Automatic Solution of Jigsaw Puzzles

Daniel J. Hoff[1]
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
d1hoff@math.ucsd.edu

Peter J. Olver[1]
School of Mathematics
University of Minnesota
Minneapolis, MN 55455
olver@math.umn.edu
http://www.math.umn.edu/∼olver

Abstract

We present a method for automatically solving apictorial jigsaw puzzles that is based on an extension of the method of differential invariant signatures. Our algorithms are designed to solve challenging puzzles, without having to impose any restrictive assumptions on the shape of the puzzle, the shapes of the individual pieces, or their intrinsic arrangement. As a demonstration, the method was successfully used to solve two commercially available puzzles. Finally we perform some preliminary investigations into scalability of the algorithm for even larger puzzles.

Keywords:
jigsaw puzzle, curvature, Euclidean signature, bivertex arc, piece fitting, piece locking

1 Introduction.

In this paper, we present a new algorithm for the automatic solution of apictorial jigsaw puzzles, meaning that there is no design or picture and the solution requires matching only the shapes of the individual pieces, cf. [10]. Our method is founded on the extended Euclidean signature method for object recognition and curve matching that we developed in [16]. We illustrate its efficacy by automatically solving two commercially available jigsaw puzzles: The Rain Forest Giant Floor Puzzle, [23], has fairly standard shaped pieces and is relatively easy to solve by hand, especially if one also uses the puzzle picture to guide in placement of the pieces. The Baffler Nonagon, [33], is considerably more challenging, as it is truly apictorial, with very irregularly shaped pieces, each of a distinct textured color.

The initial step in our procedure is to accurately photograph the puzzle pieces, in random orientations, which are then presented to the computer in the form of JPEG digital images. Following segmentation and smoothing of the boundary curves of each piece, the algorithm applies invariant numerical algorithms, [1, 4] to calculate the two simplest Euclidean differential invariants — the curvature and its derivative with respect to arc length — that are used to parametrize the Euclidean signature curve. A fundamental theorem, [4, 20], states that two sufficiently regular plane curves are equivalent under a rigid motion if and only if they

[1] *Supported in part by* NSF *Grant* DMS 08–07317.

have identical Euclidean signatures. An important feature is that, unlike, say, characterizing curves via curvature as a function of arc length, [15], such differential invariant signatures are fully local, and hence can be readily adapted both to curves under occlusion, and to puzzle pieces where one only matches a part of the boundary curves. The extension developed in [16] breaks up the complete signatures into subarcs, which corresponds to a decomposition of the original curves into "bivertex arcs" (see Section 2 for definitions). Individual bivertex arcs with the same signature are then matched by rigid motions; if these all agree, the curves are rigidly equivalent.

A key feature is that our algorithms rely solely on the shapes of the puzzle pieces, and not on any picture or design which may appear thereon. (At the opposite end of the spectrum are algorithms that deal solely with image information, on puzzles with all square pieces, [12].) It is worth emphasizing that our method is founded upon the characterization of the (approximate) bivertex arcs of the puzzle boundaries, which in turn are characterized through the two curvature invariants used to construct the differential invariant signature. With the bivertex arc signatures already known, we can efficiently compare them to determine potential fits between puzzle pieces, which are then refined using a new method we call "piece locking". With some tuning of the parameters used in the various components, the resulting method is remarkably accurate and able to automatically solve large scale, challenging, commercial puzzles.

While of limited practical use, at least outside the entertainment world, puzzle assembly has been studied with a number of more important applications in mind. In [18, 24], for instance, puzzle solution techniques are applied to broken tiles to simulate the reconstruction of archaeological artifacts. In fall, 2011, DARPA held a competition, with a $50,000 prize, to automatically reconstruct a collection of shredded documents, [8]. Recreational solving of jigsaw puzzles belongs to the class of problems for which humans have a natural aptitude but automation remains considerably more challenging. This is especially true of puzzles that combine to form a picture, in which case human solution is typically more a matter of patience than mental exertion. Because of this natural motivation, much previous work, e.g., [13, 30, 32], has focused on solving archetypical jigsaw puzzles, whose overall form is constrained by several rather restrictive assumptions, the most common being:

(1) The individual pieces have four well-defined sides, each of which contains either an "indent" or an "outdent".

(2) Each piece has at most four primary neighbors, one on each side (except, of course, for pieces on the puzzle boundary) that are fitted together via the "indents" and "outdents".

(3) The solved puzzle has pieces positioned on an (approximate) grid.

(4) The boundary of the solved puzzle is an easily identified smooth shape, e.g., a rectangle.

For example, the algorithm proposed in [34] employs bitangents and distinguished points to match simple "indents" and "outdents", and relies heavily on recognizing the boundary and corner pieces to start the assembly process. Using all four assumptions, two intermixed 104-piece puzzles were solved in [30]. A more recent work, [13], solves a 204-piece puzzle, where adjacent pieces are matched by comparing ellipses fitted to the "indents" and "outdents". However, the algorithms developed in [13, 30, 34] will not extend to more challenging

situations such as the Baffler Nonagon puzzle, shredded documents, or broken ceramics reconstruction, where none of these simplifying assumptions hold.

Our goal is to develop a method of puzzle assembly that does not require *any* of assumptions (1–4), and therefore can be readily extended beyond the realm of standard jigsaw puzzles. We do impose a mild restriction that the puzzle pieces have smooth boundary curves, of class at least C^3, that are also "v-regular", in the terminology of [16]. The latter technical assumption is defined below, and, being purely mathematical, is automatically satisfied in practical applications. One might, however, justifiably question our smoothness assumption, as many physical puzzle pieces, as well as pieces of broken pottery and tiles, have corners. Despite this limitation, in practice we are able to successfully deal with puzzle pieces with corners by applying a preliminary curve smoothing procedure that slightly rounds them off, and this has sufficed in all the examples we have tested the algorithm on. Indeed, when the images of the puzzles pieces are coarsely digitized, a preliminary smoothing step is essential for accurate computation of the required Euclidean signatures. Competing general algorithms can be found in [10], which focusses on the types of "junctions", where three or more pieces touch, [22], which bases curve matching on polar coordinate systems centered around vertices of their boundaries, i.e., local extrema of the curvature, and [18], which uses dynamic programming methods to match the curvature and arc length invariants of pairs of pieces, and then refines the result by matching piece triples.

Our approach to fitting puzzle pieces together is based on two principal tools. First, we note that the problem of matching individual piece boundaries is closely related to the recognition of planar objects under rigid motions. Based on Élie Cartan's solution to the equivalence problem for submanifolds under general Lie group actions, cf. [20], the use of differential invariant signature curves for planar object recognition was promoted in [4], and then extended to cover more general cases in [16]. The extended invariant signature method naturally lends itself to puzzle solving since it decomposes boundary curves into *bivertex arcs*, as defined below, thereby readily allowing one to compare parts of piece boundaries. In Subsection 5.1, we adapt the ideas of [16] to formulate an algorithm for finding possible piece-to-piece fits. An important point is that the piece fitting algorithm concentrates exclusively on the bivertex arcs, and hence ignores any straight line segments and circular arcs appearing in the piece boundaries. In particular, this approach frees our algorithm from reliance on the existence of a rectangular or other prescribed boundary of the entire puzzle; indeed, the boundary pieces tend to be among the last to be placed during the assembly process.

The second tool was developed after we recognized a need to increase the quality and confidence with which pieces are placed. This is desirable not only because it maximizes the overall solution confidence, but also because it minimizes the need for back-tracking and branching, allowing for larger puzzles to be successfully solved. When humans correctly place a jigsaw puzzle piece, they are rewarded with a noticeable feeling of it "snapping" into place. Moreover, it is shown in [3] that for robotic puzzle assembly, monitoring the feedback force can be a useful simulation of this sensation. In Subsection 5.2, we develop a method of *piece locking*, which is designed as a computational analogue of this feeling and based on the magnitude of a fictitious electrostatic force and torque between matching piece boundaries.

In the penultimate section, we apply our method to two commercially available puzzles, [23, 33]. Accurate photos of the individual puzzle pieces were segmented using a standard snake algorithm, [6, 7, 17]. The resulting contours were then subjected to a preliminary

smoothing using a naïve spline algorithm. After the preprocessing step, the piece fitting and locking algorithms were run until completion. Unlike most previous puzzle solving algorithms, ours work from the "inside out"; in other words, the initially selected piece tends to be located inside the puzzle, not on its boundary, and the subsequent pieces are successively fitted together, building the puzzle up from its interior, and avoiding any need to identify boundary pieces as such. With appropriate choices of adjustable parameters, effectively balancing processing speed versus accuracy of the piece placements, the Rainforest and Baffler puzzles were successfully solved in, respectively, 58 and 31 minutes on an Intel® Core™ 2 Duo E8500 3.17GHz processor.

The final section presents some preliminary investigations into scalability of our algorithm to even larger puzzles. The Cathedral jigsaw puzzle [29], is a non-standard 1000 piece puzzle, yet its individual pieces contain many standard "indents" and "outdents." This combination makes it particularly challenging, since there are many "local" near-matches, and yet simplified algorithms designed for standard puzzles cannot be applied because of the unusual structure. While we have not, as yet, achieved complete success in this much more challenging situation, the algorithm did successfully assemble 103 pieces of a 111 piece subpuzzle before terminating, in approximately 36 hours. If we had a similar sized puzzle with more irregular pieces, as in the Baffler, we would expect much greater success. Thus, while further work remains to be done, our conclusion is that the algorithm does lend itself to applicability to much larger puzzles and is eminently scalable.

2 Curves and Puzzles.

Let us introduce our basic terminology and assumptions, based on that used in [15, 16]. We will be working exclusively with plane puzzles. (Extending our methods to three-dimensional puzzles, e.g., a broken statue, would be an interesting challenge. While the mathematical machinery is in place, [20], their practical implementation remains daunting.) We will work in the Euclidean plane with the standard norm, denoted $\| z \| = \sqrt{x^2 + y^2}$ for $z = (x, y) \in \mathbb{R}^2$. We let SE(2) denote the three-dimensional *special Euclidean group* consisting of all orientation-preserving isometries, or *rigid motions*:

$$z \longmapsto R\,z + c, \quad z = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2, \quad \text{where} \quad R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \quad c = \begin{pmatrix} a \\ b \end{pmatrix}, \tag{2.1}$$

are, respectively, a 2×2 matrix representing rotation through angle θ and a translation vector. We will consistently identify planar objects up to rigid motion. All curves $C \subset \mathbb{R}^2$ are assumed to be compact, rectifiable — and hence of finite length — and simple, i.e., without self-intersections. A closed curve is called a *Jordan curve*, while a non-closed curve, with distinct endpoints, will be called an *arc*. Two arcs are said to be *non-overlapping* if they have at most one or both endpoints in common.

By definition, a *puzzle piece* is a bounded, simply connected plane domain $\mathbb{P} \subset \mathbb{R}^2$ whose boundary $C = \partial\mathbb{P}$ is a *Jordan curve* of class C^3. Two puzzle pieces $\mathbb{P}, \widetilde{\mathbb{P}}$ are *congruent*, and hence considered to be the same, if they are *rigidly equivalent*, meaning that there is a rigid motion $g \in \mathrm{SE}(2)$ such that $\widetilde{\mathbb{P}} = g \cdot \mathbb{P}$. Two puzzle pieces $\mathbb{P}, \widetilde{\mathbb{P}}$ are said to *fit* together if they share a common arc $S = \mathbb{P} \cap \widetilde{\mathbb{P}} = \partial\mathbb{P} \cap \partial\widetilde{\mathbb{P}}$, or, more generally, if they do so after a rigid

motion: $S = \widetilde{\mathbb{P}} \cap (g \cdot \mathbb{P}) = \partial\widetilde{\mathbb{P}} \cap (g \cdot \partial\mathbb{P})$ for some $g \in \mathrm{SE}(2)$. Keep in mind that pieces that fit do not overlap. In practice, the shared arc should not be too short, although this requirement need not be directly quantified as it will follow as a consequence of our fitting and locking algorithms. We can also allow two puzzle pieces to fit along more than one connected arc, although this is uncommon in real world puzzles. It is also not difficult to allow non-simply connected puzzle pieces, again rare in examples.

Remark: The key assumption used in this work is that the puzzle pieces are smooth (of class C^3), and hence do not have corners. Clearly, many physical puzzles contain pieces with corners, and it would be worth directly adapting our algorithms to cover such pieces. Such an extension will not be difficult; indeed, during the course of the assembly process, we deal with subpuzzles, that is, unions of puzzle pieces, whose boundary is only piecewise C^3 (and not necessarily simply connected). However, this adaption has proven to be unnecessary for the practical solution of all the puzzles we have tried, since the digital images of the puzzle pieces must be subjected to a preliminary smoothing operation anyways before the assembly algorithm can proceed.

By an *apictorial puzzle*, or *puzzle* for short, we mean a bounded plane domain $\mathcal{P} \subset \mathbb{R}^2$ that is the union,

$$\mathcal{P} = \bigcup_{i=1}^{n} (g_i \cdot \mathbb{P}_i), \tag{2.2}$$

of individual puzzle pieces $\mathbb{P}_i$ subject to rigid motions $g_i \in \mathrm{SE}(2)$, which we sometimes refer to as *placements*. We suppose that any two pieces, when placed, are either disjoint, $(g_i \cdot \mathbb{P}_i) \cap (g_j \cdot \mathbb{P}_j) = \varnothing$, or touch at a single point[1] $\{ z_0 \} = (g_i \cdot \mathbb{P}_i) \cap (g_j \cdot \mathbb{P}_j)$, or fit together according to the preceding definition. However, while standard jigsaw puzzles satisfy this, our algorithms don't actually require this assumption to hold, and so could, in principle, solve "multi-layered puzzles", whose pieces are allowed to overlap. (While this strikes us as an intriguing extension of standard jigsaw puzzles, we are not aware of any actual examples.) In the puzzle assembly problem, we are given the pieces $\mathcal{P}^* = \{\mathbb{P}_1, \dots, \mathbb{P}_n\}$, and our task is to determine the corresponding rigid motions $g_1, \dots, g_n$ required to assemble $\mathcal{P}$. Our methods work best when pieces that fit together do so uniquely, and only fit in accordance with their positions in the final puzzle assembly. However, more sophisticated backtracking techniques could be developed to better deal with non-uniqueness of fits (e.g., as in [18]).

3 Equivalence of Curves.

Our algorithm for fitting puzzle pieces together is based on the solution to the equivalence problem for plane curves based on extended Euclidean-invariant signatures developed in [16]. Let us review the key points, leaving the complete details to the aforementioned reference.

Given a C^3 Jordan curve $C \subset \mathbb{R}^2$, let $z(s)$, $0 \leq s \leq L$, denote its arc length parametrization, so that L is the length of C and $z(s)$ is periodic of period L. Let $\kappa(s)$ denote the signed *curvature* at the point $z(s) \in C$, and $\kappa_s(s) = d\kappa/ds$ its derivative with respect to arc

[1]More generally, one can allow pieces to touch at a finite number of points, although this is uncommon in real world examples.

length. Note that both κ and κ_s are Euclidean *differential invariants*, meaning that they are unaffected by rigid motions, [20].

A point $z(s) \in C$ is called *regular* if $\kappa_s(s) \neq 0$. An *ordinary vertex* is a local extremum of curvature, [15]. We define a *generalized vertex* to be maximal connected arc $V \subset C$ on which $\kappa_s(s) \equiv 0$. Thus, a generalized vertex is either an ordinary vertex, or a critical point of curvature, or a circular arc, or a straight line segment. In this paper, all curves are assumed to be *v-regular*, [16], meaning that they have only finitely many generalized vertices. Curves with infinitely many vertices exist in theory, but are pathological and cannot arise in real world applications.

By a *bivertex arc*, we mean an arc $B \subset C$ that has $\kappa_s = 0$ at both endpoints, but all of whose interior points are regular. A basic result of [16] states any v-regular C^3 Jordan curve that is not a circle has a unique *bivertex decomposition*,

$$C = \bigcup_{j=1}^{m} B_j \cup \bigcup_{k=1}^{n} V_k, \tag{3.1}$$

into a finite union of $m \geq 4$ non-overlapping bivertex arcs $B_1, \ldots, B_m$, and $n \geq 0$ generalized vertices $V_1, \ldots, V_n$. Note that we exclude point vertices from the bivertex decomposition, since they are accounted for by the endpoints of the bivertex arcs B_j.

Two plane curves $C, \widetilde{C} \in \mathbb{R}^2$ are said to be *rigidly equivalent*, or *congruent* for short, if there exists a rigid motion $g \in \mathrm{SE}(2)$ such that $\widetilde{C} = g \cdot C$. We extend the notion of congruence to disconnected unions of curves, keeping in mind that for two unions to be congruent, their constituent curves must be pair-wise congruent *under the same rigid motion*. The following result was established in [16].

Proposition 1. *Two v-regular plane curves $C, \widetilde{C} \subset \mathbb{R}^2$ are congruent if and only if the unions of their constituent bivertex arcs are congruent.*

Using a reformulation of Cartan's general solution to the local equivalence problem of submanifolds under Lie group actions, [5], a solution to the equivalence problem for plane curves based on their Euclidean signature was proposed in [4], and subsequently applied to object recognition and symmetry detection in a variety of contexts, [25, 26].

Definition 2. Let C be a plane curve of class C^3 and of length $L < \infty$. The *Euclidean signature* of C is the (non-simple) plane curve $S(C) = \{ (\kappa(s), \kappa_s(s)) \,|\, 0 \leq s \leq L \}$ parametrized by the curvature and its derivative with respect to arc-length.

The following theorem is a consequence of Proposition 1, combined with general results on group-invariant signatures[2] of regular submanifolds, [20].

Theorem 3. *Let $C, \widetilde{C} \subset \mathbb{R}^2$ be v-regular, non-circular Jordan curves whose bivertex decompositions contain the same number, n, of non-overlapping bivertex arcs. Assume that, for each $j = 1, \ldots, n$, the bivertex arcs B_j and $\widetilde{B}_j$ have identical signatures: $S(B_j) = S(\widetilde{B}_j)$, which implies that $B_j, \widetilde{B}_j$ are congruent, and hence there exist rigid motions $g_j \in \mathrm{SE}(2)$ such that $\widetilde{B}_j = g_j \cdot B_j$. If, in addition, all the $g_j = g$ are the same, then the entire curves are rigidly equivalent: $\widetilde{C} = g \cdot C$.*

[2]The older term "classifying submanifold" is used in place of "signature" in [20].

In practical applications, the puzzle pieces are inputted to the computer as digital images and then segmented to retrieve their boundaries. (See Section 6 for practical details.) As a result, each piece is represented by a discrete set of *sample points* $C^\Delta = \{z^1, \dots, z^N\}$, where each z^j lies on or near its boundary curve C. The actual curve can be approximately reconstructed from C^Δ by some form of interpolation, e.g., periodic splines, possibly coupled with smoothing to reduce the effect of noise. We similarly (approximately) discretize the signature curve $S(C)$ by $S^\Delta = (\sigma^1, \dots, \sigma^N)$, where each point $\sigma^j = (\kappa^j, \kappa_s^j) \in S^\Delta$ consists of suitable numerical approximations to the curvature and its arc length derivative at the corresponding sample point $z^j = (x^j, y^j) \in C^\Delta$. For example, the entries of σ^j may be found directly from the discretized curve C^Δ by employing the Euclidean-invariant numerical approximations to the curvature invariants developed in [1, 4].

According to the algorithm developed in [16], to determine if two discretized curves are rigidly equivalent, the first step is to construct their (approximate) bivertex decompositions by splitting each curve into subarcs whenever $|\,\kappa_s\,| - \delta_0$ changes sign, where $\delta_0 > 0$ is a fixed small cut-off. In order to achieve more consistent approximate Bivertex Arc Decompositions, we additionally split curves wherever $|\kappa_s|$ achieves a local minimum, provided that minimum exceeds the parameter δ_0. This added convention helps to reduce sensitivity to the value of δ_0. Additionally, instead of eliminating insignificant arcs over which curvature changes by less than δ_1 (as in [16]), we eliminate arcs on which the maximum of $|\kappa|$ is less than δ_1. Details on the selection of δ_0 and δ_1 are presented in [16], though we note that in this paper the quantity $D_\kappa(C, \widetilde{C})$ therein is taken as a maximum over all inputed pieces (rather than just two), and is denoted d_κ.

The next step is to compare individual (discrete) bivertex arcs using their (discrete) Euclidean signatures. The method, developed in detail in [16], is based, roughly, on the idea of regarding the discrete signatures as two collections of oppositely charged points, and determining the mutual electrostatic attraction, [31]. (Or, equivalently, their mutual gravitational attraction as point masses.) After some manipulation, we determine a *signature similarity coefficient* $p(B, \widetilde{B}) \in [0, 1]$ that serves to measure the closeness of two individual bivertex arc signatures, where a score of 1 reflects identical signatures, while $p(B, \widetilde{B})$ decreases to 0 as the signatures become increasingly disparate, and hence the arcs less and less likely to be congruent. To compare the two curves, we first construct approximate bivertex decompositions, and then ensure that they contain the same number of arcs; if not, there is a good chance the curves are not rigidly equivalent, but this may be due to noise, even with our selection of the cut-off parameter δ_0. In this case, the procedure for eliminating arcs from the larger collection is to delete those on which the curvature changes least. Sometimes, several possible deletions are tested. We then test the pairwise congruence of the resulting two collections of bivertex arcs, ordered as one traverses the curves with the same orientation. If they are congruent, one then checks whether the corresponding rigid motions are (approximately) identical, and if so the curves are deemed to be congruent.

4 Puzzle Assembly.

We now present the steps used in our puzzle solving algorithm. The method relies on successively fitting individual pieces by matching bivertex arcs along their boundaries and then

improving the fits via piece locking. The details of piece fitting and locking will appear in the following section.

We begin with the collection of puzzle pieces, denoted $\mathcal{P}^* = \{\mathbb{P}_1, \dots, \mathbb{P}_n\}$, whose boundaries $C_j = \partial\mathbb{P}_j$ are supplied as suitably segmented and smoothed (discrete) plane curves. The puzzle is solved by constructing a corresponding collection of rigid motions $\mathcal{G}^* = \{g_1, \dots, g_n\} \subset \mathrm{SE}(2)$ that form the completed puzzle (2.2). Without loss of generality, one of these can be fixed as the identity transformation: $g_{i_1} = e$.

At each step $k = 1, 2, \dots$, in the algorithm, we will have already assembled a *subpuzzle* $\mathcal{Q}_k \subset \mathcal{P}$ consisting of k pieces $\mathcal{Q}_k^* = \{\mathbb{P}_{i_1}, \dots, \mathbb{P}_{i_k}\} \subset \mathcal{P}^*$, along with the corresponding rigid motions $\mathcal{H}_k^* = \{g_{i_1}, \dots, g_{i_k}\}$ required to assemble it:

$$\mathcal{Q}_k = \bigcup_{\nu=1}^{k} (g_{i_\nu} \cdot \mathbb{P}_{i_\nu}) \subset \mathcal{P}. \tag{4.1}$$

Let $\mathcal{R}_k^* = \mathcal{P}^* \setminus \mathcal{Q}_k^*$ denote the set of pieces that remain to be assembled. The algorithm terminates with either a fully assembled puzzle, with $\mathcal{Q}_n = \mathcal{P}$, or, for some $k < n$, a subpuzzle $\mathcal{Q}_k \subsetneq \mathcal{P}$, for which the program is unable to fit any of the pieces remaining in $\mathcal{R}_k^*$.

For each $1 \le j \le n$, we let $\mathcal{B}_j^* = \{B_1, \dots, B_{m_j}\}$ be the set of bivertex arcs (or, more accurately, discrete approximate bivertex arcs) associated with the puzzle piece $\mathbb{P}_j$. It will be important to order the arcs in $\mathcal{B}_j^*$ consecutively as the boundary C_j is traversed in a counterclockwise manner. During the assembly process, any bivertex arcs that have already been fit to neighboring pieces, after successful piece locking, are deemed to be *inactive*, and so not available during the subsequent fitting process. We let

$$\mathcal{V}_k^* \subset \bigcup_{\nu=1}^{k} \mathcal{B}_{i_\nu}^*$$

denote the set of *active* bivertex arcs for the subpuzzle $\mathcal{Q}_k$. In principle, $\mathcal{V}_k^*$ should only contain the bivertex arcs situated on the outer boundary of $\mathcal{Q}_k$, although noise or other artifacts might prevent some interior bivertex arcs from being matched and hence remaining in $\mathcal{V}_k^*$ by "accident". Fortunately, this does not seriously affect the performance of our algorithm in practical situations.

It should be noted that for the purposes of pairing bivertex arcs, each assembled piece retains its individuality, and bivertex arcs on a piece boundary remain consecutive whether or not they are currently active. One evident limitation is that the program doesn't consider combinations of, say, two bivertex arcs from one assembled piece and three from an adjacent assembled piece, fitting five consecutive arcs on a not yet placed piece. On the other hand, the algorithm does use this extra information during piece locking, when the assembled subpuzzle is treated as a single piece. While developing a way to universally treat subpuzzles as single pieces could improve performance, it seemed unnecessary for many practical situations, including the puzzles treated here.

The first step in the assembly algorithm is to select a piece to form the initial subpuzzle $\mathcal{Q}_1^* = \{\mathbb{P}_{i_1}\}$, and, by default, set $g_{i_1} = e$. The choice of starting piece is not so important. Our convention is to choose the piece that maximizes the total curvature $\sum |\, \kappa_i \,|$, summed over all the points in the piece's discretized signature, because, as argued in [16], arcs of

high curvature tend to "better" determine a simple closed curve. Our aim is to maximize the chances of finding successful fits early on, by ensuring that the starting piece has many well-defined features. Indeed, since straight lines are of minimal curvature — and also not included in the bivertex arcs and not candidates for fitting — this initial selection is also more likely to be an interior piece of the puzzle.

To continue, at each step $k \geq 1$, we construct the next subpuzzle by finding an unattached piece $\mathbb{P}_i \in \mathcal{R}_k^*$ that fits the current subpuzzle $\mathcal{Q}_k$ under a rigid motion $g_i \in \mathrm{SE}(2)$, and then setting $\mathcal{Q}_{k+1}^* = \mathcal{Q}_k^* \cup \{\mathbb{P}_i\}$, $\mathcal{G}_{k+1}^* = \mathcal{G}_k^* \cup \{g_i\}$. We select the piece $\mathbb{P}_i$ through an adaptation of the piece fitting and locking algorithms; details appear in the following section. If no suitable piece can be found, the algorithm terminates. Otherwise, the bivertex arcs that were deemed to be matched in the piece locking stage that attaches $\mathbb{P}_i$ to $\mathcal{Q}_k$ are relabeled as inactive, and hence deleted from the collection $\mathcal{V}_k^* \cup \mathcal{B}_i^*$ in order to form the new set of active bivertex arcs $\mathcal{V}_{k+1}^*$.

Remark: A more sophisticated approach would be to allow several distinct subpuzzles to be assembled concurrently, and then deal with the problem of fitting the subpuzzles together. This proved to be not necessary for the puzzles we tried our algorithm on.

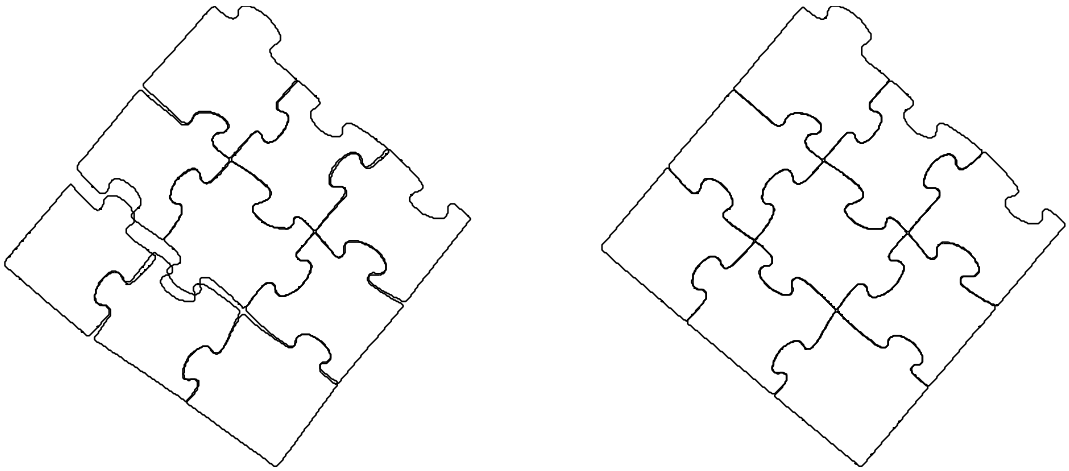
5 Fitting Puzzle Pieces Together.

In this section, we present our algorithms for fitting and locking two individual puzzle pieces. At the end, we explain how to adapt the basic algorithms to fit and lock a piece with a subpuzzle.

As above, given two puzzle pieces $\mathbb{P}, \widetilde{\mathbb{P}}$, by a *fit*, we just mean a rigid motion $g \in \mathrm{SE}(2)$. The *quality* of the fit g measures how well an arc of $g \cdot C$ approximates an arc of $\widetilde{C}$. Finding a good fit is accomplished in two steps. First, we apply the extended Euclidean-invariant signature method developed in [16] to form a stockpile of potential piece-to-piece fits. Fits are ranked according to their $\mu(\mathbb{P}, \widetilde{\mathbb{P}})$ confidence scores. We then attempt to refine the fits using the ensuing method of piece-locking. Once a lock of sufficiently high quality is found, that piece is added to the subpuzzle.

We remark that an algorithm that relies solely on selecting fits with the highest confidence scores, without the extra refinement of piece-locking, is reasonably successful at solving small puzzles. However, the resulting fit transformations tend to be insufficiently accurate, and the method can easily accumulate increasing errors that can hinder its success with larger puzzles. Figure 1(a), for instance, shows such a solution of a nine puzzle piece fragment. While the pieces have been placed with correct adjacency, it is clear from figure 1(b) that the refinement provided by piece locking generates a much better solution.

This may well inspire the reader to ask why not proceed directly to piece locking and avoid the preliminary signature-based fitting procedure? The reason is efficiency: direct piece-locking method is much slower than signature comparison, especially as the curvature invariants comprising the signature were already been computed in order to characterize the required bivertex arcs. Our combination of preliminary fitting followed by the more accurate locking procedure on potential fits appears optimal for both speed and reliability.



(a) Solution by fit finding only

(b) Solution by fit finding and piece locking

Figure 1: Two solutions of a nine piece puzzle fragment. Each has correct adjacency, but (b) improves on (a) through the use of piece locking.

5.1 Fitting.

The problem of fitting two puzzle pieces together is essentially a curve matching problem. It is thereby closely related to the recognition of plane curves under rigid motions, the primary difference being that puzzle pieces match only along subarcs. The extended Euclidean signature of [16] naturally lends itself to puzzle solving since it relies on decomposing the boundary curves into collections of bivertex arcs, and hence provides a useful, semi-local approach to curve matching.

The piece fitting algorithm begins with two puzzle pieces $\mathbb{P}, \widetilde{\mathbb{P}} \subset \mathbb{R}^2$, or, more accurately, their boundaries $C = \partial\mathbb{P}$, $\widetilde{C} = \partial\widetilde{\mathbb{P}}$. (Keep in mind that we work with the discretized boundaries and signatures throughout.) It requires the specification of a positive integer $m_0 > 0$, which governs the minimal number of matched bivertex arcs required for a fit, and a real number $0 < p_0 < 1$, which sets a minimal level required for the identification of two bivertex signatures through their p scores. The output is a *confidence score* $\mu(\mathbb{P}, \widetilde{\mathbb{P}}) \in [0, 1]$ that measures the likelihood of the two pieces fitting. The curve matching algorithm of [16] relies on specification of several parameters, denoted $\epsilon, \gamma, \alpha, \beta, C_1, C_2, \lambda_0, \lambda_1$, and we refer the reader to that paper for details on what they control. We note that for this paper, the characteristic distances D_x and D_y of that paper are computed as a maximum over all inputed pieces, rather than over a single pair of curves.

Piece Fitting Algorithm

(1) The first step is to find approximate bivertex arc decompositions of the two puzzle piece boundaries: $\mathcal{B}^* = \{B_1, \dots, B_m\} \subset C$, $\widetilde{\mathcal{B}}^* = \{B_1, \dots, B_{\tilde{m}}\} \subset \widetilde{C}$. The arcs in $\mathcal{B}^*$ are ordered according to the counterclockwise orientation of C, while those in $\widetilde{\mathcal{B}}^*$ are ordered and oriented using the *opposite*, clockwise orientation of $\widetilde{C}$.

(2) For each pair of bivertex arcs $B \in \mathcal{B}^*$ and $\widetilde{B} \in \widetilde{\mathcal{B}}^*$, compute their signature similarity coefficient $p(B, \widetilde{B}) \in [0,1]$ using the procedure presented in [16].

(3) Find maximal sequences of $m+1 \geq m_0$ consecutive bivertex arcs $\{B_i, \dots, B_{i+m}\} \subset \mathcal{B}^*$ and $\{\widetilde{B}_j, \dots, \widetilde{B}_{j+m}\} \subset \widetilde{\mathcal{B}}^*$ that satisfy $p(B_{i+l}, \widetilde{B}_{j+l}) \geq p_0$ for all $0 \leq l \leq m$.

(4) If no suitable pairs of sequences exist, set $\mu(\mathbb{P}, \widetilde{\mathbb{P}}) = 0$. Otherwise, for each such pair, use the method[3] of [16] to approximate the transformations g_k that takes B_{i+k} to $\widetilde{B}_{j+k}$, and calculate the associated $\mu(\mathbb{P}, \widetilde{\mathbb{P}})$ score. As in (2.1), we represent each rigid motion g_k by its parameters $(\theta_k, a_k, b_k) \in S^1 \times \mathbb{R}^2$. The algorithm then returns the rigid motion $g_{\mathit{fit}} = (\theta_{\mathit{fit}}, a_{\mathit{fit}}, b_{\mathit{fit}})$, whose parameters are obtained by simply averaging[4]

$$\theta_{\mathit{fit}} = \arg\left(\sum_{k=1}^{m} e^{\mathrm{i}\,\theta_k}\right), \qquad a_{\mathit{fit}} = \text{mean}\,\{a_k\}, \qquad b_{\mathit{fit}} = \text{mean}\,\{b_k\}, \tag{5.1}$$

as a possible piece-to-piece fit with confidence score $\mu(\mathbb{P}, \widetilde{\mathbb{P}})$.

5.2 Locking.

Piece locking quantifies the physical sensation a puzzle assembler experiences when two pieces are successfully fit together. Our simulation of this feeling will be based on the generalized electrostatic/gravitational attractive force that was used to compare signatures in [16], and hence underlies our computation of the p scores used in the piece fitting algorithm of the preceding subsection. In the locking phase, we now work directly with the discretized boundary curves and their constituent bivertex arcs, rather than their signatures. We view the curves C and $\widetilde{C}$ as two oppositely charged wires, or, more correctly since we use their discretizations, as two oppositely charged collections of particles. Fixing $\nu \geq 0$ and $0 < \epsilon \ll 1$, the function

$$f(z, \tilde{z}) = \frac{1}{\| \tilde{z} - z \|^{\nu} + \epsilon} \, \frac{\tilde{z} - z}{\| \tilde{z} - z \|} \tag{5.2}$$

will represent the "electrostatic force of attraction" between two individual points $z, \tilde{z} \in \mathbb{R}^2$. Observe that the closer the points are, the larger the magnitude of the force, with the small parameter ϵ included to avoid an infinite value if the points happen to coincide.

Fixing $\widetilde{C}$, we seek to improve the original fit transformation by finding a nearby rigid motion $g_{\mathit{lock}} \approx g_{\mathit{fit}}$, called a *lock*, that minimizes the total electrostatic potential energy generated by matching parts of the boundaries $g_{\mathit{lock}} \cdot C$ and $\widetilde{C}$. The result of the computation is a pair of *lock quality scores* q_1 and q_2, defined in (5.13) below, whose values tell us whether or not to assemble the two pieces.

As above, we work with the discretized boundaries, now explicitly denoted by $C^{\Delta} = \{z^1, \dots, z^n\}$ and $\widetilde{C}^{\Delta} = \{\tilde{z}^1, \dots, \tilde{z}^{\tilde{n}}\}$ with approximated signatures $S^{\Delta} = \{(\kappa^1, \kappa_s^1), \dots, (\kappa^n, \kappa_s^n)\}$

[3] We note that there is a misprint in formula (11) of [16], where the numerator should contain $p(\sigma^i, \widetilde{S}^{\Delta})$ instead of $h(\sigma^i, \widetilde{S}^{\Delta})$.

[4] By convention, $\arg 0 = 0$.

and $\widetilde{S}^{\Delta} = \{(\tilde{\kappa}^1, \tilde{\kappa}^1_s), \dots, (\tilde{\kappa}^n, \tilde{\kappa}^n_s)\}$. Let

$$d_\star = \frac{1}{n}\left[\| z^1 - z^n \| + \sum_{i=1}^{n-1} \| z^{i+1} - z^i \| \right] \tag{5.3}$$

denote the average distance between the sample points in C^{Δ}.

For the sake of computational speed, we work with various subsets of points in each discretized boundary. Namely, given $z \in C^{\Delta}$, a group element $g \in \mathrm{SE}(2)$, and a positive constant $K > 0$, define

$$\begin{aligned} &\widetilde{E}^{\Delta}(z, g, K) = \left\{ \tilde{z} \in \widetilde{C}^{\Delta} \;\middle|\; \| \tilde{z} - g \cdot z \| < K\, d_\star \right\}, \\ &\widetilde{E}^{\Delta}(g, K) = \bigcup_{z \in C^{\Delta}} \widetilde{E}^{\Delta}(z, g, K), \qquad E^{\Delta}(g, K) = \left\{ z \in C^{\Delta} \;\middle|\; \widetilde{E}^{\Delta}(z, g, K) \neq \varnothing \right\}. \end{aligned} \tag{5.4}$$

the last two of which represent the subsets of points in, respectively, $\widetilde{C}^{\Delta}$ and C^{Δ}, that are to be compared under the action of the rigid motion g. In particular, let $g_{\mathit{fit}} \in \mathrm{SE}(2)$ be the rigid motion (5.1) proposed by piece fitting of C^{Δ} and $\widetilde{C}^{\Delta}$. Fixing constants $K_1 \geq K_2 > K_3 \geq K_4 > 0$ and $\rho > 0$, the following algorithm is repeatedly applied up to some maximal number of iterations: $j_{\max} \geq 1$.

Piece Locking Algorithm

(1) Set $j = 0$, and $g_0 = g_{\mathit{fit}}$ to be the rigid motion prescribed by piece fitting.

(2) Set $d_0 = K_1\, d_\star$, which reflects the current "separation" of the pieces. Set $\theta_{-1} = c_{-1} = 0$, which reflects the current direction of "motion" of the piece C.

(3) Thinking of the points in $E^{\Delta}(g_{\mathit{fit}}, K_1)$, as defined in (5.4), as unit masses, the quantity $n_1 = \# E^{\Delta}(K_1)$ represents their total mass, while

$$z_{cm} = \frac{1}{n_1} \sum_{z \in E^{\Delta}(g_{\mathit{fit}}, K_1)} z \tag{5.5}$$

is their center of mass. Initialize $w_0 = g_0 \cdot z_{cm}$ to be its image under the rigid motion obtained from piece fitting. Further, set

$$r_2 = \sum_{z \in E^{\Delta}(g_{\mathit{fit}}, K_1)} \| z - z_{cm} \|^2, \qquad r_\infty = \max_{z \in E^{\Delta}(g_{\mathit{fit}}, K_1)} \| z - z_{cm} \|, \tag{5.6}$$

so that r_2 represents the moment of inertia of the set $E^{\Delta}(g_{\mathit{fit}}, K_1)$ around its center of mass, while r_∞ measures its overall extent.

(4) For the current iterate j, calculate the "total force"

$$f_j^{tot} = \sum_{z \in E^{\Delta}(g_{\mathit{fit}}, K_1)} f_j(z), \tag{5.7}$$

where

$$f_j(z) = \begin{cases} \displaystyle\sum_{\tilde{z}\in\widetilde{E}^{\Delta}(z,g_{fit},K_1)} f(g_j\cdot z,\tilde{z}), & \text{if } \| g_j\cdot z-\tilde{z}\| \geq K_4 d_\star \text{ for all } \tilde{z}\in\widetilde{E}^{\Delta}(z,g_{fit},K_1), \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

and the "total torque"

$$\tau_j^{tot} = \sum_{z\in E^{\Delta}(g_{fit},K_1)} (g_j\cdot z - w_j)\wedge f_j(z), \tag{5.9}$$

where $\wedge$ denotes the scalar cross product: $v\wedge w = v_1w_2 - v_2w_1$ for $v=(v_1,v_2)$, $w=(w_1,w_2)\in\mathbb{R}^2$.

(5) Calculate

$$A_j = \left\{ \min_{\tilde{z}\in\widetilde{E}^{\Delta}(z,g_{fit},K_1)} \| \tilde{z} - g_j\cdot z\| \;\middle|\; z\in E^{\Delta}(g_{fit},K_1) \right\}, \qquad \begin{aligned} d_j^{av} &= \text{mean } A_j, \\ d_j^{med} &= \text{median } A_j. \end{aligned} \tag{5.10}$$

If $d_j^{av} > d_j$ and $d_j^{med} \geq K_3 d_\star$, then the fit is poor and getting worse, and hence we terminate the algorithm by going to step (8), decrementing $j\mapsto j-1$.

(6) Let $h_j\in \mathrm{SE}(2)$ be the Euclidean motion

$$h_j\cdot z = \begin{pmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{pmatrix}(z-w_j) + w_j + c_j, \qquad \theta_j = \frac{\delta_j}{r_2}\,\tau_j^{tot}, \qquad c_j = \frac{\delta_j}{n_1}\,f_j^{tot}, \tag{5.11}$$

where

$$\delta_j = \frac{\rho\, d_j^{med}}{\max\left\{ \dfrac{\| f_j^{tot}\|}{n_1},\ \dfrac{r_\infty}{r_2}\,|\,\tau_j^{tot}\,| \right\}} \tag{5.12}$$

controls how far the transformation $h_j\in\mathrm{SE}(2)$ is allowed to stray from the identity, ensuring that the point-wise movements due to force and torque are at most a fraction ρ of the median interpoint distance d_{med}. This helps prevent the incremental motion h_j from "overshooting" the best fit.

(7) Update the key quantities by setting $g_{j+1} = h_j\cdot g_j$, $w_{j+1} = h_j\cdot w_j$, $d_{j+1} = d_j^{av}$, and then increment $j\mapsto j+1$. If $j\geq j_{\max}$, or if $\theta_j\theta_{j-1}<0$ and c_j and c_{j-1} have opposite signs in each component, terminate the loop by going to step (8), otherwise go back to step (4).

(8) The algorithm terminates with the proposed locking transformation $g_{lock} = g_j$. To measure the quality of the lock, for $K_3>0$, the sets $E^{\Delta}(g_{lock},K_3)$ and $\widetilde{E}^{\Delta}(g_{lock},K_3)$, as defined in (5.4), contain the points in C^{Δ} and $\widetilde{C}^{\Delta}$ respectively that are considered to belong to well matched arcs. Let L be the sum of the lengths of the arcs of $E^{\Delta}(g_{lock},K_3)$, and let L_{av} and n_{av} be the average perimeter and mesh size respectively amongst all inputed pieces. The *quality* of the lock is reflected in three scores:

$$q_1 = \frac{\#\widetilde{E}^{\Delta}(g_{lock},K_3)}{\#\widetilde{E}^{\Delta}(g_{lock},K_2)}, \qquad q_2 = \frac{L}{L_{av}}, \qquad \text{and} \qquad q_3 = \frac{1}{n_{av}d_\kappa} \sum_{\{i:z^i\in E^{\Delta}(g_{lock},K_3)\}} |\kappa^i|\,. \tag{5.13}$$

Here q_1 measures the fraction of those points in C^Δ that became well matched, as measured by $K_3 < K_1$, relative to those that came within $K_2 d_\star$ of $\widetilde{C}^\Delta$. q_2 gives a relative measure of the arc length that became well matched, and q_3 gives a relative measure of the significance (in terms of curvature) of the well matched portion. The lock will be deemed to be *good* when

$$\big(q_1\eta_1 + q_3\eta_2 > Q_1, \quad \text{or} \quad q_2 > Q_2^*, \big) \quad \text{and} \quad q_2 > Q_2, \quad \text{and} \quad q_3 > Q_3, \tag{5.14}$$

for some weight vector $\vec{\eta} = (\eta_1, \eta_2)$ and quality criterion $\vec{Q} = (Q_1, Q_2, Q_2^*, Q_3)$.

5.3 Fitting a Piece to a Subpuzzle.

In order to improve computational speed, we fit pieces to a subpuzzle in a series of rounds. Starting a round with a subpuzzle $\mathcal{Q}_k \subset \mathcal{P}$ and the set of remaining pieces $\mathcal{R}_k^*$, we use piece fitting and locking to try to fit a new piece $\mathbb{P}_i \in \mathcal{R}_k^*$. To do so, we apply steps (1–3) of the Piece Fitting Algorithm of Subsection 5.1 to all possible pairs of pieces $\mathbb{P}_i, \mathbb{P}_j$ with $\mathbb{P}_i \in \mathcal{R}_k^*$ and $\mathbb{P}_j \in \mathcal{Q}_k^*$ (computations that were performed previously should be saved for this purpose), requiring for computational speed that $\mathbb{P}_j$ or a piece it borders to border a piece that was placed in the current or previous round.[5] The resulting pairs of sequences of bivertex arcs are ordered according to their lengths, namely the value of $m+1$ in the language of Subsection 5.1, with pairs with equal values sorted according to their appearance.

Going through the ordered list of pairs of sequences of bivertex arcs, we apply step (4) of the piece fitting algorithm to each in turn. If, in the result, $\mu < \mu_0$ for some preselected criterion μ_0, we disregard the pair, but add to our list (if they are not already there) the two pairs of bivertex arc sequences obtained by deleting, respectively, the first and the last pairs of bivertex arcs from the discarded pair of sequences. On the other hand, if $\mu \geq \mu_0$, we perform piece locking as described in Subsection 5.2, regarding the unplaced piece $\mathbb{P}_i$ as mobile and the *active* boundary points of the subpuzzle $\mathcal{Q}_k$ (i.e. those that have not been deemed *inactive* as below) as the fixed piece.

If the quality of the resulting locking rigid motion is good, as measured by (5.14), then $\mathbb{P}_i$ is considered to be properly placed and is assigned the transformation $g_i = g_{lock}$ generated by fit finding and improved by piece locking. Any point or bivertex arc that has non-empty intersection with $E^\Delta(g_{lock}, K_2)$ or $\widetilde{E}^\Delta(g_{lock}, K_2)$, as in (5.4), is deemed to be *inactive*. The next pair of bivertex arc sequences is then analyzed, ignoring pairs that contain inactive arcs or that fit an already placed piece to the subpuzzle. If the list of pairs has been exhausted, the next round begins, provided a piece was placed during the current round; if not, the algorithm terminates as described earlier.

6 Applications.

To apply our algorithm to a physical jigsaw puzzle, we begin with individual photographs of puzzle pieces. The quality and accuracy of the photos is very important, since small errors may propagate as a large puzzle is gradually assembled, and the accumulating error may

[5] This requirement is removed for a round when the parameter sequence (see Subsection 6.3) increments.

cause the piece fitting algorithm to break down. When photographed, pieces were assigned random orientations, and then arranged in pseudo-random order.

The next step is to segment the photos in order to obtain the puzzle piece boundaries. Following segmentation, we then smooth out the noise that occurs in the segmented curves. To save time, the segmentation and smoothing operations on the individual pieces were run in parallel on different machines with various processing speeds. Once this preliminary step is completed, we are ready to run the piece fitting and locking algorithms.

6.1 Segmentation.

The segmentation process we employ is based on the method of *active contours* or *snakes*. The underlying idea is to evolve a curve on a digital image until it meets a boundary. This allows for segmentation of part of an image without concern for the content of other parts. Moreover, the result is automatically a connected boundary curve, thus avoiding the "connect the dots" problem that plagues alternative segmentation procedures. The details of the segmentation algorithm are not explored further here, and we direct the reader to [6, 7, 17] for more information. For our purposes it suffices to know that segmentation by active contours can be used to obtain reasonably accurate, discretely represented boundary curves from photographs of puzzle pieces. Our implementation of the segmentation method is based on code written by S. Lankton, [19].

6.2 Smoothing.

To reduce the computational intensity, the number of pixels in the high definition photographs must be reduced in order to perform the required segmentation in a reasonable time frame. For example, the photographs of the Baffler Nonagon used for this paper had an initial resolution of approximately 238 pixels per centimeter. Reducing the resolution by a factor of 3 enabled each piece to be segmented in around 13 minutes on the 2.4 GHz laptop. However, the reduced resolution meant that the resulting discrete segmented boundary curves were not sufficiently smooth to be able to compute meaningful values for the curvature invariants κ and κ_s using the numerical approximation methods outlined in [1, 4]. Thus, to construct usable discrete signatures and then successfully apply the methods of sections 4 and 5, we needed to subject the piece boundaries to a preliminary smoothing operation. We ended up investigating two methods of smoothing discrete curves.

The first method to be tried was smoothing by application of the *curve shortening flow*, [9, 11, 14], which moves a curve in its normal direction in proportion to its curvature. Curve shortening acts like a nonlinear diffusion equation, smoothing out noise and small-scale features, and eventually contracting a closed curve to a "round point". While this method of smoothing has many nice properties, and is extensively used in applications, we discovered one significant disadvantage. Namely, since curve shortening contracts the curve at all points of non-zero curvature, it proved difficult to preserve distinguishing features until such time as the boundary curve was sufficiently smooth to be able to compute a meaningful Euclidean signature. While, as observed in [4], the smoothed signatures continue to encode remnants of these features, this causes major difficulties with puzzle assembly. Additionally, identical boundary curves may well become noticeably non-congruent if smoothed by different

amounts.

To overcome the observed difficulties with curve shortening, we sought a method of smoothing that led to a meaningful Euclidean signature while still preserving the distinguishing features. We ended up employing a rather simple spline-based smoothing method. A periodic spline of the points of the curve is calculated. The points are then redistributed evenly by arc-length along the spline in a way that is offset from the original mesh. We will refer to this method as the *spline-and-respace* method of smoothing. In applications, spline-and-respace is carried out a number of times; in our examples 1500 iterations seemed optimal. We did not try to conduct a rigorous analysis of this method, and we have only empirical evidence to support its use. More sophisticated methods based on smoothing splines, e.g., [28], could well be faster and more accurate, but this remains for future investigation. In all of our applications, spline-and-respace smoothing led to a meaningful signature while retaining all the relevant curve features, and so became our method of choice to smooth the segmentations of puzzle pieces.

6.3 Generating Solutions.

By applying the segmentation and smoothing algorithms discussed in the preceding subsections, we are able to straightforwardly generate discrete representations of puzzle pieces from photographs. We can then apply the puzzle solving algorithm described in section 4. The performance of the algorithm depends upon the choice of the following parameters

$$\alpha,\ \beta,\ \gamma,\ C_1,\ C_2,\ p_0,\ m_0,\ \mu_0,\ K_1,\ K_2,\ K_3,\ K_4,\ \lambda_0,\ \lambda_1,\ \nu,\ \epsilon,\ \rho,\ j_{\max},\ \vec{\eta},\ \vec{Q}, \tag{6.1}$$

some of which are introduced in [16]. (The parameter l in that paper only applies to closed curves, and so does not play a role here.) Of these, p_0, m_0, and μ_0 will be called *depth parameters* (since for example, decreasing the value of μ_0 may increase the number of fits that are checked by piece locking), while the constants K_2, K_3, and $\vec{Q}$ will be called *quality parameters* since they determine how well pieces must fit to be considered correctly placed. Thus the choice of parameters affects the speed of the algorithm as well as its accuracy.

We observed significant gains in processing speed by allowing the depth parameters to vary during the computation. We search first at a low depth, where correct matches tend to occur most frequently. But, if the algorithm is unable to place any additional pieces, we then allow it to search at greater depths. To this end, we introduce a finite sequence of parameters $(p_{0,j}, m_{0,j}, \mu_{0,j}, K_{3,j})$ for $1 \le j \le j_\star$. The algorithm is first applied with $j = 1$, but, whenever it dead-ends, we increment j, terminating the algorithm once $j > j_\star$. After a round in which a piece is successfully placed, we return to $j = 1$. This allows the algorithm to run at shallow depths whenever possible and only search deeply in order to get "unstuck."

We applied the puzzle solution algorithm to the pieces of the Baffler Nonagon with parameters

$$\begin{gathered}\alpha = 2, \quad \beta = 5, \quad \gamma = 5, \quad C_1 = 1000, \quad C_2 = 2, \quad K_1 = 15, \quad K_2 = 4, \quad K_4 = \tfrac{1}{2}, \quad \lambda_0 = 20, \\ \lambda_1 = 10, \quad \nu = 3, \quad \epsilon = 10^{-4}, \quad \rho = \tfrac{1}{3}, \quad j_{\max} = 50, \quad \vec{\eta} = (1, 1.5), \quad \vec{Q} = (.9, .2, .3, 0),\end{gathered} \tag{6.2}$$

and parameter sequence

$$\{\, (p_{0,j}, m_{0,j}, \mu_{0,j}, K_{3,j}) \mid\ j = 1, 2 \,\} = \big\{(.94, 3, .6, 1.118),\ (.9, 2, .6, 2)\big\}, \tag{6.3}$$

so $j_\star = 2$. Following segmentation and smoothing, the puzzle was correctly solved in 31 minutes. We remark that the solution time should be judged relative to the complexity of the pieces. Puzzles that obey the assumptions (1–4) in section 1 allow for simpler solving algorithms, which are inherently faster, since the assumptions largely restrict the possible relative positions pieces can take (e.g. the rectangular nature of pieces reduces the the number of (relative) rotations to be checked to the four that align the piece's straight edges with those of the other pieces). Figure 2 displays the pieces as inputed and the solved puzzle.

While the unusual shapes of the pieces in the Baffler Nonagon make its solution more challenging, the fact that they are so different makes errors in piece placement less likely. In order to show that the method is also effective on puzzles with many similar pieces, we applied it to the simpler (at least to a human) Rain Forest Puzzle [23], with parameters

$$\begin{gathered}\alpha = 2, \quad \beta = 5, \quad \gamma = 5, \quad C_1 = 1000, \quad C_2 = 2, \quad K_1 = 15, \quad K_2 = 4, \quad K_4 = \tfrac{1}{2}, \quad \lambda_0 = 30, \\ \lambda_1 = 10, \quad \nu = 3, \quad \epsilon = 10^{-4}, \quad \rho = \tfrac{1}{3}, \quad j_{\max} = 50, \quad \vec{\eta} = (1, 1.5), \quad \vec{Q} = (.9, .2, .3, 0),\end{gathered} \tag{6.4}$$

and parameter sequence

$$\big\{\, (p_{0,j}, m_{0,j}, \mu_{0,j}, K_{3,j}) \;\big|\; j = 1, 2 \,\big\} = \big\{(.94, 2, .6, .707),\ (.9, 2, .6, 2)\big\}. \tag{6.5}$$

Note that the only difference with the Baffler parameters (6.2), (6.3) is an increase in λ_0 and decrease in $m_{0,1}$ and $K_{3,1}$. This is done in order to maintain a better quality control of the process, given that the Rain Forest pieces are both significantly larger, and hence of higher resolution, as well as more alike, and so requiring a better quality of fit to ensure proper placement. Following segmentation and smoothing, the puzzle was correctly solved in 58 minutes. Figure 3 shows the pieces as inputed and the solved puzzle. Again, we emphasize that our method makes no a priori assumptions on the shape of the puzzle boundary.

The software used to compute these examples is available for downloading from the second author's web page: `http://math.umn.edu/~olver/matlab.html`

7 Scalability.

In order to test the scalability of our algorithm, we experimented on portions of the 1000 piece Cathedral jigsaw puzzle [29]. This puzzle is non-standard, yet contains many standard "indents" and "outdents." This combination makes it particularly difficult to solve by computer, since there are many "local" near-matches, and yet simplified algorithms cannot be applied because of the unusual structure.

When addressing larger numbers of puzzles pieces that contained many near-matches, we found that additional precision is required of the input data. Toward this end, we scanned the backside of 111 pieces of the Cathedral against a black background on a flatbed scanner at 600 dpi. As above, the images were segmented using active contours. However, in order to minimize the effects of fringes on the pieces, we then applied the following refinement to the segmentation: First, we fill in (turn to foreground) any patches of background pixels whose boundary contour is comprised of fewer than 1000 pixels. Then we form the 7-by-7 square pixel grid centered at each foreground pixel, and change that pixel to background if it lies

halfway between any two background pixels in the grid. This step is applied recursively; after it terminates, the contours are extracted.

Using the resulting data, and parameters

$$\alpha = 2, \quad \beta = 5, \quad \gamma = 5, \quad C_1 = 1000, \quad C_2 = 2, \quad K_1 = 15, \quad K_2 = 4, \quad K_4 = \tfrac{1}{2}, \quad \lambda_0 = 20,$$
$$\lambda_1 = 8, \quad \nu = 3, \quad \epsilon = 10^{-4}, \quad \rho = \tfrac{1}{3}, \quad j_{\max} = 50, \quad \vec{\eta} = (1, 1.5), \quad \vec{Q} = (.976, .24, .3, .029), \tag{7.1}$$

with (length one) parameter sequence

$$\{\, (p_{0,j}, m_{0,j}, \mu_{0,j}, K_{3,j}) \mid \ j = 1 \,\} = \big\{(.94, 2, .6, .707)\big\}. \tag{7.2}$$

the algorithm assembled 103 of the 111 pieces before terminating. Following segmentation and smoothing, the puzzle was solved to this extent in approximately 36 hours. Figure 4 shows the pieces as inputed and the partially solved puzzle.

The combination of many near-matches and error in the image processing made high-confidence matching (i.e. the use of high values for Q_1, Q_2, and Q_3) necessary for accuracy in this computation. Hence correct matches may be not be accepted the first time they are tried, but are later once more neighboring pieces have been placed (resulting in higher q_1, q_2, and q_3). This extra care notably increases the computation time, but this effect could be reduced with higher fidelity data. It is also likely that this puzzle fragment would be solved completely if lower values of Q_1, Q_2 and Q_3 were employed after the algorithm "dead-ends" as part of a modified parameter sequence.

Another factor that entered into play with the larger puzzle was accumulation of errors. In particular, we noted some differences in long-term performance on machines running different operating systems. Although at this time we have not explored it fully, we believe this is largely due to the repeated composition of Euclidean transformations stored in different forms (compare (2.1) and (5.11)) using sine and cosine functions compiled to varying accuracies.

8 Further Directions.

Our results indicate a number of different directions that are worth pursuing. The first are concerned with improving the puzzle solving algorithms as developed so far. One immediate issue worth further investigation is the specification of the various parameters (6.1). While some work was already done in this direction, particularly our realization that the depth parameters could be profitably adjusted during the course of the assembly algorithms, we did not conduct a systematic investigation into fine tuning the parameter values for optimal performance, which may well depend upon the nature of the puzzle, the accuracy of the photographs of the pieces and their segmented/smoothed boundaries, and the desired speed of computation. A future research project will be to better understand the optimal parameter values, perhaps through automatic learning on a larger training set. It would be interesting to see our program compete with human puzzle solvers, particularly in challenging cases like the Baffler Nonagon, or puzzles that allow overlapping pieces, if such can be devised.

We were pleasantly surprised by the algorithm's ability to accurately place pieces, and then completely solve the challenging large-scale puzzles we tried it on. We had initially expected to require it to do a fair amount of backtracking in the event that a piece is

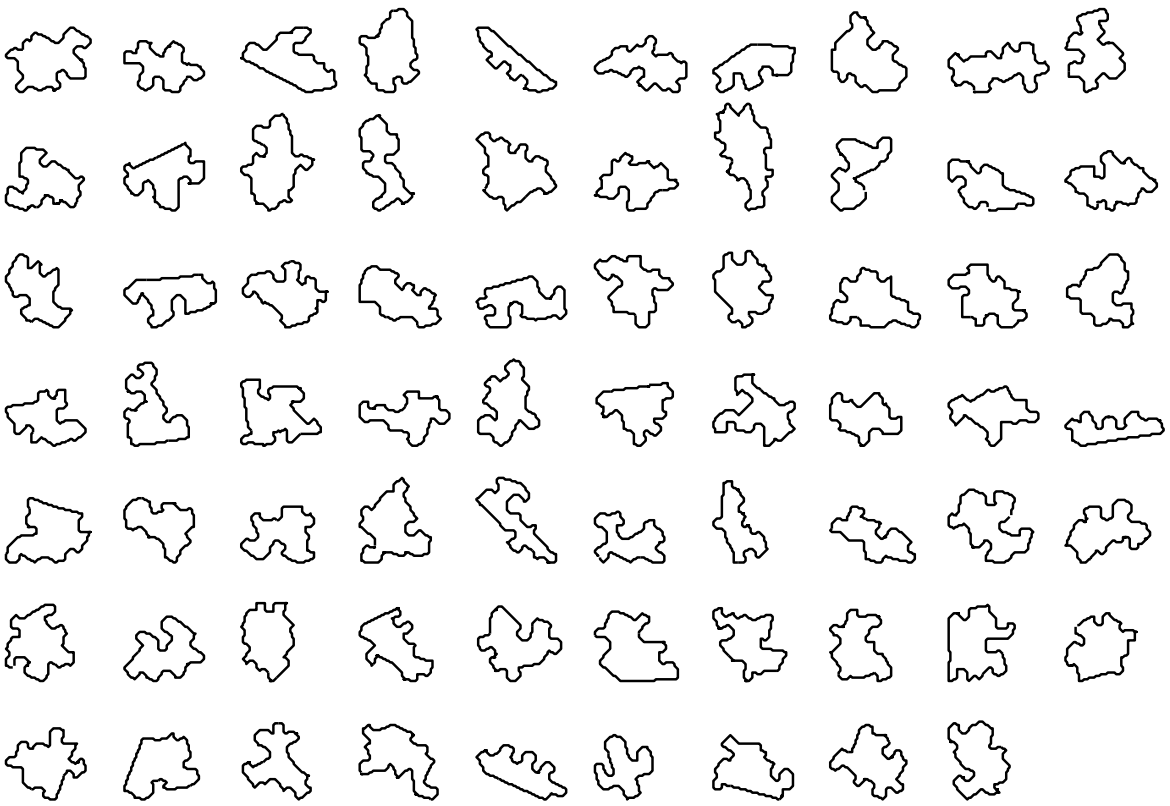
wrongly placed, and allowed for that possibility to be incorporated in our code. If more challenging examples require revisiting this part of the algorithm, a careful analysis of the relevant backtracking and branching rules would be worth doing. This would be particularly useful for dealing with puzzles whose pieces fit together in multiple ways, not all of which are compatible with the final assembly. We had also envisioned having to readjust the earlier piece placements if their accumulated errors prevented completing the puzzle, even when correctly placed. One idea, motivated by the numerical algorithm of simulated annealing, [21], would be to employ random perturbations to "jiggle" all the pieces in a subpuzzle in order to improve its overall fit quality.

Provided the photographs of the individual pieces are sufficiently accurate, the segmentation algorithm does a fine job of locating a decent approximation to their boundaries. On the other hand, the simple spline-and-respace smoothing method could certainly be improved, perhaps by using the more sophisticated approaches developed in [28]. Spline smoothing might also be enhanced by linking it to the approximate bivertex decomposition of the piece boundaries, e.g., by respacing the nodes in a non-uniform manner. For instance, we found that weighting the mesh density by κ_s gives some intriguing results, but this requires further investigation before being useful.

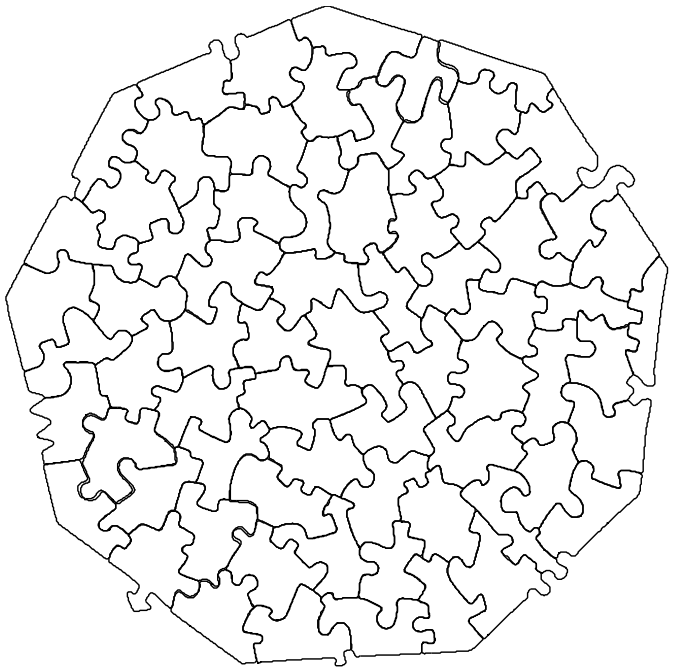
One question that might occur to the reader is why use two steps to place the pieces: piece fitting based on the bivertex arcs followed by piece locking. We did contemplate bypassing the signature matching step, and using piece-locking alone, but the latter algorithm is much slower than signature comparison, and hence, as it stands, would not produce the desired matching results as efficiently. Moreover, in order to characterize the bivertex arcs, we have effectively already computed their signatures, and so by this stage signature matching is extremely efficient.

It would certainly be worthwhile extending our algorithms, both for fitting and smoothing, to be more attuned to corners, but this was not pursued as it turned out to be unnecessary to successfully solve the puzzles we considered. Furthermore, developing a way to universally treat subpuzzles as single pieces, while unnecessary in the examples treated here, could make the algorithm more robust. Building on this, a more sophisticated approach would be to allow several distinct subpuzzles to be assembled concurrently, and then deal with the problem of fitting them together. Finally, methods that incorporate overall designs or pictures on the puzzle with our shape matching algorithms will be left to future research. In a practical direction, it would be very revealing to test our algorithms on other types of assembly problems, such as broken archaeological artifacts, e.g., ceramics or pottery, or shredded documents.

Since our extended signature methods are semi-local, based on subarcs of boundary curves, they could also be readily adapted to the problems of recognizing and reconstructing objects under partial occlusion, [2, 27]. Indeed, our piece locking refinement could be applied back to the original problem of object recognition, [4, 16], with or without occlusion, to, we expect, great effect. Finally, in a more speculative vein, one might try to extend these methods to solving fully three-dimensional puzzles, e.g., broken statues. The underlying Cartan equivalence method applies to general submanifolds, and the corresponding differential invariant signatures that uniquely characterize (sufficiently regular) surfaces under rigid motions are known, [20]. However, there are major practical, numerical, and computational issues that need resolving before such an extension can be contemplated.

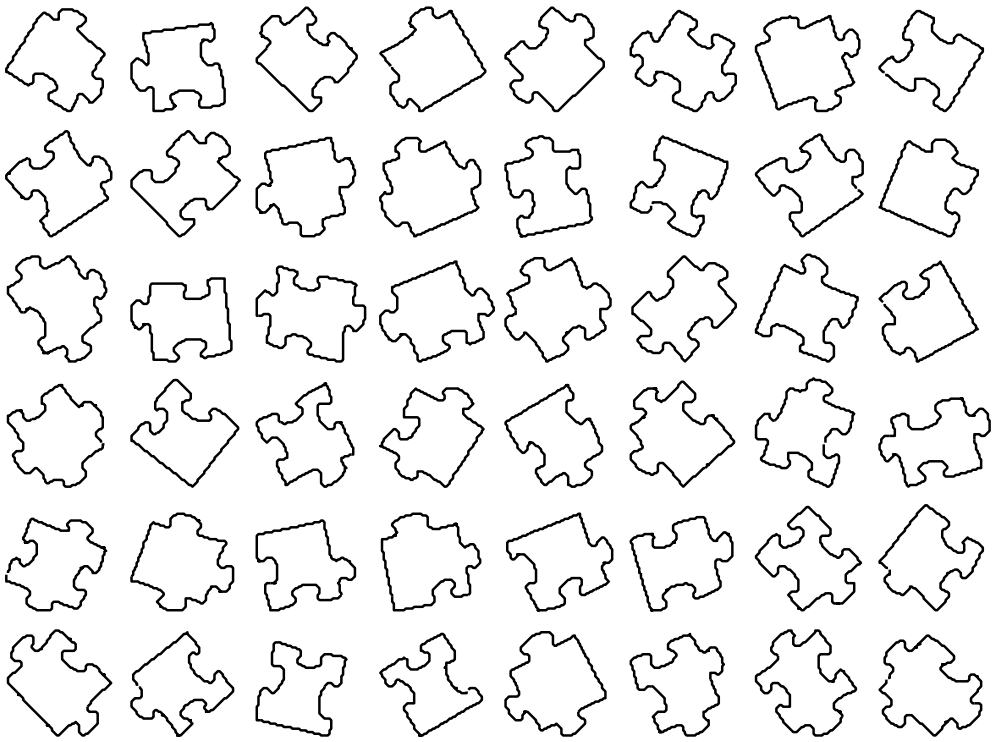


(a) Pieces of the Baffler Nonagon

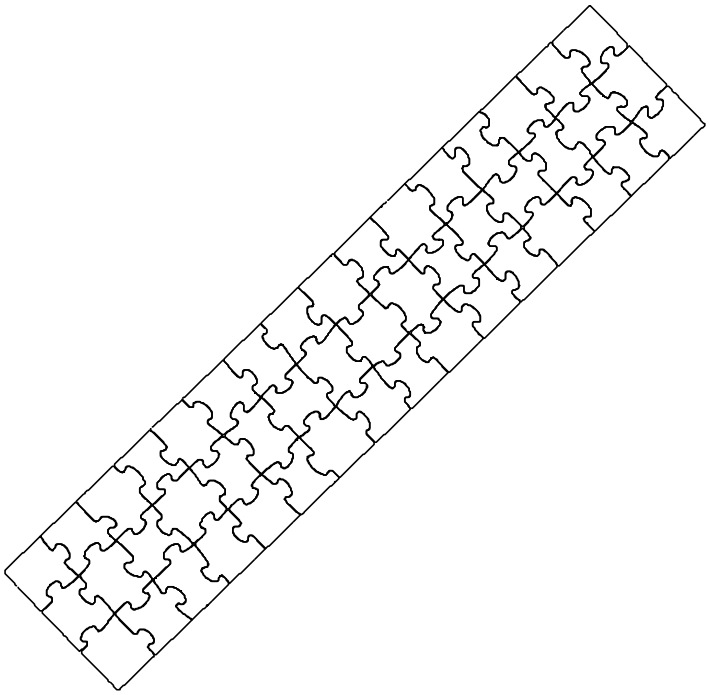


(b) The Solved Baffler Nonagon

Figure 2: The pieces (a) and solution (b) of the Baffler Nonagon [33]. The pieces in (a) are displayed in the orientations and pseudo-random order (left to right, top to bottom) as inputed to the solving algorithm. The solution method produced (b) from the data in (a) in 31 minutes.

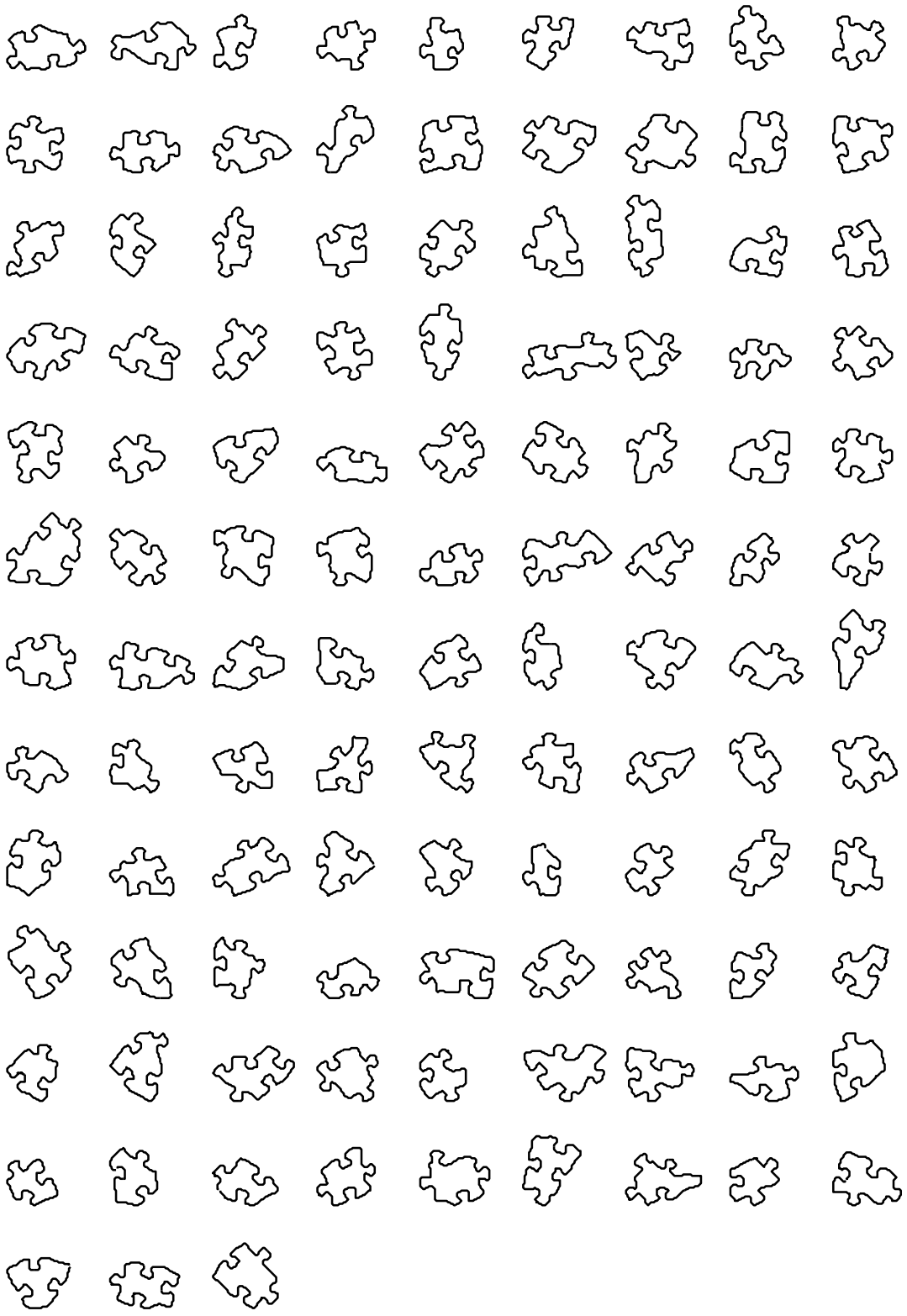


(a) Pieces of the Rain Forest Giant Floor Puzzle



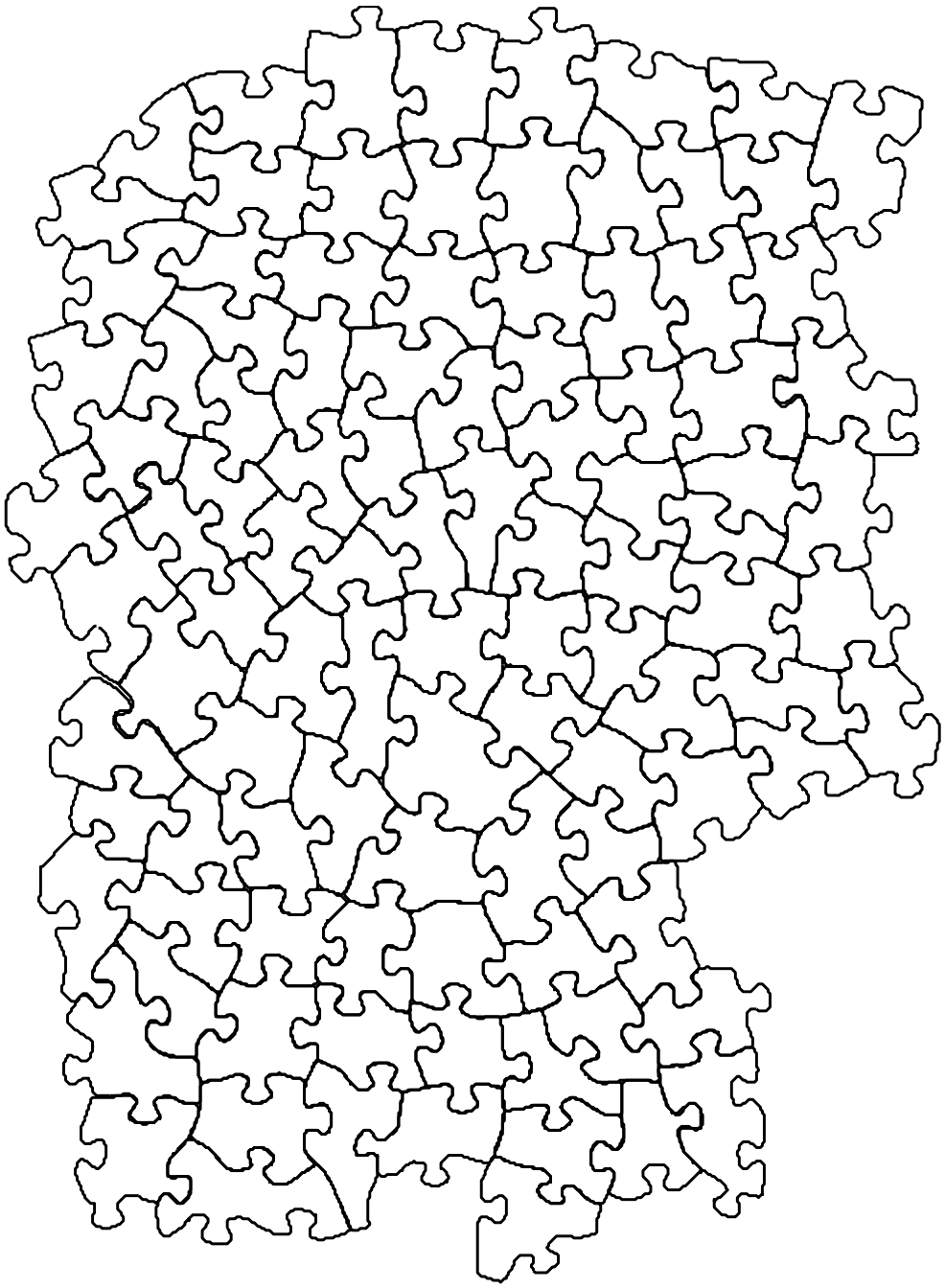
(b) The Solved Rain Forest Giant Floor Puzzle

Figure 3: The pieces (a) and solution (b) of the Rain Forest Giant Floor Puzzle [23]. The pieces in (a) are displayed in the orientations and pseudo-random order (left to right, top to bottom) as inputed to the solving algorithm. The solution method produced (b) from the data in (a) in 58 minutes.



(a) Pieces of the Cathedral Jigsaw Puzzle

Figure 4



(b) Partial Solution to the Cathedral Jigsaw Puzzle

Figure 4: The pieces (a) and 103 piece partial solution (b) of the Cathedral Jigsaw Puzzle [29]. The pieces in (a) are displayed in the orientations and pseudo-random order (left to right, top to bottom) as inputed to the solving algorithm. The solution method produced (b) from the data in (a) in approximately 36 hours. We note that (b) has been rotated 90 degrees clockwise from the output in order to better fit the page.

References

[1] Boutin, M., Numerically invariant signature curves, *Int. J. Computer Vision* **40** (2000), 235–248.

[2] Bruckstein, A.M., Holt, R.J., Netravali, A.N., and Richardson, T.J., Invariant signatures for planar shape recognition under partial occlusion, *CVGIP: Image Understanding* **58** (1993), 49–65.

[3] Burdea, G.C., and Wolfson, H.J., Solving jigsaw puzzles by a robot, *IEEE Transactions on Robotics and Automation* **5** (1989), 752–764.

[4] Calabi, E., Olver, P.J., Shakiban, C., Tannenbaum, A., and Haker, S., Differential and numerically invariant signature curves applied to object recognition, *Int. J. Computer Vision* **26** (1998), 107–135.

[5] Cartan, É., Les problèmes d'équivalence, *in*: *Oeuvres Complètes*, part. II, vol. 2, Gauthier–Villars, Paris, 1953, pp. 1311–1334.

[6] Caselles, V., Kimmel, R., and Sapiro, G., Geodesic active contours, *Int. J. Computer Vision* **22** (1997), 61–79.

[7] Chan, T., and Vese, L., Active contours without edges, *IEEE Transactions on Image Processing* **10** (2001), 266–277.

[8] The DARPA Shredder Challenge, `http://www.shredderchallenge.com/`, 2011.

[9] Deckelnick, K., Dziuk, G., and Elliott, C.M., Computation of geometric partial differential equations and mean curvature flow, *Acta Numer.* **14** (2005), 139–232.

[10] Freeman, H., and Carder, L., Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition, *IEEE Trans. Elec. Comp.* **13** (1964), 118–127.

[11] Gage, M., and Hamilton, R.S., The heat equation shrinking convex plane curves, *J. Diff. Geom.* **23** (1986), 69–96.

[12] Gallagher, A.C., Jigsaw puzzles with pieces of unknown orientation, *in*: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Comp. Soc. Press, Los Alamitos, CA, 2012, pp. 382–389.

[13] Goldberg, D., Malon, C., and Bern, M., A global approach to the automatic solution of jigsaw puzzles, *Computational Geometry* **28** (2004), 165–174.

[14] Grayson, M., The heat equation shrinks embedded plane curves to round points, *J. Diff. Geom.* **26** (1987), 285–314.

[15] Guggenheimer, H.W., *Differential Geometry*, McGraw–Hill, New York, 1963.

[16] Hoff, D., and Olver, P.J., Extensions of invariant signatures for object recognition, *J. Math. Imaging Vision* **45** (2013), 176–185.

[17] Kichenassamy, S., Kumar, A., Olver, P.J., Tannenbaum, A., and Yezzi, A., Conformal curvature flows: from phase transitions to active vision, *Arch. Rat. Mech. Anal.* **134** (1996), 275–301.

[18] Kong, W., and Kimia, B.B., On solving 2D and 3D puzzles using curve matching, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2** (2001), 583–590.

[19] Lankton, S., Active Contours, `http://www.shawnlankton.com/2007/05/active-contours/`, 2007.

[20] Olver, P.J., *Equivalence, Invariants, and Symmetry*, Cambridge University Press, Cambridge, 1995.

[21] Pham, D.T., and Karaboga, D., *Intelligent Optimisation Techniques*: *Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer–Verlag, New York, 2000.

[22] Radack, G.M., and Badler, N.I., Jigsaw puzzle matching using a boundary-centered polar encoding, *Comp. Graphics Image Proc.* **19** (1982), 1–17.

[23] The Rain Forest Giant Floor Puzzle, Frank Schaffer Publications, Inc., Torrance, CA, 1992.

[24] Sağiroğlu, M.S., and Erçil, A., A texture based approach to reconstruction of archaeological finds, *in*: *Proceedings of the 6th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage, VAST05*, Mudge, M., Ryan, N., Scopigno, R., eds., Eurographics Assoc., Aire-la-Ville, Switzerland, 2005, pp. 137–142.

[25] Shakiban, C., and Lloyd, P., Signature curves statistics of DNA supercoils, *in*: *Geometry, Integrability and Quantization*, vol. 5, I.M. Mladenov and A.C. Hirschfeld, eds., Softex, Sofia, Bulgaria, 2004, pp. 203–210.

[26] Shakiban, C., and Lloyd, R., Classification of signature curves using latent semantic analysis, *in*: *Computer Algebra and Geometric Algebra with Applications*, H. Li, P.J. Olver, and G. Sommer, eds., Lecture Notes in Computer Science, vol. 3519, Springer–Verlag, New York, 2005, pp. 152–162.

[27] Turney, J.L., Mudge, T.N., and Volz, R.A., Recognizing partially occluded parts, *IEEE Trans. Pattern Anal. Machine Intel.* **7** (1985), 410–421.

[28] Wang, Y., *Smoothing Splines*: *Methods and Applications*, Monographs Stat. Appl. Probability, vol. 121, CRC Press, Boca Raton, FL, 2011.

[29] Wesley, R., Cathedral, Product #51015, SunsOut, Inc., Costa Mesa, CA.

[30] Wolfson, H., Schonberg, E., Kalvin, A., and Lamdan, Y., Solving jigsaw puzzles by computer, *Annals of Operations Research* **12** (1988), 51–64.

[31] Wu, K., and Levine, M.D., 3D part segmentation using simulated electrical charge distributions, *IEEE Trans. Pattern Anal. Machine Intel.* **19** (1997), 1223–1235.

[32] Feng-Hui Yao, F.–H., and Shao, G.–F., A shape and image merging technique to solve jigsaw puzzles, *Pattern Recognition Lett.* **24** (2003), 1819–1835.

[33] Yates, C., The Baffler: the nonagon, Ceaco, Newton, MA, 2010.

[34] Zisserman, A., Forsyth, D.A., Mundy, J.L., and Rothwell, C.A., Recognizing general curved objects efficiently, *in*: *Geometric Invariance in Computer Vision*, J.L. Mundy and A. Zisserman, eds., MIT Press, Cambridge, Mass., 1992, pp. 228–251.