Jigsaw Puzzles with Pieces of Unknown Orientation

Andrew C. Gallagher
Eastman Kodak Research Laboratories
Rochester, New York
andrew.c.gallagher@gmail.com

Abstract

This paper introduces new types of square-piece jigsaw puzzles: those for which the orientation of each jigsaw piece is unknown. We propose a tree-based reassembly that greedily merges components while respecting the geometric constraints of the puzzle problem. The algorithm has state-of-the-art performance for puzzle assembly, whether or not the orientation of the pieces is known. Our algorithm makes fewer assumptions than past work, and success is shown even when pieces from multiple puzzles are mixed together. For solving puzzles where jigsaw piece location is known but orientation is unknown, we propose a pairwise MRF where each node represents a jigsaw piece's orientation. Other contributions of the paper include an improved measure (MGC) for quantifying the compatibility of potential jigsaw piece matches based on expecting smoothness in gradient distributions across boundaries.

1. Introduction

For hundreds of years, people have been entertained by the challenge of assembling the pieces of a jigsaw puzzle into a complete picture. One imagines that the same strategies employed by human solvers today were used to solve those first puzzles produced by British mapmaker John Spilsbury in the 18th century, relying on the puzzle piece shapes and textures. Certainly, this combinatorial challenge is one that inspires developments in computer science. The computational problem of jigsaw puzzle assembly was first introduced nearly fifty years ago in a fundamental work by Freeman and Gardner [7]. As with a physical jigsaw puzzle, the object of the computational problem is to adjoin a number of smaller jigsaw pieces to form a complete picture.

There are two essential components for computationally solving a jigsaw puzzle, a measure of jigsaw piece compatibility for adjoining a pair of jigsaw pieces and a strategy for puzzle assembly. In this paper, we propose advances in both categories: this paper introduces a new measure for quantifying the compatibility of adjacent jigsaw pieces, and a new constrained tree-based assembly.

(a) 3456 Jigsaw Pieces

(b) 9600 Jigsaw Pieces

Figure 1: In this paper, we introduce square-piece puzzles where the orientation of the jigsaw pieces is unknown. We solve puzzles using a constrained minimal spanning tree algorithm. We often achieve perfect reassembly of very large puzzles; the assembled puzzle in (a) has 3456 pieces (right) from its jigsaw pieces (left), and the puzzle in (b) has 9600 jigsaw pieces. We believe these are the largest automatically solved puzzles to date, and certainly the largest with pieces of unknown orientation.

As noted by [9], the intriguing nature of the puzzle assembly problem is enough to justify research on the topic. In addition to being an interesting problem in its own right, computational jigsaw assembly has applications in reassembling archaeological artifacts [10] and recovering shredded documents or photographs [2, 17, 11].

We follow the lead of recent work [1, 3, 24, 19] and consider jigsaw puzzles with square pieces. This allows us to focus our efforts exclusively on image content. Our contributions to the state-of-the-art are as follows: First, we introduce two new types of puzzles having pieces with unknown orientation. To solve puzzles with jigsaw pieces of unknown location and orientation, we propose a greedy tree-based algorithm. We relax the assumption that the puzzle dimensions must be known at the time the puzzle is assembled. To solve puzzles with jigsaw pieces of known lo-

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

cation and unknown orientation, we propose a graph model solution. Second, we define a Mahalanobis-inspired jigsaw piece compatibility measure and show that it improves over others, and allows us to tackle more difficult puzzles.

2. Related Work

Puzzle assembly has a rich literature of exploration. Following Freeman *et al*. [7], several other early works explore aspects of using jigsaw piece shape information and contour matching to find likely matching pieces. For example, in [22], the well-known human strategy of solving the jigsaw puzzle boundary first is employed by identifying edge pieces with shape information, and posing edge assembly as a traveling salesman problem. With square-piece puzzles, [3] proposes an MRF, but enforcing the global constraints (e.g. that each piece should appear once), proves difficult. Recently, Pomeranz *et al*. [19] showed improved performance with a greedy algorithm that segments a partial solution (by growing a single component) and then shifts assembled portions, looking for improved fits. Inevitably, tough decisions (during component growing) are made earlier than absolutely necessary. Both of the aforementioned assume oriented jigsaw pieces, and both assume that the dimensions of the assembled puzzle are known.

Demaine and Demaine [6] show that when there is uncertainty in the jigsaw piece compatibility, puzzle assembly is an NP-hard problem. Therefore, we cannot define a global energy function that can be efficiently optimized. In other words, there are too many ways in which the jigsaw pieces could be assembled to evaluate them all. Instead, researchers have proposed a variety of assembly strategies, including greedy selection of puzzle pieces, edge identification and assembly, and Markov Random Fields.

Kosiba *et al*. [13] was the first to use both jigsaw piece shape *and* image information by encouraging adjacent jigsaw pieces to have similar colors by computing color compatibility along the matching contour. Chung *et al*. follow, penalizing squared color disagreement across the boundary, and later in [3, 24] this compatibility (in LAB color space) is confirmed as a good choice from among a set of options. Additional works that consider color [1, 16, 18, 20, 25] use subtle variations of this dissimilarity measure. Exceptions include [1], where the abutting profiles of two jigsaw pieces are matched with dynamic time warping, [20], where inpainting [5] is used to hallucinate the content across a boundary, and [19], where a compatibility measure based on predicting the values across the boundary is proposed.

We summarize a selection of works related to puzzle assembly in Table 1 according to whether they use shape features or color features, and the number of pieces in the puzzle. Further, we note that several other papers on jigsaw assembly have thorough reviews of the related work, including [3, 9, 25]. We believe our paper is the first to describe algorithms for solving jigsaws with square pieces of unknown orientation.

Author	Year	Color	Shape	Square Pieces	Puzzle Size
Freeman [7]	1964		✓		9
Wolfson [22]	1988		✓		104
Webster [21]	1991		✓		9
Kosiba [13]	1994	✓	✓		54
Chung [4]	1998	✓	✓		54
Kong [12]	2001		✓		32
Goldberg [9]	2002		✓		204
Yao [25]	2003	✓	✓		12
Makridis [16]	2006	✓	✓		7
Sağiroğlu [20]	2006	✓	✓		21
Nielsen [18]	2008	✓	✓		320
Alajlan [1]	2009	✓		✓	100
Cho [3]	2010	✓		✓	432
Yang [24]	2011	✓		✓	108
Pomeranz [19]	2011	✓		✓	3300
This work	2012	✓		✓	9600

Table 1: A selection of works on solving jigsaw puzzles, and whether image-based features or shape features are considered. Also, the size of the largest reconstructed puzzle (number of pieces) is given.

3. Square-Piece Jigsaw Puzzles

In several recent works, puzzles with square pieces have recently been explored [1, 3, 19, 24]. In the past, there have been several implicit assumptions for square-piece puzzles. First, it is assumed that the puzzle dimensions are known. Anchor pieces are optional in [3] and a single anchor is required in [24]. The past work assumes that the orientation of each jigsaw piece is known, and only the location of each piece in the completed puzzle is unknown. Consequently, a pair of puzzle pieces can only fit together in four different ways. We call this type of square-piece jigsaw puzzle a "Type 1" puzzle.

In the following sections, we introduce two types of square-piece puzzles where the pieces have unknown orientation. Examples of puzzles of all three types are shown in Figure 6.

3.1. Type 2: Unknown Rotation and Location

In this twist on the square-piece jigsaw puzzle, neither the location, nor the orientation, of any piece is known. This increases the complexity of the problem in several ways. First, a pair of pieces can fit together in any of 16 configurations (the second piece can be above, to the left or right of, or beneath the first piece, and the second jigsaw piece can have any of four orientations.) This is not a trivial extension; the number of possible solutions versus Type 1 is multiplied by a factor of 4^K in a K-piece puzzle. Second, the assembly problem is more complicated. An algorithm must consider both rotation and translation of pieces or components. Third, the puzzle dimensions of a rectangular puzzle are less useful because it is not known whether the completed puzzle is in portrait or landscape orientation (absent

some additional image-based inference). In Section 4.2, we introduce our tree-based reassembly algorithm that is used to solve square-piece puzzles of Types 1 and 2.

3.2. Type 3: Unknown Rotation, Known Location

In this puzzle, the global geometry and position of every jigsaw piece is known. Only the orientation of each piece is unknown. There are 4^K possible solutions. The problem amounts to determining which, of the four possible orientations for each piece, is the correct one. While this problem is the least computationally complex of the three, it leads to an elegant graph model solution and is included for completeness.

4. Solving Puzzles

There are two essential components for computationally solving a jigsaw puzzle, a measure of jigsaw piece compatibility for adjoining a pair of jigsaw pieces and a strategy for puzzle assembly. We propose contributions for each component. In Section 4.1, we propose a new jigsaw piece compatibility score called MGC. In Section 4.2 we propose a tree-based reassembly algorithm for solving puzzles of Types 1 and 2, and finally in Section 4.3 an MRF-based framework for solving Type 3 puzzles.

4.1. Measuring Pairwise Compatibility

In a correctly solved jigsaw puzzle, adjoining jigsaw pieces tend to share similar colors along their common edge. This observation motivates the dissimilarity measures in previous work that penalize intensity differences along the adjacent pixel boundaries.

While this compatibility measure largely accomplishes the goal of image content consistency across jigsaw pieces, it can fail in instances where gradients or edges occur in the neighborhood of jigsaw piece edges.

4.1.1 Mahalanobis Gradient Compatibility

We propose a measure called Mahalanobis Gradient Compatibility (MGC) that describes the local gradients near the boundary of a puzzle piece by making two improvements over the standard compatibility measure. First, we propose to penalize changes in intensity gradients, rather than penalizing changes to intensity. In other words, if a jigsaw piece has a gradient near its edge, we expect that the adjoining puzzle piece will continue the gradient. Second, rather than penalizing all deviations from a constant gradient uniformly (i.e. with Euclidean distance), we learn the covariance between the color channels and use the Mahalanobis distance. In essence, we want the boundary of two adjoining jigsaw pieces to have a similar gradient distribution to the gradient (within a jigsaw piece) on either side of the boundary.

When computing the compatibility $D_{LR}(x_i, x_j)$ of a jigsaw piece x_j on the right side of piece x_i, we find the distribution of the color gradients near the right edge of piece x_i. We define an array of gradients G_{iL} with 3 columns (one each for the red, green, and blue color channels), and P rows (where P is the pixel dimension of the jigsaw piece). G_{iL} describes the intensity changes along the right size of the jigsaw piece x_i (since it will be on the left of the pair). Entries in G_{iL} are given as:

$$G_{iL}(p, c) = x_i(p, P, c) - x_i(p, P-1, c) \tag{1}$$

The mean distribution of those gradients on the right side of jigsaw piece x_i is found as:

$$\mu_{iL}(c) = \frac{1}{P}\sum_{p=1}^{P} G_{iL}(p, c) \tag{2}$$

For each color channel, μ_{iR} is the mean difference between the final two columns of x_i. Similarly, the 3×3 covariance S_{iL} estimated from G_{iL} captures the relationship of the gradients near the edge of the jigsaw piece between the color channels. To avoid numerical problems related to the inversion of S and the inherent issues of quantized pixel values, we include nine "dummy gradients" in the calculations (e.g. [0 0 1],[1 1 1]).

At this point, we define the compatibility measure from jigsaw piece x_i to x_j as follows:

$$D_{LR}(x_i, x_j) = \sum_{p=1}^{P}(G_{ijLR}(p) - \mu_{iL})S_{iL}^{-1}(G_{ijLR}(p) - \mu_{iL})^T \tag{3}$$

where:

$G_{ijLR}(p)$ is the gradient from the right side of piece x_i to the left side of piece x_j, at row position p. Explicitly,

$$G_{ijLR}(p, c) = x_j(p, 1, c) - x_i(p, P, c) \tag{4}$$

The dissimilarity $D_{LR}(x_i, x_j)$ is not symmetric because the junction between pieces x_i and x_j is evaluated based on the distributions estimated from the x_i side of the boundary. Modified Equations, in the spirit of (1) to (4), are used to define $D_{RL}(x_j, x_i)$. Finally, the symmetric compatibility measure $C_{LR}(x_i, x_j)$ for placing x_i and x_j as left-right neighbors is:

$$C_{LR}(x_i, x_j) = D_{LR}(x_i, x_j) + D_{RL}(x_j, x_i) \tag{5}$$

These equations are appropriately modified for analyzing each of the 16 configurations of two adjacent jigsaw pieces with rotation. In practice, the dissimilarities between all pairs of jigsaw pieces for all pairwise configurations are computed. Next, the ratio is taken between it

(a) (b) RGB Putative Matches (c) MGC Putative Matches

(d) (e) RGB Putative Matches (f) MGC Putative Matches

Figure 2: Our MGC compatibility measure has better performance than simply summing color differences across the boundary (RGB). For each query jigsaw piece in the left column, the compatibility measure is used to examine all other jigsaw pieces in the puzzle for finding the match to the right side of the query piece. The top four matches with RGB SSD are shown in order in columns 2-5, and columns 6-9 show the top four results using MGC. Putative matches are shown adjacent to the query pieces to allow the reader to judge the match. Correct matches are outlined in green. Correct matches tend to have lower rank when MGC is used. Specifically, for matching (a), the RGB SSD chooses as its top match a jigsaw piece with almost the same boundary pixels, but breaks the natural curves that are intersecting the right boundary. On the other hand, MGC learns the distribution of the gradients near the right edge of (a), and the correct match (c, left) is the top ranked piece.

	P=14			P=28		
	K= 221	K= 432	K= 1064	K= 221	K= 432	K= 1064
RGB SSD	0.682	0.649	0.621	0.828	0.790	0.863
LAB SSD	0.676	0.634	0.606	0.826	0.788	0.859
MGC	**0.816**	**0.785**	**0.771**	**0.919**	**0.902**	**0.942**

Table 2: Similarity performance for Type 1 puzzles: Across a variety of puzzle sizes (K) and jigsaw piece sizes (P), the correct jigsaw matches are found for larger portion of jigsaw pieces with MGC than with other measures of jigsaw piece compatibility. Note that RGB and LAB SSD have similar performance.

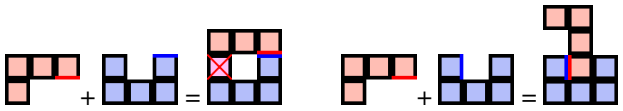
	P=14			P=28		
	K= 221	K= 432	K= 1064	K= 221	K= 432	K= 1064
RGB SSD	0.596	0.569	0.542	0.782	0.740	0.832
LAB SSD	0.591	0.554	0.525	0.780	0.738	0.827
MGC	**0.757**	**0.712**	**0.703**	**0.902**	**0.879**	**0.933**

Table 3: Similarity performance for Type 2 puzzles with pieces of unknown orientation: Across a variety of puzzle sizes (K) and jigsaw piece sizes (P), the correct jigsaw matches are found for larger portion of jigsaw pieces with MGC than with other measures of jigsaw piece compatibility.

and the second-smallest dissimilarity measure for that jigsaw piece's edge (akin to SIFT feature matching [15]). The logic behind this ratio is that a confident true match is one that is *much* better than any alternative. An unsure match tends to have other jigsaw pieces with almost the same compatibility score and ratio value near 1.0. The confidences are stored in a 3D array $S(x_i, x_j, r)$ of size $K \times K \times 16$ where r indicates the pairwise configuration. The number of counter-clockwise turns for x_j is given as $\lfloor \frac{r-1}{4} \rfloor + 1$, and $r_c = \mod(r-1, 4) + 1$ indicates whether x_j is $\{\text{above, to the right, below, to the left}\}$ of x_i.

4.1.2 Evaluation in Puzzle Assembly

In the context of assembling a jigsaw puzzle, the important question is whether a proposed measure can be used to find the correct matching jigsaw piece out of all the potential matches. Over all 20 images from [3], we compute the fraction of pieces for which the jigsaw piece having the best compatibility score is the correct match. We compare the proposed compatibility measure (MGC), as well as previously proposed dissimilarities RGB and LAB. For visual inspection, ranked potential matches are shown in Figure 2.



(a) Merge causes collision (b) Successful merge

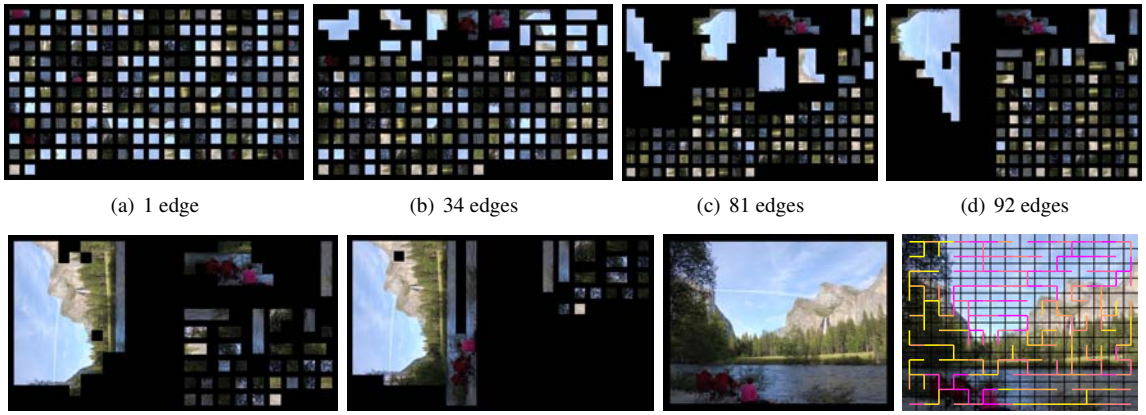
Figure 3: Example of collisions when merging together two forests of jigsaw pieces in the "constrained tree stage" by making the red and blue edges of the respective forests adjacent. In (a), a collision occurs where two jigsaw pieces overlap (red X), so merging the two forests is abandoned. In (b), the merge is successful.

Results are reported in Tables 2 and 3 for different numbers of puzzle pieces in the puzzle ($K = \{221, 432, 1064\}$), and for different size pieces (either $P = \{14, 28\}$. In all cases, the proposed MGC measure outperforms the others by a large margin. For example, for 79.0% of the jigsaw pieces with K =432 and P=28 pixels, RGB SSD retrieves the correctly matching jigsaw piece. With MGC, that figure increases to 90.2%, reducing the error rate by over 50%. The gap in performance between MGC and the other compatibility measures is greater when the resolution of each jigsaw piece is smaller. On 432 piece puzzles, our MGC measure achieves 90.2% and surpasses the predictive dissimilarity of [19] (86% accuracy) and the LAB dissimilarity used by [3] (79%). This is a significant improvement, with a 29% reduction in errors over [19].

4.2. Tree-Based Reassembly for Types 1 and 2

In this section, we introduce our greedy assembly algorithm for square-piece puzzles of unknown orientation (Types 1 and 2). The algorithm is inspired by Kruskal's Algorithm [14] for finding a minimal spanning tree (MST) of a graph $G = (V_G, E_G)$.

The puzzle assembly problem emits a graph where each jigsaw piece is a vertex, and edge weights (from $S(x_i, x_j, r)$) correspond to the compatibilities (i.e. matching costs) between pairs of pieces. Each graph edge also has an associated geometric configuration r between the pair of



(a) 1 edge (b) 34 edges (c) 81 edges (d) 92 edges

(e) 156 edges (f) 174 edges (g) Assembled Puzzle (h) Spanning Tree

Figure 4: Puzzle assembly begins with each jigsaw piece as its own forest. Forests are merged (by combining and rotating, if necessary) to create assembled subclusters according to the compatibility score that we introduce (MGC), until the puzzle is assembled (a)-(g). The spanning tree in (h) shows the final representation of the solution as well as the order in which edges were added, from early (magenta) to later (yellow). This example has 192 jigsaw pieces.

jigsaw pieces. The MST of this graph would include every jigsaw piece, and certainly the MST is the cheapest possible configuration that could be used to assemble the pieces into a single connected component. However, there is a problem. Nothing prevents the MST from being a graph that results in an assembled puzzle that overlaps onto itself (e.g. if edges of the MST indicate that two different pieces should each be positioned to the right of a third piece). Therefore, we desire the MST that is constrained to meet our geometric requirement that the assembled puzzle should be flat (pieces should not overlap).

Efficient methods for finding the MST have been discovered, but virtually any constraint to the problem results in a NP-hard variant. Specifically, constraining the degree of vertexes in the MST results in an NP-hard problem [8]. The geometric requirements above at least constrain the problem by that much, as the flat assembly requires that no vertex have degree greater than four (in addition to the tighter constraints on edge and corner pieces). Therefore, our MST problem with geometric constraints is NP-hard as well, and we propose a heuristic based on Kruskal's algorithm.

As a review, Kruskal's algorithm for finding the MST begins by considering each vertex in V as a separate forest (i.e. a forest is a vertex subset). From the set E of edges, the minimum weight edge is found. If the vertexes associated with that edge belong to separate forests, they are joined into a single forest. Otherwise (i.e. the edge forms a loop in one forest), the edge is discarded. The algorithm terminates when all vertexes belong to the same forest, and the edge set of the MST is the collection of non-discarded edges.

Our tree-based reassembly algorithm has three stages:

1. The constrained tree stage: In this stage, we perform a constrained version of Kruskal's algorithm to find a tree in E that constructs a flat puzzle assembly. Each jigsaw piece begins as its own forest in upright (non-rotated) orientation, and forests record the relative spatial locations of the member vertexes (jigsaw pieces) as well as the absolute rotation to apply to each jigsaw piece. Entries of E are examined, and the lowest cost edge e_{min} is found and removed from the set of remaining edges. If the vertexes of e_{min} belong to the same forest, e_{min} is discarded because otherwise a loop would be formed. If e_{min} passes that test, the forests joined by e_{min} are merged according to the geometric relationship associated r with e_{min}. Merging the forests include updating the absolute rotations for each jigsaw piece. If, in merging the forests, two jigsaw pieces occupy the same position, then a collision has occurred and e_{min} is discarded without merging the forests (see Figure 3). Otherwise, e_{min} is added to the set of edges in the tree. The procedure is described by Figure 4, which shows a jigsaw puzzle in various stages of assembly.

2. Trimming: Occasionally, the tree resulting from stage 1 does not fit neatly into a rectangular frame. If the dimensions (number of jigsaw pieces on each edge) of the puzzle are known, then the assembled tree is trimmed. Trimming is performed by finding the position of the frame that trims off the fewest pieces. Trimming results in a single assembled forest (those pieces within the frame) and the trimmed pieces (which are returned to the set of candidate piece forests). When orientation is unknown, the trimming procedure must consider both orientations of the frame.

3. Filling: After trimming, the puzzle frame can have unoccupied holes. Holes are filled by order of the number of occupied adjacent neighbors. For each hole, the candidate piece with a given rotation is selection that has the minimum total dissimilarity score across all neighbors. We enforce the requirement that pieces can only appear once in the assembled puzzle; so if the correct match for a given hole is elsewhere, then all available choices to fill a hole may be poor. Figure 5 shows an example result of the trimming and filling steps.

(a) Assembled Puzzle Tree (b) After Trimming (c) After Filling

Figure 5: Our constrained tree stage does not enforce a specific shape or dimensions to an assembled puzzle (a), so pieces (especially when clipped blocks are present) can extend beyond the bounds of the image. By trimming the assembled puzzle to known dimensions (b) and filling (c) the reconstructed puzzle can often be improved. This puzzle has 1176 pieces of unknown orientation.

4.3. An MRF for Solving Type 3 Puzzles

For puzzles with in-place rotated pieces, the task is to rotate each jigsaw piece to its original orientation. A natural function to minimize is the total sum of the cost across the boundaries of any two pieces. Let X_i represent one of the four possible orientations of the i^{th} jigsaw piece, and $\mathbf{X}$ be the set of all X_i. Thus, we seek to minimize:

$$E(\mathbf{X}) = \sum_{(X_i, X_j) \in N} \Phi(X_i, X_j) \tag{6}$$

over the possible labels of X_i, where N indicates the set of sets of neighbors in the puzzle. This equation is a pairwise Markov Random Field. The pairwise cost term $\Phi(X_i, X_j)$ is the cost incurred when two adjacent pieces have a given orientation. This term has 16 elements, populated by the entries of $S(x_i, x_j, r)$ with each of the 16 possible pairwise configurations r. Interestingly, there is no unary term in the model because we have no way of predicting the orientation from a small image block alone. Also, the model is non-submodular because we there is no reason to believe that neighboring blocks should be mis-orientated in the same way. Consequently, finding the global minimum is intractable, so we use approximate inference (TRW-S [23]) to find a good labeling for the nodes.

5. Experiments

We apply our puzzle assembly algorithms to the set of 20 images from [3]. Several aspects of the problem are explored. We vary the number of pieces in the puzzle, including the sizes of 221, 432 from [3]. We also explore a larger puzzle with 1064 pieces. Also, we vary the size of each jigsaw piece from the 28×28 pixels from [3], and a more challenging 14×14 pixels. We stress that these 20 images were not used at all in the development of the algorithm. Computational cost of MGC scales with the square of the number of pieces. Solving a 432 piece Type 2 puzzle requires about 100 seconds on a modern PC versus about 40 seconds when RGB SSD is used. On Type 1 puzzles with 432 pieces, we compare our results with Cho *et al*. [3] and Pomeranz *et al*. [19].

	Direct	Neighbor	Comp.	Perfect
Cho *et al*. [3]	0.10	0.55	-	0
Pomeranz *et al*. [19]	0.94	**0.95**	-	**13**
Tree-based + LAB SSD	0.814	0.892	0.853	8
Tree-based + MGC	**0.953**	**0.951**	**0.953**	12

Table 4: Performance comparison for Type 1 puzzle assembly (oriented jigsaw pieces) for puzzles with 432 pieces.

Overall, we find that our jigsaw piece compatibility score provides a significant improvement in puzzle assembly performance, and occasionally, perfect reconstruction of jigsaw puzzles is achieved. We evaluate an assembled puzzle with the following measures, the first two from [3]:

Direct comparison: measures the fraction of pieces in the assembled puzzle that are in the correct absolute position.

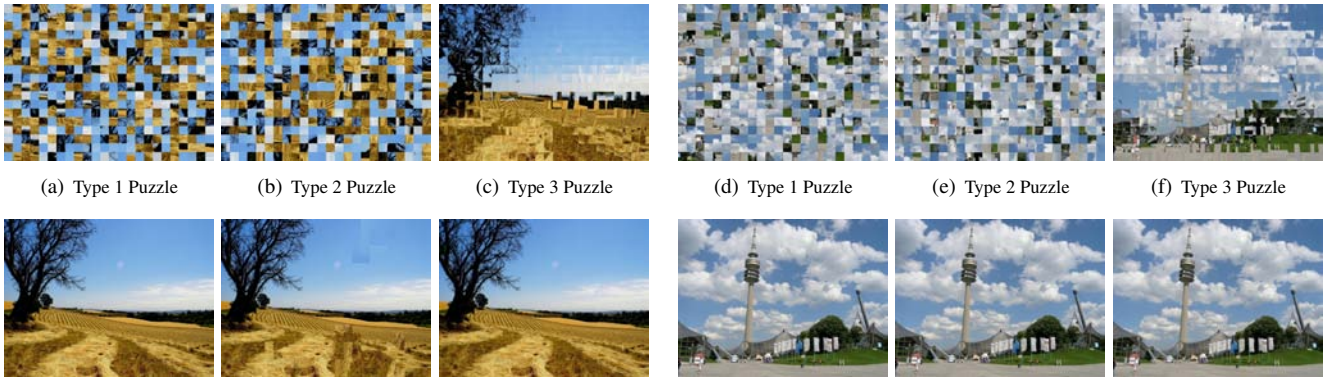
Neighbor comparison: measures the fraction of pairwise piece adjacencies that are correct. For a jigsaw puzzle with $m \times n$ jigsaw pieces, there are a total of $2mn - m - n$ possible adjacencies.

Largest Component: measures the fraction of jigsaw pieces in the largest connected component of jigsaw pieces that have correct pairwise adjacencies with other jigsaw pieces in the component. Essentially, this measures the size of the largest correct portion of the assembled jigsaw, without regard to its position in the assembled puzzle.

Perfect Reconstruction: a binary indication of whether every piece of the assembled puzzle is in the correct position with correct rotation.

Figure 6 shows the visual results and Tables 4 to 6 report reassembly accuracies for various experiments.

Type 1 Puzzles: Table 4 reports our results for solving the 432 jigsaw piece puzzle. We exactly solve 12 of the 20 puzzles, and have high accuracy (95.3%) under direct comparison. Our performance is at least comparable to the state of the art ([19]). The effectiveness of the MGC dissimilarity measure is shown in two ways. First, Table 4 shows that the performance of our tree-based assembly is improved in all categories when MGC is used versus the sum-of-squared differences in LAB (Table 5 shows a similar result for Type 2 puzzles). Second, we use the method of [19], substituting our MGC measure for their predictive compatibility with no other changes. We used 10 repetitions and selected the rep-



(a) Type 1 Puzzle (b) Type 2 Puzzle (c) Type 3 Puzzle (d) Type 1 Puzzle (e) Type 2 Puzzle (f) Type 3 Puzzle

(g) Tree-Based Solution (h) Tree-Based Solution (i) MFR Solution (j) Tree-Based Solution (k) Tree-Based Solution (l) MRF Solution

Figure 6: Results from our algorithms on the types of square-piece puzzles are shown. Each puzzle has 432 pieces, each of size 28×28. Type 1 puzzles scramble the location of the pieces, Type 3 puzzles scramble the orientation, and both location and orientation of pieces must be found for Type 2 puzzles. These results are all perfect reassemblies, except for the solution shown in (h), where there are some mistakes in both the wheat and the sky. In general, errors involve regions that are extremely uniform, in texture (e.g. an area of uniform foliage) or gradient (e.g. smooth sky, or clipped pixels). When the assembly algorithm runs out of "easy choices" it must make more difficult choices, and mistakes can occur.

	Direct	Neighbor	Comp.	Perfect
Tree-based + LAB SSD	0.423	0.682	0.636	1
Tree-based + MGC	**0.822**	**0.904**	**0.889**	**9**

Table 5: Our MGC similarity leads to better puzzle assembly versus traditional dissimilarities. This table shows accuracy at assembling Type 2 puzzles with pieces with unknown location and orientation. The puzzles have 432 pieces, each with 28×28 pixels.

	P=14			P=28		
	K= 221	K= 432	K= 1064	K= 221	K= 432	K= 1064
Direct	0.333	0.377	0.294	0.803	0.822	0.906
Neighbor	0.603	0.626	0.568	0.885	0.904	0.936
Component	0.525	0.551	0.534	0.882	0.889	0.936
Perfect	0	0	0	8	9	14

Table 6: Solving various sizes of Type 2 puzzles (pieces with unknown location and orientation). This table reports accuracy scores for assembled jigsaw puzzles with our tree-based reassembly algorithm different numbers of pieces (K) and different jigsaw piece sizes (P, in pixels square).

etition with the highest "best buddy" score. Due to the random component of their algorithm, the overall results were lower than reported in Table 4, with a direct score of 0.856. However, when MGC was used, higher scores resulted for the direct score (0.875, an increase by an absolute 1.9%) and in the best buddy score (by an absolute 2.7%), although with a decrease in the neighbor score of 1.1%.

Type 2 Puzzles: Tables 5 and 6 report the assembly accuracy for puzzles where neither the position nor orientation of pieces are known. As expected, due to the increased problem complexity, the overall reconstruction accuracies suffer somewhat compared to the Type 1 puzzles. However, it is somewhat surprising that nine puzzles are still perfectly reconstructed.[1] As there are no other algorithms that tackle this problem, we explore the results under different parameter settings in Table 6. Even for small pieces (14×14), a large portion of pieces are correctly positioned. Surprisingly, performance is not very sensitive to the number of pieces in the puzzle, and even puzzles with over 1000 pieces are often solved perfectly. Perhaps this is because increasing the number of puzzle pieces does not radically increase the number of pieces that can be confused as a possible match (on this point, see also Table 3).

Finally, we note that sometimes the largest correct connected component score is much larger than the direct matching comparison score. This can be a result of early mistakes in the assembly. Because the assembly algorithm has no means for backtracking or correcting mistakes (other than the trim-fill steps), these mistakes can degrade the direct matching score.

Type 3 Puzzles: Finding the correct orientation of jigsaw pieces with known location using the proposed MRF is highly reliable. Across the 20 images, the orientation accuracy is 97.2% when considering puzzles with 432 pieces each with 28×28 pixels. The main reason for failure is when blocks have a constant value (i.e. clipped), which is true for 3.3% of the blocks. The algorithm executes quickly, requiring about two seconds for computing the pairwise compatibility terms, and negligible time for inference.

Mixed-Bag Puzzles: We emphasize that the initial tree stage of our reassembly algorithm requires no information about the dimensions of the resulting puzzle, as opposed to prior work [3, 19]. This allows us to assemble puzzles even with extra jigsaw pieces that the other methods may have difficulty handling. To illustrate this point, we combine the jigsaw pieces from two or three puzzles and then perform only the first step of our reassembly (stage 1). Note that no information other than the pieces is given to our algorithm, so it knows neither the number of puzzles that are

[1] All reconstructions for Type 1 and 2 puzzles with 432 pieces of size 28×28 can be found at: http://tinyurl.com/7udcpps

(a) 2 Mixed Puzzles (1200 pieces) (b) Tree-Based Solution of (a) (c) 3 Mixed Puzzles (1800 pieces) (d) Tree-Based Solution of (c)

Figure 7: Building Mixed Puzzles. In (a) and (c), the jigsaw pieces from 2 and 3 puzzles, respectively, are mixed together. Each puzzle has 600 jigsaw pieces, each jigsaw piece is 28×28, and is a Type 2 puzzle having rotated jigsaw pieces. Our tree-based reassembly is still able to assemble the puzzles, even with no information about the number of puzzles mixed together, or their dimensions. Perfect reassembly is achieved in (b), but a few mistakes occur in two of the component puzzles in (d), some of which could be fixed with our trimming step.

present, nor the number of pieces that are in a given puzzle. The results (Figure 7) show not only that the algorithm can properly group the jigsaw pieces into the component puzzles, but that the reassembly is also effective.

Finally, in an effort to push our proposed system to the limit, we attempt a Type 2 jigsaw puzzle with 9600 jigsaw pieces. Perfect reconstruction is achieved (Figure 1), probably aided in part by the image content that contains a variety of textures and colors. Processing required 23.5 hours on a modern PC.

6. Conclusion

In summary, this paper introduces a new class of square-piece jigsaw puzzles: those having pieces with unknown orientations. For solving jigsaw puzzles where neither the piece location nor orientation is known, we propose a tree-based reassembly that greedily merges components while respecting the geometric constraints of the problem. For solving puzzles where location is known, we propose a pairwise MRF where each node represents a jigsaw piece's orientation.

We also propose a new measure (MGC) for quantifying the compatibility of a potential jigsaw piece matches based on image analysis alone. We show that MGC has superior performance to previously proposed measures of jigsaw piece compatibility.

Using MGC and the proposed assembly strategies, we achieve state-of-the-art results at the task of assembling jigsaw puzzles with unknown jigsaw piece orientations. In particular, we have achieved perfect reconstruction of jigsaw puzzles containing up to 9600 pieces, the largest automatically solved jigsaw puzzle to date.

References

[1] N. Alajlan. Solving square jigsaw puzzles using dynamic programming and the hungarian procedure. *Amer. Journ. Applied Sciences*, 2009.

[2] S. Cao, H. Liu, and S. Yan. Automated assembly of shredded pieces from multiple photos. In *Proc. ICME*, 2010.

[3] T. S. Cho, S. Avidan, and W. T. Freeman. A probabilistic image jigsaw puzzle solver. In *Proc. CVPR*, 2010.

[4] M. G. Chung, M. M. Fleck, and D. A. Forsyth. Jigsaw puzzle solver using shape and color. In *Proc. ICSP*, 1998.

[5] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based impainting. In *Proc. CVPR*, 2002.

[6] E. Demaine and M. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23, 2007.

[7] H. Freeman and L. Gardner. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE. Trans. on Electronic Computers*, 1964.

[8] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman, 1979.

[9] D. Goldberg, C. Malon, and M. Bern. A global approach to solution of jigsaw puzzles. *Symposium on Computational Geometry*, 2002.

[10] K. Hori, M. Imai, and T. Ogasawara. Joint detection for potshards of broken earthware. In *Proc. CVPR*, 1999.

[11] E. Justino, L. Oliveria, and C. Freitas. Reconstructing shredded documents through feature matching. *Forensic Science International*, 2006.

[12] W. Kong and B. Kimia. On solving 2d and 3d puzzles using curve matching. In *Proc. CVPR*, 2001.

[13] D. A. Kosiba, P. M. Devaux, S. Balasubramanian, T. L. Gandhi, and K. Kasturi. An automatic jigsaw puzzle solver. In *Proc. ICPR*, 1994.

[14] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proc. Amer. Math. Soc.*, 1956.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[16] M. Makridis and N. Papamarkos. A new technique for solving a jigsaw puzzle. In *Image Processing, 2006 IEEE International Conference on*, 2006.

[17] M. Marques and C. Freitas. Reconstructing strip-shredded documents using color as feature matching. *Proc. ACM Symposium on Applied Computing*, 2009.

[18] T. R. Nielsen, P. Drewsen, and K. Hansen. Solving jigsaw puzzles using image features. *Pattern Recogn. Lett.*, 29, October 2008.

[19] D. Pomeranz, M. Shemesh, and O. Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *Proc. CVPR*, 2011.

[20] M. Sağiroğlu and A. Erçil. A texture based matching approach for automated assembly of puzzles. In *Proc. ICPR*, 2006.

[21] R. Webster, P. LaFollette, and R. Stafford. Isthmus critical points for solving jigsaw puzzles in computer vision. In *Proc. IEEE Trans. Systems, Man and Cybernetics*, 1991.

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

[23] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *CVPR*, 2008.

[24] X. Yang, N. Adluru, and L. Latecki. Particle filter with state permutations for solving image jigsaw puzzles. In *Proc. CVPR*, 2011.

[25] F.-H. Yao and G.-F. Shao. A shape and image merging technique to solve jigsaw puzzles. *Pattern Recognition Letters*, 2003.