AN IMPROVED HEURISTIC-BASED APPROACH FOR SOLVING SQUARE JIGSAW PUZZLES

Naif Alajlan

Electrical Engineering Dept.
King Saud University, P.O.Box 800, Riyadh 11421, Saudi Arabia.

ABSTRACT

An algorithm for assembling square jigsaw puzzles is presented. We commence by introducing criteria that govern the selection of proper jigsaw puzzle solving method for a given application. Our algorithm uses the gray level profiles of border pixels for local matching of the puzzle pieces. Unlike the classical best-first search, the algorithm simultaneously locates the neighbors of a puzzle piece during the search. To improve the search for a global solution, every puzzle piece is considered as starting piece at various starting locations. Experiments using well-known images demonstrate the effectiveness of the proposed approach over the classical piece-by-piece matching approach. The performance evaluation is based on a more precise new performance measure.

Keywords Jigsaw Puzzle Solving, Image Descrambling, Image Restoration, Square Puzzle Assembly and Hungarian Method.

1. 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 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 Gardner more than four decades ago [3]. Most existing techniques for solving jigsaw puzzles assume curved canonical shapes, which have concavities and convexities, of the puzzle pieces [12]. 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. [10] 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 [11, 1] 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) [13, 5]. 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 [12, 1]. Genetic algorithms, which is an evolutionary optimization approach, has also been used to solve the jigsaw puzzle problem [10].

In this paper, 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 profiles of borders at the four piece sides are employed. The main contribution of the work presented in this paper 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 is repeated 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. In the following, the proposed method is explained in more details.

2. THE PROPOSED ALGORITHM

Given an \( m \times n \) 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 reassemble the original image. In this paper, 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 any type of features such as color or texture and to arbitrary shaped puzzle pieces.

A pseudo code of the proposed algorithm \textit{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 \( m \times n \) 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 \( D(i,j) \) is computed as the sum of absolute differences (\( SAD \)):
where \( Q \) 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 grayscale profiles 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 \( 2mn - m - n \) 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, 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 it’s neighbors is minimum. The \text{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 \text{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. 2. The table in the figure 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 it’s 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.

\begin{equation}
SAD(R_i, L_j) = \sum_{q=1}^{Q} |R_i(q) - L_j(q)|
\end{equation}

**Algorithm 1** Pseudo code of the puzzle solving algorithm:

\begin{verbatim}
ImgOut = PuzzleSolve(P, m, n)

Initialization:

1: \( m \) and \( n \) are the numbers of rows and columns of the puzzle grid, respectively.
2: \( P = \{P_i\} \) is a group of \( mn \) square puzzle pieces.
3: \( L \) is an \( m \times n \) matrix of located pieces indices.
4: \( D \) is an \( mn \times mn \) distance matrix where \( D(i, j) \) is a quadruple of right, left, top, and bottom distances between \( P_i \) and \( P_j \).
5: \text{hungarian}(Z) \) is an optimal assignment function that assigns the rows to the columns of the cost matrix \( Z \).

1: \( c \leftarrow 0 \)
2: for every internal location of the puzzle grid \((k, l)\) do
3: for every \( P_i \) do
4: \( c \leftarrow c + 1 \)
5: \( I \leftarrow P - \{P_i\} \)
6: \( L \leftarrow 0 \)
7: \( H \leftarrow \) all \( D(i, j) \), \( j \in I \) arranged as rows.
8: \( \text{[assign, cost]} \leftarrow \text{hungarian}(H) \)
9: \( L(k, l) \leftarrow I(\text{assign}) \)
10: \( I \leftarrow I - \{I(\text{assign})\} \)
11: while \( I \neq \emptyset \) do
12: \( \text{[assign, cost]} \leftarrow \text{hungarian}(H) \)
13: \( L(k_p, l_p) \leftarrow I(\text{assign}) \)
14: \( I \leftarrow I - \{I(\text{assign})\} \)
15: end while
16: \( \text{cost}(c) \leftarrow \) sum of all \( 2mn - m - n \) border distances between the arranged puzzles in \( L \).
17: \( S \{c\} \leftarrow L \)
18: end for
19: \( \text{return ImgOut} \) such that \( \text{cost}(r) = \min(\text{cost}) \).
\end{verbatim}
3. EXPERIMENTAL RESULTS

To test our algorithm, three well-known images are used, namely, Cameraman, Lena and Circuit as shown in Figs. 1 and 3. All images are in grayscale format with 8-bit quantization. The images are artificially divided into square pieces of various sizes ranging from $4 \times 4$ to $9 \times 9$. To measure the accuracy of the algorithm, a new precision measure is proposed which is given as:

$$\text{Precision} = \frac{C}{T}$$  \hspace{1cm} (2)

where $C$ is the number of correctly located pieces and $T$ is the total number of puzzle pieces. This measure is suitable for error-tolerant applications and overcomes the drawback of "correct/false" decision used for evaluating most current methods.

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.

Figs. 4, 5 and 6 show the precisions of the proposed algorithm and the classical method on Cameraman, Lena and Circuit images, respectively, 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 64.

4. CONCLUSIONS AND FUTURE WORK

This paper 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, the grayscale profiles of border pixels are used. 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 three standard 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.
Fig. 4. Precisions of the proposed and classical methods at different puzzle sizes on Cameraman image.

Fig. 5. Precisions of the proposed and classical methods at different puzzle sizes on Lena image.

Fig. 6. Precisions of the proposed and classical methods at different puzzle sizes on Circuit image.

Acknowledgement

The author would like to acknowledge the support of the Research Center of the Engineering College at King Saud University under grant number 12/428.

5. REFERENCES


