sum_\alpha \sum_i w_{\alpha,i} ||\mathbf{b}_{\alpha,i} - \mathbf{t}_{\alpha,i}||^2$, where $w_{\alpha,i}$ is the preservation weight of edge i on input shape α.

Input Shape Editing Suggestions. Our approach targets input shapes that have a fuzzy geometric specification. Thus, we allow the user to alter the input shapes to simplify the optimizer's job. Such edits are suggested to the user by overlaying dissection piece boundary intervals on top of their corresponding input shape boundary intervals as red curves. The user can follow the suggestions by moving the input shape edges towards these curves. The user often makes additional edits to preserve the shape's identity or coherency. Fig. 18 shows

(a) Initial Design (b) Suggested Edits



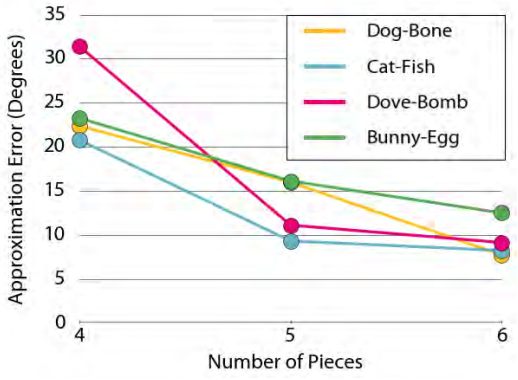
(c) Edited Input Shapes (d) Final Design

Fig. 18. A workflow in our user interface. (a) The initial design from the automatic approach. Problematic areas are circled. The details of the face have been smoothed out and the region around the state of Florida is distorted. (b) The original input shape with edit suggestions in red. (c) The input shapes after the editing and painting of salient regions (red). The original input shape is shown in gray. (d) The final design after user edits. The details of the face have been restored and Florida is no longer distorted.

an example of this process where the user reshapes the bottom and side of the head.

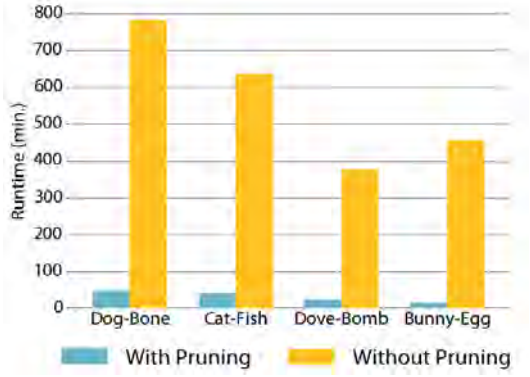
Salient Region Painting. Some areas on the input shapes are *salient* in that human perception is especially sensitive to distortions in them. The user can paint salient parts of the input shape and the optimizer will prioritize them by increasing the preservation weights for their edges. In Fig. 18(c) the user prioritizes the preservation of the protrusions forming the states of Texas and Florida as well as the nose and lips of the head.

Direct Editing. The user can also directly edit the dissection pieces. To satisfy the connection constraints, these edits will propagate to the geometry of other pieces. This propagation is visualized interactively.



(a) Approximation Error vs. Piece Count

(b) Dog-bone dissection with four and five pieces



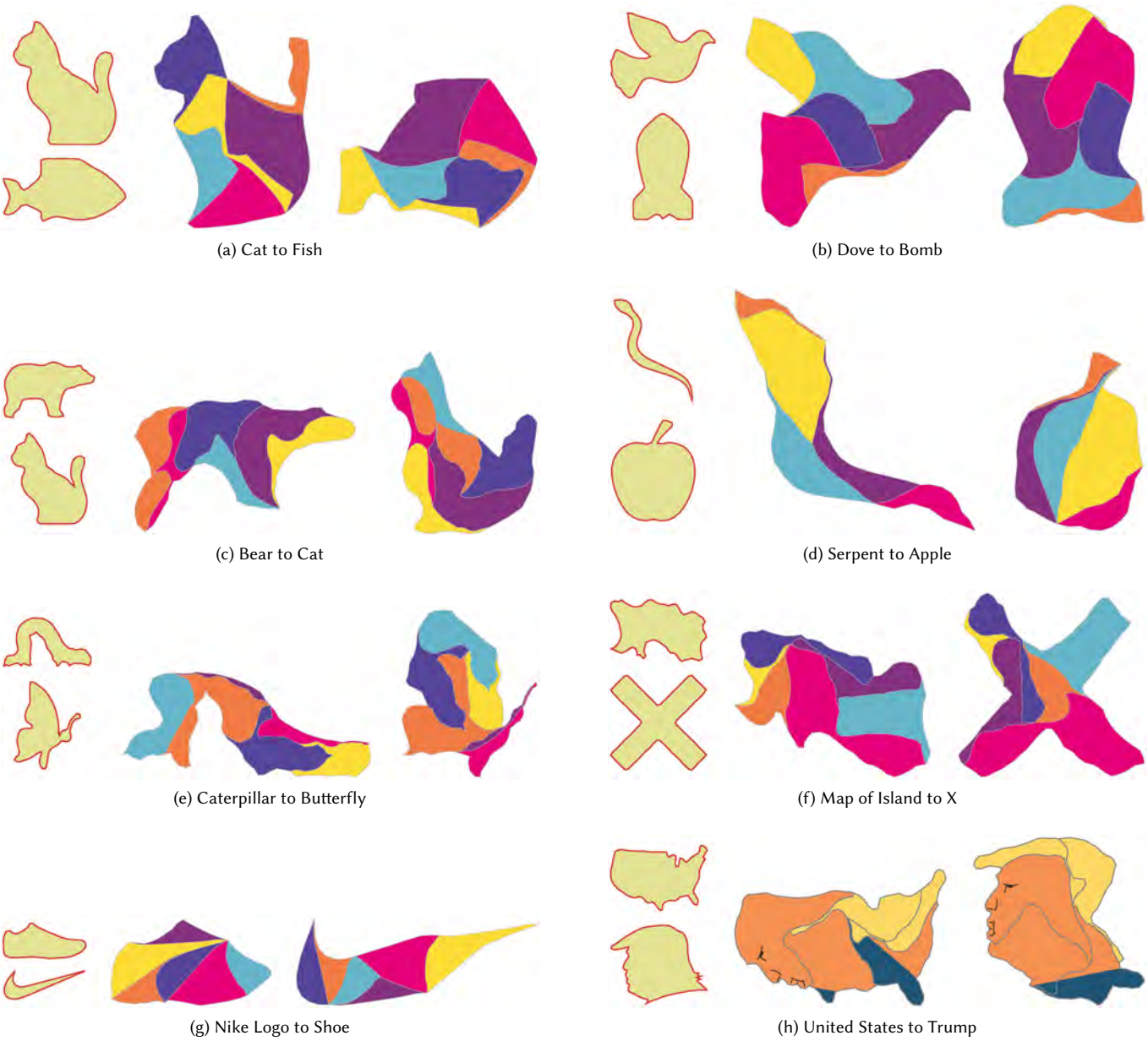
(c) Effect of Pruning For 5-Piece Dissections

Fig. 19. (a) Decline in approximation error as the number of pieces used is increased. (b) The best solutions for the dog-bone dissection obtained with four and five pieces. (c) Performance gains obtained from the pruning tests.

8 EXPERIMENTS AND RESULTS

8.1 Performance

We implemented our method in C++ and tested it on a 2.6 GHz laptop. Due to the exponential nature of the problem, our solution search takes a substantial amount of time. However, the search procedure described in Section 6.3 is embarrassingly parallelizable, and we would expect our runtimes to decrease substantially with the addition of more processors.



(a) Cat to Fish

(b) Dove to Bomb

(c) Bear to Cat

(d) Serpent to Apple

(e) Caterpillar to Butterfly

(f) Map of Island to X

(g) Nike Logo to Shoe

(h) United States to Trump

Fig. 20. Generating dissections between different shapes. Input shapes are shown on the left. Assembled pieces are shown on the right.

	Pieces	J. Constraints	Runtime
Dog-Bone	6	8	438 min
Cat-Fish	6	8	410 min
Dove-Bomb	6	8	368 min
Bunny-Egg	6	9	583 min
Serpent-Apple	5	8	96 min

Table 1. Performance on several inputs. For each test, we list the number of pieces used, the number of connection constraints, and the total runtime.

Table 1 summarizes the performance of our method. We note that it is not unusual to see similar performance characteristics when solving geometric problems in a vast combinatorial search space. For example, recent work by Kwan et al. [2016] that tackled the 2D collage problem had comparable timing, taking about 12 hours to generate their most complex result.

(a) Arrangement Outlines (b) Dog Arrangement (c) Bone Arrangement (d) Incorrect Arrangement

Fig. 21. The fabricated dog bone puzzle. (a) Outlines of the arranged pieces that were shown to the participant.

8.2 Approximation Accuracy

We tested how increasing the piece count improves the accuracy of approximating the input shapes. Fig. 19(a) shows the results across four dissections. We used the maximum angular error from our optimization (1) to measure the approximation accuracy. For the three piece counts tested, we observed a roughly linear relationship.

8.3 Pruning Efficiency Gain

We measured the extent to which the pruning tests (Section 6) improve the performance of the solution search. Fig. 19(b) compares the runtimes for pruning versus no pruning, for four dissections, using a piece count of five. At this piece count, the pruning improves the performance by a factor of roughly 15. For six pieces, all the trials run without pruning did not finish after 24 hours.

8.4 Results

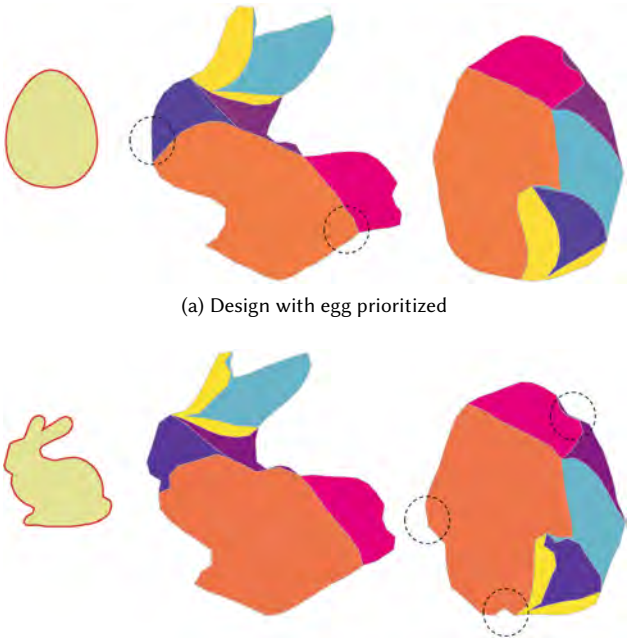
We used our approach to generate several dissections, which are shown in Fig. 20. All the input shapes were obtained by taking one of the first results from a Google image search.

Our method supports organic objects (Fig. 20(a)–(e)), man-made objects (Fig. 20(b), (g)), the outlines of geographic entities (Fig. 20(f), (h)), and abstract shapes (Fig. 20(f), (g)). In most cases, six pieces were needed to form a satisfactory approximation of the input shapes. The two exceptions were the serpent-apple and Trump-USA dissections, which used only five pieces. The Trump-USA result (Fig. 20(h)) shows how simple texturing can enhance a dissection's appearance.

8.5 Application to Puzzles

Our dissections can create a type of puzzle as shown in Fig. 21. We fabricated several of our results and conducted an informal user study in which participants were shown outlines of the two shapes and instructed to assemble the pieces into those shapes. We observed that the puzzles are substantially, albeit not overwhelmingly, difficult. On average, it took a participant about twenty minutes to solve the puzzles for both shapes. The pieces can form coherent shapes that are not one of the intended ones (Fig. 21(d)), which increases the difficulty of the puzzle.

We can introduce a post-processing step that partitions the original set of dissection pieces into smaller ones, allowing the user to tune the difficulty of the puzzle. The partition could aim for high similarity between the pieces, as in traditional jigsaw puzzles.



(a) Design with egg prioritized

(b) Design with bunny prioritized

Fig. 22. A failure. The input shapes are shown in gold. Even with user interaction, problematic regions (circled) in both shapes cannot be eliminated in a single design.

8.6 Limitations

In some cases our approach fails to generate a satisfactory dissection. Fig. 22 shows an example. Even with manual editing, the constraints of the design preclude capturing the details of the Stanford bunny without causing unsightly bulges and cavities on the egg. In such cases, the trade-off between the two options is left to the user.

Our method does not consider the physical properties of the pieces, so it can generate pieces with extremely narrow sections that would be weak if fabricated. In our current implementation,

(a) (b) (c)

Fig. 23. The application of our method to a horror movie scene. (a) A plate on a countertop. (b) The plate begins to vibrate and splits into pieces. (c) The pieces rearrange themselves to form a devilish visage.

the user may manually edit the solution to remove these sections. A term penalizing such sections may be added to the optimization (1).

We also ignore the semantic significance of the pieces. For example, in the cat-fish dissection (Fig. 20(a)), one of the pieces corresponds almost exactly to the cat's head. These cases may be undesirable when designing a puzzle as they reduce its difficulty.

Finally, allowing pieces to flip between arrangements, in addition to rotating, would enable us to generate a wider range of solutions.

9 SUMMARY AND FUTURE WORK

We have introduced the approximate dissection problem, a relaxation of the exact dissection problem, which allows us to capture the essence of a pair of complex shapes using a small number of pieces. We developed a combinatorial search strategy for the problem that prunes the solution space to identify high-quality dissection solutions in a tractable amount of time.

Since our geometric approach cannot take into account the perceptual significance of a given deformation, we developed a novel user interface that allows a casual user to refine the dissection solution. Our interface suggests alterations to the input shapes that would make the dissection generation easier, while allowing the user to add additional alterations that preserve the identity of the input shape.

Our system enables the creation of dissections that are qualitatively different than previous ones. Dissections between complex, naturalistic shapes possess a visceral quality that is lacking in dissections between abstracted or geometrically ideal shapes. These dissections can be used to convey a message in a unique fashion (Fig. 23) or as a puzzle.

Our approach offers several opportunities for future work. First, we could generalize it to support dissections between more than two shapes. Second, the search time for the solution is substantial, even with the pruning tests. Since we currently use a commodity solver [Wächter and Biegler 2006], we can potentially boost performance by developing a special procedure for the dissection geometry optimization (Section 4). Developing additional ways of pruning partial solutions could provide even bigger gains. Finally, we could tackle the approximate dissection problem in three dimensions. Our approach does not naturally generalize to 3D, because we represent dissection pieces as a ring of one-dimensional boundary intervals and do not take piece interlocking into account. Hence, a substantially different method would be needed.

ACKNOWLEDGMENTS

We thank Michael S. Brown for narrating the video and Quang Trung Truong for assistance in fabricating the results. Yu was supported in part by National Science Foundation award number 1565978 and by the Joseph P. Healey Research Grant Program provided by the Office of the Vice Provost for Research and Strategic Initiatives and by the Dean of Graduate Studies of UMass Boston. Yeung was supported in part by Singapore MOE Academic Research Fund MOE2016-T2-2-154 and by a grant from the National Heritage Board of Singapore. Duncan was supported in part by the SUTD Digital Manufacturing and Design (DManD) Centre, which is supported by the National Research Foundation (NRF) of Singapore. This research was also supported by the NRF under its IDM Futures Funding Initiative and Virtual Singapore Award No. NRF2015VSG-AA3DCM001-014. We are grateful to the anonymous reviewers for their constructive comments.

REFERENCES

J. Bosboom, E.D. Demaine, M.L. Demaine, J. Lynch, P. Manurangsi, M. Rudoy, and A. Yodpinyanee. 2015. k-Piece dissection is NP-hard. In *Abstracts from the 18th Japan Conf. on Discrete and Computational Geometry and Graphs*. 2.

M.J. Cohn. 1975. Economical triangle-square dissection. *Geometriae Dedicata* 3, 4 (1975), 447–467.

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

G.N. Frederickson. 2003. *Dissections: Plane and Fancy.* Cambridge University Press.

R.J. Gardner. 1985. A problem of Sallee on equidecomposable convex bodies. *Proc. American Mathematical Society* 94, 2 (1985), 329–332.

Gurobi Optimization, Inc. 2016. Gurobi Optimizer Reference Manual. (2016). http://www.gurobi.com

Y.-J. Huang, S.-Y. Chan, W.-C. Lin, and S.-Y. Chuang. 2016. Making and animating transformable 3D models. *Computers & Graphics* 54 (2016), 127–134.

C.S. Kaplan and D.H. Salesin. 2000. Escherization. In *Proc. ACM SIGGRAPH '00 Conf.* 499–510.

E. Kranakis, D. Krizanc, and J. Urrutia. 2000. Efficient regular polygon dissections. *Geometriae Dedicata* 80, 1 (2000), 247–262.

K.C. Kwan, L.T. Sinn, C. Han, T.-T. Wong, and C.-W. Fu. 2016. Pyramid of arclength descriptor for generating collage of shapes. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 229.

M. Löffler, M. Kaiser, T. van Kapel, G. Klappe, M. van Kreveld, and F. Staals. 2014. The Connect-The-Dots family of puzzles: Design and automatic generation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 72.

P. Manurangsi, M. Rudoy, and A. Yodpinyanee. 2016. Dissection with the fewest pieces is hard, even to approximate. In *Discrete and Computational Geometry and Graphs: 18th Japan Conf. (JCDCGG 2015)*, Vol. 9943. 37.

P. Song, C.-W. Fu, Y. Jin, H. Xu, L. Liu, P.-A. Heng, and D. Cohen-or. 2017. Reconfigurable interlocking furniture. *ACM Transactions on Graphics (TOG)* 36, 6, Article 174 (2017), 14 pages.

A. Wächter and L.T. Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming* 106, 1 (2006), 25–57.

J. Xu and C.S. Kaplan. 2007. Image-guided maze construction. *ACM Transactions on Graphics (TOG)* 26, 3, Article 29 (2007), 9 pages.

X. Xu, L. Zhang, and T.-T. Wong. 2010. Structure-based ASCII art. *ACM Transactions on Graphics (TOG)* 29, 4, Article 52 (2010), 9 pages.

Y. Zhou, S. Sueda, W. Matusik, and A. Shamir. 2014. Boxelization: Folding 3D objects into boxes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 71.

Y. Zhou, R. Wang, et al. 2012. An algorithm for creating geometric dissection puzzles. In *Proc. Bridges Conf.* 49–58.

C. Zou, J. Cao, W. Ranaweera, I. Alhashim, P. Tan, A. Sheffer, and H. Zhang. 2016. Legible compact calligrams. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 122.