American Journal of Applied Sciences 6 (11): 1941-1947, 2009
ISSN 1546-9239

Solving Square Jigsaw Puzzles Using Dynamic Programming and the Hungarian Procedure

Naif Alajlan
Advanced Lab for Intelligent Systems Research, Department of Computer Engineering,
King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

Abstract: Problem statement: To design, implement, and test an algorithm for solving the square jigsaw puzzle problem, which has many applications in image processing, pattern recognition, and computer vision such as restoration of archeological artifacts and image descrambling. **Approach:** The algorithm used the gray level profiles of border pixels for local matching of the puzzle pieces, which was performed using dynamic programming to facilitate non-rigid alignment of pixels of two gray level profiles. Unlike the classical best-first search, the algorithm simultaneously located the neighbors of a puzzle piece during the search using the well-known Hungarian procedure, which is an optimal assignment procedure. To improve the search for a global solution, every puzzle piece was considered as starting piece at various starting locations. **Results:** Experiments using four well-known images demonstrated the effectiveness of the proposed approach over the classical piece-by-piece matching approach. The performance evaluation was based on a new precision performance measure. For all four test images, the proposed algorithm achieved 100% precision rate for puzzles up to 8×8. **Conclusion:** The proposed search mechanism based on simultaneous allocation of puzzle pieces using the Hungarian procedure provided better performance than piece-by-piece used in classical methods.

Key words: Jigsaw puzzle solving, image descrambling, image restoration, square puzzle assembly, dynamic programming and Hungarian method

INTRODUCTION

Automatic solving of jigsaw puzzles suggests finding a subjectively correct spatial arrangement of the puzzle pieces (or sub-images) in order to reassemble a larger and complete image. Over the past three decades, the jigsaw puzzle problem has attracted researchers from various fields including pattern recognition, image processing, computer vision, combinatorial optimization and many other fields of mathematics. Automatic puzzle solving has many application domains and can provide interesting solutions to some problems. For instance, speech scrambling in the frequency domain is usually made by dividing the spectrogram into pieces and, then, rearranging them in such a way that the speech cannot be recognized when the inverse transformation is applied. Clearly, puzzle solving is suited for speech descrambling in this case. Other examples of application domains include assembly of cracked art paintings, restoration of archeological artifacts and image descrambling.

Below are some criteria that govern the selection of a puzzle solving algorithm for a specific application:

Accuracy: An algorithm should assemble puzzles with a high degree of accuracy.

Invariance: An algorithm should be invariant to rotating and translation of the puzzle pieces.

Robustness: An algorithm should perform well when some pieces are missing, extra, or overlapping.
Scalability: An algorithm's performance should be invariant as the number of puzzle pieces increases.

Generality: Can be applied to different types of images such as binary, grayscale and colored images.

Computational complexity: An algorithm should be computationally efficient in order to be suitable for real-time applications.

One of the earliest attempts to solve the jigsaw puzzle problem was due by Freeman and Garder[3] more than four decades ago. Most existing techniques for solving jig-saw puzzles assume curved canonical shapes, which have concavities and convexities, of the puzzle pieces[14]. This assumption usually leads to a clear distinction between border and internal pieces which reduces the search space for the solution and makes it tractable. A few other techniques work on

puzzle pieces with arbitrary shapes and treat assembling pieces as partial shape matching problem[4,6]. Toyama *et al*.[11] proposed a method for solving rectangular puzzle of binary images using a genetic algorithm approach. Regarding the features considered for piece matching, some methods use other information in addition to shape such as color[1,13] or texture[9].

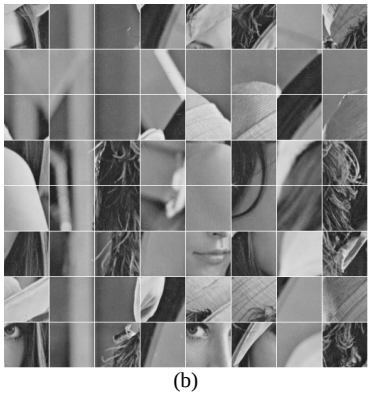
Local matching of puzzle pieces, although essential, is usually not sufficient to solve the puzzle problem efficiently. A global search that seeks the minimum sum of distances
across the entire assembly of the puzzle pieces is required. However, such global search is known to be NP-complete problem[2]. To overcome this difficulty, some methods rely on the high discrimination ability of the local matching function which makes the basins of attraction of the global solution quite large especially when the number of pieces is moderate (below 100)[5,15]. Other methods use local search with backtracking to avoid local minima and improve the global search[4]. When the border pieces can be identified, their arrangement becomes similar to the well-known traveling salesman problem due to their closed loop nature[1,14]. Genetic algorithms, which is an evolutionary optimization approach, has also been used to solve the jigsaw puzzle problem[11].

In this study, we present a method for solving the square jigsaw puzzle problem as shown in Fig. 1. All puzzle pieces have square shape; therefore, it is not possible to identify the border pieces. For matching pieces, the gray level pro-files of borders at the four piece sides are employed. The main contribution of the work presented in this study is the enhancement of the local search by using the Hungarian method[7], which is an optimal assignment procedure. Unlike in most previous methods where local search is proceeded in a piece-by-piece manner, all neighbors of a puzzle piece are located simultaneously during the puzzle assembly. The algorithm searches for the solution with different starting pieces at various locations to improve the global search. This search mechanism can be applied to any type of features such as color or texture and to arbitrary shaped puzzle pieces.

Problem description: Given an m × n location grid of mn subimages of square puzzle pieces of an image, the aim is to place a puzzle piece at each location of the grid such that the arranged pieces subjectively reassemble an original image. In this study, grayscale images are considered and puzzle pieces are obtained artificially. Although no rotation of the puzzle pieces is allowed, the search for a global minimum is still NP-complete. Since all puzzle pieces have the same square shape, no priori knowledge can be used to differentiate between border and internal pieces. In addition, it is not possible to use partial shape matching. Instead, the gray level profiles of border pixels at the four piece sides are employed for matching the puzzle pieces.

In our method, the local search is enhanced by using the Hungarian method[7,8], which is an optimal assignment procedure. Unlike most previous methods where local search is proceeded in a piece-by-piece manner, all neighbors of a puzzle piece are located simultaneously during the puzzle assembly. This can be viewed as an alternative to backtracking in the sense that both approaches avoid the best-first piece in order to obtain a local minimum in a larger neighborhood. In our method, the global search is performed implicitly by repeating the search with different starting pieces at various locations which more likely enables finding the global solution. This approach can be applied to arbitrary types of features such as color or texture and to arbitrary shaped puzzle pieces, which constitutes a main direction of our future research in this area.

(a)



(b)

Fig. 1: An example of 8×8 puzzle. (a) the original Lena image and (b) a scrambled version

MATERIALS AND METHODS

Local matching via dynamic programming: Here, a dynamic programming approach for matching border gray level profiles of puzzle pieces is presented.

A border gray level profile can be viewed as a one-dimensional sequence where matching two sequences is based on finding the optimal (i.e., least cost) correspondence between their points. Consider the border gray level profiles of Fig. 2. Panel (a) shows the profiles of two neighboring borders. Unlike the non-neighboring profiles in panel (b), neighboring profiles usually exhibit similarity in the overall shape with small deformations and displacement. Therefore, a dissimilarity distance that handles such transformations is employed, which is based on Dynamic Time Warping (DTW)[10,12]. Unlike the Euclidean distance that provides one-to-one alignment, nonlinear alignment can be achieved by the DTW, where one point on the sequence can be aligned to one or more points on another sequence, as illustrated in Fig. 3.

Let A(n) and B(n) be two sequences of border pro-files, where $n \in <1, N>$ is the index of the sequence points. Then, an N×N distance table, DT, is constructed to find the optimal correspondence between the points of the two sequences.

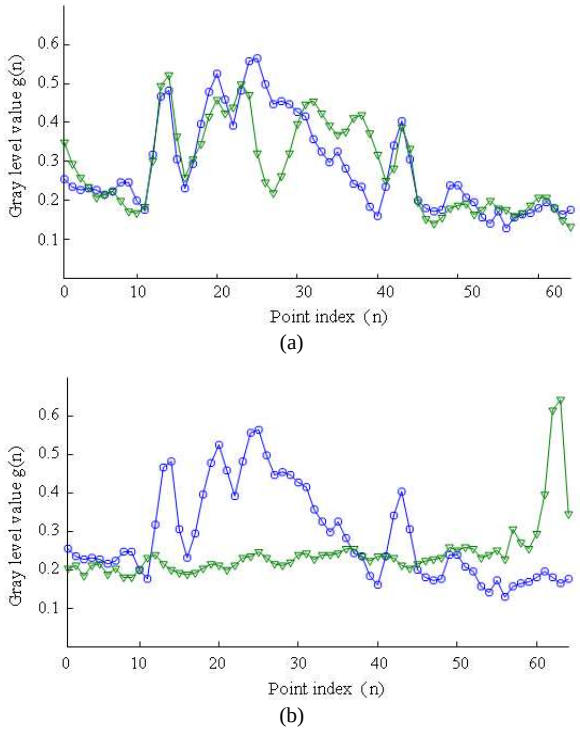


Fig. 2: Two examples of border gray level profiles: (a) neighboring and (b) non-neighboring

The columns of DT represent the points of one sequence and the rows represent the points of the other. Initially, the elements of DT are set as:

$$DT_{initial}(n,m) = \begin{cases} 0 & \max(1, n-w+1) \le \\ & m \le \min(N, n+w-1) \\ \infty & \text{otherwise} \end{cases} \quad (1)$$

where, $n,m \in <1, N>$, w is a predefined diagonal width for DT as illustrated in Fig. 4 and max(a, b) and min(a, b) are the maximum and minimum values of a and b, respectively. Only the elements of DT that fall within w are up-dated during the DTW search. In our implementation, w is set to approximately 10% of the sequence length, as in[10]. This initialization of DT avoids computing the distances between all the points of two sequences and restricts the distance computation to only those points which are more likely correspond to each other. Therefore, the computational complexity is largely reduced while more meaningful correspondences are obtained.

Starting at the first points for sequences A and B, the distance table DT is updated, through the diagonal window of width w, left-to-right and up-to-bottom starting from the upper-left element, as shown in Fig. 4. The first row and first column elements are initialized as the absolute difference between the corresponding points. Then, the rest of the zero-valued elements of DT are updated as:

$$DT(n,m) = |A(n) - B(m)| + \min \begin{cases} DT(n-1,m) \\ DT(n-1,m-1) \\ DT(n,m-1) \end{cases} \quad (2)$$

The least cost path through the distance table is the value of element DT (N, N), which corresponds to the best matching between the two sequences.

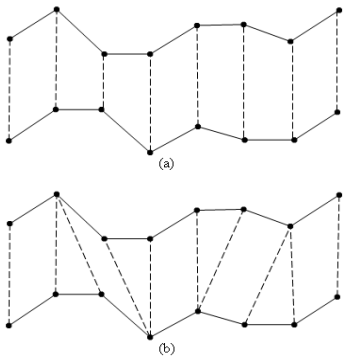


Fig. 3: An illustration of aligning the points of two sequences using (a) the Euclidean distance and (b) the DTW algorithm

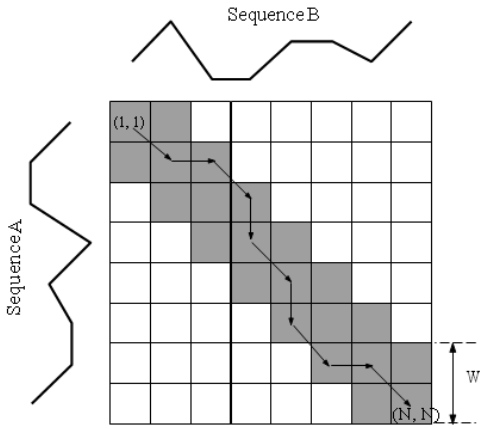


Fig. 4: DTW table. See text for details

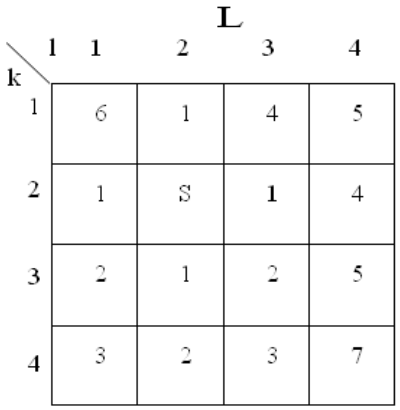


Fig. 5: An illustrative example of Algorithm 1 execution steps on 4 × 4 puzzle. See text for details

The proposed algorithm: A pseudo code of the proposed algorithm PuzzleSolve() is shown in Algorithm 1. The algorithm accepts as inputs a group of square puzzle pieces P and the dimensions of the rectangular grid m ×n where the pieces will be located.

Since each square piece has 4 sides, 4-connectedness is used where internal, border and corner locations have 4, 3 and 2 neighbors, respectively. At first, the algorithm initializes a distance matrix D which stores the pairwise distances between the puzzle pieces. The distance D(i, j) is a quadruple representing the right, left, top and bottom neighboring relations between pieces i and j. Let R_i and L_j be the grayscale profiles of the right side of piece P_i and the left side of piece P_j, respectively. Then, the first element of D(i, j) = DT (N, N), which is computed using equation (1),where N is the number of pixels in each side of a square puzzle piece. Note that Eq. 1 also represents the second element of D(j, i). Top and bottom distances are computed in the same manner using the top and bottom gray scale pro-files of the puzzle pieces.

The algorithm repeats the search for every puzzle piece at every internal location as the starting piece and returns the solution that yields the minimum sum of border distances. There are mn puzzle pieces, (m-2)(n-2) internal locations and 2 mn-m-n borders in the grid. The requirement for repeating the search comes from the fact that it is not possible to discriminate between border and internal pieces due to their square shape nature. At every search with starting piece P_i at location (k, l), the location matrix L is initialized with zeros and a group I includes all pieces except P_i. Then, the Hungarian method[7], which is an optimal assignment procedure, locates four pieces at the neighboring locations to P_i such that the sum of border distances between P_i and its neighbors is minimum.

The Hungarian function is passed with a matrix H of the distances between P_i and all other puzzles in I, i.e., the elements of the i-th row in D (each element is a quadruple) are arranged as rows of H. The pieces selected by the assignment procedure assign are located in L and removed from I. Next, the algorithm selects a nonzero location in L, (k_p, l_p), with greatest number of zero (or empty) neighbors N. When multiple locations exist, the location with minimum sum of neighbor distances is selected. Intuitively, as the number of pieces located at once increases, a global minimum is more likely achieved. The Hungarian procedure is applied to assign pieces in the empty neighbors as described earlier except that only elements of D corresponding to the empty locations are considered, i.e., the columns of H correspond to the zero neighbors. This process is repeated until I becomes empty (all pieces are located).

An example illustrating the execution sequence of the algorithm is shown in Fig. 5, which represents L and the numbers in the table reflect the sequence of assigning each location. At first, the starting piece is located in (2, 2) (denoted by S). Then, the first execution of the Hungarian procedure assigns the neighboring locations denoted by 1. At this time, there are two nonzero locations in L that have three zero neighbors, (3, 2) and (2, 3). The former is selected since it has lower sum of distances with its neighbors. Then, the neighbors of the piece located in (3, 2) are assigned (denoted by 2). Note that only right, left and bottom distances are considered in the distance matrix H since the top neighbor is already located. The algorithm continues until all locations are assigned.

RESULTS AND DISCUSSION

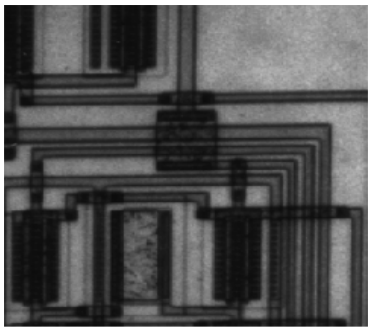
To test our algorithm, four well-known images are used, namely, Lena, Cameraman, Taxi and Circuit as shown in Fig. 1 and 6. All images are in gray scale format with 8 bit quantization. The images are artificially divided into square pieces of various sizes ranging from 4×4-10×10. To measure the accuracy of the algorithm, a new precision measure is proposed which is given as:

$$\text{Precision} = \frac{C}{T} \quad (3)$$

Where:
C = The number of correctly located pieces
T = The total number of puzzle pieces

(a)

(b)

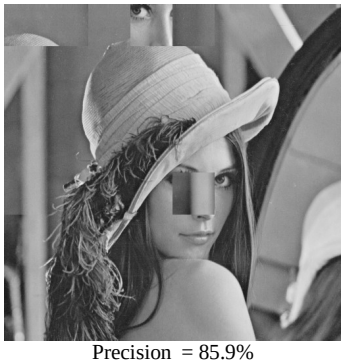


(c)

Fig. 6: Test images. (a) cameraman; (b) taxi; (c) circuit

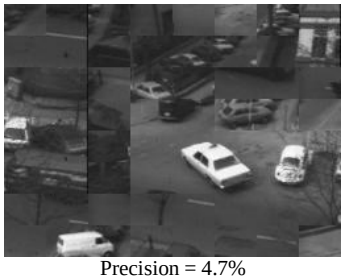
This measure is suitable for error-tolerant applications and overcomes the drawback of "correct/false" decision used for evaluating most current methods. Figure 7 shows three demonstrations of this performance measure.

A classical method for solving the jigsaw puzzle problem is to proceed the search in a piece-by-piece manner as follows. After a starting piece is located, it's best match is located at a neighboring location following left-to-right, top-to-bottom direction. This process is repeated with every puzzle piece as the starting piece and the minimum global solution is selected. We use the same distance measure for the classical method since the aim is to test the proposed search mechanism.



Precision = 85.9%

Precision = 46.8%



Precision = 4.7%

Fig. 7: Demonstration of precision measure

Table 1: Precision (%) of the proposed and classical methods at different puzzle sizes on the Lena image

Puzzle size	Proposed	Classical
4×4	100	100.0
5×5	100	100.0
6×6	100	100.0
7×7	100	89.8
8×8	100	12.5
9×9	100	0.0
10×10	53	0.0

Table 2: Precision (%) of the proposed and classical methods at different puzzle sizes on the cameraman image

Puzzle size	Proposed	Classical
4×4	100	100.0
5×5	100	100.0
6×6	100	100.0
7×7	100	91.8
8×8	100	18.8
9×9	100	0.0
10×10	59	3.0

Table 3: Precision (%) of the proposed and classical methods at different puzzle sizes on the taxi image

Puzzle size	Proposed	Classical
4×4	100	100.0
5×5	100	100.0
6×6	100	100.0
7×7	100	87.8
8×8	100	25.0
9×9	100	7.4
10×10	64	0.0

Table 4: Precision (%) of the proposed and classical methods at different puzzle sizes on the circuit image

Puzzle size	Proposed	Classical
4 × 4	100.0	100.0
5 × 5	100.0	100.0
6 × 6	100.0	100.0
7 × 7	100.0	77.6
8 × 8	100.0	9.4
9 × 9	95.1	0.0
10 × 10	44.0	0.0

Table 1-4 show the precisions of the proposed algorithm and the classical method on the four test images, at various numbers of puzzle pieces. Clearly, the proposed algorithm outperforms the classical method with a good margin. However, the performance of the proposed algorithm deteriorates as the number of puzzles exceeds 100.

Algorithm 1: Pseudo code of the puzzle solving algorithm:

ImgOut = PuzzleSolve(P , m, n)

Initialization:

m and n are the numbers of rows and columns of the puzzle grid, respectively.

$P = \{P_i\}$ is a group of mn square puzzle pieces.

L is an is an $m \times n$ matrix of located pieces indices.

D is an $m\,n \times m\,n$ distance matrix where D(i, j) is a quadruple of right, left, top and bottom distances between P_i and P_j .

hungarian(Z) is an optimal assignment function that assigns the rows to the columns of the cost matrix Z.

```
c←0
for every internal location of the puzzle grid (k, l) do
    for every Pi do
        c←c+1
        I←P−{Pi}
        L←0 _
        L(k, l) ←i
        H←all D(i, j), j ∈ I arranged as rows.
        [assign, cost] = hungarian(H)
        L(k, l±1), L(k±1, l)←I(assign)
        I←I − {I(assign)}
        while I ≠ Ø do
            [kp, lp]←nonzero location in L with greatest number of zero neighbors N.
            H←elements of D(L(kp, lp), j), j ∈ I according to N, arranged as rows.
            [assign, cost] = hungarian(H)
            L(N)←I(assign)
            I←I – {I(assign)}
        end while
        cost(c)←sum of all 2 mn-m-n border distances between the arranged puzzles in L.
        S{c}←L
    end for
end for
ImgOut ← S{r} such that cost(r) = min(cost).
return ImgOut
```

CONCLUSION

This study presented an algorithm for solving square jigsaw puzzles. The main contribution of the algorithm is focused on the search mechanism which is based on simultaneous allocation of puzzle pieces using the Hungarian procedure, rather than piece-by-piece used in classical methods. For matching puzzles, dynamic time warping is used to mea-sure the dissimilarity of the gray scale profiles of border pixels. Global solution is more likely obtained by repeating the search with different starting pieces at various locations. The proposed algorithm demonstrated better performance over the classical approach using four standard test images.

As most puzzle solving methods in the literature, the main limitation of our algorithm is the inability to solve puzzles with large number of pieces. Our future research in this area includes using other features such as color and texture for matching puzzle pieces. Another future direction is to apply the Hungarian procedure along with a partial shape matching technique for solving arbitrary shaped puzzles.

ACKNOWLEDGEMENT

The researcher would like to acknowledge the support of the Advanced Lab for Intelligent Systems Research at King Saud University.

REFERENCES

1. Chung, M.G., M.M. Fleck and D.A. Forsyth, 1998. Jigsaw puzzle solver using shape and color. Proceedings of IEEE 4th International Conference on Signal Processing, Oct. 12-16, IEEE Xplore Press, Beijing, China, pp: 877-880. DOI: 10.1109/ICOSP.1998.770751
2. Demaine, E.D. and M.L. Demaine, 2007. Jigsaw puzzles, edge matching and polyomino packing: Connections and complexity. Graphs Combinator, 23: 195-208. DOI: 10.1007/s00373-007-0713-4.
3. Freeman, H. and L. Garder, 1964. Apictorial Jigsaw puzzles: The computer solution of a problem in pattern recognition. IEEE Trans. Elect. Comput., 13: 118-127. DOI: 10.1109/PGEC.1964.263780
4. Kong, W. and B.B. Kimia, 2001. On solving 2D and 3D puzzles using curve matching. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'01), IEEE Xplore Press, USA., pp: 583-590. DOI: 10.1109/CVPR.2001.991015
5. Kosiba, D.A., P.M. Devaux, S. Balasubramanian, T.L. Gandhi and K. Kasturi, 1994. An automatic jigsaw puzzle solver. Proceedings of IEEE 12th International Conference on Pattern Recognition, Oct. 9-13, IEEE Xplore Press, Jerusalem, Israel, pp: 616-618. DOI: 10.1109/ICPR.1994.576377
6. Makridis, M. and N. Papamarkos, 2006. A new technique for solving a jigsaw puzzle. Proceedings of IEEE International Conference on Image Processing, Oct. 8-11, IEEE Xplore Press, Atlanta, GA., pp: 2001-2004. DOI: 10.1109/ICIP.2006.312891
7. Munkres, J., 1957. Algorithms for assignment and transportation problems. J. Soc. Ind. Applied Math., 5: 32-38.
8. Papadimitriou, C. and K. Steiglitz, 1988. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, New Jersey, ISBN: 0-486-40258-4.
9. Sagiroglu, M.S. and A. Ercil, 2006. A texture based matching approach for automated assembly of puzzles. Proceedings of 18th International Conference on Pattern Recognition, (PR'06), IEEE Xplore Press, Hong Kong, pp: 1036-1041. DOI: 10.1109/ICPR.2006.184
10. Sakoe, H. and S. Chiba, 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process., 26: 43-49.
11. Toyama, F., Y. Fujiki, K. Shoji and J. Miyamichi, 2002. Assembly of puzzles using a genetic algorithm. Proceedings of 16th IEEE International Conference on Pattern Recognition, (PR'06), IEEE Xplore Press, USA., pp: 389-392. DOI: 10.1109/ICPR.2002.1047477.
12. Wang, K. and T. Gasser, 1997. Alignment of curves by dynamic time warping. Ann. Stat., 25: 1251-1276.
13. Weiss-Cohen, M. and Y. Halevi, 2005. Knowledge retrieval for automatic solving of jigsaw puzzles. Proceedings of International Conference on Intelligent Agents, Web Technologies and Internet Commerce and International Conference on Computational Intelligence for Modeling, Control and Automation, Nov. 28-30, IEEE Xplore Press, Vienna, pp: 379-383. DOI: 10.1109/CIMCA.2005.1631498
14. Wolfson, H., E. Schonberg, A. Kalvin and Y. Lamdan, 1988. Soving jigsaw puzzles by computer. Ann. Operat. Res., 12: 51-64. DOI: 10.1007/BF02186360
15. Yao, F.H. and G.F. Shao, 2003. A shape and image merging technique to solve jigsaw puzzles. Patt. Recog. Lett., 24: 1819-1835. DOI: 10.1016/S0167-8655(03)00006-0