Solving Image Puzzles with a Simple Quadratic Programming Formulation

Conference Paper · August 2012

DOI: 10.1109/SIBGRAPI.2012.18

3 authors, including:

Fernanda A. Andaló
LEGO / SciPet
35 PUBLICATIONS **238** CITATIONS
SEE PROFILE

Gabriel Taubin
Brown University
231 PUBLICATIONS **11,133** CITATIONS
SEE PROFILE

Solving Image Puzzles with a Simple Quadratic Programming Formulation

Fernanda A. Andaló[1], Gabriel Taubin[2], Siome Goldenstein[1]
[1] Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil
{feandalo,siome}@ic.unicamp.br
[2]Division of Engineering, Brown University, Providence, RI, USA
taubin@brown.edu

***Abstract*—We present a new formulation to automatically solve jigsaw puzzles considering only the information contained on the image. Our formulation maps the problem of solving a jigsaw puzzle to the maximization of a constrained quadratic function that can be solved by a numerical method. The proposed method is deterministic and it can handle arbitrary rectangular pieces. We tested the validity of the method to solve problems up to 3300 puzzle pieces, and we compared our results to the current state-of-the-art, obtaining superior accuracy.**

***Keywords*-image puzzle; jigsaw puzzle; image analysis; quadratic programming**

I. Introduction

In the well-known *jigsaw puzzle* problem, numerous non-overlapping tiles have to be assembled together, according to a color pattern or shape fitting, with the goal of reconstructing a single plane or image. Although this problem has been proven to be *NP*-complete when the affinity between the tiles is uncertain [1], several scientific challenges such as the reconstruction of documents from shredded paper [2], and re-assembling broken archeological artifacts from fragments [3], can be reformulated as 2D or 3D jigsaw puzzle problems.

In this paper we focus on the problem of reconstructing images from identically shaped rectangular tiles placed without repetition within a regular rectangular grid of known dimensions. Contrary to what occurs in traditional jigsaw puzzles, here the tile shape does not provide any information, making the problem even more challenging.

In solving this kind of problem, we first need to deal with its combinatorial nature: since a tiling can be described as a permutation of the tiles within the rectangular grid, the number of possible tilings grows exponentially as a function of the number of tiles. In addition, since the problem is global in nature we seek local measures of pairwise tile matching to help reduce the complexity of the search. However, no such rule based solely on local boundary tile similarity is known to date.

Automatic solvers for jigsaw puzzles have been proposed since 1954, when the first method was presented by Freeman and Garder [4]. It was able to solve apictorial puzzles with 9 fragments by analyzing critical points on the border of the puzzle pieces.

Since then and based on the first method, several other methods focused on matching the shape of the tiles only [5]. Kosiba et al. [6] were the first to consider not only the shape of the tiles, but also the content of the image. In their method, the matching process between the tiles considers many characteristics: color samples along the borders, curvature parameters, and the concavity and convexity of the tiles.

Nielsen et al. [7] proposed the first solver to assemble successfully puzzles without shape information. The method was able to solve puzzles with 320 square tiles using a greedy approach.

Two recently proposed methods [8], [9] solve the square jigsaw puzzle considering a pairwise compatibility metric between tiles which compares the color information on the shared boundary of two tiles.

Cho et al. [8] presents a solver for puzzles with 432 tiles based on maximizing a probability function via loopy belief propagation. Since they don't have local evidence for their graphical model, they rely on knowing the placement of some tiles to solve the problem.

Pomeranz et al. [9] presented the current state-of-the-art solver. By assuming a puzzle with square tiles, they solve problems with up to 3300 tiles using a greedy approach. However, this method requires solving each puzzle several times starting from different random seeds to obtain good results up to certain accuracy.

In this paper we propose a simple quadratic programming formulation to solve jigsaw puzzles with identically shaped rectangular tiles. We show that, for the same image sets, we can achieve superior accuracy over the current state-of-the-art method. Our method will be referenced as *PSQP – Puzzle Solving by Quadratic Programming*.

II. Problem definition

First consider an image partitioned into a regular 2D grid of size $N_{cols} \times N_{rows}$, forming N tiles $t_1, \ldots, t_N$, of identical dimensions. Now consider an empty grid of the same size as the previous one with N locations labeled $1, \ldots, N$. The problem is to determine a one-to-one correspondence between the N tiles and the N locations, optimal with respect to certain properly constructed global matching function. Since this correspondence can be described by a permutation π of the N tiles, the problem to be solved reduces to a discrete optimization problem over the finite group of permutations of N elements.

We organize the locations as a directed graph $G = \{V, E = E_H \cup E_V\}$, where the vertices are the tile locations, $V = \{1, \ldots, N\}$, and the set of edges E comprises all pairs of neighboring tile locations. E_H and E_V denote the set of horizontal and vertical neighboring locations, respectively. G must be a direct graph because in general, swapping two tiles from neighboring locations should result in a change in the global matching function.

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

We consider the following global matching function of a permutation π

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

where $e = (i, j)$ is the edge connecting the neighboring locations i and j, and $\pi(i)$ can be regarded as a 1-1 mapping which assigns the tile $t_{\pi(i)}$ to the location i.

Our goal is to maximize this function over all the permutations π of N elements. Since this is a hard combinatorial optimization problem, we first extend the domain of the global matching function to the set of doubly stochastic matrices, and we reformulate the problem as a constrained continuous optimization problem, which we solve using numerical methods. Then we describe how the coefficients of the matrices C_H and C_V are computed so that the solution of the continuous optimization problem correlates well with the original combinatorial optimization problem.

III. Global Matching Function

In this section we show how Equation 1 can be reformulated as a homogeneous quadratic function of a square matrix, which allows us to relate the problem to a simpler continuous optimization problem. First of all, each permutation π of N elements can be represented as a permutation matrix, i.e., a binary square matrix P with exactly one entry equal to 1 in each row and in each column:

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

With this notation we can reformulate the global function as follows

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

where a generic term $(P^\top CP)_{ij}$, corresponding to the 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 of $\varepsilon(P)$ is also a homogeneous non-negative quadratic function of P. If we represent the columns of the $N \times N$ matrix P as a vector p of dimension N^2, we get

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

where p is the vertical concatenation of the columns, $p_1, \ldots, p_N$ of P. We can reformulate Equation 4 in the canonical form $p^\top Ap$, where A is a symmetric $N^2 \times N^2$ matrix, representing the Hessian of $\varepsilon(P)$. In vector form and in coordinates:

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

Even though in practice we never construct the matrix A explicitly, an analytic expression can be obtained. Since for a generic matrix C we can write

$$(P^\top CP)_{ij} = \sum_{k=1}^{N}\sum_{l=1}^{N} P_{ki} C_{kl} P_{lj}, \tag{6}$$

we have

$$\begin{aligned} \varepsilon(P) = &\sum_{(i,j)\in E_H}\sum_{k=1}^{N}\sum_{l=1}^{N} P_{ki} C_{H_{kl}} P_{lj} \\ &+ \sum_{(i,j)\in E_V}\sum_{k=1}^{N}\sum_{l=1}^{N} P_{ki} C_{V_{kl}} P_{lj}. \end{aligned} \tag{7}$$

And it follows that the coefficients of the matrix A can be accumulated by a simple linear traversal of matrices C_H and C_V. Fig. 1 illustrates the problem formulation.

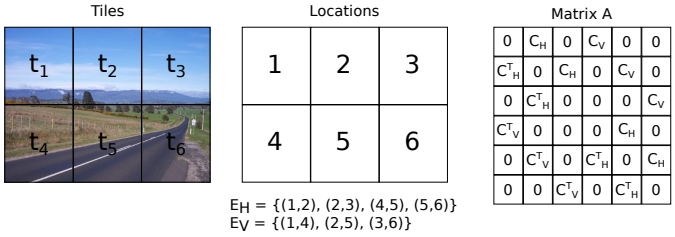


Fig. 1. Problem formulation. From left to right: tiles, locations with the two edge sets, and matrix A represented as a block matrix. $C_H^\top$ and $C_V^\top$ are the transpose of matrices C_H and C_V, respectively, and each block of matrix A is a 6×6 matrix.

IV. Constrained Gradient Ascent

Permutation matrices are special cases of doubly stochastic matrices [10]. A doubly stochastic matrix is a non-negative matrix such that the sum of all the elements in each row is equal to 1, and the sum of all the elements in each column is also equal to 1. In fact, the set of doubly stochastic matrices is the convex hull of the permutation matrices within the set of $N \times N$ matrices. Each doubly stochastic matrix satisfies N^2 inequality constraints, which specify that the elements P_{ij} of P are non-negative; and $2N$ equality constraints, which specify that the sum of the rows and columns of P are equal to 1. By extending the domain of $\varepsilon(P)$ to all the

doubly stochastic matrices, the problem reduces to solving the following quadratic optimization problem

$$\begin{aligned} &\text{Maximize } f(p) = p^\top A p, \\ &\text{subject to } P\mathbb{1} = \mathbb{1}, P^\top \mathbb{1} = \mathbb{1}, \text{ and } p_{ij} \geq 0, \end{aligned} \tag{8}$$

where $\mathbb{1}$ is a column vector of size N with all elements equal to one. We use a constrained gradient ascent algorithm, with gradient projection [11], to search for local maxima of this problem.

Note that even though the objective function $f(p)$ is positive on the feasible set, it is not necessarily concave because matrix A is not positive definite: all the diagonal values of A are 0 in our formulation, which violates the necessary conditions for positive definiteness, and also for positive semi-definiteness. Therefore, we cannot guarantee $f(p)$ to attain a maximum at a permutation matrix. But in practice, we observe that we can get as close as possible to a solution by working with the constraints.

To maximize Equation 8, we propose a modified constrained gradient ascent approach, with gradient projection [11]. To locate a local maxima of a function, we need to update the variables in steps proportional to the gradient at the current point, while projecting the gradient.

In our approach we maintain a set of active variables. An inactive variable is one that is at the boundary of the feasible region and cannot be further updated. The method starts from $p_{kl} = \frac{1}{N}, 1 \leq k, l \leq N$, and with all variables active, $active_{kl} = true$, where $active$ indicates if a variable is active or not. The ascent direction, $d = \nabla f(p) = A * p$, at the current estimate p, in general does not satisfy the linear constraints, so we must project it onto the space orthogonal to the subspace defined by the linear equality constraints [11], resulting in the constrained ascent direction c. Fig. 2 illustrates a 2D simplification of the process of projecting the gradient.

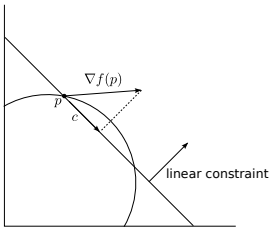


Fig. 2. Projection of the gradient onto the space orthogonal to the space defined by the linear equality constraints.

With the constrained ascent direction c, the method can update the previously feasible point to a new feasible point p: $p_{kl} = p_{kl} + step * c$, for $1 \leq k, l \leq N$, and $active_{kl} = true$, where $step$ is the maximum value so that $0 \leq p_{kl} \leq 1$.

When one of the variables reaches the boundary of the feasible region, we should update the constraints so that this variable stays at the boundary. However, in practice, maintaining a group of modifying and orthogonal constraints implies high computational costs and storage. Instead, we re-initialize p every time there is no direction to maximize the energy inside the feasible region. In order to do this, we deactivate the variables that are on the limit of the feasible region, i.e., the ones that are equal to either zero or one. The process of deactivating a variable on the upper limit corresponds to assigning the corresponding tile to the most probable location. We then restart p without the inactive variables, i.e., $p_{kl} = \frac{1}{N - nFixedTiles}$, for $1 \leq k, l \leq N$, and $active_{kl} = true$, where $nFixedTiles$ is the number of tiles that have been assigned to a location. Then we repeat these steps until all the tiles have been assigned to a location.

Algorithm 1 shows the pseudo-code for the ascent gradient approach that maximizes Equation 8.

Algorithm 1 Constrained Gradient Ascent.

Input: Compatibility matrices C_H and C_V; and the number of tiles N.

Output: Permutation π of tiles.

```
nFixedTiles ← 0; # Number of tiles that have been assigned to a location.
active_kl ← true, for 1 ≤ k, l ≤ N; # Active variables.
while nFixedTiles < N do
    p_kl ← 1/(N − nFixedTiles), for 1 ≤ k, l ≤ N and active_kl = true;
    d ← ∇f(p) ← A * p; # Ascent direction.
    c ← Kd; # Ascent direction c. K is the projection matrix.
    p_kl ← p_kl + step * c, for 1 ≤ k, l ≤ N and active_kl = true;
    for all p_kl = 0, with 1 ≤ k, l ≤ N and active_kl = true do
        active_kl ← false;
    end for
    for all p_kl = 1, with 1 ≤ k, l ≤ N and active_kl = true do
        active_kl ← false;
        π(l) ← k;
        nFixedTiles ← nFixedTiles + 1;
    end for
end while
```

V. Compatibility between tiles

The compatibility between pairs of tiles has been studied before [8], [9], and plays an important role in solving the image puzzle. Demaine et al. [1] showed that if it is locally possible to tell whether two tiles fit together in the final solution, then trying to join together all pairs of tiles in a greedy manner solves the puzzle in polynomial time. But, in natural images, it is easy to find examples of tiles with ambiguous neighboring tiles. In this work, we consider the prediction-based compatibility proposed by Pomeranz et al. [9]. The horizontal dissimilarity between a left hand side tile t_i and

right hand side tile t_j is defined as

$$Pred_{H_{ij}} = \left[\sum_{k=1}^{T} \sum_{l=1}^{3} (\mid 2 * t_i(k,T,l) - t_i(k,T-1,l) - t_j(k,1,l) \mid)^p + (\mid 2 * t_j(k,1,l) - t_j(k,2,l) - t_i(k,T,l) \mid)^p \right]^{\frac{q}{p}}, \quad (9)$$

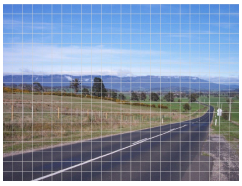
where tiles t_i and t_j are regarded as $T \times T \times 3$ matrices, variables p and q are tunable parameters, and the color difference is measured in the normalized LAB color space. The vertical matching error $D_{V_{ij}}$ is computed in a similar fashion.

Based on the predicted values, the compatibility between tiles t_i and t_j is defined as [9]

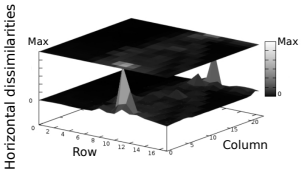
$$C_{H_{ij}} \propto exp\left(-\frac{Pred_{H_{ij}}}{quartile(i)}\right), \quad (10)$$

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

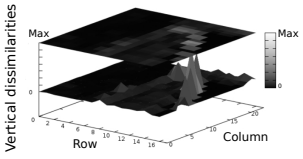
Unfortunately, the local matching dissimilarity (Equation 9) does not provide enough information to solve the puzzle globally. This is illustrated in Fig. 3, where it is shown the dissimilarities considering only correctly assigned neighboring tiles. We can see that in some constant parts of the image (the sky, for example), the dissimilarity among all tiles is lower that in other non-constant parts and thus they are not comparable. The problem gets worse when we consider the errors between every possible pair of tiles.



(a) Correctly assigned tiles.



(b) Horizontal dissimilarities.



(c) Vertical dissimilarities.

Fig. 3. Dissimilarity between correctly assigned tiles.

Due to this difficulty, we present a new compatibility measure, based on [9], that imposes a stronger global order to the tiles' dissimilarities. The horizontal compatibility between tiles t_i and t_j is defined as

$$C_{H_{ij}} \propto exp\left(-\varphi(i) - \frac{Pred_{H_{ij}}}{quartil(i)}\right), \quad (11)$$

where $\varphi(i)$ is used to impose a global order to the dissimilarities. It is the position of tile t_j in respect to t_i, when all dissimilarities $Pred_{H_{ik}}$, for $1 \leq k \leq N$, are ordered, added to the position of tile t_i in respect to t_j, when all dissimilarities $Pred_{H_{kj}}$, for $1 \leq k \leq N$, are ordered. For example, if tile t_j is the second tile closest to t_i (based on their dissimilarity) and t_i is the first tile closest to t_j on the opposite border, then $\varphi(i) = 3$. The vertical compatibility is computed using the same idea.

There are two other differences on the predictions' computation compared to the work of Pomeranz et al. [9]. First, the color differences are computed in the YIQ color space instead of the LAB space. For best results, the channels have been normalized to have the same variance.

Pomeranz et al. [9] studied the dissimilarity measure varying parameters p and q and concluded that there are optimal values to be fixed and used across all image puzzles. However we observed that, for our method, there are optimal p and q for each image and the values can greatly influence the final permutation. For this reason, in this work we test several sets of parameters, $p \in \left[\frac{3}{10}, 3\right]$ and $q \in \left[\frac{1}{20}, \frac{10}{3}\right]$, and we choose the set that yields the highest global matching value. This range of parameters was defined running the method *PSQP* on 20 training images randomly chosen from the internet.

VI. Implementation

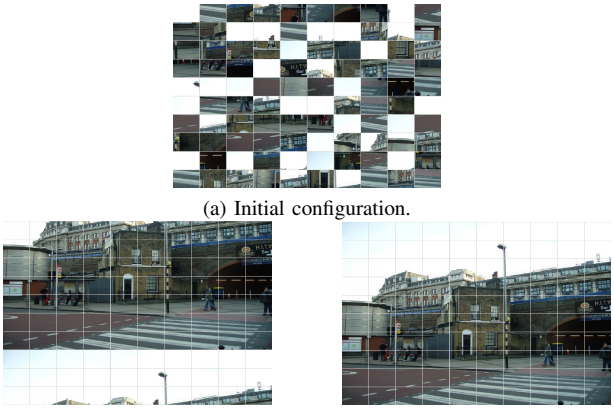
In the implementation of the gradient ascent method, the memory footprint is a major concern, because matrices C_H and C_V, vector p, and the ascent direction c have $N \times N$ entries each. To save up memory, we observed that the term $\varphi(i)$ (Equation 11) makes the compatibility values really small when distant neighbors of t_i are considered. Thus, using a safe threshold (10^{-6}) we can zero out compatibility values that are already almost zero. By doing this, matrices C_H and C_V become sparse. In the optimal case, C_H will have $N_{rows}(N_{cols} - 1)$ non-zero entries and C_V will have $N_{cols}(N_{rows} - 1)$ non-zero entries, a drastically reduction on memory usage.

In terms of computational complexity, our algorithm runs in quadratic time in the number of tiles, i.e, *PSQP* is $O(n^2)$. The ascent direction computation is done by traversing matrices C_H and C_V, and the projection of the ascent direction is done by traversing the corresponding vector two times.

In practice, we have two problems that were not discussed before. First, the constant tiles – group of tiles that have equal feature vectors on all sides – impose a hard problem to solve. These tiles have total compatibility among them and to the neighboring non-constant tiles. To address this, we simply do not take them into account by zeroing out their compatibility. By doing this, the constant tiles will fit in the holes left by the optimization process. Note that, because they are equal, it doesn't matter which permutation will be adopted among them.

The second problem is the non-concavity property of the energy function that we want to maximize. There is no guarantee that the maximum provided by the Constrained Gradient Ascent algorithm is the global maximum, only a local one. For a few puzzles, especially the ones that contain

constant tiles (and the compatibility values were altered), the final permutation does not represent a global maximum, but a local maximum that is a shift of the global permutation (Fig. 4).



(a) Initial configuration.

(b) Permutation associated with a local maxima.

(c) Permutation associated with the global maxima.

Fig. 4. Example of the non-concavity property of the global matching function. Note that this puzzle contains several constant (white) tiles.

In such cases, the method performs an ultimate step to adjust the permutation shift. The global matching function $\varepsilon(\pi)$ is computed for every possible permutation resulting from shifting the final π in every row and column, and with p and q equal to one.

The formulation of the entire method is present in Algorithm 2, where function $ConstrainedGradientAscent$ gives the permutation π according to Algorithm 1; $GlobalMatching$ computes the global matching value $\varepsilon(\pi)$ for permutation π after shifting it by s_H horizontally and s_V vertically; and finally function $Shift$ applies, to π, the shift that generates the highest global matching value horizontally ($s_{H_{max}}$) and vertically ($s_{V_{max}}$).

Algorithm 2 *PSQP* method.

$\pi \leftarrow ConstrainedGradientAscent(C_H, C_V, N)$;

$s_{H_{max}}, s_{V_{max}} \leftarrow \underset{\substack{s_H \in [0, N_{cols}-1], \\ s_V \in [0, N_{rows}-1]}}{\text{argmax}} GlobalMatching(\pi, s_H, s_V)$;

$\pi_{final} \leftarrow Shift(\pi, s_{H_{max}}, s_{V_{max}})$

Note that this last step is linear on the number of tiles.

VII. Experimental Results

To compare *PSQP* to the recently proposed state-of-the-art method [9], we use the same database of 20 images provided by [8], where each puzzle consists of 432 tiles of 28×28 pixels. We also consider two performance metrics presented in [8]:

Direct comparison: the obtained permutation is compared directly to the ground-truth permutation. This metric calculates the ratio between the number of tiles in the obtained solution that are assigned to the correct location and the total number of tiles.

Neighbor comparison: for each assigned tile, this metric computes the fraction of its correctly assigned neighboring tiles (tiles that are adjacent in the correct assignment). The reconstruction accuracy is the average fraction of correct neighboring tiles.

PSQP was implemented in *C++* and, for each image, we executed it several times to test different sets of parameters: $p = \frac{3}{10}$ and $q = \frac{1}{20}$; $p = 1$ and $q = \frac{1}{6}$; $p = 1$ and $q = 1$; $p = 1$ and $q = \frac{10}{3}$; and $p = 3$ and $q = 3$. The sets are tested in order and, when the same result is achieved with different sets, the best solution so far is considered (there is no need to continue testing other sets in this case). For each execution, the two performance metrics were computed. Table I shows the results with the best set of parameters for each image. The experiment was executed in a 2 GHz machine with 6 GB of RAM memory.

TABLE I
PERFORMANCE METRICS COMPUTED FOR EACH OF THE TWENTY IMAGES. D STANDS FOR *Direct comparison* AND N FOR *neighbor comparison*. THE RESULTS OF POMERANZ ET AL.'S METHOD ARE REPORTED IN THE SUPPLEMENTAL MATERIAL OF THE RELATED PUBLICATION [9].

	PSQP		**Pomeranz et al. [9]**	
Imagem	**D (%)**	**N (%)**	**D (%)**	**N (%)**
1	88.5	85.5	77	80
2	83.2	82.1	82	81
3	100.0	100.0	100	100
4	65.9	65.0	2	67
5	100.0	100.0	100	100
6	98.4	98.3	100	100
7	100.0	100.0	84	86
8	100.0	100.0	100	100
9	100.0	100.0	100	100
10	100.0	100.0	100	100
11	100.0	100.0	100	100
12	99.5	99.4	100	100
13	88.9	87.8	87	86
14	100.0	100.0	100	100
15	95.6	94.2	90	89
16	100.0	100.0	100	100
17	100.0	100.0	97	96
18	100.0	100.0	100	100
19	100.0	100.0	100	100
20	100.0	100.0	100	100
Mean	96.0	95.6	91	94

The average performance is 96.0% under Direct comparison and 95.6% under Neighbor comparison. The average running time to obtain the final permutation with the correct parameters is 3.5 minutes per execution. To perform the parameters testing and to obtain the final permutation, the average time per image is 12 minutes.

It is important to note that higher accuracy can be obtained if other parameters are tested. But if fewer sets are tested, for example only $p = 1$ and $q = \frac{1}{6}$, we still get high accuracy: 91.7% and 94.3%. If we add another set, for example $p = 1$ and $q = \frac{10}{3}$, we get 95.1% and 95.3%.

The reported accuracy for the method of Pomeranz et al. [9] is 91% and 94%[1]. To achieve this accuracy, the method needs to be executed 10 times with random seeds, and the best result according to a metric is selected. The reported execution time is 1.2 minute in average per image in a 3.2 GHz machine with 4 GB of RAM memory. To compare the running times, we re-executed their experiments in the same machine used here. With the MATLAB code provided by the authors, we obtained an average of 1.5 minute per execution, adding up to 15 minutes to obtain the final solution for each image.

The method of Pomeranz et al. [9] is probabilistic, i.e, with 10 executions of the method, there is no guarantee that the reported accuracy will be achieved. For example, to obtain the same level of accuracy, we had to run the method more than 10 times for images 9 and 12. *PSQP* is deterministic, i.e, it yields the same result independent of the initial configuration of the puzzle[2] and without random seeds.

Fig. 5 shows some image puzzles in which *PSQP* is more accurate according to both metrics, and Fig. 6 shows some image puzzles in which the method of Pomeranz et al. [9] is more accurate.

Another advantage of *PSQP* is that it can solve puzzles with rectangular (not only square) tiles. This is an important characteristic when using automatic solvers for shredded document reconstruction, for example. The same experiment was repeated, but now with puzzles consisting of 432 tiles of 56×14 pixels each. The average performance is 89.7% under Direct comparison and 95.2% under Neighbor comparison. Fig. 7 shows some of the results obtained with non-square tiles.

PSQP can also reconstruct larger jigsaw puzzles. We used 40 additional images provided by [9] to test the method with 540 and 805-piece puzzles. The overall performance for 540-piece puzzles is 90.6% and 95.3%, and for 805-piece puzzles the performance is 82.5% and 93.4%, under direct and neighbor comparison, respectively. Pomeranz et al. [9] reported the accuracy of 83.5% and 90.9% for 504-pieces, and 80.3% and 89.7% for 805-pieces, under direct and neighbor comparison, respectively, using the same database. Fig. 8 shows reconstructed images for two of these images.

Also using images provided by [9], we tested *PSQP* with 2360 and 3300-piece puzzles (Fig. 9), which resulted in reconstructions with 100% accuracy.

VIII. Conclusion

We introduced a new formulation for solving image jigsaw puzzle problems, the method *PSQP – Puzzle Solving by Quadratic Programming*. In our formulation, a solved puzzle is a one-to-one assignment of tiles to locations, according to a energy function. Since this is a hard combinatorial problem, we reformulate it as a quadratic programming approach, where we can find an approximate solution by means of a gradient ascent algorithm.

We compared *PSQP* to the current state-of-the-art and it provided superior results according to the used metrics. *PSQP* also has some advantages. First, it can solve puzzles not only with square tiles, but also with rectangular ones. Second, it is deterministic and although several parameter sets have to be tested, the method always yields the same results, while the current state-of-the-art method has to be executed several times to attain a certain accuracy. For the size of the puzzles tested, *PSQP* is faster, considering all the necessary executions in both methods.

By analyzing the results, we observed that image puzzles that contain constant tiles are a weakness of *PSQP*. Constant tiles are difficult to order in a global sense, so we cannot consider them as a normal piece. We also observed that the right parameter set for each image may be determined *a priori* by analyzing the image and tiles properties. These two observations will be included in future studies.

[1]These accuracy values were reported in the supplemental material of [9] and confirmed by the re-execution of the experiments.

[2]This is true for all images, except the ones with constant tiles. The permutation of the constant tiles is taken into consideration when computing the performance metrics.

Acknowledgment

This work is primarily supported by CNPq grant 201238/2010-1, with additional funding from NSF (grants IIS-0808718, CCF-0729126, and CCF-0915661), CNPq (grants 309254/2007-8, 551007/2007-9, 551623/2009-8 and 200717/2010-3), FAPESP, and CAPES.

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] E. Justino, L. Oliveira, and C. Freitas, "Reconstructing shredded documents through feature matching," *Forensic science international*, vol. 160, no. 2, pp. 140–147, 2006.

[3] J. McBride and B. Kimia, "Archaeological fragment reconstruction using curve-matching," in *Conference on Computer Vision and Pattern Recognition Workshop. (CVPRW)*, vol. 1, 2003, pp. 3–3.

[4] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Transactions on Electronic Computers*, no. 2, pp. 118–127, 1964.

[5] D. Goldberg, C. Malon, and M. Bern, "A global approach to automatic solution of jigsaw puzzles," in *Proceedings of the eighteenth annual symposium on Computational geometry*, 2002, pp. 82–87.

[6] D. Kosiba, P. Devaux, S. Balasubramanian, T. Gandhi, and K. Kasturi, "An automatic jigsaw puzzle solver," in *Proceedings of the 12th International Conference on Pattern Recognition (IAPR)*, vol. 1, 1994, pp. 616–618.

[7] T. Nielsen, P. Drewsen, and K. Hansen, "Solving jigsaw puzzles using image features," *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1924–1933, 2008.

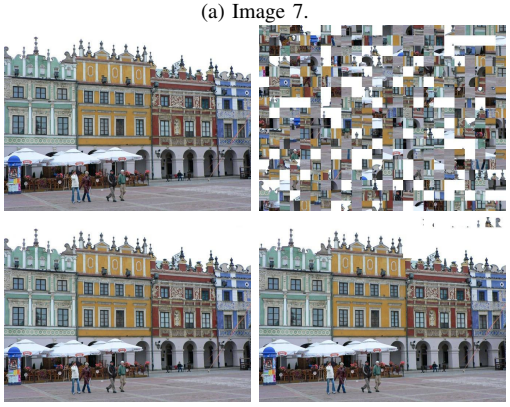
[8] T. Cho, S. Avidan, and W. Freeman, "A probabilistic image jigsaw puzzle solver," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 183–190.

[9] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 9–16.

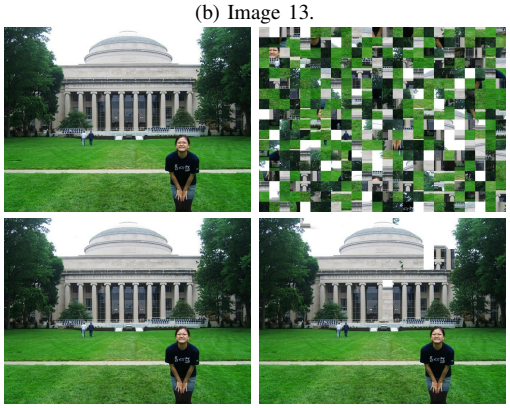
[10] E. Seneta, *Non-negative matrices and Markov chains*. Springer Verlag, 2006.

[11] J. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960.

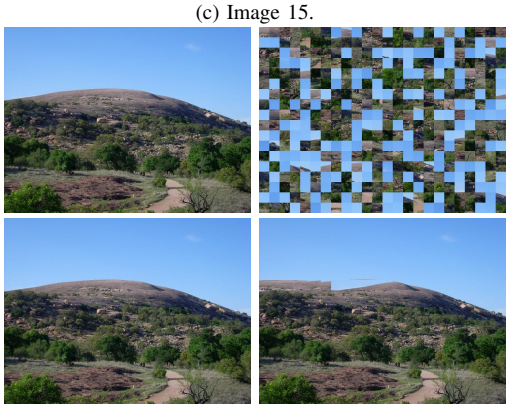
(a) Image 7.



(b) Image 13.

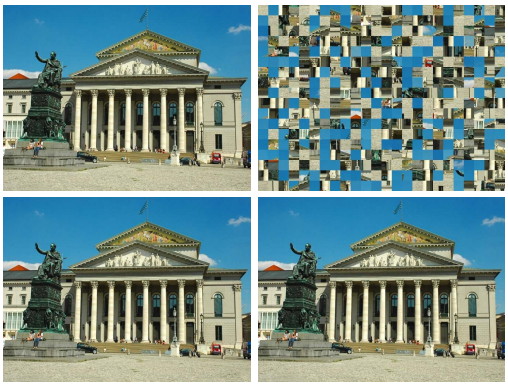


(c) Image 15.

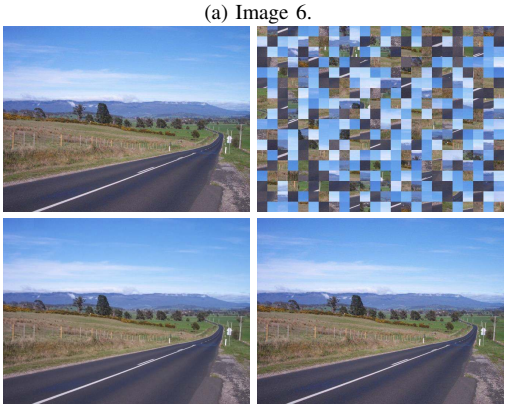


(d) Image 17.

Fig. 5. Image puzzles with 432 tiles of 28×28 pixels each. For each sub-image, the upper left is the original image, the upper right is the initial configuration of the puzzle for *PSQP*, the lower left is the final result for *PSQP*, and the lower right is Pomeranz et al.'s result.

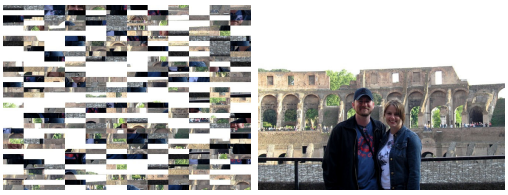


(a) Image 6.



(b) Image 12.

Fig. 6. Image puzzles with 432 tiles of 28×28 pixels each. For each sub-image, the upper left is the original image, the upper right is the initial configuration of the puzzle for *PSQP*, the lower left is the final result for *PSQP*, and the lower right is Pomeranz et al.'s result.



(a) Image 4. Performance: 66.0% under Direct comparison and 65.6% under Neighbor comparison. The non-constant part of the image is perfectly reconstructed.

(b) Image 18. Performance: 100% accuracy under both metrics.

Fig. 7. Image puzzles with 432 tiles of 56×14 pixels each. The first image represents the initial configuration of the puzzle and the second image is the final result for *PSQP*.

(a) Reconstructed image puzzle with 540 tiles, with 28×28 pixels each. Left: *PSQP* result with 100% accuracy under both metrics. Right: Pomeranz et al. [9], with 1.0% under direct comparison and 64% under neighbor comparison.

(b) Reconstructed image puzzle with 805 tiles, with 28×28 pixels each. Left: *PSQP* result with 91.9% accuracy under direct comparison and 90.6% under neighbor comparison. Right: Pomeranz et al. [9], with 83.0% under both metrics.

Fig. 8. Image puzzles with 540 and 805 tiles.

Fig. 9. *PSQP* applied to larger puzzles. First row shows the initial configuration and the reconstructed image puzzle with 2360 tiles of 28×28 pixels each. The second row shows the initial configuration and the reconstructed image puzzle with 3300 tiles of 28×28 pixels each.