KNN-BASED JIGSAW PUZZLE SOLVER

Farzana S, Shaikh Haris Jamal

International Institute of Information Technology, Hyderabad

ABSTRACT

We present two approaches for solving square jigsaw puzzles with known dimensions: a KNN-based compatibility model with greedy assembly and a Paikin-Tal inspired solver with dimensional constraints. The first approach extracts multi-modal features from puzzle pieces including boundary statistics, gradient information, and texture descriptors, then uses k-nearest neighbors to learn compatibility between adjacent pieces. Multiple assembly strategies are evaluated including random initialization, center-fixed, boundary-fixed, and random-fixed configurations. The second approach implements gradient-based dissimilarity measures with best-buddy relationships and compatibility scoring. Both methods are evaluated on an 8x8 grid using direct accuracy, neighbor accuracy, and structural similarity metrics. The source code for both solvers is publicly available at `github.com/farzanashaju/puzzle-solver`.

***Index Terms*—** Image Reconstruction, Jigsaw Puzzle Solver

1. INTRODUCTION

Jigsaw puzzle solving is a challenging computer vision problem with applications in image forensics, archaeological artifact reconstruction, and document restoration. The problem involves reassembling a complete image from shuffled fragments without knowledge of their correct positions. This work focuses on square puzzles with known dimensions, where pieces form a regular 8x8 grid.

We present two distinct approaches to this problem. The first method employs a machine learning strategy using k-nearest neighbors to learn compatibility patterns from positive and negative adjacency examples. The second implements a gradient-based dissimilarity measure inspired by the Paikin-Tal algorithm with dimensional constraints. Both approaches are evaluated comprehensively across multiple images with various assembly strategies.

2. METHOD 1: KNN-BASED SOLVER

2.1. Feature Extraction

Each puzzle piece is processed to extract numerical features describing its colour, texture, and edge properties. These features are later used to determine whether two pieces are likely to be adjacent.

Boundary Features: For each of the four sides of a piece (top, bottom, left, right), we extract a narrow strip of 5 pixels width from the boundary. From this strip:

- For each RGB channel, we compute the mean, standard deviation, and median of pixel values.
- We convert the boundary to grayscale and compute horizontal and vertical Sobel gradients. From each gradient map, we record the mean absolute value and standard deviation.
- We extract texture information using Local Binary Patterns (LBP) with 8 neighbours and radius 1. We form a 10-bin histogram of LBP values and normalise it.

Patch Features: To represent the piece as a whole rather than only its boundaries, we extract:

- The mean and standard deviation of each RGB channel across the entire patch.
- Three dominant colours obtained by applying K-Means clustering ($k = 3$) to the pixel values, and storing the resulting cluster centroids.
- The standard deviation of the grayscale version of the full piece, capturing global texture variation.

These boundary and patch features together produce a descriptor that distinguishes pieces based on both their internal appearance and their potential compatibility with neighbouring pieces.

2.2. Compatibility Model

The goal of the compatibility model is to estimate how likely it is that one piece belongs next to another. Instead of learning a parametric classifier, we use a K-Nearest Neighbour (KNN) model, which stores examples and compares new pairs directly to known cases.

We build two separate models-a horizontal model for left–right adjacency and a vertical model for top–bottom adjacency. The true puzzle layout is known during training. We utilize both positive pairs (real neighbours from the ground

truth image) and negative pairs (randomly sampled piece pairs that are not adjacent). A 2:1 negative-to-positive ratio is used. For each ordered pair, we create a feature vector by concatenating the boundary features of the relevant touching sides, the patch features of both pieces and the absolute difference between the two boundary feature vectors.

We apply Principal Component Analysis (PCA) to reduce the dimensionality of the pairwise feature vectors. The transformed feature vectors are stored in the KNN model using Euclidean distance. No parameter optimisation occurs — the model simply indexes these examples for future lookup.

Given two pieces during assembly, we extract their pair-feature vector, apply PCA, and query the k nearest stored examples ($k = 5$ in our implementation). Each neighbour has a binary label indicating whether it was a true adjacent pair. The compatibility score combines the proportion of positive neighbours and a distance-weighted version of that proportion:

$$C = 0.6 \cdot \frac{\sum w_i \cdot y_i}{\sum w_i} + 0.4 \cdot \frac{\sum y_i}{k}, \tag{1}$$

where $y_i \in \{0, 1\}$ is the label of neighbour i, d_i is its distance to the query point, and $w_i = 1/(d_i + \epsilon)$ is an inverse-distance weight. Higher scores indicate stronger evidence that the two pieces should be adjacent.

2.3. Assembly Strategies

The puzzle is assembled using a greedy search guided by the compatibility model. All strategies iteratively place pieces onto an initially empty 8×8 grid. At each step, the solver evaluates all unplaced pieces at all empty positions and chooses the placement with the highest local compatibility score relative to already placed neighbours.

We evaluate four initialization strategies:

- **Random**: A random piece is placed at the centre of the grid, and assembly proceeds outward.
- **Center-Fixed**: The true centre piece is placed first. This anchors the solution spatially.
- **Boundary-Fixed**: All pieces belonging to the puzzle border are placed in their correct positions before greedy assembly begins.
- **Random-Fixed**: A number (n) of random pieces are fixed at their correct locations. We test $n = 3, 6, 9$ to analyse how increasing constraints affect reconstruction.

If the greedy process encounters a situation where no placement yields a meaningful compatibility score, the solver inserts a remaining piece randomly to prevent deadlock and continues assembly.

3. METHOD 2: PAIKIN–TAL INSPIRED SOLVER

This method adapts the core ideas of Paikin and Tal [1], who proposed a fast, fully-automatic, deterministic, greedy solver for square jigsaw puzzles. Their approach is notable for being general enough to handle difficult puzzle scenarios, such as unknown puzzle dimensions, missing pieces, unknown orientations, and mixed-piece inputs. The effectiveness of the method relies on establishing not just how similar two pieces appear, but how reliable that similarity is, enabling confident early placements and avoiding typical greedy failures.

The solver progresses through three conceptual stages: (i) quantifying how well pairs of pieces fit together, (ii) selecting a distinctive and reliable starting region, and (iii) iteratively assembling the puzzle using the most confident match available at each step.

3.1. Dissimilarity Calculation

The first step assesses how likely two pieces belong next to each other. Rather than matching boundaries directly, the method predicts what the boundary of one piece should look like if a true neighboring piece were present. Each piece is converted to LAB color space to better reflect perceptual differences, and boundary pixels are represented as narrow one-pixel-wide strips along each edge.

For a given relation direction r (right, down, left, up), the dissimilarity between piece i and piece j is computed using:

$$D(i, j, r) = \sum |\text{pred}_i(r) - \text{bound}_j(r^{-1})| \tag{2}$$

The predicted boundary is obtained through linear extrapolation of the inner boundary:

$$\text{pred}_i(r) = 2 \cdot \text{bound}_i(r) - \text{bound}_i^{\text{inner}}(r) \tag{3}$$

This prediction-based model simplifies earlier approaches, yet remains highly effective: it captures subtle edge transitions and leverages the L1 norm, which is computationally efficient and empirically robust. Importantly, D is asymmetric, meaning $D(i, j, r) \neq D(j, i, r^{-1})$, reflecting the directional nature of prediction.

3.2. Compatibility and Best Buddies

Although low dissimilarity suggests that two pieces may match, this alone is not sufficient—particularly in smooth or uniform areas where many pieces appear similar. To address this, the solver introduces a notion of reliability via a normalized compatibility score:

$$C(i, j, r) = \max\left(0, 1 - \frac{D(i, j, r)}{D(i, j_2, r)}\right) \tag{4}$$

Here, j_2 denotes the second-best neighbor for piece i in direction r. A compatibility score close to 1 indicates that j

(a) Original (b) KNN: Random (c) KNN: Boundary-fixed (d) Paikin–Tal

Fig. 1: Comparison of reconstruction results.

is not only the best match, but also significantly better than all alternatives. This discourages premature placements in ambiguous textures and prioritizes distinctive, confidently matched regions.

A particularly useful construct is the notion of **"best buddy"**: two pieces form a best-buddy pair if each selects the other as its highest-compatibility neighbor in opposing directions.

3.3. Assembly with Constraints

The assembly proceeds in a deterministic, greedy manner. Unlike prior solvers that selected an arbitrary seed, this method begins with a highly reliable starting piece—one that has best buddies in all four directions, and whose neighbors are themselves strongly supported. We rank candidate placements based on mutual compatibility, and the solver iteratively attaches the most reliable neighbor to the current configuration.

4. EXPERIMENTAL SETUP

Both methods are evaluated on images converted to 8x8 grids (64 pieces total). Images are resized to exact multiples of the grid dimensions, divided into square pieces, and shuffled randomly. For reproducibility, random seeds are fixed per image while varying across the dataset.

4.1. Evaluation Metrics

We employ three complementary metrics:

- **Direct Accuracy**: The fraction of pieces placed at their correct grid positions.
- **Neighbor Accuracy**: The fraction of adjacent piece pairs that are true neighbors in the original image.
- **SSIM**: Structural similarity index between reconstructed and original images.

Table 1: Performance comparison across methods

Method / Strategy	Direct	Neighbor	SSIM
KNN-Based Solver			
Random	0.03	0.25	0.38
Center-Fixed	0.12	0.26	0.47
Boundary-Fixed	**0.65**	**0.55**	**0.78**
Random-Fixed (3)	0.20	0.26	0.52
Random-Fixed (6)	0.32	0.29	0.60
Random-Fixed (9)	0.41	0.34	0.66
Paikin-Tal Solver			
Optimal	**0.81**	**0.92**	**0.90**

5. RESULTS AND DISCUSSION

Table 1 presents performance across both methods and all initialization strategies. The KNN-based solver shows strong dependence on initialization constraints. Random initialization achieves only 3% direct accuracy (SSIM 0.38), while center-fixed provides marginal improvement at 12% (SSIM 0.47). Boundary-fixed initialization yields 65% direct accuracy and 0.78 SSIM by establishing a reliable frame. Random-fixed strategies show monotonic improvement with additional constraints: 20%, 32%, and 41% direct accuracy for 3, 6, and 9 fixed pieces respectively.

The Paikin-Tal solver substantially outperforms all KNN variants, achieving 81% direct accuracy, 92% neighbor accuracy, and 0.90 SSIM. The high neighbor accuracy indicates the method correctly identifies adjacencies even when global positioning errors occur. This 16-point performance gap over the best KNN strategy suggests gradient-based dissimilarity measures capture piece compatibility more effectively than learned features. The prediction-based approach directly models boundary continuity, while the best-buddy criterion prevents premature placements in ambiguous regions. Additionally, the gradient method requires no training phase and operates deterministically.

6. CONCLUSION

We presented two approaches for solving 8x8 square jigsaw puzzles. The KNN-based method achieves 65% direct accuracy with boundary-fixed initialization, demonstrating that learned compatibility models require substantial spatial constraints for effective greedy assembly. The Paikin-Tal inspired solver outperforms all KNN variants with 81% direct accuracy and 92% neighbor accuracy through gradient-based dissimilarity and best-buddy relationships.

Key findings include: initialization strategy critically impacts performance (3% for random vs 65% for boundary-fixed); constraint level shows monotonic improvement in KNN assembly; and gradient-based methods fundamentally outperform feature-based learning for this domain. Future work should explore unknown dimensions, missing pieces, hybrid approaches combining learned features with gradient measures, and global optimization to correct locally-correct but globally-misplaced regions.

7. REFERENCES

[1] Genady Paikin and Ayellet Tal, "Solving multiple square jigsaw puzzles with missing pieces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4832–4839.