EG UK Computer Graphics & Visual Computing (2019)
G. K. L. Tam and J. C. Roberts (Editors)

Robust and Flexible Puzzle Solving with Corner-based Cycle Consistent Correspondences

T. Wang[†1], K. Vladimirov[1], S. Goh[1], Y.-K. Lai[2], X. Xie[1] and G. K.L. Tam[1]

[1]Department of Computer Science, Swansea University, UK
[2]School of Computer Science and Informatics, Cardiff University, UK

Abstract

Solving jigsaw puzzles is a classic problem in computer vision with various applications. Over the past decades, many useful approaches have been introduced. Most existing works use edge-wise similarity measures for assembling puzzles with square pieces of the same size, and recent work innovates to use the loop constraint to improve efficiency and accuracy. We observe that most existing techniques cannot be easily extended to puzzles with rectangular pieces of arbitrary sizes, and no existing loop constraints can be used to model such challenging scenarios. In this paper, we propose a new corner-wise matching approach, modelled using the MatchLift framework to solve square puzzles with cycle consistency. We further show one exciting example illustrating how puzzles with rectangular pieces of arbitrary sizes would be solved by our technique.

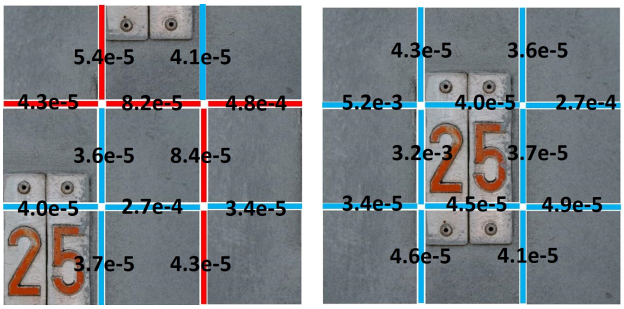
CCS Concepts

• ***Computing methodologies*** → *Image comparison;* • ***Applied computing*** → *Image composition;* • ***Theory of computation*** → *Semidefinite programming;*

1. Introduction

Solving jigsaw puzzles is a classic problem in computer vision. In 1964, [FG64] introduced the first algorithm for matching puzzle pieces. Since then, approaches have focused on using shape and colour information [WLS91, KDB*94] for puzzle solving. Puzzle solving has great applications in many research areas, like forensics [RRCL13, LZZC14] and archaeology [PPE*02, HFG*06], to recover documents or art works from small fragments.

Techniques to solve a jigsaw puzzle consist of two steps: i) computing constraints (e.g. colour-based similarity between puzzle pieces) and ii) assembling puzzle pieces via some optimisation technique. Notable examples include [Gal12] which introduces the novel Mahalanobis Gradient Compatibility (MGC) measure to compute the similarity between puzzle pieces, and a minimal spanning tree (MST) [BKJ56] approach to assemble similar pieces in a greedy manner (Figure 1a). Based on colour space normalisation, [CAF10] proposes a global approach to assembling similar puzzle pieces. Their compatibility measure is based on a thin region (often 1 column of pixels of the edge) of each piece. These two measures are frequently used in subsequent works for puzzle solving [SHC14, PT15, SMHC16, Zan16]. More recently, the loop constraint [SHC14, YRA16, SHC18] was proposed to enforce cycle consistency when pieces are matched, and good performance was demonstrated.



(a) *Assembled result by [Gal12]. Red edges show incorrect matching pairs in greedy assembly.* **(b)** *Correct assembled result obtained by our proposed technique.*

Figure 1: *Comparison between [Gal12] and our proposed technique on square puzzle solving. The number on each edge shows the MGC similarity score between a pair of pieces.*

From the literature, we made two observations. First, much of the previous work focuses on puzzles with square pieces of the same size but they may not apply to puzzle solving with rectangle pieces of arbitrary sizes (Figure 2). The problem of solving such puzzles is arguably harder with a larger search space because of the arbitrary edge lengths. It challenges most of the existing edge-wise similarity measures. Second, even though the loop constraint is power-

† Chairman Eurographics Publications Board

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

DOI: 10.2312/cgvc.20191266

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

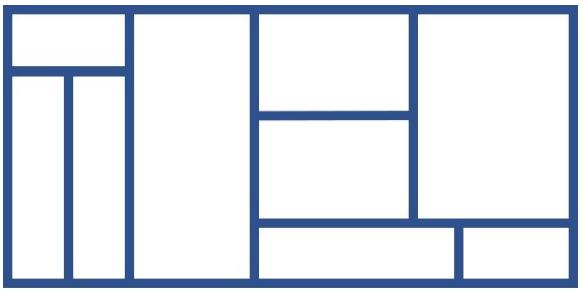


Figure 2: *Rectangular pieces with arbitrary sizes are challenging for edge-wise similarity measures and assembly techniques.*

ful, we observe that many of the existing works assume some form of input regularity, and either build loops explicitly from square pieces which would be slow, or use the loop constraint by casting puzzle assembly in a sophisticated optimisation. These techniques however are not easy to extend to arbitrarily shaped puzzle pieces. These observations motivate the following research question:

- Can we develop a more flexible technique that can leverage the strength of loop constraint for puzzle solving of rectangle pieces of arbitrary sizes?

Instead of using the whole edges of pieces for puzzle assembly like in existing work, we investigate if corners of puzzle pieces can be used. Next we cast the problem of discovering loops in possible puzzle pieces as a cycle consistent correspondence problem [CGH14]. Once we identify good pairwise corner-wise correspondences, we adapt minimum spanning tree [Gal12] for puzzle solving. Our results show that the approach can improve the performance of [Gal12] which uses MGC alone. Our contributions are:

- We innovate to use corner-wise correspondences for the puzzle solving task — we demonstrate its usefulness for square puzzle solving, and illustrate one example of how it can be adopted for rectangular pieces of arbitrary sizes.
- We propose a loop discovery technique for puzzle solving by modelling it as a cycle consistent correspondence problem, which allows to use the MatchLift framework [CGH14] for puzzle solving.

We discuss related work in Section 2. After providing an overview of our method in Section 3, we show how we model corner-wise matching in the MatchLift framework [CGH14] for square puzzle solving in Sections 4 and 5. We evaluate our method in Section 6. Section 7 illustrates one example how puzzle with rectangular pieces of arbitrary sizes can be solved. We discuss limitations and future work in Section 8, and conclude in Section 9.

2. Related Work

We discuss existing puzzle solving techniques in three sections. Section 2.1 discusses similarity measures for piece matching. Section 2.2 summarises assembly techniques for puzzle solving. Finally, Section 2.3 discusses correspondence techniques that use cycle consistency which inspires the technique in this paper.

2.1. Similarity Measures for Piece Matching

Pairwise similarity measures of puzzle pieces have been widely used for puzzle solving. [Gal12, MWD13, SHC14, Zan16] use Mahalanobis Gradient Compatibility (MGC) to compute the pairwise similarity of puzzle pieces. MGC is a dissimilarity metric. It uses gradients to determine the boundary similarity of puzzle pieces. [CAF10, YAL11, MWD13, SHC14, DN16, Zan16] use another dissimilarity measure SSD (Sum of Squared Differences) as pairwise measure. SSD sums the squared distances on pixels along the boundary to determine colour dissimilarity of puzzle pieces. From our experiments when image resolution is low or image content is not distinctive, dissimilarity-based approach can provide incorrect results (as shown in Figure 1a). [PSB11, PT15] use a probabilistic approach to determining the similarity between puzzle pieces. The methods use colour and size constraints on puzzle pieces. These colour similarity measures (e.g. SSD, MGC) are not global constraints and may lead to inconsistent results.

2.2. Assembling Puzzle Pieces

Puzzle solving is a hard problem due to the large search space. Many works use greedy approaches for puzzle assembly from pieces [PSB11, Gal12, PT15, Zan16, SMHC16]. In general, a greedy approach uses designated constraints (in terms of placement of pieces and similarity measure) to find suitable results. They often begin from a small confident region and gradually expand it by accepting new pieces. Since adjacent pieces are locally consistent, these techniques often do not refine or rectify incorrect assembly results. The final assembled results may not be globally consistent.

Greedy approaches with loop constraints show good performance. [SHC14] introduces a novel 4-piece-loop constraint for finding small cycles. Each cycle can be considered as an assembled region that contains 4 puzzle pieces. They first compute all pairwise MGC scores as the similarity measure between puzzle pieces. Based on the MGC scores they find small cycles. Next, they merge small cycles to build larger cycles, which form larger assembled regions. [SHC18] builds on the idea but merges small cycles in a hierarchical manner. When incorrect pieces are matched, loop constraints provide a mechanism to examine piece neighbours and remove inconsistent ones. This improves puzzle assembling results. [YRA16] models puzzle assembly as a linear programming (LP) problem. They iteratively optimise pieces and increase the size of assembled results. Each iteration of LP optimisation can be considered as a general loop constraint optimisation. It shows that LP can perform better than [SHC14].

Global approaches [WK01, GMB04, CAF10, MWD13] assemble puzzles by optimising a global objective function. [ATG12] proposed to use quadratic programming (QP) to globally optimise piece placement. [AACT*16] shows QP performs good assembled results even when there are missing puzzle pieces from the input.

Though loop constraints have been used in the literature, we observe that these techniques are mostly tailored for solving square puzzles only. They are not flexible to extend and handle rectangular pieces of arbitrary sizes. This inspires us to tackle this challenge, and the use of corners and cycle consistency for puzzle solving.

2.3. Cycle Consistency

Cycle consistency is a useful constraint for matching problems of multiple objects. For example, given three objects A, B and C, cycle consistency enforces matchings from A to B, and from

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

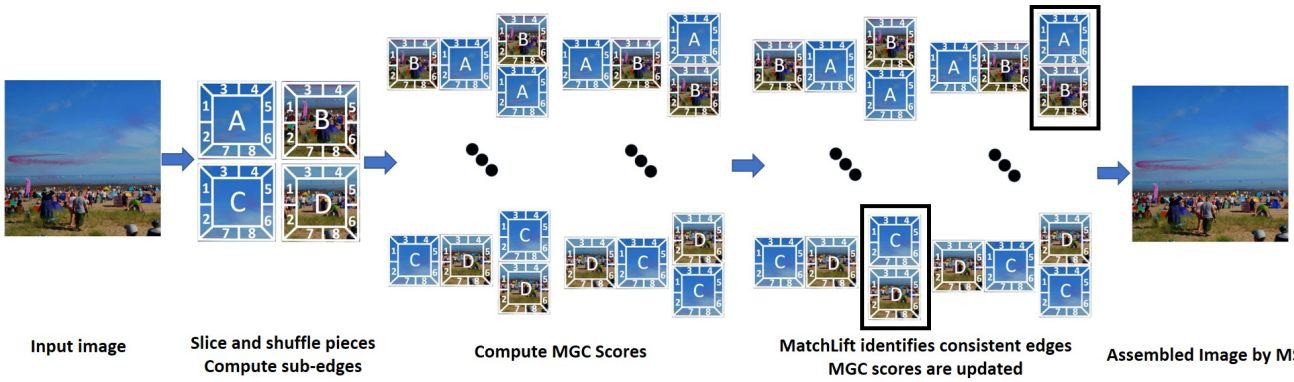


Figure 3: *Pipeline of our method. In this example, we slice the input image into four pieces for simpler illustration. For each piece, we break each edge into two sub-edges. We use sub-edges to represent four corners on each square piece. Next, we use MGC to determine the similarities between all possible pairs of sub-edge based corners. We then treat these pairs as correspondences, use MatchLift to identify cycle-consistent correspondences and update their MGC scores. Finally we apply minimum spanning tree [Gal12] on the updated MGC scores for puzzle assembly.*

B to C such that C to A is also meaningful. In recent years, [ZKP10, RSSS11, ZLXYE15, ZPIE17, WZD18] have successfully applied cycle consistency to obtain globally consistent matchings in the image domain. [HG13] proposed a semi-definite programming (SDP) approach for solving the cycle consistent matching problem in the 3D domain. It shows that SDP can provide up to 50% error tolerance of pairwise matchings between input objects. Building on [HG13], [CGH14] introduces MatchLift for solving globally consistent matching in a general setting, with a tolerance rate of $1-\Theta(\log^2 n/\sqrt{n})$ to random outliers.

Inspired by [CGH14], we suggest that the puzzle solving can be cast in the MatchLift framework, which helps discover loop correspondences. To do so, we propose to use *corners* of pieces as the basic unit while most existing techniques use edges (e.g. MGC). This provides a flexible framework for solving both square and rectangular puzzles of arbitrary sizes. The high tolerance to input errors of our method (due to the MatchLift framework) helps improve the precision of MGC matchings, making our method more robust for challenging inputs.

3. Method Overview

Figure 3 shows the pipeline of our technique for solving puzzles. The input image to our method is first sliced and shuffled into (e.g. square) puzzle pieces. Our method further breaks each puzzle piece into four (2-by-2) corners, by subdividing each edge of a piece into 2 sub-edges (Figure 4, Section 4.2). Then we use MGC to compute the similarities between all possible pairs of sub-edges. We treat these pairs as correspondences. Section 4.3 presents how we use MatchLift to identify cycle consistent correspondences. Section 5 discusses how we refine the respective MGC scores of correspondences identified by MatchLift, and finally solve the puzzle using minimum spanning tree. Further, in Section 7, we discuss how we extend our technique to solve puzzles consisting of rectangular pieces of arbitrary sizes. In this paper, we assume all pieces have known orientation with unknown position (so called Type I puzzle problem [Gal12]).

4. MatchLift and Puzzle Solving

MatchLift [CGH14] is a convex optimisation technique to find cycle-consistent correspondences from a set of noisy input. For example, in 3D reconstruction of a chair, it is critical to estimate depth by computing reliable point-to-point correspondences across a collection of images of the same chair from different views. Correspondences can be established by SIFT key point descriptors, but inconsistent correspondences cannot be avoided. [CGH14] can identify cycle-consistent correspondences across multiple images. The idea is to encode all pairwise correspondences between images in a permutation matrix. Then it applies semi-definite programming with relaxed binary constraint and sparsity to enforce cycle consistency. Due to page limit, we would refer readers to [CGH14] for the mathematical details.

Here, we use MatchLift to find reliable cycle-consistent correspondences for puzzle solving. In our modelling, we treat each corner as an object (similar to one of the images in the chair reconstruction example), and pairwise matching of sub-edges as correspondence between corners (similar to point-to-point correspondences between images). Our contribution is to model piece matching in the puzzle problem as a corner-wise cycle-consistent correspondence problem in the MatchLift framework. We show that it can handle square and rectangular puzzles of arbitrary sizes.

4.1. Computing MGC Scores

Our technique builds on MGC scores [Gal12] which we briefly discuss here. MGC is a gradient-based compatibility measurement between puzzle piece edges (all pieces must have the same size). For an edge, it first defines a matrix of colour distribution with dimensions $px \times 3$, where px is the number of pixels of a piece edge with 3 colour channels (red, green, blue). For a pair of edges on two square pieces, MGC determines a compatibility score by computing Mahalanobis distance between their colour distribution matrices.

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

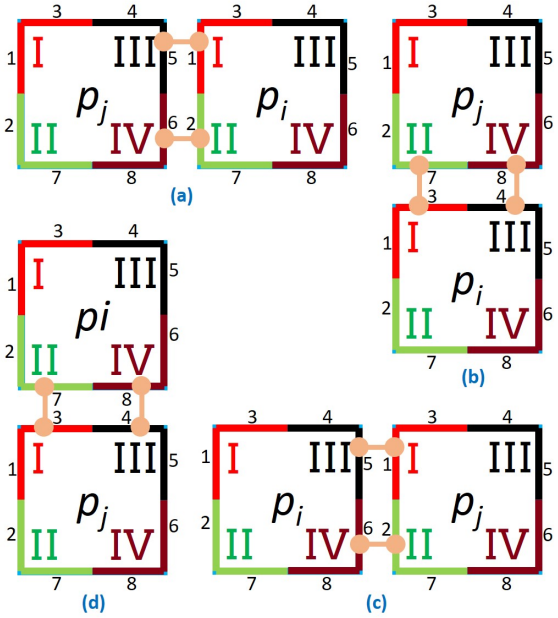


Figure 4: *An ordering scheme to generate correspondences of sub-edges between puzzle pieces.*

4.2. Modelling Puzzle Pieces by Corners

Next we introduce the sub-edge. Let $P = \{p_1, ..., p_n\}$ be the set of all input puzzle pieces. For each puzzle piece $p_i \in P$, we break each edge into two sub-edges. There are four edges and in total eight sub-edges per piece. We label each sub-edge in a fixed order as shown in Figure 4. We further define $e_a(p_i)$ as an operator to return the sub-edge from p_i where $1 \leq a \leq 8$. For each pair of pieces p_i, p_j, we consider eight possible correspondences associated to the sub-edges of p_i, p_j based on an ordering scheme as shown in Figures 4 (a)-(d). Beginning from the left two sub-edges of p_i and the right two sub-edges of p_j, we define correspondences $c_k = (e_1(p_i), e_5(p_j))$ and $c_l = (e_2(p_i), e_6(p_j))$ (shown as tan coloured correspondences in Figure 4 (a)). Following the ordering scheme, we can define eight correspondences for p_i and p_j, and we repeat the procedure for all pairs of pieces to compute the set of input correspondences C. For each correspondence $c_k \in C$, where $1 \leq k \leq 8n(n-1)$, we define the similarity between the two sub-edges using MGC score. MGC scores have a large range (the maximum value might be ten thousand times larger than minimum). We normalise them into [0,1]. After normalisation, scores close to 1 mean two sub-edges are highly similar. Take c_k for example, our measure is thus $sim(c_k) = MGC_{normalised}(e_1(p_i), e_5(p_j))$. Other cases can be similarly defined.

Next, we define the corners of pieces as units for puzzle solving. We use $p_i^\alpha \in P$ to indicate a corner on a piece, where $\alpha \in \{I, II, III, IV\}$, as shown in Figure 4. For example, assuming there are ten pieces in a puzzle, the corner II on the tenth piece is labelled as p_{10}^{II}, and it contains two sub-edges $e_2(p_{10})$, $e_7(p_{10})$. We define $v(p_i^\alpha, p_j^\beta)$, where $\alpha, \beta \in \{I, II, III, IV\}$, as the corner-wise similarity score of two corners p_i^α and p_j^β. Since the orientation of input pieces is known (Type I puzzle), some corners are incompatible

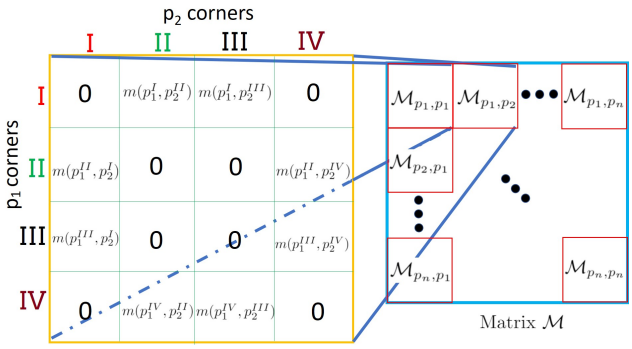


Figure 5: *Example of matrix $\mathcal{M}$ with n puzzle pieces. All diagonal red blocks are self matching between a pair of the same piece. Non-diagonal red blocks contain corner-wise similarity measures of pieces.*

with each other such as p_i^I and p_j^I. For incompatible corners, we set $v(p_i^\alpha, p_j^\beta) = 0$. $v(p_i^\alpha, p_j^\beta)$ can be summarised as

$$v(p_i^\alpha, p_j^\beta) = \begin{cases} sim(c_k), & \text{if } c_k \in C \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

We encode the corner-wise similarity in a block matrix as the input of MatchLift. Let $\mathcal{M}_{p_i,p_j}$ be a 4×4 matrix, which is shown in Figure 5 (left). Given a puzzle with n pieces, we can encode all $\mathcal{M}_{p_i,p_j}$ blocks into a piece-wise similarity matrix $\mathcal{M}$ of dimension $4n \times 4n$ (i.e. $\mathcal{M}_{p_i,p_j} \subset \mathcal{M}$ in Figure 5 (right)). It is arranged such that the non-diagonal block $\mathcal{M}_{p_i,p_j} \subset \mathcal{M}$, where $i \neq j$ contains all corner-wise MGC scores $m(p_i^\alpha, p_j^\beta) \in \mathcal{M}_{p_i,p_j}$ between pieces p_i and p_j. The diagonal elements $m(p_i^\alpha, p_i^\alpha) \in \mathcal{M}$ represent self-matching between a pair of the same corner p_i^α and p_i^α. We set those elements as 1. In summary, we define the element $m(p_i^\alpha, p_i^\alpha) \in \mathcal{M}$ as

$$m(p_i^\alpha, p_j^\beta) = \begin{cases} v(p_i^\alpha, p_j^\beta), & i \neq j,\ v(p_i^\alpha, p_j^\beta) \geq t_1 \\ 1, & i = j,\ \alpha = \beta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and t_1 is a user defined threshold to accept correspondences with good MGC scores. $\mathcal{M}_{p_i,p_j}$ represents partial matching whilst $\mathcal{M}$ represents the full matching of input pieces. This matrix can then be optimised using MatchLift framework, using SDP [CGH14].

4.3. Corners and Cycle Consistency

Our intuition of using corners in the MatchLift framework to handle square puzzle pieces is that it can find two-cycle (direct correspondence, white) and four-cycle consistent correspondences (yellow) as shown in Figure 6. We mark the positions of corners to indicate the matching between sub-edges. For example, in Figure 6 each bottom-right corner contains sub-edges 6 and 8 and each bottom-left corner contains sub-edges 2 and 7. If there is a matching between sub-edges 6 and 2, then it means corner IV and corner II have been matched. Since we have cycle consistency as a constraint there will not be displaced-matching, such as sub-edge 6 will not match to sub-edge 1.

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

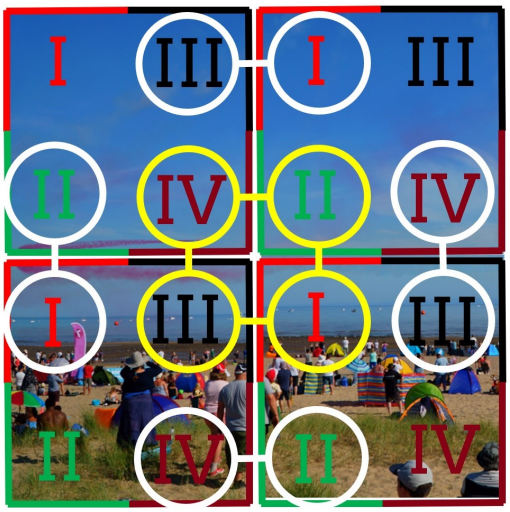


Figure 6: *Yellow lines and circles indicate a four-cycle correspondences between four corners. White lines and circles are two-cycle correspondences between two corners.*

Compared with sub-edges, using corners can reduce the size of $\mathcal{M}$. For example, for n square pieces and we break each edge into f sub-edges, the dimensions of the resulting matrix $\mathcal{M}$ based sub-edge matching will be $4fn \times 4fn$. By adopting this corner-wise approach our $\mathcal{M}$ is only $4n \times 4n$.

5. Assembling Pieces

After running MatchLift, the matrix $\mathcal{M}$ will be updated. The elements $m(p_i^\alpha, p_j^\beta) \in [0,1]$ with 1 indicating a confident correspondence that forms a cycle whilst 0 means the associated corner matching is not cycle consistent. Confident correspondences with $m(p_i^\alpha, p_j^\beta) \geq t_2$ are then returned and t_2 is a user defined threshold.

We follow [Gal12] to assemble puzzle pieces using minimum spanning tree (MST), which is a greedy technique. Based on the extracted corners from MatchLift we can infer the matching between sub-edges. If two sub-edges are matched we set the MGC score of the whole corresponding edge with a small value (by multiplying 0.000001 to the MGC scores) so that MST can prioritise the matching for piece assembling earlier. For example, in Figure 6 sub-edges 6 and 2 of the bottom pieces are matched. The MGC score between the entire right edge (containing sub-edges 5 and 6) and the entire left edge (containing sub-edges 1 and 2) is reduced. This allows MST to prioritise such matching to be considered first leading to correct assembly. The detailed information about MST and how to use MST to assemble puzzle pieces can be found in [BKJ56, Gal12].

6. Evaluation

We evaluate our method against [Gal12] in this section. First, we perform a quantitative evaluation on a small collection of images in Section 6.1 to compare the success assembly rate of our technique against MGC alone. Section 6.2 shows some assembled results from both methods as qualitative evaluation. For both methods we use the same number of puzzle pieces and images. We also evaluate with puzzle pieces of different resolutions in our experiments.

Image	100	144	196	assemble	t_1 t_2
1	100 79	94 90	100 41	6.3s 17.9s	0.2 0.2
2	100 81	100 55	100 37	5.0s 18.5s	0.1 0.9
3	91 94	50 51	31 47	15.9s 16.5s	0.4 0.9
4	73 93	57 39	41 84	17.4s 16.4s	0.2 0.7
5	54 32	46 37	51 35	17.2s 17.0s	0.3 0.9
6	96 83	76 74	49 52	12.8s 12.8s	0.5 0.8
7	92 100	64 62	49 58	11.5s 10.7s	0.3 0.9
avg (%)	87 80	70 58	60 51	12.3s 15.7s	

Table 1: *We compare our method and [Gal12] by showing percentage of correctly assembled pieces with 100, 144 and 196 pieces input. The assemble column shows the time requires to run MST for assembling. t_1, t_2 are parameters we used in our method. Our and [Gal12] results are shown in red and blue respectively.*

6.1. Quantitative Evaluation

We evaluate our method against [Gal12] on seven images of varying numbers of pieces and resolutions. We use five of our own (high and low resolution) images and two images from public data set [CAF10] (low resolution). We slice each image into 100, 144, and 196 pieces as the input of both methods. The higher number of pieces leads to lower resolution of each piece. Though MatchLift [CGH14] in theory has good tolerance to random outliers, the stability of MGC is low. When there are too many incorrect correspondences, it would lead to poor results. We therefore need to adjust two of our parameters t_1 and t_2. t_1 controls the number of correspondences accepted as input to MatchLift (more correspondences mean more input noises). t_2 controls how confident we accept the matching results from MatchLift. These parameters are somehow dependent on the resolution of images and stability of MGC. For t_1 we try 20 values $0.6 \leq t_1 \leq 1$ and 9 values for t_2 where $0.5 \leq t_2 \leq 0.95$ and report the best assembled results in Table 1. The overall process is time consuming. On average, it takes five hours (i7-6700 4.0GHz CPU with 32GB memory) per image in this experiment.

We use the ground truth coordinates of each piece to evaluate the assembled results, so-called the direct comparison [Gal12]. When an assembling technique misaligned a large assembled region, the percentage of correctly assembled pieces will reduce significantly. The evaluation results are shown in Table 1. Our initial results show that our method can produce better results than [Gal12] with the proper parameters. Because our technique recovers better piece matching, the MST assembling step is faster than using MGC alone. Nevertheless, we hope to discover the best parameter settings automatically for our technique in the future, for example, to investigate the spectrum of the matrix $\mathcal{M}$ [CGH14].

6.2. Image Resolution and Puzzle Solving

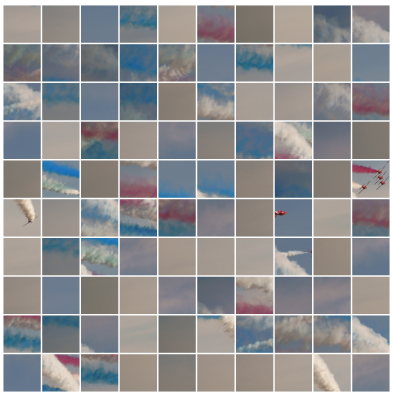
Next, we qualitatively evaluate our technique on high resolution images (all input images have a resolution above 2700 by 2700) in

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

(a) *Original image*

(b) *Assembled result by directly using MGC*

(c) *Our method result*



(d) *Input 100 puzzle pieces*

(e) *Assembled result by directly using MGC*

(f) *Our method result*

(g) *Input image*

(h) *Assembled result by directly using MGC*

(i) *Our method result*

(j) *Input image*

(k) *Assembled result by directly using MGC*

(l) *Our method result*

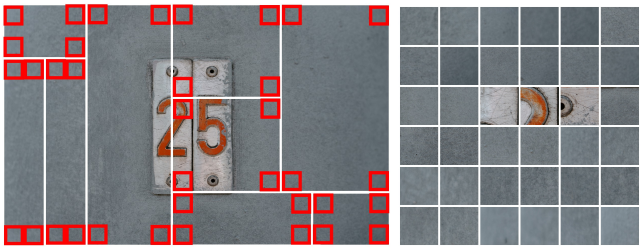
Figure 7: *Experimental results on puzzles built from high resolution images.*

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

(a) *Input image* **(b)** *Assembled result by directly using MGC* **(c)** *Our method assembled result*

(d) *Input image* **(e)** *Assembled result by directly using MGC* **(f)** *Our method assembled result*

Figure 8: *Directly using MGC causes more incorrect assembled results in low resolution images or images containing indistinctive pieces. Our method maintains 100% correctness in assembled images.*



(a) *Rectangular pieces of arbitrary sizes. We compute pairwise similarity by using corners as labelled in the red boxes.* **(b)** *We compute all possible corner-wise similarities.*

Figure 9: *We use corners on rectangular pieces with arbitrary sizes.*

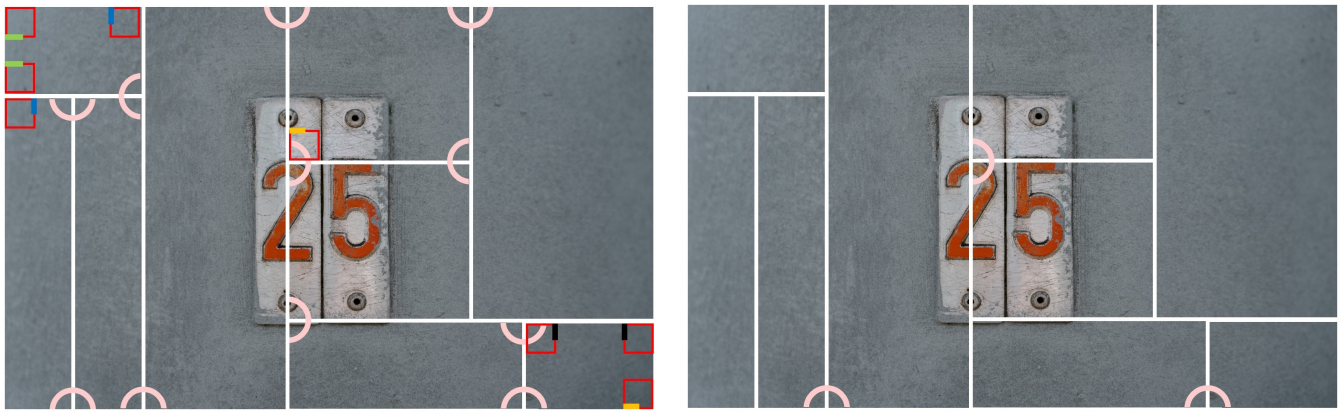
Figure 7. In Figures 7b and 7c both images are assembled from 49 pieces. For regions with distinctive texture, such as clouds at the low part of the image, MGC and MST perform well and produce good assembled results. However, MGC produces unreliable scores around the white smoke and cloud at the top. This leads to incorrect assembled results. In our case, after MatchLift refinement, MST can assemble 100% correct results. When we increase the number of puzzle pieces to 100, MGC becomes unreliable (Figure 7e). Meanwhile, our technique can still discover confident matching. This allows MST to assemble 100% correct results (Figure 7f).

Low distinctive regions are challenging for MGC. The resolution of Figure 7g is 2700 by 2700, and there are 196 puzzle pieces. Similar to Figure 7a the sky is difficult to be assembled by MGC. When puzzle pieces become smaller (since the number of pieces increased), the number of pixels to compute in MGC is fewer. MGC will return more unreliable scores. For example, in the red highlighted region of Figure 7h, MGC considers the sky and cloth are highly similar.

Figure 7j is another high resolution image of resolution 3840 by 3840. It consists of 64 pieces. Though the resolution is higher with fewer pieces, MGC does not perform well on under-exposed regions and leads to incorrectly assembled top-left region.

Figure 8 evaluates the two methods with images of low resolution. Figure 8a has a resolution of 1200 by 1200, and size of 289KB. We slice it into 144 pieces. In Figure 8b, without MatchLift refinement, the technique struggles to assemble regions around the deck, gun and road pieces. Our method models puzzle solving based on corners and cycle consistency constraint. We can better handle unreliable MGC scores of such pieces and assemble the correct results in Figure 8c. Similar situation appears in Figures 8e and 8f. With-

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.



(a) *Result of our method. Correct corner-wise matchings are labelled as pink half-circles. Incorrect matchings are labelled as other colours.*

(b) *All correct matchings obtained by [Gal12]. These matchings are insufficient to solve the puzzle. To avoid clutter, we do not show incorrect matchings.*

Figure 10: *By using MatchLift on corners we can find reasonable matching between rectangular pieces with arbitrary sizes.*

out MatchLift refinement, MST cannot find a correct assembling of the gun barrel and the camouflage netting behind the vehicles.

7. Puzzle Solving for Rectangular Pieces of Arbitrary Sizes

We show one interesting example of applying our technique to solve puzzles of rectangular pieces with arbitrary sizes. To our knowledge no existing techniques can handle such challenging case. Since the pieces have arbitrary sizes, our earlier square puzzle slicer does not apply. To produce the input puzzle pieces, we manually slice the image as shown in Figure 9a into 9 pieces. Next, we manually select 36 local regions (Figure 9b) to represent the four corners of all 9 pieces (red boxes in Figure 9a). Similar to square puzzle examples, for each corner, our technique breaks each edge into two sub-edges and computes similarity to other corners/pieces. We encode all similarity scores and pass them to MatchLift to obtain corner-wise matchings on these rectangular pieces.

Our method outputs 15 corner-wise correspondences. 11 of them are correct and are visualised as pink half circles in Figure 10a. The four incorrect corner-wise matchings are visualised as coloured bars with associated local regions (the red boxes) in Figure 10a. Among these four mismatched pairs, the green pair and the black pair are respectively from the same rectangular piece and can be removed as it is not possible to assemble corners/sub-edges from the same rectangular piece. The matched sub-edges of the blue pair are located inside the two rectangular pieces. The sub-edges/whole edges that are inside pieces should not be used in the assembling, because an assembling is based on the borders of each piece. Similarly, one of the matched sub-edges in the orange pair is also inside the rectangular piece. These four mismatched pairs can be easily removed in a pruning scheme as post-processing. On close inspection, we argue that such a puzzle with rectangular pieces of arbitrary sizes can be assembled correctly using the returned matched corners (visualised as the pink half circles) as shown in Figure 10a.

Figure 10b shows all the correct matchings obtained by [Gal12] with the same input of Figure 9b. To avoid clutter, we visualise all (only three) correct assembled corners (the pink half-circles). Since most corners are incorrectly assembled, we cannot refine/infer the results as we did for Figure 10a.

8. Limitation and Further Work

Long computational time is an issue for our current technique. MatchLift requires multiple eigen-decompositions which can be slow for puzzles with a large number of pieces. Another problem is that due to the nature of the images (e.g. distinctiveness, texture, resolution), our technique requires some parameter adjustment to obtain the best results, tailoring to the image properties. We hope to investigate and develop a parameter-free technique.

Currently, we are using square puzzle pieces with known rotations. We can use the same modelling idea to solve puzzle pieces with unknown rotation (Type II puzzle [Gal12]). In that way, the matrix $\mathcal{M}$ will be denser than the current configuration. We also would like to try non-rectangular pieces, or a mixture of square, triangle and polygonal pieces. Since our method models the puzzle problem with corners, it can be extended to such challenging examples, which existing techniques cannot solve. Given our promising cycle consistent corner constraint, we hope to develop a fully automatic technique to solve such problems.

9. Conclusion

In this paper, we try to solve square puzzle problems by considering two novel ideas. First, we use corner-wise correspondences, rather than edge-wise correspondences. Second, we model the subsequent puzzle problem into the MatchLift framework, solved via a semi-definite programming approach to recover cycle-consistent correspondences. We then refine the confident scores of these correspondences to promote their use for piece assembling early via a minimum spanning tree puzzle solver. Experimental results show that our technique can achieve better results than the non-refined cases. Finally we show that our technique can be extended to puzzles consisting of rectangular pieces of arbitrary sizes. It is an exciting and arguably more challenging problem. Our technique can still show promising initial results.

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.

References

[AACT*16] A. ANDALÃŞ F., CARNEIRO G., TAUBIN G., GOLDENSTEIN S., VELHO L.: Automatic reconstruction of ancient portuguese tile panels. *IEEE Computer Graphics and Applications* (07 2016). ii

[ATG12] ANDALÃŞ F., TAUBIN G., GOLDENSTEIN S.: Solving image puzzles with a simple quadratic programming formulation. pp. 63–70. doi:10.1109/SIBGRAPI.2012.18. ii

[BKJ56] B KRUSKAL JR J.: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society 7* (02 1956), 48–50. doi:10.2307/2033241. i, v

[CAF10] CHO T. S., AVIDAN S., FREEMAN W. T.: A probabilistic image jigsaw puzzle solver. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2010), pp. 183–190. doi:10.1109/CVPR.2010.5540212. i, ii, v

[CGH14] CHEN Y., GUIBAS L., HUANG Q.: Near-optimal joint object matching via convex relaxation. In *Proceedings of the 31st International Conference on Machine Learning* (Bejing, China, 22–24 Jun 2014), Xing E. P., Jebara T., (Eds.), vol. 32 of *Proceedings of Machine Learning Research*, PMLR, pp. 100–108. URL: http://proceedings.mlr.press/v32/chend14.html. ii, iii, iv, v

[DN16] D., DAVID O. E., NETANYAHU N. S.: Dnn-buddies: A deep neural network-based estimation metric for the jigsaw puzzle problem. In *Artificial Neural Networks and Machine Learning – ICANN 2016* (Cham, 2016), Villa A. E., Masulli P., Pons Rivero A. J., (Eds.), Springer International Publishing, pp. 170–178. ii

[FG64] FREEMAN H., GARDER L.: Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers EC-13*, 2 (April 1964), 118–127. doi:10.1109/PGEC.1964.263781. i

[Gal12] GALLAGHER A.: Jigsaw puzzles with pieces of unknown orientation. In *Proc. CVPR* (2012). i, ii, iii, v, viii

[GMB04] GOLDBERG D., MALON C., BERN M.: A global approach to automatic solution of jigsaw puzzles. *Computational Geometry 28*, 2 (2004), 165 – 174. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002. URL: http://www.sciencedirect.com/science/article/pii/S0925772104000239, doi:https://doi.org/10.1016/j.comgeo.2004.03.007. ii

[HFG*06] HUANG Q.-X., FLÖRY S., GELFAND N., HOFER M., POTTMANN H.: Reassembling fractured objects by geometric matching. *ACM Trans. Graph. 25*, 3 (July 2006), 569–578. i

[HG13] HUANG Q.-X., GUIBAS L.: Consistent shape maps via semidefinite programming. *Computer Graphics Forum 32*, 5 (2013), 177–186. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12184, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12184, doi:10.1111/cgf.12184. iii

[KDB*94] KOSIBA D., DEVAUX P., BALASUBRAMANIAN S., GANDHI T., KASTURI K.: An automatic jigsaw puzzle solver. vol. 1, pp. 616 – 618 vol.1. doi:10.1109/ICPR.1994.576377. i

[LZZC14] LI H., ZHENG Y., ZHANG S., CHENG J.: Solving a special type of jigsaw puzzles: Banknote reconstruction from a large number of fragments. *IEEE Transactions on Multimedia 16*, 2 (2014), 571–578. i

[MWD13] MONDAL D., WANG Y., DUROCHER S.: Robust solvers for square jigsaw puzzles. In *2013 International Conference on Computer and Robot Vision* (May 2013), pp. 249–256. doi:10.1109/CRV.2013.54. ii

[PPE*02] PAPAODYSSEUS C., PANAGOPOULOS T., EXARHOS M., TRIANTAFILLOU C., FRAGOULIS D., DOUMAS C.: Contour-shape based reconstruction of fragmented, 1600 bc wall paintings. *IEEE Transactions on Signal Processing 50*, 6 (June 2002), 1277–1288. i

[PSB11] POMERANZ D., SHEMESH M., BEN-SHAHAR O.: A fully automated greedy square jigsaw puzzle solver. In *CVPR 2011* (June 2011), pp. 9–16. doi:10.1109/CVPR.2011.5995331. ii

[PT15] PAIKIN G., TAL A.: Solving multiple square jigsaw puzzles with missing pieces. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 4832–4839. doi:10.1109/CVPR.2015.7299116. i, ii

[RRCL13] RICHTER F., RIES C. X., CEBRON N., LIENHART R.: Learning to reassemble shredded documents. *IEEE Transactions on Multimedia 15*, 3 (2013), 582–593. i

[RSSS11] ROBERTS R., SINHA S. N., SZELISKI R., STEEDLY D.: Structure from motion for scenes with large duplicate structures. In *CVPR 2011* (June 2011), pp. 3137–3144. doi:10.1109/CVPR.2011.5995549. iii

[SHC14] SON K., HAYS J., COOPER D. B.: Solving square jigsaw puzzles with loop constraints. In *Computer Vision – ECCV 2014* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 32–46. i, ii

[SHC18] SON K., HAYS J., COOPER D. B.: Solving square jigsaw puzzle by hierarchical loop constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018), 1–1. doi:10.1109/TPAMI.2018.2857776. i, ii

[SMHC16] SON K., MORENO D., HAYS J., COOPER D. B.: Solving small-piece jigsaw puzzles by growing consensus. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 1193–1201. doi:10.1109/CVPR.2016.134. i, ii

[WK01] WEIXIN KONG, KIMIA B. B.: On solving 2d and 3d puzzles using curve matching. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Dec 2001), vol. 2, pp. II–II. doi:10.1109/CVPR.2001.991015. ii

[WLS91] WEBSTER R., LAFOLLETTE P., STAFFORD R.: Isthmus critical points for solving jigsaw puzzles in computer vision. *Systems, Man and Cybernetics, IEEE Transactions on 21* (10 1991), 1271 – 1278. doi:10.1109/21.120080. i

[WZD18] WANG Q., ZHOU X., DANIILIDIS K.: Multi-image semantic matching by mining consistent features. pp. 685–694. doi:10.1109/CVPR.2018.00078. iii

[YAL11] YANG X., ADLURU N., LATECKI L. J.: Particle filter with state permutations for solving image jigsaw puzzles. In *CVPR 2011* (June 2011), pp. 2873–2880. doi:10.1109/CVPR.2011.5995535. ii

[YRA16] YU R., RUSSELL C., AGAPITO L.: Solving jigsaw puzzles with linear programming. *CoRR abs/1511.04472* (2016). i, ii

[Zan16] ZANOCI C.: Making puzzles less puzzling : An automatic jigsaw puzzle solver. i, ii

[ZKP10] ZACH C., KLOPSCHITZ M., POLLEFEYS M.: Disambiguating visual relations using loop constraints. pp. 1426–1433. doi:10.1109/CVPR.2010.5539801. iii

[ZLXYE15] ZHOU T., LEE Y. J., X. YU S., EFROS A.: Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. pp. 1191–1200. doi:10.1109/CVPR.2015.7298723. iii

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2242–2251. iii

© 2019 The Author(s)
Eurographics Proceedings © 2019 The Eurographics Association.