


## **Genetic-based square jigsaw puzzle solver using the combined color+texture compatibility criterion**


**Atefeh Parvin**

(University of Sistan and Baluchestan, Electrical and Computer Faculty, Department of Communications Engineering, Zahedan, Iran,

 <https://orcid.org/0009-0004-9994-9239>, [atefeh.parvin@pgs.usb.ac.ir](mailto:atefeh.parvin@pgs.usb.ac.ir))


**Farahnaz Mohanna**

(University of Sistan and Baluchestan, Electrical and Computer Faculty, Department of Communications Engineering, Zahedan, Iran

 <https://orcid.org/0000-0002-8689-6201>, [f\\_mohanna@ece.usb.ac.ir](mailto:f_mohanna@ece.usb.ac.ir))

**Masoumeh Rezaei**

(University of Sistan and Baluchestan, Electrical and Computer Faculty, Department of Computer Engineering, Zahedan, Iran

 <https://orcid.org/0000-0001-8184-1773>, [mrezaei@ece.usb.ac.ir](mailto:mrezaei@ece.usb.ac.ir))

**Abstract:** When reconstructing jigsaw puzzles, the state-of-the-art algorithms struggle to distinguish between identically colored pieces that belong to different objects. This limitation significantly impacts the accuracy of puzzle solvers, especially in complex images with repetitive colors or textures. To address this issue, we propose a new GA-based square jigsaw puzzle solver. A combined color and texture discriminator is incorporated into the proposed solver to prevent pieces that have the same color but come from distinct objects from being joined together incorrectly. Color and texture features are extracted separately using the sum of square distances and Gabor filter. To evaluate the performance of the proposed solver, we used a dataset consisting of 66 images: 20 puzzles with 432 pieces from the MIT collection, 20 puzzles with 540 pieces, and 20 puzzles with 805 pieces from the McGill collection, and 3 puzzles with 2360 pieces, and 3 puzzles with 3300 pieces from the Pomeranz collection. For the direct, neighbor, and largest component comparisons, the proposed method's accuracy is 92.91%, 96.66%, and 90.83%, respectively. The proposed method demonstrates an improvement of 11.9%, and 3.65% in accuracy based on direct and neighbor comparison criteria, on the database images when compared to current state-of-the-art GA-based square jigsaw puzzle solver.

**Keywords:** Jigsaw puzzle solving, Genetic algorithm, Compatibility criterion, Texture feature, Gabor filter

**Categories:** H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

**DOI:** 10.3897/jucs.129768

## **1 Introduction**

A square jigsaw puzzle consists of an image divided into  $N$  non-overlapping square pieces. By disrupting the order of these pieces, a player attempts to reconstruct the image using the shape and color information of each piece. Solving a square jigsaw puzzle is essential for restoration of ancient pieces, recovery of shredded photos, and repairing of fractured bones [Sholomon, 13]. Various methods have been used to solve

a square jigsaw puzzle by computer. Among them, approaches utilizing Genetic Algorithm (GA) are more feasible to apply in the real world scenarios [Sholomon, 13, Sholomon, 14]. GA-based techniques outperform deep neural network-based techniques [Chen, 23a, Chen, 23b, Chen, 24a, Chen, 24b, Chen, 24c] in terms of accuracy for puzzles with more pieces [Sholomon, 14], and do not require a vast number of database images. Two essential components for reconstructing a jigsaw puzzle based on the GA are a compatibility criterion, which measures how well a pair of pieces fit together, and the reconstruction strategy [Sholomon, 14]. Jigsaw puzzles are generally classified into four types based on specific characteristics. In type 1 puzzles, the direction of each piece is known, but their locations are not. Therefore, a pair of pieces can be joined in only four different ways. In type 2 puzzles, neither the location nor the direction of any piece is known, resulting in 16 possible ways to fit a pair of pieces together. In type 3 puzzles, each piece location is known, but its direction is not. Finally, in type 4 puzzles, neither the location nor the direction of any piece is known, and a piece may need to be flipped, meaning the face of the puzzle is also unknown [Sholomon, 14]. According to the previous work, the average accuracy of methods for solving the type 1 jigsaw puzzles, with 432 pieces to 22834 pieces, has been reported 97.12%, and 98.36% based on direct, and neighbor comparison criteria, respectively. Also, the average accuracy of methods solving the type 2 jigsaw puzzles, with 432 pieces to 22755 pieces, has been reported 96.16% based on the neighbor comparison criterion. The direct comparison criterion measures the number of pieces in the correct position in a solved puzzle [Sholomon, 14]. The neighbor comparison criterion counts the number of correctly positioned adjacent pairs of pieces in a solved puzzle [Sholomon, 14]. The largest component comparison identifies the largest contiguous area of correctly assembled pieces, regardless of their overall position in the puzzle [Sholomon, 14]. Additionally, the complete reconstruction criterion assesses whether every piece in a solved puzzle is in the correct position and orientation [Sholomon, 14]. In addition, the compatibility criterion of Sum of Squared Distances (SSD) [Sholomon, 13] [Sholomon, 14] [Sholomon, 16], Mahalanobis Gradient Compatibility (MGC) [Sholomon, 16], SSD+MGC [Guo, 20], and  $L_q^p$  compatibility (LPQ) [Bezulj, 18] have been used as the compatibility measures for adjoining a pair of pieces in solving a square jigsaw puzzle. A three-phase procedure has also been employed as the crossover operator in the most of the GA-based puzzles solvers [Sholomon, 13] [Sholomon, 14].

A review of previous GA-based square jigsaw puzzle solvers reveals that these solvers primarily used color criteria to measure the compatibility of puzzle pieces. However, relying solely on color criteria can lead to inaccuracies, particularly when two pieces from different objects have similar colors. This limitation reduces the accuracy of these solvers.

In this paper, we propose a new compatibility criterion that combines color and texture features. By using the joint color and texture compatibility criterion, the proposed GA-based square jigsaw puzzle solver (from now on, we call it the proposed GA-based solver) has achieved a higher accuracy on the database including 66 selected square jigsaw puzzles. The selected square jigsaw puzzles consist of, 20 puzzles with 432 pieces of the MIT dataset [Cho, 10], 20 puzzles with 540 pieces, and 20 puzzles with 805 pieces from the McGill dataset [Olmos, 15], three puzzles with 2360 pieces,

and three puzzles with 3300 pieces from the Pomeranz dataset [Pomeranz, 11]. For this paper, the main contributions are as follows:

1. The proposed GA-based solver achieves higher accuracy compared to the other similar GA-based solvers.
2. It can easily be implemented in the real world.
3. The proposed solver introduces the first hybrid color+texture discriminator for GA-based puzzle solving.
4. Unlike deep neural network-based solvers, the proposed solver does not require a large number of puzzles. It offers a data-efficient, training-free alternative to neural networks.
5. The proposed method is capable of solving any type 1, 2, and 3 square jigsaw puzzles. It also is size-independent.

However, the proposed solver has some drawbacks:

- 1- It is a moderate time consuming for puzzles in the RGB color space.
- 2- It cannot reconstruct the type 4 square jigsaw puzzles.
- 3- It cannot reconstruct the non-square jigsaw puzzles.

In continue, the related work is presented in section 2. The proposed GA-based solver is introduced in section 3. The results, comparisons, and evaluation are reported in section 4. The paper concluded in section 5.

## 2 Related Work

The GA-based type 1 square jigsaw puzzles solver was introduced in [Sholomon, 13] based on the compatibility criterion of Sum of Squared Distances (SSD) in the  $L^*a^*b$  color space, and employs a three-phase procedure as the crossover operator. In the three-phase procedure for piece selection, and assignment, first, the agreed pieces, second, the best-buddy pieces, and finally the most compatible pieces available were placed. The average accuracy of this solver with 10 runs on the database was reported 87.89%, and 93.46% according to the direct, and neighbor comparisons criteria, respectively.

The GA-based type 2 square jigsaw puzzles solver introduced in [Sholomon, 14] uses the SSD compatibility criterion in the  $L^*a^*b$  color space, and employs the Prim algorithm as the crossover operator. The average accuracy of this solver with 30 runs on the database was reported 84.23%, and 93.00% according to the direct, and neighbor comparison criteria, respectively.

The GA-based type 2 square jigsaw puzzles solver was presented in [Sholomon, 16] to solve the square jigsaw puzzles type 2 up to 30745 pieces. This solver employs both the SSD, and Mahalanobis Gradient Compatibility (MGC) fitness functions, and uses the three-phase procedure as the crossover operator. The average accuracy of this solver on 20 puzzles with 432 pieces was reported 96.16%, and 96.03 according to the neighbor comparison using the SSD, and MGC compatibility criteria, respectively, demonstrating the superiority of SSD over MGC.

The GA-based jigsaw puzzles solver was presented in [Guo, 20] to solve the small-scale puzzles. The fitness function was based on the SSD+MGC compatibility criterion, and the crossover operator was the three-phase procedure. Additionally, two new mutation operators were introduced. The database included 25 square jigsaw puzzles

with 256, 646, and 1196 pieces. The average accuracy of this solver was reported 94.31% based on the direct comparison.

The GA-based type 2 jigsaw puzzles solver was presented in [Bezulj, 18] giving a weight to each GA solution to use in the roulette wheel. The fitness function was the LPQ compatibility and the crossover operator was the three-phase procedure without its first phase. The accuracy of this solver on 20 MIT puzzles with 432 pieces was 92.6% according to the direct comparison.

An image super-resolution reconstruction method was introduced in [Chen, 24a] leveraging the multi-level information compensation and U-Net network. First, the U-Net network performed multi-level feature extraction and channel compression on input features. Next, correlation features from different channels were extracted and integrated. A multi-level information compensation model was then devised to address information loss during the compression process.

The image inpainting network using multi-scale feature module was introduced in [Chen, 24b] and enhanced attention module. First, a multi-scale fusion module was presented based on dilated convolution to reduce information loss. To ensure consistency in image inpainting details and style, style loss and perceptual loss functions were incorporated.

A lightweight method presented in [Chen, 23a] integrates group convolution and attention mechanism to enhance and replace the traditional convolution module.

The image inpainting method was introduced in [Chen, 24c] according to partial multi-scale channel attention mechanism and deep neural networks to process on images with large missing sections. The method employed a Res-U-Net module as its generator, where the U-Net backbone handled the encoding and decoding steps for damaged images. The residual network enhanced the network's capability to extract the features from the damaged images.

The image restoration method was introduced in [Chen, 23b] consisting of the semantic priors, deep attention residual group, and full-scale skip connection. The semantic priors network learned comprehensive semantic information about visual elements in missing regions to facilitate completion. The deep attention residual group enabled the generator to focus on both missing regions and adaptively learn channel features. The full-scale skip connection combined the low-level feature maps containing image boundaries with the high-level feature maps containing image textures and details to repair the missing regions.

In [Markaki, 23], the problem of jigsaw puzzle solvers has been thoroughly discussed, including their evaluation and applications.

The jigsaw puzzles solver introduced in [Li, 22] based on the Generative Adversarial Network (GAN). The multi-task pipeline was designed including a classification, and the GAN branches that were connected by the flow-based warp module. The average accuracy of this solver was reported 79.0% on the database including 7639 puzzles.

The type 2 large square jigsaw puzzles solver introduced in [Huroyan, 20] utilizes the MGC to match puzzle pieces. Additionally, a graph connection Laplacian is employed to recover the puzzle pieces rotation. The average accuracy of this solver was reported 89.1%, 91.4%, and 90.62% respectively based on the direct, neighbor, and largest component comparisons on the database.

The pairwise compatibility measure was presented in [Guerroui, 18] using the Gist and color distance to solve the jigsaw puzzles. The rotation-based strategy was

presented to work on the multiple parts, and solve puzzles from the local matching candidates. The average accuracy of this solver on 20 MIT type 2 square jigsaw puzzles with 432 pieces was reported 93%, and 96% respectively based on the direct, and neighbor comparisons. For 20 MIT type 1 square jigsaw puzzles, the reported average accuracy was 94.16%, and 96.44% based on direct and neighbor comparisons respectively.

The deep learning model was introduced in [Ahmad, 22] to determine the different states of jigsaw puzzle. The task was represented as a classification problem, where each state of puzzle was considered as a class. The model can also serve as a fitness function for other GA-based jigsaw puzzles solvers.

The design was introduced in [Chen, 22] to enhance the search throughput by evaluating newly generated inputs with the JIT-compiled path constraints. The model achieved three orders of magnitude higher search throughput than the existing fuzzers and could scale to the multiple cores.

The computational puzzles solver presented in [Son, 18] aims to maximize geometric consensus among hierarchical piece loops for square jigsaw puzzles without prior information. Initially, loops of four pieces are searched, and then aggregated into higher-level piece loops. The average accuracy of this solver was reported as follows: for type 1 square jigsaw puzzles, 94.9%, 95.7%, and 93.42% based on direct, neighbor, and largest component comparisons respectively, and for four type 2 square jigsaw puzzles, 93.46%, and 95.3% based on direct, neighbor comparisons respectively.

The square jigsaw puzzles solver presented in [McGill-Smith, 19] utilizes the ORB feature detector in conjunction the FLANN-based matcher. Initially, an image of the square jigsaw puzzle is segmented into regions of interest using contour detection. These regions undergo a feature matching process, followed by filtering using Lowe's ratio test to refine matches. Outliers are subsequently identified and removed using the homography transformation matrix. The accuracy of the solver was reported 81.2% on a jigsaw puzzle with 24 pieces.

The square jigsaw puzzles solver introduced in [Paikin, 15] employs a greedy approach for comparing the compatibility of puzzle pieces. However, the greedy solvers depended on the initial placement of puzzle pieces. This solver is capable of reconstructing puzzles with missing pieces and handling puzzles contained multiple puzzles pieces.

A jigsaw puzzle-solving task as a self-supervised pre-training mechanism was presented in [Peng, 20] for fine-grained sketch-based image retrieval (FG-SBIR). However, it primarily focused on small-grid puzzles (e.g., 2×2 or 3×3 pieces) rather than large-scale puzzles.

Table 1 showed the summary results of the state-of-the-art GA-based square jigsaw solver so far.

Time	Crossover	Standard Deviation		Average Accuracy		pieces	Data Sets	Type	Fitness	Ref.
48.73 sec.	Three phase	0.34	2.62	95.70	82.94	432	MIT & Pomeranz & McGill	1	SSD	[Sholomon, 13]
64.04 sec.		0.40	0.65	95.38	91.65	540				
116.18 sec.		0.31	0.62	95.85	93.63	805				
17.60 min.		0.38	0.86	88.00	84.62	2360				

30.24 min.		0.27	7.19	92.37	86.62	3300				
61.06 min.		0.11	1.74	95.06	92.04	5015				
3.21 hours		0.08	0.45	98.36	97.12	10375				
13.19 hours		0.05	0.28	96.22	91.74	22834				
8 sec.	Prim	0.39	10.47	94.44	81.51	432	MIT & Pomeranz	2	SSD	[Sholomon, 14]
11.90 sec.		0.49	12.49	91.33	74.55	540				
22.10 sec.		0.48	10.56	91.45	79.20	805				
3.67 min.		0.18	0.90	97.33	92.75	2360				
4.82 min.		0.79	1.14	94.93	93.14	3300				
	Three-phase			96.16		432	MIT	2	SSD	[Sholomon, 16]
				96.03					MGC	
	Three phase		0.61		99.37	256		2	SSD+MGC	[Guo, 20]
			0.68		98.42	646				
			2.72		84.94	1196				
	Three-phase without its first phase			92.60		432	MIT	2	LPQ	[Bezulj, 18]

Table 1: The summary results of all the state-of-art GA-based square jigsaw solver

### 3 The Proposed GA-based Solver

The block diagram of the proposed method is shown in Fig. 1, and the pseudocode of the proposed GA-based solver is shown in Table 2.

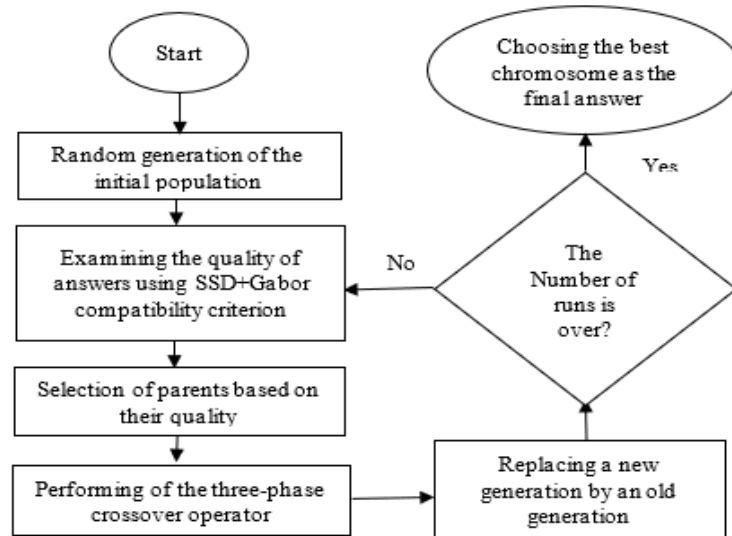


Figure 1: The block diagram of the proposed GA-based square jigsaw puzzles solver

First, the puzzle pieces are randomly assembled to form an initial population of chromosomes. Second, the fitness score for each chromosomes is calculated using a combination of the SSD compatibility criterion and the Gabor texture, referred to as the SSD+Gabor compatibility criterion. Third, some of the best chromosomes are selected as parents to generate a new population. Fourth, the crossover operation is performed on two selected parents to create a new child chromosome. Fifth, the children chromosomes, along with the parent chromosomes, form a new generation, which replaces the old generation. If the number of runs has not yet been completed, the proposed GA-based solver returns to step 2 and continues the process. Otherwise, if the number of runs is completed, the best chromosome at this stage represents the solved puzzle. It is mentioned, the fitness function for the proposed method combines the SSD compatibility criterion between each pair of puzzle pieces with the extracted Gabor features from the same two pieces. The color dissimilarity between two puzzle pieces of  $x_i$ , and  $x_j$  is computed by (1) [Sholomon, 14].

#### Algorithm

- 1:  $population \leftarrow$  generate 1000 random chromosomes
- 2: **for** generation number = 1  $\rightarrow$  20 **do**
- 3: evaluate all chromosomes using the SSD+Gabor fitness function
- 4:  $new\ population \leftarrow NULL$
- 5: copy 4 best chromosomes to  $new\ population$
- 6: **while** size ( $new\ population$ )  $\leq$  1000 **do**
- 7:  $parent1 \leftarrow$  select chromosome
- 8:  $parent2 \leftarrow$  select chromosome
- 9:  $child \leftarrow$  Three phase procedure ( $parent1, parent2$ )
- 10: add child to  $new\ population$
- 11: **end while**
- 12:  $population \leftarrow new\ population$

- 13: end for  
 14: solution ← best population

Table 2: The pseudocode of the proposed GA-based solver

$$D(x_i, x_j, r) = \sqrt{\sum_{k=1}^K \sum_{b=1}^3 (x_i(k, K, b) - x_j(k, 1, b))^2} \quad (1)$$

where,  $K$  is a piece dimension,  $b$  is a piece component,  $r$  means right and  $(x_i, x_j, r)$  indicates piece of  $x_j$  is in the right side of piece of  $x_i$  in the puzzle.  $D$  means the dissimilarity and  $D(x_i, x_j, r)$  indicates the dissimilarity between two pieces of  $x_i$ , and  $x_j$ , that  $x_j$  is in the right side of piece of  $x_i$ . In the proposed solver, the RGB color space is used, so the number of color components ( $b$ ) is 3, and each piece dimension is considered  $28 \times 28$ , so  $K = 28 \times 28$ .

The sum of dissimilarity on all a puzzle pieces by considering all the neighbor pieces in the right and down directions in the puzzle is computed by (2) [Sholomon, 14].

$$\sum_{i=1}^N \sum_{j=1}^{M-1} (D(x_{i,j}, x_{i,j+1}, r)) + \sum_{i=1}^{N-1} \sum_{j=1}^M (D(x_{i,j}, x_{i+1,j}, d)) \quad (2)$$

where,  $r$  and  $d$  denote the right and down directions, respectively.  $M \times N$  is the chromosome size.  $x_{i,j}$  is a single puzzle piece at coordinates  $(x, y)$ .  $D$  is the dissimilarity between two pieces.

To extract the Gabor texture from each puzzle piece, the Gabor filter is applied as descriptor in equations (3) to (6) [Otomo, 21].

$$G(x, y) = \exp\left(-\frac{1}{2} \left[ \frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2} \right] \cos\left(\frac{2\pi x_\theta}{\lambda} + \psi\right)\right) \quad (3)$$

$$x_\theta = x \cos(\theta) + y \sin(\theta) \quad (4)$$

$$y_\theta = -x \sin(\theta) + y \cos(\theta) \quad (5)$$

$$\sigma_x = \sigma, \sigma_y = \frac{\sigma}{Y} \quad (6)$$

where, the output of Gabor filter is  $G(x, y)$ .  $(x, y)$  is the spatial coordinates.  $\sigma$  is the standard deviation of the Gaussian distribution.  $\theta$  is the rotation angle.  $\lambda$  is the sinusoidal operator wavelength.  $Y$  is the spatial dimension ratio.  $\psi$  is the phase offset. In the proposed solver, values  $\psi=0.2$ ,  $\Theta=0.67$ ,  $\sigma=0.65$ ,  $\lambda=0.57$ , and  $Y=0.35$  were chosen based on extensive experiments conducted on the puzzles database.

In the proposed solver, after computing the SSD and Gabor values for each puzzle pieces individually, these two values are added and listed for each puzzle piece. Then, for each chromosome, a fitness score is computed using equation (2) based on the combined SSD and Gabor (SSD+Gabor) values. Following this, two chromosomes are selected using the Roulette wheel selection method to participate in the crossover operator, which generates a child chromosome. The new generation, comprising the children and selected parent chromosomes, replaces the old generation. This procedure iterates until the predetermined number of iterations is completed.

The three-phase crossover operator [Sholomon, 13] begins by selecting the first puzzle randomly. It then checks if there is an edge where both parents agree, meaning both contain the piece in the same orientation. If such a piece exists, it is placed in the correct position. If the parents agree on two or more edges, one of these edges is randomly selected, and the corresponding piece is assigned. If no common edge is

found, the operator enters the second phase and searches for the best matching piece according to equations (7) and (8).

$$\forall x_k \in Pieces. C(x_i, x_j, R_1) \geq C(x_i, x_k, R_1) \quad (7)$$

$$\forall x_p \in Pieces. C(x_j, x_i, R_2) \geq C(x_j, x_p, R_2) \quad (8)$$

where *Pieces.* is a set of all the pieces of a puzzle and  $R_1$  and  $R_2$  are “complementary” spatial relationships (for example, if  $R_1$ =right,  $R_2$ =left and vice versa) and  $C(x_j, x_i, R)$  is the adjacency probability of two pieces in the  $R$  direction.

In the second phase, the operator checks whether one of the parents has a piece in the spatial relation  $RR$  to  $PP$ , which is also the best match for  $PP$  in that relation. If such a piece is found, it is selected and assigned. If there are several best matches, one is randomly selected. If a best match is found but has already been assigned to  $PP$ , it is ignored, and the search for other best matches continues. If no best match is found, the operator proceeds to the third phase.

In the third phase, the operator follows Prim’s algorithm to find a Minimum Spanning Tree (MST). The algorithm starts with an empty MST and maintains two sets of vertices (puzzle pieces). The first set contains the vertices already included in the MST, while the other set contains the vertices not yet included. At each step, the algorithm considers all edges connecting the two sets and picks the minimum weight edge. After selecting the edges, it moves the other endpoint of the edge to the set containing the MST. Thus, at every step of Prim’s algorithm, a cut is selected, and the corresponding vertex is included in the MST.

To select the Gabor compatibility, we investigated alternative methods such as the ORB descriptor [Chen, 22], SIFT descriptor [Rublee, 11], Co-occurrence matrix [Humeau-Heurtier, 19], Tamura texture [Humeau-Heurtier, 19], and Variogram texture [Humeau-Heurtier, 19]. However, results indicated that descriptors invariant to the rotation and scaling are unsuitable for the compatibility criterion. This is because areas of the same color in two adjacent pieces should not be considered adjacent if they differ in direction and size. The Co-occurrence matrix, Tamura, and Variogram texture extraction algorithms, which are well-known texture extractors, have also failed to improve the accuracy of the proposed solver either alone or when combined with the SSD criterion.

#### 4 Simulation Results, Comparisons, and Evaluation

The proposed GA-based solver simulation has been performed on a system with GPU Quadro P400 with 8GB memory, and CPU Core i7-8700 3.2GHz 12 processors. The initial generation comprised 1000 randomly selected chromosomes.

To obtain results, the proposed solver was executed 10 times on the database puzzles, then the average, best, and worst accuracies, along with their standard deviation, were reported based on criteria comparisons of the direct, neighbor, largest component, and complete reconstruction. Furthermore, to ensure a fair comparison of the proposed solver’s performance with similar solvers [Sholomon, 13], [Sholomon, 14], [Sholomon, 16], [Guo, 20], and [Bezulj, 18], we implemented all solvers with an initial random selection of 1000 chromosomes, and each solver was executed 10 times on the system. The accuracy comparisons of the proposed solver with the solver in [Sholomon, 14] are shown in Table 3, and Table 4.

Pieces number					Neighbor	Solver
3300	2360	805	540	432		
95.64	97.88	95.01	93.96	97.06	Average (%)	Proposed
94.93	97.33	91.45	91.33	94.44		in [Sholomon, 14]
96.11	98.05	95.76	94.83	98.8	Best (%)	Proposed
95.43	97.51	92.07	91.98	94.86		in [Sholomon, 14]
95.20	97.47	94.40	93.31	96.85	Worst (%)	Proposed
94.02	97.14	90.72	90/60	93.79		in [Sholomon, 14]
0.45	0.15	0.45	0.49	0.38	Standard Deviation (SD)	Proposed
0.79	0.18	0.48	0.49	0.39		in [Sholomon, 14]
Pieces number					Direct	GA-based solver
3300	2360	805	540	432		
94.75	93.61	89.32	86.45	93.12	Average (%)	Proposed
93.14	92.75	79.20	74.55	81.51		in [Sholomon, 14]
95.27	94.15	92.05	92.77	97.45	Best (%)	Proposed
94.45	93.43	87.76	89.57	94.58		in [Sholomon, 14]
94.33	92.84	80.64	84.00	89.00	Worst (%)	Proposed
92.39	91.74	76.15	74.26	77.77		in [Sholomon, 14]
0.49	0.69	3.59	3.82	3.03	SD	Proposed
1.14	0.90	10.56	12.49	10.47		in [Sholomon,14]
Pieces number					Largest component	Solver
3300	2360	805	540	432		
94.75	93.61	86.27	86.50	93.02	Average (%)	Proposed
93.14	92.63	74.91	71.28	81.49		in [Sholomon, 14]
95.27	94.19	91.92	92.78	97.45	Best (%)	Proposed
94.45	93.20	87.00	86.06	94.44		in [Sholomon, 14]
94.30	92.84	80.50	82.96	87.73	Worst (%)	Proposed
92.39	91.74	73.13	67.82	75.21		in [Sholomon,14]
0.49	0.69	3.60	3.63	3.36	SD	Proposed
0.69	0.85	14.30	13.43	13.1		in [Sholomon, 14]

Table 3: The accuracy comparison of the proposed solver with solver in [Sholomon, 14]

According to Table 3, the average accuracy of the proposed method with 10 independent runs, increased by 11.90%, 3.56%, and 15.22% compared to the solver in [Sholomon, 14] based on direct, neighbor, and largest component comparison criteria.

Pieces number			Criterion	Solver
805	540	432		
7	8	11	Completed reconstruction	Proposed
6	8	10		in [Sholomon, 14]

Table 4: The average accuracy comparison of the proposed solver with solver in [Sholomon, 14] on puzzles with 432, 540, and 805 pieces by using the completed reconstruction criterion.

According to Table 4, in the reconstruction of 20 puzzles with 432 pieces, 20 puzzles with 540 pieces, and 20 puzzles with 805 pieces with 10 runs, respectively, the proposed method completely solved 11, 8, and 7 puzzles. In all these cases, the proposed solver outperformed the solver in [Sholomon, 14].

Average accuracy by neighbor comparison (%)	Fitness	Crossover operator	Type 2 pieces number	Solver
92.60	LPQ	Three-phase procedure without phase 1	432	in [Bezulj, 18]
97.06	SSD+Gabor	Three-phase procedure		Proposed

Table 5: The average accuracy comparison of the proposed solver with solver in [Bezulj, 18].

The average accuracy comparison of the proposed method with the solver in [Bezulj, 18] on 20 puzzles with 432 pieces, with 10 runs, is shown in Table 5. It can be observed that the average accuracy of the proposed solver increased by 4.46% based on the neighbor comparison compared to the solver in [Bezulj, 18].

Average accuracy by neighbor comparison (%)	Fitness	Crossover operator	Type 2 pieces number	Solver
96.03	MGC	Three-phase procedure without first phase	432	in [Sholomon, 16]
96.60	SSD			Proposed
97.06	SSD+Gabor	Three-phase procedure		

Table 6: The average accuracy comparison of the proposed solver with solver in [Sholomon, 16].

The average accuracy comparison of the proposed GA-based solver with the solver in [Sholomon, 16] on 20 puzzles with 432 pieces, with 10 independent runs, is shown in Table 6. According to Table 6, the average accuracy of the proposed method compared to the solver in [Sholomon, 16] increased by 0.46%, and 1.03% based on the neighbor comparison using the MGC, and SSD criteria, respectively.

Type 2 pieces number			Direct comparison	Solver
1197	646	256	Average (%)	Proposed
87.32	98.76	99.61		in [Guo, 20]
84.70	98.42	99.37	Best (%)	Proposed
91.13	99.69	100		in [Guo, 20]
86.79	99.07	100		

85.12	98.14	99.21	Worst (%)	Proposed
79.43	97.21	98.44		in [Guo, 20]
2.44	0.60	0.42	SD	Proposed
2.72	0.68	0.61		in [Guo, 20]

Table 7: The accuracy comparison of the proposed solver with solver in [Guo, 20].

The comparison of the average, best, and worst accuracy of the proposed solver with the solver in [Guo, 20] on 25 puzzles with 256, 646, and 1196 pieces based on the direct comparison is shown in Table 7. It can be observed that the average accuracy of the proposed method increased by 2.62% based on direct comparisons compared to the solver in [Guo, 20].

Type 1 pieces number					Neighbor comparison	Solver
3300	2360	805	540	432		
96.57	97.91	97.73	96.66	98.21	Average (%)	Proposed
92.37	88.00	95.85	95.38	95.70		in [Sholomon, 13]
97.14	98.77	98.19	97.48	99.06	Best (%)	Proposed
92.76	88.86	96.26	95.96	96.16		in [Sholomon, 13]
96.38	97.40	96.97	95.93	97.67	Worst (%)	Proposed
91.91	87.52	95.35	94.65	95.21		in [Sholomon, 13]
0.41	0.74	0.35	0.44	0.41	SD	Proposed
0.27	0.38	0.31	0.40	0.34		in [Sholomon, 13]
Type 1 pieces number					Direct comparison	Solver
3300	2360	805	540	432		
94.81	93.48	95.35	93.10	95.18	Average (%)	Proposed
86.62	84.62	93.63	91.65	82.94		in [Sholomon, 13]
95.44	94.29	96.15	94.37	97.20	Best (%)	Proposed
89.92	85.73	94.67	92.75	86.19		in [Sholomon, 13]
93.65	93.06	94.04	91.85	91.92	Worst (%)	Proposed
65.42	82.73	92.79	90.57	80.56		in [Sholomon, 13]
1.01	0.69	0.65	0.64	1.27	SD	Proposed
7.19	0.86	0.62	0.65	2.62		in [Sholomon, 13]

Table 8: The accuracy comparison of the proposed solver with solver in [Sholomon, 13].

The average accuracy comparison of the proposed solver with the solver in [Sholomon, 13] on the database with 10 independent runs based on the neighbor, and

direct comparisons is shown in Table 8. It can be observed that the average accuracy of the proposed method increased by 9.91%, and 12.24%, respectively, based on neighbor, and direct comparisons compared to the solver in [Sholomon, 13].

solver in [Sholomon,14]	proposed solver based on the SSD	proposed solver based on the SSD+Gabor	Number of Pieces
7.71 sec.	7.71 sec.	16.10 sec.	432
10.94 sec.	10.94 sec.	22.40 sec.	540
17.60 sec.	17.60 sec.	38.23 sec.	805
2.27 min.	2.27 min.	5.04 min.	2360
3.30 min.	3.30 min.	8.10 min.	3300

Table 9: The execution time comparison between the proposed solver and solver in [Sholomon, 14]

Based on our investigation of state-of-the-art GA-based square jigsaw solvers mentioned in section 2, it was found that the solver in [Sholomon,14] has a shorter execution time compared to others. Therefore, for a fair comparison of execution times, the proposed solver and the solver in [Sholomon,14] were implemented on the system under identical conditions, and their execution times were recorded. The execution time comparison between the proposed solver and solver in [Sholomon,14] is shown in Table 9.

It is evident that the execution time of the proposed method based on SSD is less than that based on SSD+Gabor. The SSD computation time includes calculating the SSD values between all pairs of pieces of a puzzle and storing these values in a list. On the other hand, the SSD+Gabor computation time includes calculating the SSD values between all pairs of puzzle pieces, calculating the Gabor values for each pair, adding these two values together, and then sorting these sums in a list. As expected and as seen in Table 9, the execution time of the proposed method is comparable to that of the solver in [Sholomon, 14]. Both solvers solve square jigsaw puzzles the same as the execution time of solver in [Sholomon, 14], because both are solving the square jigsaw puzzles using GA with SSD fitness and a three-phase crossover operator. However, when using SSD+Gabor fitness in the proposed solver, the execution time increases, which reflects the trade-off for improving puzzle-solving accuracy.

The average accuracy comparison of the proposed method with state-of-the-art GA-based square jigsaw solvers mentioned in section 2, conducted on the same database and under identical conditions, is shown in Fig. 2. The horizontal axis shows the direct, neighbor, and largest component comparison criteria. As it is seen, the proposed method has a higher average accuracy in comparison with all these solvers. The minimum accuracy increase achieved by the proposed method is 3.56%. However, the maximum increase in execution time is 4.8 minutes. The computational workload and complexity of a puzzle solver are determined by the number of system clock cycles used during the solving process. Therefore, higher execution times indicate greater complexity and computational volume for a puzzle solver. Two examples of the solved jigsaw puzzles by the proposed solver are shown in Fig. 3.

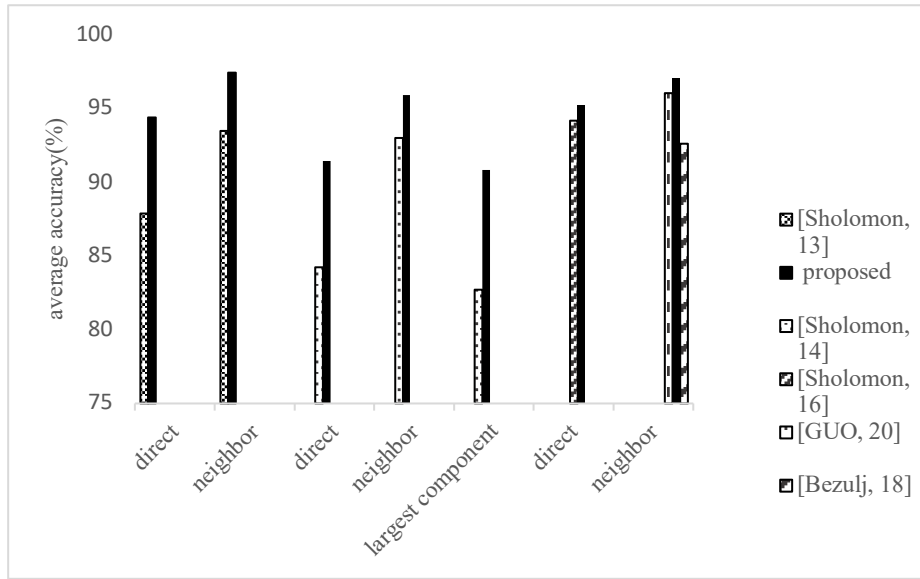


Figure 2: The average accuracy comparison of the proposed solver with state-of-the-art GA-based solvers



a



b

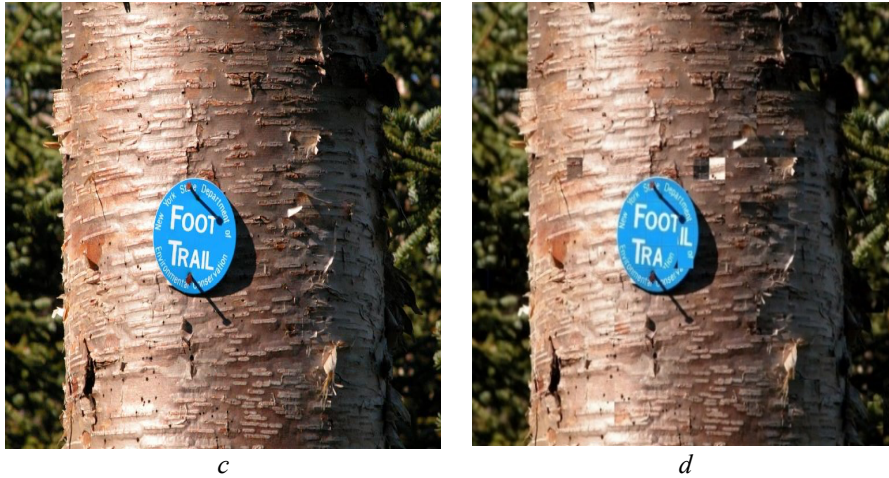


Figure 3: a) a puzzle with 432 pieces, solved in (b), c) a puzzle with 540 pieces, solved in (d)

## 5 Conclusions

The GA-based square jigsaw puzzle solver was developed with a fitness function that integrates the Gabor texture extraction and RGB color dissimilarity from puzzle pieces, along with a three-phase crossover operator for puzzle piece selection and assignment. The proposed solver was tested on the database with 66 images including puzzle images from the MIT, Pomeranz, and McGill databases with different number of pieces. The validation of simulation results included, the criteria of direct, neighbor, and largest component comparisons, and the complete reconstruction.

The SSD+Gabor fitness was chosen for the proposed method over alternatives such as ORB, SIFT, Co-occurrence, Tamura, and Variogram textures due to its significant improvement in accuracy observed across a large number of experiments.

The simulation results showed the accuracy of proposed GA-based solver has increased by 11.9%, 3.56%, and 15.22% according to the direct, neighbor, and largest component comparison criteria, respectively, compared to the state-of-art GA-based square jigsaw solvers. Additionally, the proposed solver successfully completed 11, 8, and 7 puzzles out of 20 puzzles with 432 pieces, 20 puzzles with 540 pieces, and 20 puzzles with 805 pieces, respectively, in 10 runs. These results suggest that the proposed solver can be easily implemented in real-world scenarios to solve square jigsaw puzzles of types 1, 2, and 3 with varying numbers of pieces.

As future work, we recommend investigating the performance of the proposed solver on the database puzzles using the HSV color space. If we restrict the color dissimilarity calculation to only the H component of puzzle pieces for the SSD+Gabor method, it could potentially lead to a significant decrease in execution time for the proposed solver.

## References

- [Ahmad, 22] Ahmad, I., Hwang, S. and Shin, S., "Determining jigsaw puzzle state from an image based on deep learning," in proceeding of 2022 International Conference on Artificial Intelligence in Information and Communication. Korea, p. 30-32 (2022) [doi: 10.1109/ICAIIIC54071.2022.9722672].
- [Bezulj, 18] Bezulj, M.P. and Jovicic, N.S., "Genetic Algorithm-Based Solver for Jigsaw Puzzles – Analysis and Improvement," in proceeding of 26th Telecommunications Forum (TELFOR). Serbia, p. 1-4 (2018) [doi: 10.1109/TELFOT.2018.8612038].
- [Chen, 22] Chen, J., Wang, J., Song, C., and Yin, H., "JIGSAW: Efficient and scalable path constraints fuzzing," in proceeding of 2022 IEEE Symposium on Security and Privacy. CA, (2022) [doi: 10.1109/SP46214.2022.9833796].
- [Chen, 23a] Chen, Y., Xia, R., Yang, K. and Zeu, K., "GCAM: lightweight image inpainting via group convolution and attention mechanism," International Journal of Machine Learning and Cybernetics (2023) [doi: 10.1007/s13042-023-01999-z].
- [Chen, 23b] Chen, Y., Xia, R., Yang, K. and Zeu, K., "DARGS: Image inpainting algorithm via deep attention residuals group and semantics," Journal of King Saud University - Computer and Information Sciences 35(6) (2023)
- [Chen, 24a] Chen, Y., Xia, R., Yang, K. and Zeu, K., "MICU: Image super-resolution via multi-level information compensation and U-Net," Expert System with Applications 245 (2024) [doi: 10.1016/j.eswa.2023.123111].
- [Chen, 24b] Chen, Y., Xia, R., Yang, K. and Zeu, K., "MFMAM: Image inpainting via multi-scale feature module with attention module," Computer Vision and Image Understanding 238 (2024) [doi: 10.1016/j.cviu.2023.103883].
- [Chen, 24c] Chen, Y., Xia, R., Yang, K. and Zeu, K., "DNNAM: Image inpainting algorithm via deep neural networks and attention mechanism," Applied Soft Computing 154 (2024) [doi: 10.1016/j.asoc.2024.111392].
- [Cho, 10] Cho, T.S., Avidan, S. and Freeman, W., "A probabilistic image jigsaw puzzle solver," in proceeding of IEEE Conference on Computer Vision and Pattern Recognition. USA, p. 183-190 (2010) [doi: 10.1109/CVPR.2010.5540212].
- [Guerroui, 18] Guerroui, N., and Seridi, H., "Solving computational square jigsaw puzzles with a novel pairwise compatibility measure," Journal of King Saud University-Computer and Information Sciences 32(8), 928-939 (2018).
- [Guo, 20] Guo, W., Wei, W., Zhang, Y., and Fu, A., "A Genetic Algorithm-Based Solver for Small-Scale Jigsaw Puzzles," In Proceeding of International Conference on Swarm Intelligence. p. 362-373 (2020) [doi: 10.1007/978-3-030-53956-6\_32].
- [<https://www.sciencedirect.com/science/article/pii/S1319157823001210>].
- [Humeau-Heurtier, 19] Humeau-Heurtier, A., "Texture feature extraction methods: A Survey," IEEE Access. 7, 8975-9000 (2019) [doi: 10.1109/ACCESS.2018.2890743].
- [Huroyan, 20] Huroyan, V., Lerman, G., and Wu, H.T., "Solving jigsaw puzzles by the graph connection Laplacian," SIAM journal on Imaging Sciences 13(4), 717-1753 (2020).
- [Li, 22] Li, R., Liu, S., Wang, G., Liu, G., and Zang, B., "Jigsaw GAN: Auxiliary learning for solving jigsaw puzzles with generative adversarial networks," IEEE Transactions on Image Processing 31, 513-524 (2022).

- [Markaki, 23] Markaki, S., and Panagiotakis, C., "Jigsaw puzzles solving techniques and applications: a survey," *visual Computer* 39(1), 4405-4421 (2023).
- [McGill-Smith, 19] McGill-Smith, S., and Green, R., "Jigsaw puzzle solver to locate piece position," in *proceeding of 2019 International Conference on Image and Vision Computing NewZealand*. NewZealand, p. 1-6 (2019) [doi: 10.1109/IVCNZ48456.2019.8961019].
- [Olmos, 15] Olmos, A., and Kingdom, F.A.A., "Mcgill calibrated color database. Website designed by Adriana Olmos," Last update: Nov. (2015) [URL: <http://tabby.vision.mcgill.ca/>].
- [Otomo, 21] Otomo, Y., and Igarashi, H., "Topology Optimization Using Gabor Filter: Application to Synchronous Reluctance Motor," *IEEE Transaction on Magnetics*. 57(6), (2021) [doi: 10.1109/TMAG.2021.3057402].
- [Paikin, 15] Paikin, G., and Tal, A., "Solving multiple square jigsaw puzzles with missing pieces," in *proceeding of 2015 IEEE Conference on Computer Vision and Pattern Recognition*. USA, p. 4832-4839 (2015) [doi: 10.1109/CVPR.2015.7299116].
- [Pang, 20] Pang, K., Yang, Y., Hospedales, T. M., Xiang, T., and Song, Y. Z., "Solving mixed-model jigsaw puzzle for fine-grained sketch-based image retrieval," in *proceeding of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. USA, p. 10347-10355 (2020) [doi: 10.1109/CVPR42600.2020.01036].
- [Pomeranz, 11] Pomeranz, D., Shemesh, M., and Ben-Shahar, O., "A fully automated greedy square jigsaw puzzle solver," in *proceeding of IEEE Conference on Computer Vision and Pattern Recognition*. USA, p. 9-16 (2011) [doi: 10.1109/CVPR.2011.5995331].
- [Rublee, 11] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An efficient alternative to SIFT or SURF," in *proceeding of 2011 International Conference on Computer Vision*. Spain, (2011) [doi: 10.1109/ICCV.2011.6126544].
- [Sholomon, 13] Sholomon, D., David, E., and Netanyahu, N.S., "A genetic algorithm-based solver for very large jigsaw puzzles," in *Proceeding of IEEE International Conference on Computer Vision and Pattern Recognition*. OR, p.1767-1774 (2013) [doi: 10.1109/CVPR.2013.231].
- [Sholomon, 14] Sholomon, D., David, E., and Netanyahu, N.S., "A generalized genetic algorithm-based solver for very large jigsaw puzzles of complex types," in *Proceeding of Twenty-Eighth AAAI Conference on Artificial Intelligence*. CA, p.2839-2845 (2014) [doi: 10.48550/arXiv.1711.06768].
- [Sholomon, 16] Sholomon, D., David, E., and Netanyahu, N.S., "An automatic solver for very large jigsaw puzzle using genetic algorithms," *Genetic Programming and Evolvable Machines* 17(3), 291-313 (2016).
- [Son, 18] Son, K., Hays, J., and Cooper, D.B., "Solving square jigsaw puzzle by hierarchical loop constraints," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41(9), 2222-2235 (2018).