PSQP – Puzzle Solving by Quadratic Programming

Fernanda A. Andaló, *Member, IEEE,* Gabriel Taubin, *Fellow, IEEE,* and Siome Goldenstein, *Senior Member, IEEE*

***Abstract*—In this article we present the first effective global method for the reconstruction of image puzzles comprising rectangle pieces – Puzzle Solving by Quadratic Programming (PSQP). The proposed novel mathematical formulation reduces the problem to the maximization of a constrained quadratic function, which is solved via a gradient ascent approach. The proposed method is deterministic and can deal with arbitrary identical rectangular pieces. We provide experimental results showing its effectiveness when compared to state-of-the-art approaches. Although the method was developed to solve image puzzles, we also show how to apply it to the reconstruction of simulated strip-shredded documents, broadening its applicability.**

***Index Terms*—image puzzle, jigsaw puzzle, image analysis, tile compatibility, quadratic programming.**

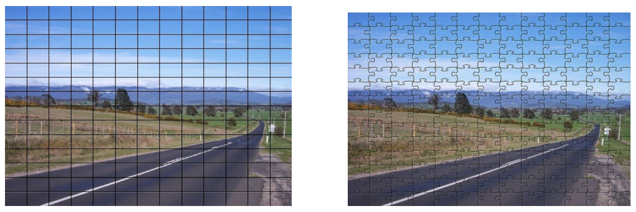

(a) (b)

Fig. 1. Different formulations of the jigsaw puzzle problem: (a) formulation considered in this work, in which pieces have linear boundaries, and (b) jigsaw puzzle formed by traditional puzzle pieces.

I. Introduction

The traditional *Jigsaw Puzzle* is the problem of assembling several non-overlapping puzzle pieces that can be combined following a fitting and/or a color pattern logic, with the final goal of obtaining a single plane or image.

Although it has been proved to be *NP*-complete when the affinity between the pieces in uncertain [1], much effort has been devoted to solve the problem and the scientific challenges that can be reformulated as 2D or 3D puzzles, such as: reassembling broken archaeological artifacts [2], [3], reconstruction of shredded documents [4], [5], speech recognition [6], DNA/RNA modeling [7], and image editing [8], among others. However, the primary interest in solving these puzzles is probably the simple and challenging nature of the problem, which captures people's imagination.

This article addresses the problem of reconstructing images from rectangular non-overlapping tiles of identical shape and size (Figure 1a), placed without repetition within a regular grid of the same dimension as the original image. Unlike the traditional jigsaw puzzle (Figure 1b), in our formulation the linear boundaries of the tiles do not provide additional geometric information, making the problem resolution even more challenging. We also assume that the puzzle pieces have known orientation.

To solve the problem, it is necessary to overcome two main difficulties. The first difficulty is the inherent combinatorial complexity of the problem. Since a feasible solution can be described by a permutation of the tiles in the rectangular grid, the number of feasible solutions increases exponentially as a function of the number of tiles. There exist $(M \times N)!$ possible permutations in a problem with $M \times N$ tiles. The second difficulty is the global nature of the problem. Several local measures have been studied to compute a compatibility measure between tiles and consequently to decrease the complexity of the global search. Nevertheless, an exact measure based solely on the similarity of the tiles' borders is unknown to date.

The proposed *Puzzle Solving by Quadratic Programming* (PSQP) method addresses these difficulties, comprising a global compatibility function, which is maximized to find the best permutation among tiles, and a local compatibility function, which gives a compatibility measure of tiles being assigned to neighboring locations.

Experiments are divided into two groups. First PSQP is compared to related methods that address the same problem [9], [10], [11], according to three measures: Direct comparison, in which the obtained permutation is compared directly to the ground-truth permutation; Neighbor comparison, in which the reconstruction accuracy is the average fraction of correct neighboring tiles; and Perfect reconstruction, a binary indicator of perfectly reconstructing the puzzle. PSQP is accurate considering these metrics, in comparison with the related methods. The second group of experiments aims at demonstrating the effectiveness of PSQP in another application – reconstruction of simulated strip-shredded documents. We also obtain good results, showing that PSQP applicability is not only limited to solving image puzzles with identically shaped rectangular tiles.

The outline of this article is as follows. Section II gives a brief history of automatic puzzle solving, classifying PSQP in comparison to relevant related works. The formulation of the PSQP method is presented in Section III. Some implementation issues and their solutions are discussed in Section IV. Experimental results and analysis are presented in Section V. Finally, conclusions are presented in Section VI, along with directions for future work.

F.A. Andaló and S. Goldenstein are with the the Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil e-mail: feandalo,siome@ic.unicamp.br.

G. Taubin is with the Division of Engineering, Brown University, Providence, RI, USA e-mail: taubin@brown.edu.

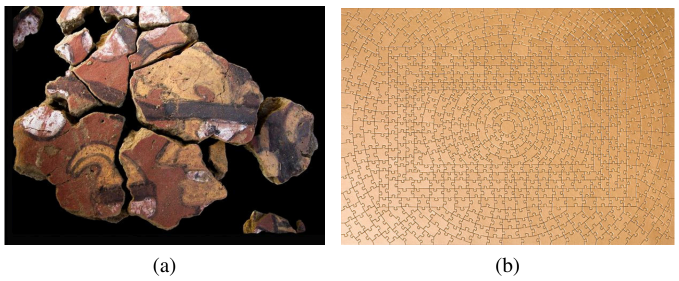


Fig. 2. Examples of real puzzles. (a) Fragments from a mural found at Huaca Bandera, 2010, photograph courtesy National Archaeological Museum of Brüning, Peru. (b) Apictorial puzzle manufactured by *Ravensburger*.

II. Related Work

Methods for automatically solving image puzzles can be classified according to whether or not they use chromatic information, by the shape of the puzzle pieces, and by the choice of two main strategies: matching between the pieces, and the puzzle assembling itself (i.e. how the pieces are combined to form the result).

In general, there are pictorial puzzles, where the puzzle pieces are obtained by cutting an image; and apictorial puzzles, where the pieces are obtained by cutting a constant color plane, without any chromatic variation. In pictorial puzzles, the pieces can be identical size rectangles (Figure 1a), can be a traditional shape (Figure 1b), or have totally irregular shapes (Figure 2a). In these puzzles, the matching strategy between pieces relies on the analysis of their color and/or shape. In apictorial puzzles, the pieces should be matched only by their shape, disallowing the use or regular pieces (Figure 2b). We proceed to classify and analyze the main methods for automatic jigsaw puzzle solving, with respect to these important characteristics.

A. Apictorial puzzles

The first method for solving apictorial puzzles with traditional pieces was proposed in 1964 by Freeman and Gardner [12]. It was able to solve problems with only 9 pieces. Even though it had no implementation, it was a foundation for subsequent works. The method identified critical points along the boundary of the pieces, which were then used to compute a matching measure.

Burdea and Wolfson [13] developed a program to find corresponding pieces in a traditional puzzle, and even showed how to control a robot arm to position the pieces next to each other according to the problem solution. With this method they were able to solve puzzles with 104 pieces. The method used a *Schwartz–Sharir* [14] bi-dimensional curve matching algorithm to compute the affinity between the pieces, and optimized search trees for assembling the puzzle.

Curve matching was also the strategy used for the comparing pieces in the method proposed by Goldberg et al. [15]. Nearly 40 years after the publication of the first solution, this method was able to solve problems with 204 pieces. The method first computes the centers of the ellipses that fit on the indents and outdents of the pieces. Then, by comparing these points, the method finds the best translation and rotation for each puzzle piece. To assemble the puzzle, the authors use a greedy approach, starting with the edge pieces.

All these techniques rely on special features of the fragments, as smooth edges and sharp corners. These characteristics are typically not available in applications such as the reconstruction of archaeological material, where the fragments are completely irregular, the boundaries are eroded, and many pieces may be missing.

Leitão and Stolfi [16] address the problem of irregular pieces by applying a multi-scale approach directly to the comparison of the contours, without the need of identifying critical points. They demonstrate the ability of automatically distinguishing adjacent fragments, by re-sampling the contours in low-level details, which facilitates the global search. Considering the formed pairs of fragments, only the best are kept and analyzed at higher scales.

Using a similar multi-scale approach and a greedy search – global best-first – McBride and Kimia [2] were able to reconstruct an artifact of approximately 20 fragments.

B. Pictorial puzzles

Kosiba et al. [17] proposed the first method that considers the chromatic information of the pieces, although the size of the puzzle shown to be successfully assembled is small – only 54 pieces with a greedy strategy. The process of matching the pieces takes into account many of their features: color samples along the edges, curvature parameters and their convexities and concavities.

The type of piece used in the PSQP method is different from the ones described so far. We consider identical rectangular pieces, or tiles, where the matching characteristic is the chromatic information along their borders. The literature for this classification is somewhat recent and not extensive, although the problem is very important in practice. Here we present a real application for this kind of formulation: the reconstruction of shredded documents (text or image) by paper shredders.

The available literature for image reconstruction from identical rectangular tiles comprises mainly a few recent works [18], [19], [9], [10], [11], [20].

Nielsen et al. [18] presented the first method to successfully solve these image puzzles. The method was shown to reconstruct puzzles with up to 320 pieces, assembled by a greedy strategy.

Cho et al. [19] obtained approximate reconstruction of the original image using graphical models and a probabilistic function maximized by Loopy Belief Propagation. Because the method needs information about the layout of the original image, they employed two strategies to explore *a priori* knowledge: estimation of image in low resolution from a few tiles, to serve as local evidence in the graphical model; or the correct fixation of some tiles, called anchors, by the user. Although being semi-automatic, this strategy allowed the assembling of puzzles with up to 432 tiles.

Focusing on the disadvantage of the method by Cho et al. [19], Pomeranz et al. [9] presented a greedy method which does not require any user intervention. First, a compatibility

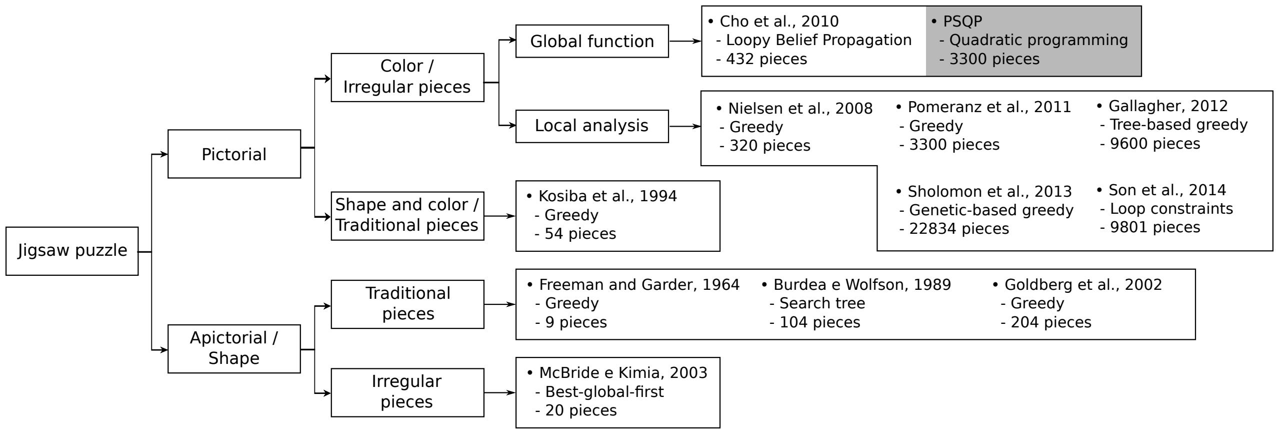

Jigsaw puzzle
Pictorial
Apictorial / Shape
Color / Irregular pieces
Shape and color / Traditional pieces
Global function
Local analysis
Traditional pieces
Irregular pieces
• Cho et al., 2010 - Loopy Belief Propagation - 432 pieces
• PSQP - Quadratic programming - 3300 pieces
• Nielsen et al., 2008 - Greedy - 320 pieces
• Pomeranz et al., 2011 - Greedy - 3300 pieces
• Gallagher, 2012 - Tree-based greedy - 9600 pieces
• Sholomon et al., 2013 - Genetic-based greedy - 22834 pieces
• Son et al., 2014 - Loop constraints - 9801 pieces
• Kosiba et al., 1994 - Greedy - 54 pieces
• Freeman and Garder, 1964 - Greedy - 9 pieces
• Burdea e Wolfson, 1989 - Search tree - 104 pieces
• Goldberg et al., 2002 - Greedy - 204 pieces
• McBride e Kimia, 2003 - Best-global-first - 20 pieces

Fig. 3. Classification of relevant puzzle solving methods.

function which measures the affinity between two tiles is computed and evaluated on every pair of tiles. Then the method comprises three modules: positioning, segmentation, and translation. The positioning module puts all tiles on the grid following a predetermined logic and considering randomly selected seeds; the segmentation module identifies the regions that are more likely to be assembled correctly; and the translation module reallocates regions and tiles to produce the result. With this strategy, they achieved the considerable improvement of solving puzzle with up to 3,300 tiles.

The method proposed by Gallagher [10] uses a new compatibility measure based on local gradients near the boundary of the tiles and a tree-based greedy assembling approach. The method is able to assemble puzzles with up to 9,600 square pieces with unknown orientation and location (using specific images).

Sholomon et al. [11] advanced the state-of-the-art, proposing a genetic algorithm to solve very large puzzles up to 22,834 puzzle pieces with known tile rotation and puzzle dimension.

Son et al. [20] proposed a novel algorithm based on “loop constraints” for assembling jigsaw puzzles where the rotation and the position of each piece are unknown. Their key idea is to find all small loops, or cycles of pieces, and group these small loops into higher order “loops of loops”, in a bottom-up fashion, to form a structure with no geometric conflict.

These discussed related methods can work in the same scenario considered here: known puzzle dimension and tile orientation. Some of them and other methods in the literature are also extended to work in more complex scenarios, but we are not covering them here as they are out of scope.

In this article we present an effective global optimization method to the solution of rectangular-piece jigsaw puzzles, with known tile rotation and puzzle dimension. We provide a new mathematical formulation to reformulate a hard combinatorial problem into a constrained continuous optimization problem, permitting the application of numerical methods.

PSQP is effective in most image puzzles up to $3,300$ tiles, considering the studied metrics. It also can solve puzzles with arbitrary rectangular pieces directly.

Figure 3 shows a diagram summarizing the methods presented in this section, classified first by the characteristic used to match the pieces (shape and/or color) and subsequently by the piece type (rectangular, traditional or irregular), and assembling strategy. Along with each method we specify its specific assembling strategy when possible and the maximum number of puzzle pieces that were tested.

III. Image puzzle solving by quadratic programming

The explanation of the PSQP – Puzzle Solving by Quadratic Programming – can be divided into four stages. First the global compatibility function is presented in Subsection III-A and in Subsection III-B we show how to reformulate it as a quadratic homogeneous function. The numerical method used to solve the optimization problem is presented in Subsection III-C, and the local compatibility function is defined in Subsection III-D.

A. *Global compatibility function*

The method formulation consists of an image partitioned into a regular 2D grid of size $N_{columns} \times N_{rows}$ forming N tiles, $t_1, \dots, t_N$, of identical dimensions; and an empty similar grid with N locations labeled $1, \dots, N$. We have to determine a biunivocal correspondence between the N tiles and the N locations, optimized with respect to a properly constructed global compatibility function.

Since the biunivocal correspondence can be described as a permutation π of N tiles, the problem can be reduced to a discrete optimization over the finite group of N-element permutations.

We organize this formulation in a directed graph $G = \{V, E = E_H \bigcup E_V\}$, where the vertices represent the locations, $V = 1, \dots, N$, and the edge set E comprehends all pairs of neighboring locations. Sets E_H and E_V denote horizontally and vertically neighboring locations, respectively. Graph G must be directed since, in general, swapping two tiles should result in a global compatibility value change.

For each pair of tiles (t_i, t_j), for $1 \leq i, j \leq N$ and $i \neq j$, we define two local compatibility measures, $C_{H_{i,j}} \geq 0$ and $C_{V_{i,j}} \geq 0$, which correspond to the compatibility of the two tiles being associated with locations connected by any horizontal edge $e \in E_H$ or vertical edge $e \in E_V$, respectively.

The global compatibility function of a permutation π is

$$\varepsilon(\pi) = \sum_{(i,j)\in E_H} C_{H_{\pi(i)\pi(j)}} + \sum_{(i,j)\in E_V} C_{V_{\pi(i)\pi(j)}}, \tag{1}$$

where $e = (i, j)$ is the edge connecting the neighboring locations i and j; and $\pi(i)$ corresponds to the tile permuted to location i.

The goal is to maximize this function (Equation 1) considering all permutations π of N elements. Since this is a hard combinatorial problem, we reformulate it as a constrained continuous optimization problem which, in turn, can be solved by numerical methods.

B. Reformulation the global compatibility function

In this section, the defined global compatibility function (Equation 1) will be reformulated as a quadratic homogeneous function of a square matrix.

Each permutation π of N elements can be represented as a permutation matrix, i.e., a binary square matrix P, with one entry 1 in each row and column:

$$P_{ik} = \begin{cases} 1, & \text{if } k = \pi(i), \\ 0, & \text{if } k \neq \pi(i). \end{cases} \tag{2}$$

Using this notation, we can reformulate the global compatibility function as

$$\varepsilon(P) = \sum_{(i,j)\in E_H} (P^\top C_H P)_{ij} + \sum_{(i,j)\in E_V} (P^\top C_V P)_{ij}, \tag{3}$$

where a generic term $(P^\top CP)_{ij}$, corresponding to edge $e = (i, j)$, is the element (ij) of the square matrix $(P^\top CP)$.

Note that, for each edge $e = (i, j)$, the term $(P^\top CP)$ is a homogeneous non-negative quadratic function of elements of matrix P. It follows that the sum of all terms in $\varepsilon(P)$ is also a homogeneous non-negative quadratic function of P. If the columns $p_1, \ldots, p_N$ of matrix P, which has $N \times N$ elements, are stacked up in a column-vector p of dimension N^2, we have

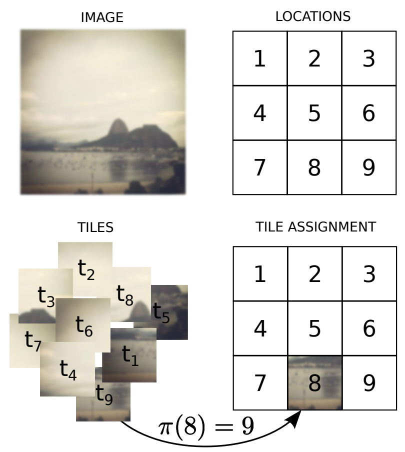
$$\varepsilon(P) = \sum_{(i,j)\in E_H} p_i^\top C_H p_j + \sum_{(i,j)\in E_V} p_i^\top C_V p_j. \tag{4}$$

We can rewrite Equation 4 in the canonical quadratic form $p^\top Ap$, where A is a symmetric non-negative matrix of dimension $N^2 \times N^2$ that represents the Hessian of $\varepsilon(P)$. In vector form and in coordinates,

$$\varepsilon(P) = p^\top Ap = \sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{j=1}^{N} P_{ki} A_{(ki)(lj)} P_{lj}, \tag{5}$$

where $A_{(ki)(lj)}$ is element (lj) of block (ki) of matrix A. Block (ki) of matrix A is the second-order partial derivative of $\varepsilon(P)$ with respect to edge (k, i), which will be either a null matrix, C_H, C_V, or their transposes.

Figure 4 illustrates the proposed formulation: tile assignment via permutation of the tiles, edge sets representing neighboring locations, and the final form of matrix A. Note that in this example A is a block-matrix, where each block is a 9×9 matrix.



$$\boxed{\begin{aligned} E_H &= \{(1,2),(2,3),(4,5),(5,6),(7,8),(8,9)\} \\ E_V &= \{(1,4),(2,5),(3,6),(4,7),(5,8),(6,9)\} \end{aligned}}$$

MATRIX A

$$\begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & C_H & 0 & C_V & 0 & 0 & 0 & 0 & 0 \\ \hline C^T_H & 0 & C_H & 0 & C_V & 0 & 0 & 0 & 0 \\ \hline 0 & C^T_H & 0 & 0 & 0 & C_V & 0 & 0 & 0 \\ \hline C^T_V & 0 & 0 & 0 & C_H & 0 & C_V & 0 & 0 \\ \hline 0 & C^T_V & 0 & C^T_H & 0 & C_H & 0 & C_V & 0 \\ \hline 0 & 0 & C^T_V & 0 & C^T_H & 0 & 0 & 0 & C_V \\ \hline 0 & 0 & 0 & C^T_V & 0 & 0 & 0 & C_H & 0 \\ \hline 0 & 0 & 0 & 0 & C^T_V & 0 & C^T_H & 0 & C_H \\ \hline 0 & 0 & 0 & 0 & 0 & C^T_V & 0 & C^T_H & 0 \\ \hline \end{array}$$

Fig. 4. Problem formulation. From top to bottom: each tile t_i is assigned to a location j. The result is represented by a permutation π of the $N = 9$ tiles; edge sets E_H and E_V representing horizontal and vertical neighboring locations, respectively; and the final form of matrix A for this example.

C. Constrained Gradient Ascent

Permutation matrices are a special case of doubly stochastic matrices [21], which are non-negative matrices such that the sum of all elements in each row and in each column is equal to 1. In fact, the set of doubly stochastic matrices is the convex hull of $N \times N$ permutation matrices. Each doubly stochastic matrix satisfies N^2 inequality constraints, which specify that the elements of P are non-negative, and $2N$ equality constraints, which specify that the sum of elements of each row and each column of P is equal to 1.

Extending the domain of $\varepsilon(P)$ for all doubly stochastic matrices, the problem reduces to the solution of the following quadratic optimization problem:

$$\begin{aligned} &\text{Maximize } f(p) = p^\top Ap, \\ &\text{subject to } P\mathbb{1} = \mathbb{1}, P^\top\mathbb{1} = \mathbb{1}, \text{ and } p_{ij} \geq 0, \end{aligned} \tag{6}$$

where $\mathbb{1}$ is an $N-$column vector with all elements equal to 1, and $p_{ij} = P_{ij}$, i.e., the element (ij) of P.

Although the objective function $f(p)$ is positive in the feasible set, it is not necessarily concave, since matrix A is not positive definite. All diagonal values of A are equal to 0, which

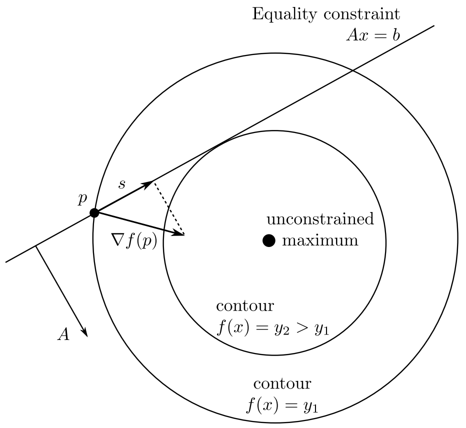


Fig. 5. Simplified 2D gradient projection of the gradient $\nabla f(p)$ onto the space defined by the linear equality constraint, during a maximization process, generating a constrained ascent direction s.

violates the necessary conditions for the matrix to be positive definite or even positive semi-definite. Therefore we cannot guarantee that $f(p)$ reaches its maximum in a permutation matrix. However, in practice, we observe that we can get very close to a solution by working with the constraints.

To search for a local maximum of Equation 6, we propose a simple constrained gradient ascent algorithm, with gradient projection [22]. To find the local maximum, we update the variables in steps proportional to the gradient of the function in the current point, while the gradient is projected.

Although there are several optimization algorithms to maximize quadratic functions in literature, we propose a specific gradient ascent to deal with two particular difficulties. First, the function to be maximized is not concave. Second, the number of variables is quadratic on N, e.g., for a puzzle with $3,300$ tiles, we have to work with $10,890,000$ variables.

The optimization algorithm needs to keep a set of active variables – the ones that are inside the feasible region. A variable is inactivated when it reaches the boundary of the feasible region and cannot be further updated, meaning that the dimensionality of the problem is reduced by one. All variables are initialized as $p_{kl} = \frac{1}{N}$, for $1 \leq k, l \leq N$, and set as active, i.e., $active_{kl} = true$, where $active_{kl}$ indicates whether p_{kl} is active or not.

The ascent direction is computed as $d = \nabla f(p) = A * p$ at current p. Note that to compute d it is not necessary to explicitly construct matrix A, because it is composed by known blocks that are either formed by zeros, C_H, C_V, or their transposes.

It is possible, however, that the ascent direction d does not reside in the space defined by the linear equality constraints. Therefore, the ascent direction must be projected onto the space defined by the linear equality constraints [22], yielding a constrained ascent direction s. Figure 5 illustrates a simplified 2D gradient projection during a maximization process.

Considering the constrained ascent direction s, the method updates p to a new feasible point: $p_{kl} = p_{kl} + step * s$, for $1 \leq k, l \leq N$ and $active_{kl} = true$, where $step$ is the maximum value such that $0 \leq p_{kl} \leq 1$.

When one of the variables reaches the boundary of the feasible region, the constraints should be modified so that the variable no longer gets updated and remains at the boundary. In practice, however, maintaining a group of mutable and orthogonal constraints implies high computational costs. For this reason, p must be reinitialized when there is no ascent direction to maximize the energy inside the feasible region.

In order to reinitialize p, firstly the variables at the boundary of the feasible region, i.e, the ones equal to 0 or 1, must be deactivated. Deactivating a variable which is in the upper limit is equivalent to associate the corresponding tile to the most probable location. Then p is initialized without the inactive variables: $p_{kl} = \frac{1}{N-N_a}$, for $1 \leq k, l \leq N$ and $active_{kl} = true$, where N_a is the number of tiles that have already been assigned to a location. The algorithm iterates until all tiles have been assigned to a location.

Algorithm 1 describes the proposed Constrained Gradient Ascent optimization method.

Algorithm 1 Constrained Gradient Ascent.
Input: Local compatibility matrices C_H and C_V; and the number of tiles N.
Output: Permutation π of tiles.

```
$N_a \leftarrow 0$;
$active_{kl} \leftarrow true$, for $1 \leq k, l \leq N$;
while $N_a < N$ do
    $p_{kl} \leftarrow \frac{1}{N - N_a}$, for $1 \leq k, l \leq N$ and $active_{kl} = true$;
    $d \leftarrow \nabla f(p) \leftarrow A * p$;
    $s \leftarrow Kd$, where $K$ is the projection matrix;
    $p_{kl} \leftarrow p_{kl} + step * s$, for $1 \leq k, l \leq N$ and $active_{kl} = true$;
    for $1 \leq k, l \leq N$ and $active_{kl} = true$ do
        if $p_{kl} = 0$ then
            $active_{kl} \leftarrow false$;
        end if
        if $p_{kl} = 1$ then
            $active_{kl} \leftarrow false$;
            $\pi(l) \leftarrow k$;
            $N_a \leftarrow N_a + 1$;
        end if
    end for
end while
```

D. Local compatibility function

The compatibility between pairs of fragments has been studied previously [19], [9] and plays a crucial role in the solution of image puzzles. Demaine et al. [1] showed that if it is possible to locally identify whether two fragments must be neighbors in the solution, then an attempt to assemble all tiles using a greedy method solves the problem in polynomial time. But in natural images it is trivial to find examples of tiles with ambiguous neighbors.

In the method by Cho et al. [19], one of the considered compatibility functions is based on dissimilarity. The horizontal dissimilarity between two tiles t_i and t_j is computed as

$$D_{H_{ij}} = \sum_{k=1}^{T} \sum_{l=1}^{3} (t_i(k,T,l) - t_j(k,1,l))^2, \tag{7}$$

where tiles t_i and t_j are $T \times T \times 3$ matrices and the color difference is computed in the normalized L*a*b* color space. The vertical dissimilarity $D_{V_{ij}}$ is computed similarly.

Based on the dissimilarity measure, Cho et al. [19] calculate the local horizontal compatibility between two tiles t_i and t_j as

$$C_{H_{ij}} \propto \exp\left(-\frac{D_{H_{ij}}}{2{\sigma_c}^2}\right), \tag{8}$$

where σ_c is adaptively defined as the difference between the lowest and the second lowest $D_{H_{ij}}$, for $1 \leq j \leq N$.

Pomeranz et al. [9] observed that the local compatibility function proposed by [19] (Equation 7) is related to the L_2 norm of the vector of differences across tiles borders, suggesting that other norms could provide better results. To analyze the distribution of dissimilarity values when computing compatibilities, several powers q of the L_p norm were studied. The horizontal dissimilarity with norm $(L_p)^q$ for tiles t_i and t_j is defined as [9]

$$D_{H_{ij}} = \left(\sum_{k=1}^{T} \sum_{l=1}^{3} (\mid t_i(k,T,l) - t_j(k,1,l) \mid)^p\right)^{\frac{q}{p}}. \tag{9}$$

While Cho et al. [19] consider $p = 2$ and $q = 2$, Pomeranz et al. [9] found their best results with values $p = \frac{3}{10}$ and $q = \frac{1}{16}$.

Pomeranz et al. [9] also studied a compatibility function based on prediction. Instead of computing dissimilarities, it quantifies how well the content of one tile border can be predicted based on the content of the other tile border. For the predictions, they used Taylor expansion. The derivative is obtained by computing the difference between the two last pixels, near the border, of every row of the tile. This value is used as a prediction for the first pixel of the corresponding row in the second tile. The prediction is then compared to the actual value of the first pixel of the second tile. That is, the horizontal prediction of t_i over t_j is

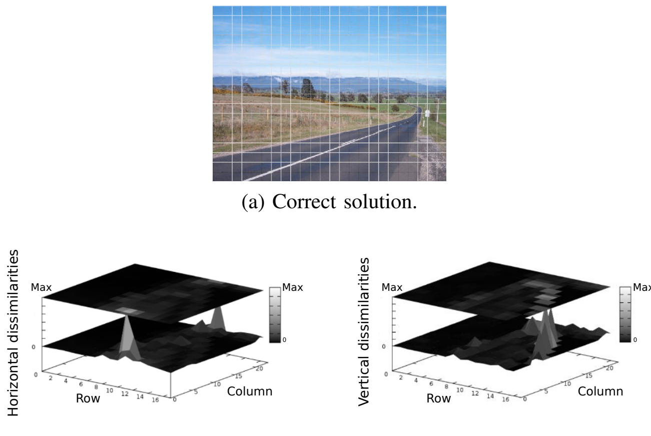
$$\begin{aligned} D_{H_{ij}} &= \\ \Bigg[&\sum_{k=1}^{T} \sum_{l=1}^{3} (\mid 2 * t_i(k,T,l) - t_i(k,T-1,l) - t_j(k,1,l) \mid)^p \\ &+ (\mid 2 * t_j(k,1,l) - t_j(k,2,l) - t_i(k,T,l) \mid)^p \Bigg]^{\frac{q}{p}}, \end{aligned} \tag{10}$$

Their compatibility measure is defined as

$$C_{H_{ij}} \propto \exp\left(-\frac{D_{H_{ij}}}{quartile(i)}\right), \tag{11}$$

where $quartile(i)$ is the quartile of the dissimilarity values among tile t_i and all other tiles.

Gallagher [10] proposed a measure called Mahalanobis Gradient Compatibility (MGC) that describes the local gradients near the boundary of a tile. It penalizes changes in intensity



(a) Correct solution.

(b) Horizontal dissimilarities.

(c) Vertical dissimilarities.

Fig. 6. Dissimilarities among tiles assigned to their correct location.

gradients and considers the covariance between color channels. The horizontal dissimilarity between tiles t_i and t_j is

$$D_{LR}(t_i,t_j) = \sum_{k=1}^{T} (G_{ijLR}(k) - \mu_{iL}) S_{iL}^{-1} (G_{ijLR}(k) - \mu_{iL})^T, \tag{12}$$

where $G_{ijLR}(k)$ is the gradient from the right side of t_i to the left side of t_j, at row k; μ_{iL} is the mean difference between the final two columns of t_i; and S_{iL} is a 3×3 covariance matrix estimated from the difference between the last column of t_i and the first column of t_j.

Because dissimilarity D_{LR} is not symmetric, the final horizontal symmetric dissimilarity is

$$D_{H_{ij}} = ((D_{LR}(t_i,t_j))^p + (D_{RL}(x_j,x_i))^p)^q, \tag{13}$$

where, in [10], the parameters are fixed: $p = q = 1$. Equations 12 and 13 are modified appropriately to analyze the other configurations.

The compatibility functions studied so far make use of components that account for global information, as σ_c and $quartile(i)$. Such components are important because dissimilarities between tiles have an intrinsic ambiguity and are only comparable in local small regions. As an example, Figure 6 shows the dissimilarities computed by Equation 7, considering only neighboring tiles in the correct solution. In those areas of the image where there is little information, such as cloudless skies, the dissimilarity between the tiles is lower than the dissimilarity between tiles in non-constant parts. This problem is even worse when all possible pairs of tiles are considered.

Because PSQP is a global method, where all variables are taken into consideration together in each iteration, the ambiguity difficulty has to be further addressed. We present a new way of computing compatibilities between tiles, based on the previous ones, which imposes a stronger global order. The horizontal compatibility between tiles t_i and t_j is defined as

$$C_{H_{ij}} \propto \exp\left(-\varphi(i,j) - \frac{D_{H_{ij}}}{quartile(i)}\right), \tag{14}$$

where $\varphi(i,j)$ is used to impose a stronger global meaning to the compatibilities. The value of $\varphi(i,j)$ is determined by the position/ranking of $D_{H_{ij}}$ in an increasingly ordered set of values $D_{H_{ik}}$, for $1 \leq k \leq N$, summed up with the position/ranking of $D_{H_{ij}}$ in an increasingly ordered set of values $D_{H_{kj}}$, for $1 \leq k \leq N$. For example, if tile t_j is the second candidate neighbor of t_i, and t_i is the first candidate neighbor of t_j in the opposite border, according to D_H, then $\varphi(i, j) = 3$. The same idea is applied to compute vertical compatibility values.

We have studied Equation 14 with two different dissimilarity functions: the prediction-based function (Equation 10) and MGC (Equation 13). In Equation 10, the color differences were computed in the YIQ color space and the channels were normalized to attain the same variance. In Equation 13, the RGB space was considered. These color spaces were chosen based on the recommendation of the original works.

While in [9], [10] the values of p and q in $(L_p)^q$ are fixed, we observed that there are optimal p and q for each image. For this reason, we do not apply the same values for all images. In practice, some sets of parameters p and q are tested and the method chooses the set that yields the highest normalized global compatibility (Equation 1).

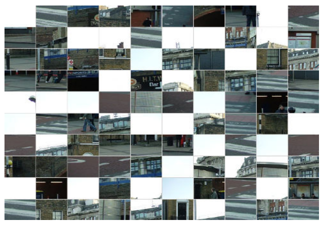
IV. Implementation

A major concern in the implementation of PSQP is memory usage. Vector p, the ascent direction s, and matrices C_H and C_V have N^2 elements each. To save up memory, we observed that the term $\varphi(i, j)$ in Equation 14 induces the compatibility values to decrease rapidly, becoming very close to 0 when distant neighbors of t_i are considered. Under these circumstances, it is possible to zero out compatibility values that are already close to 0 using a safe threshold (10^{-6}), which makes matrices C_H and C_V sparse. In the optimal case, C_H will have $N_{rows} \times (N_{columns} - 1)$ non-negative entries and C_V will have $N_{columns} \times (N_{rows} - 1)$ non-negative entries, drastically reducing memory usage.

The computational complexity of PSQP in $O(N^4)$. The ascent direction is computed by traversing matrices C_H and C_V, and gradient projection is done by traversing the corresponding vector twice.

There are two new issues not discussed so far. First, the constant tiles – group of tiles that have borders with identical color information – pose a difficult problem to be solved. These tiles have total compatibility among them and the same compatibility with all other non-constant tiles. To overcome this issue, we ignore constant tiles. If the total compatibility is replaced by zero compatibility, these tiles will not be analyzed by the method, finally being assigned to empty locations by the end of the optimization method. Note that it does not matter the permutation adopted among constant tiles, because their borders are equal. Our method will not distinguish between two or more sets of distinct constant tiles.

The second issue is associated with the non-concave property of the global compatibility function. There is no guarantee that the maximum obtained by the Gradient Ascent method is the global optimum. In practice, we observed that for the majority of images, PSQP works towards finding the correct solution. However, in some few cases, especially in images


(a) Initial configuration (random permutation).

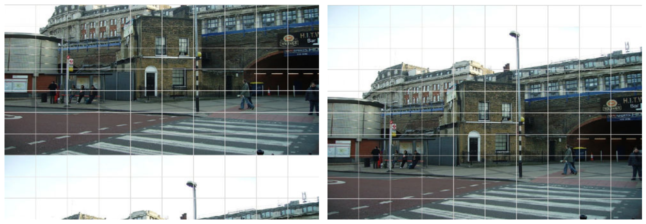

(b) Final permutation: displacement of the correct permutation. (c) Correct permutation.

Fig. 7. Example of the non-concave property of the global compatibility function. Note that the image contains several constant (white) tiles.

with constant tiles, the final permutation is a displacement of the correct permutation (Figure 7).

To correct the displacement, PSQP adopts a post-processing step: the global compatibility function $\varepsilon(\pi)$ is computed for the final permutation π changed by every possible cyclical shift, considering each row and each column, and with p and q equal to 1. The algorithm picks the shift that increases the global compatibility – a linear-time operation on the number of tiles. The formulation of PSQP is presented in Algorithm 2.

Algorithm 2 PSQP method.

```
$\pi \leftarrow GradientAscent(C_H, C_V, N)$;
$st_{H_{max}}, st_{V_{max}} \leftarrow \underset{st_H \in [1, N_{cols}],\ st_V \in [1, N_{rows}]}{\operatorname{argmax}} GlobalComp(\pi, st_H, st_V)$;
$\pi_{final} \leftarrow Shift(\pi, st_{H_{max}}, st_{V_{max}})$;
```

In Algorithm 2, function $GradientAscent$ returns the permutation π given by Algorithm 1; $GlobalComp$ computes the global compatibility $\varepsilon(\pi)$ after applying to π cyclical shifts of st_H horizontally and st_V vertically; and finally $Shift$ applies to π the shifts that generate the higher global compatibility value ($st_{H_{max}}$ horizontally and $st_{V_{max}}$ vertically).

V. Experimental results

We use twenty jigsaw puzzles [19], each with 432 tiles of size 28×28 pixels, to compare PSQP to the three most recent methods in the literature [9], [10], [11]. The accuracy of the solutions are measured according to three different metrics [19], [10]:

Direct comparison: the final permutation is compared directly to the ground-truth permutation. This metric computes the ration between the number of tiles that are assigned to the correct location and the total of tiles.

Neighbor comparison: for each tile, this metric computes the fraction of its neighboring tiles that are also

TABLE I
SETS OF PARAMETERS FOR PSQP, DEPENDING ON THE CONSIDERED DISSIMILARITY MEASURE.

Equation 10		*Equation 13*	
p	***q***	***p***	***q***
0.3	1.8	0.5	1.0
1.0	6.0	0.5	2.0
1.0	1.0		
1.0	0.3		
3.0	3.0		

TABLE II
METRICS COMPUTED FOR EACH OF THE TWENTY IMAGES USING EQUATION 10. D STANDS FOR *Direct comparison* AND N FOR *Neighbor comparison*.

	PSQP	
Image #	**D (%)**	**N (%)**
1	88.5	85.5
2	83.2	82.1
3	100.0	100.0
4	65.9	65.0
5	100.0	100.0
6	98.4	98.3
7	100.0	100.0
8	100.0	100.0
9	100.0	100.0
10	100.0	100.0
11	100.0	100.0
12	99.5	99.4
13	88.9	87.8
14	100.0	100.0
15	95.6	94.2
16	100.0	100.0
17	100.0	100.0
18	100.0	100.0
19	100.0	100.0
20	100.0	100.0
Mean	96.0	95.6

its neighbors in the correct solution. The accuracy is the mean fraction of correctly assigned neighbors.

Perfect reconstruction: binary indication of whether every tile is assigned to the correct location in a puzzle.

Direct comparison, however, is not a good metric [11], being too sensitive to shifted solutions. For the sake of completeness, we reported the results considering all metrics, including direct comparison.

The method PSQP was implemented in C++ and the experiments were conducted on a 3.4 GHz machine with 8 GB of RAM.

In order to find out the optimal parameters p and q, PSQP was previously trained with a set of 50 images randomly chosen from the Internet, with the two considered dissimilarity measures (Equations 10 and 13). The training step was performed by a grid search in an interval of possible parameters. Table I shows the optimal sets of parameters.

PSQP was executed with the optimal sets of parameters and the best solution – the one that yields the highest normalized compatibility measure (Equation 1) – is chosen as the solution. For all executions, the accuracy metrics were computed and are reported in Table II.

By using the prediction-based dissimilarity (Equation 10), the mean accuracy was 96.0% under direct comparison, 95.6% under neighbor comparison, and 13 perfect reconstructions.

TABLE III
ACCURACY COMPARISON.

Method	**D (%)**	**N (%)**	**# of perfect**
432-tile puzzles (28×28 pixels)			
PSQP with Eq. 10	96.0	95.6	13
PSQP with Eq. 13	95.6	95.4	13
Pomeranz et al. [9]	91.0	94.0	13
Gallagher [10]	95.3	95.1	12
Sholomon et al. [11] (avg.)	82.9	95.7	7

By using MGC (Equation 13), the mean accuracy was 95.6% under direct comparison, 95.4% under neighbor comparison, and 13 perfect reconstructions

The mean time to obtain the final permutation with the appropriate parameters is 1.8 minute per execution. Note that several sets of parameters can be tested in parallel, because the executions are independent. In this case, the total run-time for an image is 1.8 minute. In case sets are tested sequentially, the total run-time is 7 minutes. If the same result is obtained with consecutive sets, the best solution (the one with higher overall compatibility) so far is considered, without the need of testing other sets.

Higher accuracy can be obtained if other parameters sets are considered. However, if fewer sets are considered, for example, only $p = 1$ and $q = 1$, we also obtain good accuracy: 87.4% under direct comparison and 94.1% under neighbor comparison. Besides $p = 1$ and $q = 1$, if one more set is considered, for example, $p = 1$ and $q = 0.3$, we obtain 91.4% and 95.0%.

Table III summarizes the results obtained by related methods in the literature [9], [10], [11]. Some of these methods are not deterministic, needing 10 executions with random seeds, for each image, to yield the reported average accuracy. PSQP is deterministic because it does not require random seeds and it is independent of the initial permutation of tiles[1]. Some puzzles in which PSQP is superior are presented in Figure 8.

A. Puzzles with rectangular tiles

Another advantage of *PSQP* is the possibility of directly solving puzzles with rectangular tiles, not only square ones. For applications such as the reconstruction of fragmented paper, for example, this is an essential characteristic.

To test the accuracy of the method concerning the use of non-square tiles, we conducted an experiment with the same 20 images provided by [19], divided into 432 tiles, but now with 56×14 pixels each.

The obtained mean accuracy using the prediction-based dissimilarity was 89.7% under direct comparison and 95.2% under neighbor comparison. Figure 9 shows examples of puzzles with rectangular tiles solved by PSQP.

B. Puzzles with more tiles

We also executed PSQP with 40 additional images provided by [9], in order to test puzzles with 540 and 805 tiles, each one with 28×28 pixels.

[1]This is true for every image, except the ones with constant tiles. The permutation among them is taken into account to compute the metrics.

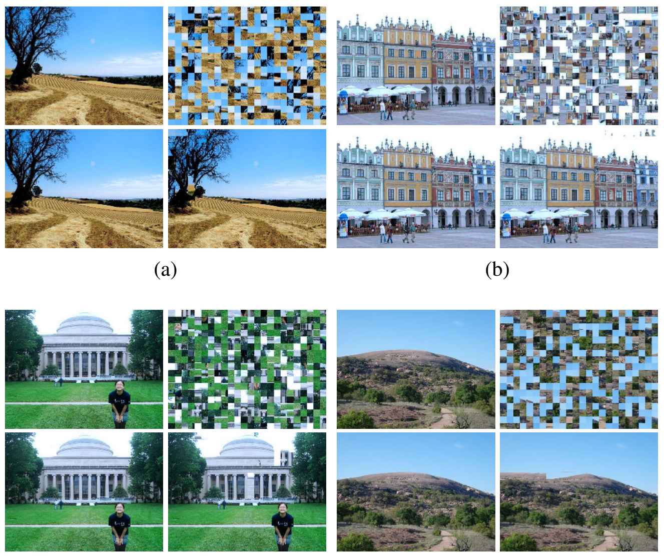
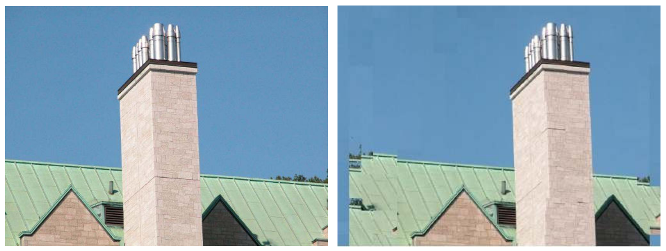

(a) (b) (c) (d)

Fig. 8. Jigsaw puzzles with 432 tiles of size 28×28 pixels. In each subfigure, we present the original image, the initial permutation for PSQP, PSQP final permutation, and the result obtained by [9].


(a) Puzzle with 540 tiles of size 28×28 pixels. Left: PSQP, with 100% accuracy; Right: Pomeranz et al. [9], 1% under direct comparison and 64% under neighbor comparison.

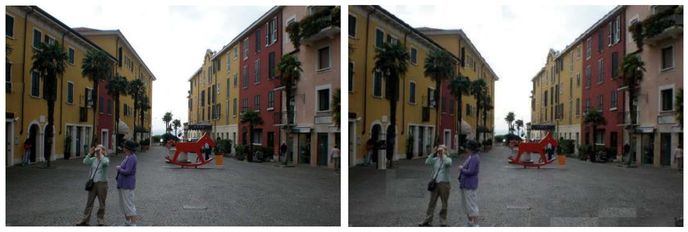
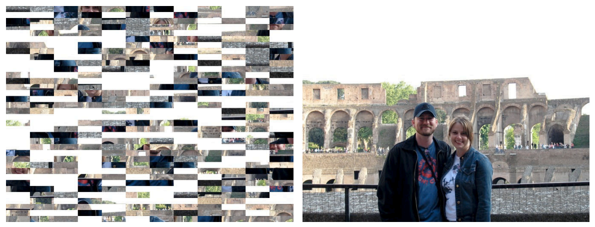

(b) Puzzle with 805 tiles of size 28×28 pixels. Left: PSQP, 91.9% under direct comparison and 90.6% under neighbor comparison; Right: Pomeranz et al. [9], 83.0% under both metrics.

Fig. 10. Puzzles with 540 and 805 tiles.


(a) Image 4. Accuracy: 66.15% under direct comparison and 65.50% under neighbor comparison. The non-constant part is perfectly reconstructed.

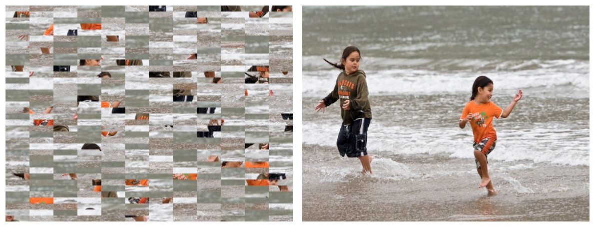
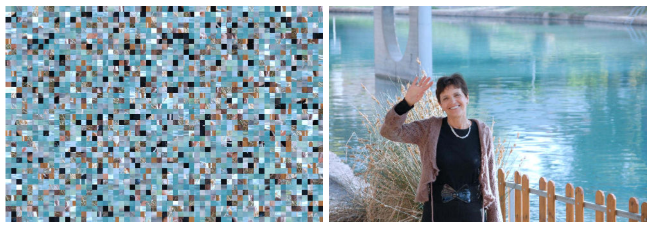

(b) Imagem 12. Accuracy: 100% under both metrics.

Fig. 9. Jigsaw puzzles with 432 tiles of size 56×14 pixels. In each subfigure, we present the initial permutation and the result.


(a) Puzzle with 2, 360 tiles.

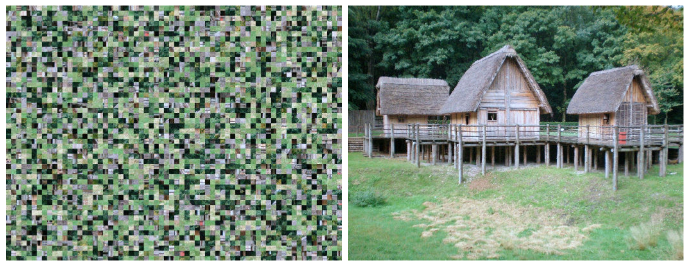

(b) Puzzle with 3, 300 tiles.

Fig. 11. Puzzles with tiles of size 28×28 pixels each. In each subfigure, we present the initial permutation and the result.

The obtained accuracy for 540-tile puzzles using Equation 10 was 90.6% under direct comparison, 95.3% under neighbor comparison, and 13 perfect reconstructions; and for 805-tile puzzles, 82.5%, 93.4%, and 8 perfect reconstructions.

Figure 10 presents some results for puzzles with 540 and 805 tiles. The results were reported for PSQP using only the prediction-based dissimilarity, since MGC did not provide further improvements.

Also considering images provided by [9], PSQP was tested with 2, 360- and 3, 300-tile puzzles, 28×28 pixels each tile. Figure 11 shows some reconstructions with 100% accuracy. Table IV summarizes all of the obtained results.

C. *Simulated strip-shredded paper*

PSQP is not limited to solving image puzzles. We demonstrate its usage in another domain: reconstruction of simulated strip-shredded documents.

In forensic investigations, investigators depend frequently on the preservation quality of a document, or image, to analyze or identify its contents. In some cases, such documents may be damaged, torn, or obliterated. The reconstruction process when they are torn, for example, can be done manually, which suggests a tedious and laborious work.

Recently, the DARPA challenge [23] fomented the formulation of novel methods to reconstruct real shredded papers, most of them requiring some kind of human intervention.

TABLE IV
ACCURACY COMPARISON.

Method	D (%)	N (%)	# of perfect
540-tile puzzles (28×28 pixels)			
PSQP	90.8	95.3	13
Pomeranz et al. [9]	83.5	90.9	9
Sholomon et al. [11] (avg.)	91.6	95.4	8
805-tile puzzles (28×28 pixels)			
PSQP	86.9	93.1	8
Pomeranz et al. [9]	80.3	89.7	7
Sholomon et al. [11] (avg.)	93.6	95.9	8
$2,360$-tile puzzles (28×28 pixels)			
PSQP	92.7	95.8	2
Pomeranz et al. [9]	33.4	84.7	1
Sholomon et al. [11] (avg.)	84.6	88.0	-
$3,300$-tile puzzles (28×28 pixels)			
PSQP	88.1	93.0	1
Pomeranz et al. [9]	80.7	85.0	1
Sholomon et al. [11] (avg.)	86.6	92.8	-

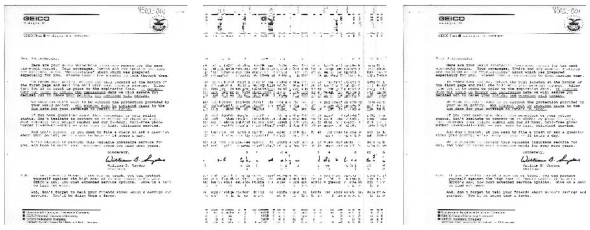

(a) Original image. (b) Initial permutation. (c) Result.

Fig. 12. A business letter divided into 90. Accuracy of 3.3% under direct comparison and 95.5% under neighbor comparison.

Although modern scenarios require the construction of documents cut with cross-cut shredders, or even by hand, many industrial-strength fast shredders are still strip-cut.

Here we perform a simulation and apply PSQP to automatically reconstruct documents fragmented into strips, improving the efficiency of the reconstruction when compared to the manual process.

We use 30 scans of documents divided into three categories: business letters, legal documents, and magazine pages. The scans were selected from the *ISRI-OCRtk* database [24], originally constructed to evaluate OCR methods.

Each scan had its background without text removed to contain only the document itself, and was divided into 28-pixel width strips, resulting into 80 to 90 strips per document (depending on its dimensions).

The method PSQP was executed for each image, using only two sets of parameters, $p = 1$ and $q = 0.3$; and $p = 3$ and $q = 3$, and using the prediction-based dissimilarity. Considering the whole document, the accuracy was 67.9% under direct comparison, 99.1% under neighbor comparison, and 19 perfect reconstructions. Note that the majority of documents contains a white background which generates constant strips. If we consider only the blocks of text, the accuracy is 100%, i.e, all reconstructions all readable. Figures 12 to 14 show some of the results obtained for each document category.

We then considered mixed strips from four business documents. All the documents are readable after running PSQP, as

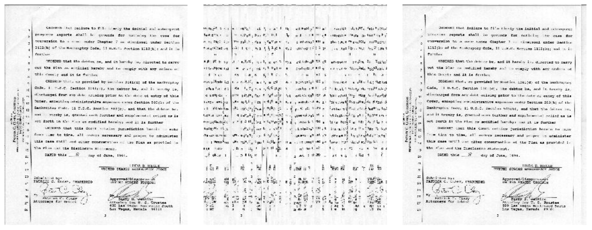

(a) Original image. (b) Initial permutation. (c) Result.

Fig. 13. A legal document divided into 90 strips. Accuracy of 0% under direct comparison and 98.9% under neighbor comparison.

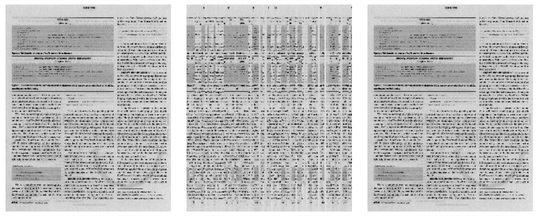

(a) Original image. (b) Initial permutation. (c) Result.

Fig. 14. A magazine page divided into 81 strips. Accuracy of 100% under direct comparison and 100% under neighbor comparison.

shown on Figure 15.

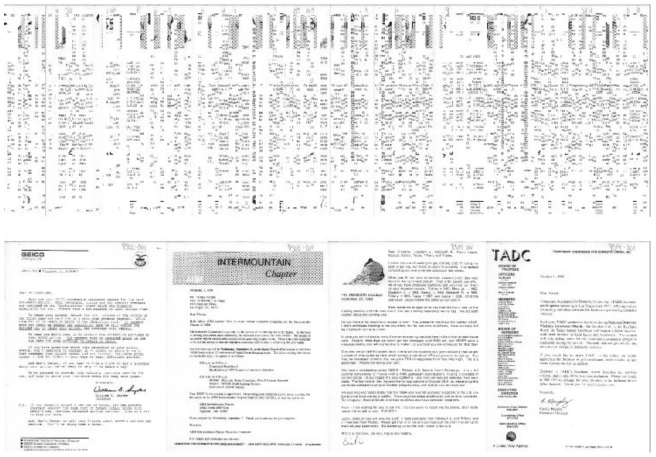


Fig. 15. PSQP applied to the reconstruction of mixed strips from four business documents. The top row shows the mixed strips and the bottom row shows the reconstruction.

VI. CONCLUSION

We presented the *Puzzle Solving by Quadratic Programming* (PSQP) method as a new formulation to solve image puzzles. In our formulation, a solution corresponds to the biunivocal association of tiles to locations, according to an energy function. The complexity of this hard combinatorial problem makes it infeasible in practice. So we reformulate it as a quadratic programming problem, which we solve using a Constrained Gradient Ascent algorithm.

PSQP was compared to related methods [9], [10], [11], providing, in several image puzzles, superior results according to the studied metrics. The sizes of puzzles that have been tested are: 432 square tiles, 432 rectangular tiles, 540, 805, $2,360$ and $3,300$ square tiles. We have also applied PSQP to the reconstruction of simulated stripped documents, which illustrated its versatility.

The proposed approach is fully automatic, and can solve puzzles with rectangular tiles of arbitrary shape. The method is deterministic and the reported accuracy is always guaranteed.

By analyzing the results, we observed that puzzles with constant color tiles (where all the pixels in the tile have exactly the same color) represent a fragility of PSQP method. Constant tiles are difficult to order globally, and so they cannot be considered as normal pieces in the resolution.

It was also observed that optimal parameters p and q for each image may be defined *a priori*, by analyzing the image and the tiles properties. These last two observations should be incorporated into future studies.

As future extensions to the method, we plan to incorporate some of these challenges to the problem formulation:

- In a puzzle some tiles may be missing. PSQP does not contemplate this challenge, because it does a biunivocal association between locations and tiles.
- Tiles from several puzzles may be mixed together. The method should be capable of presenting the resolution of every puzzle involved.
- Tiles may be rotated from its correct orientation. The method should be capable of determining the correct orientation.

Acknowledgment

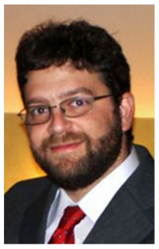
This work as primarily supported by CNPq (grants 201238/2010-1 and 308882/2013-0), with additional support by FAPERJ and CAPES (grant E-26/103.665/2012), NSF (grants IIS-0808718, CCF-0915661, and IIP-1330139), and FAPESP (grant 2012/50468-6).

References

[1] E. Demaine and M. Demaine, "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity," *Graphs and Combinatorics*, vol. 23, pp. 195–208, 2007.
[2] J. McBride and B. Kimia, "Archaeological fragment reconstruction using curve-matching," in *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2003.
[3] B. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich, "A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings," in *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008*, vol. 27, no. 3, 2008, pp. 84:1–84:9.
[4] E. Justino, L. Oliveira, and C. Freitas, "Reconstructing shredded documents through feature matching," *Forensic Science International*, vol. 160, no. 2, pp. 140–147, 2006.
[5] L. Zhu, Z. Zhou, and D. Hu, "Globally consistent reconstruction of ripped-up documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 1–13, 2008.
[6] Y. Zhao, M. Su, Z. Chou, and J. Lee, "A puzzle solver and its application in speech descrambling," in *International Conference on Computer Engineering and Applications (CEA)*, 2007, pp. 171–176.
[7] W. Marande and G. Burger, "Mitochondrial dna as a genomic jigsaw puzzle," *Science*, vol. 318, no. 5849, p. 415, 2007.
[8] T. Cho, S. Avidan, and W. Freeman, "The patch transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1489–1501, 2010.
[9] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 9–16.
[10] A. Gallagher, "Jigsaw puzzles with pieces of unknown orientation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 382–389.
[11] D. Sholomon, O. David, and N. Netanyahu, "A genetic algorithm-based solver for very large jigsaw puzzles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1767–1774.
[12] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 2, pp. 118–127, 1964.
[13] G. Burdea and H. Wolfson, "Solving jigsaw puzzles by a robot," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 6, pp. 752–764, 1989.
[14] J. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 29–44, 1987.
[15] D. Goldberg, C. Malon, and M. Bern, "A global approach to automatic solution of jigsaw puzzles," in *18th Annual ACM Symposium on Computational Geometry (SoCG)*, 2002, pp. 82–87.
[16] H. Leitão and J. Stolfi, "A multi-scale method for the re-assembly of fragmented objects," in *British Machine Vision Conference (BMVC)*, 2000, pp. 705–714.
[17] D. Kosiba, P. Devaux, S. Balasubramanian, T. Gandhi, and K. Kasturi, "An automatic jigsaw puzzle solver," in *12th International Conference on Pattern Recognition (IAPR)*, 1994, pp. 616–618.
[18] T. Nielsen, P. Drewsen, and K. Hansen, "Solving jigsaw puzzles using image features," *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1924–1933, 2008.
[19] T. Cho, S. Avidan, and W. Freeman, "A probabilistic image jigsaw puzzle solver," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 183–190.
[20] K. Son, J. Hays, and D. Cooper, "Solving square jigsaw puzzles with loop constraints," in *European Computer Vision Conference (ECCV)*, 2014, pp. 32–46.
[21] E. Seneta, *Non-negative matrices and Markov chains*. Springer Verlag, 2006.
[22] J. Rosen, "The gradient projection method for nonlinear programming," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960.
[23] "DARPA shredder challenge," http://www.shredderchallenge.com/, 2008.
[24] T. Nartker, S. Rice, and S. Lumos, "Software tools and test data for research and testing of page-reading ocr systems," in *Document Recognition and Retrieval XII*, 2005, pp. 37–47.

Fernanda A. Andaló is a Postdoctoral Researcher at the Institute of Computing, University of Campinas (UNICAMP), Brazil. She received a B.Sc. in Computer Science from University of Brasília (UnB) in 2004, a M.Sc. in Computer Science from UNICAMP in 2007, and a Ph.D. in Computer Science from the same university in 2012, during which she was a visiting researcher at the Division of Engineering, Brown University, for a year. She was a postdoctoral researcher at VISGRAF Laboratory, National Institute of Pure and Applied Mathematics (IMPA) in 2013, and during 2014-2015 she worked at the Samsung Research Institute Brazil as a research scientist. Her research interests include Computer Vision, Image Processing, and Computer Graphics.

Gabriel Taubin earned a Licenciado en Ciencias Matematicas degree from the University of Buenos Aires, Argentina, and a Ph.D. degree in Electrical Engineering from Brown University. In 1990 he joined the IBM Research Division, where he held various positions, including Research Staff Member and Research Manager. In 2003 Taubin joined the Brown University School of Engineering as an Associate Professor of Engineering and Computer Science. During the 2000-2001 academic year, on sabbatical from IBM, he was Visiting Professor of Electrical Engineering at the California Institute of Technology. During the Spring semester of 2010, on sabbatical from Brown, he was Visiting Associate Professor of Media Arts and Sciences at MIT. Prof. Taubin was the Editor-in-Chief of the IEEE Computer Graphics and Applications Magazine from 2010 to 2013, and has served as a member of the Editorial Board of the Geometric Models journal, and as associate editor of the IEEE Transactions of Visualization and Computer Graphics. Prof. Taubin was named IEEE Fellow for His Contributions to the development of three-dimensional geometry compression technology and multimedia standards, won the Eurographics 2002 Gnter Enderle Best Paper Award, and was named IBM Master Inventor. He has contributed to the field called Digital Geometry Processing with methods to capture 3D shape, for surface reconstruction, geometric modeling, geometry compression, progressive transmission, signal processing, and display of discrete surfaces. The 3D geometry compression technology that he have developed at IBM was incorporated into the MPEG-4 standard, and became an integral part of IBM products.



Siome K. Goldenstein is an Associate Professor at the Institute of Computing, University of Campinas, UNICAMP, Brazil, and a senior IEEE member. He received an Electronic Engineering degree from the Federal University of Rio de Janeiro in 1995, an M.Sc. in Computer Science from the Pontifical Catholic University of Rio de Janeiro in 1997, and a Ph.D. in Computer and Information Science from University of Pennsylvania in 2002. In 2003, he was a postdoctoral fellow at the CBIM Center, at Rutgers University, and during 2010–2012 he was a Visiting Associate professor at the Division of Engineering, Brown University. He is an Area Editor of IEEE Transactions on Information Forensics and Security (T.IFS), Elsevier's Computer Vision and Image Understanding (CVIU), and Elsevier's Graphical Models (GMOD). His interests lie in computational forensics, computer vision, computer graphics, and machine learning.