Jigsaw Puzzle Solver Milestone 2 Report

Team No: **5**

Contributors:

- **Karim Samer Mostafa** **23p0439**
- **Youssef Maged Morees** **2301081**
- **Omar Gamil Anwar Osman** **23P0438**
- **Paula Hanna Naguib Hanna** **23P0440**
- **Ahmed Mohamed Elian** **23p0446**

Course: **Image Processing** CSE381 (UG2023)

Prof. **Mahmoud Khalil**

TA. **Eng. Ali Ahmed, Eng. Dina Zakaria**

Project GitHub Repo : https://github.com/karimsamer100/Image-Processing-Jigsaw-Puzzle-Solver-Project

1. Overview

Milestone 2 requires using Milestone 1 outputs to **assemble** or **identify likely neighboring pieces** ("fix the picture"), using **classical image processing only (no AI/ML)**, and to provide: (1) assembly implementation, (2) visualization of matches, (3) documentation, (4) demo on clean + challenging cases, and (5) discussion/reflection.

---

2. Inputs and Outputs

2.1 Inputs (from MS1 outputs)

- **Piece images:** processed_output/puzzle_{size}/pieces/*.png|jpg
- **Piece naming:** convention used to parse puzzle ID and true position (row/col) ex: **11_r0c0.png**

2.2 Outputs (MS2 artifacts)

1. Assembled puzzle image (reconstructed grid).
2. Visualization figure including:
    - Our assembled result
    - Ground truth full image
    - Per-cell correctness visualization using SSIM thresholding (green/red borders)
3. Saved images under:
    - ms2_results_puzzle_2x2
    - ms2_results_puzzle_4x4
    - ms2_results_puzzle_8x8

3. System Design Overview (Pipeline)

3.1 Piece Representation (Edge Features)

Each piece is represented by lightweight boundary descriptors computed from the MS1 piece image:

1. **Edge color profile** : Mean color along a narrow strip (depth 3–4 px) on each side.
2. **Edge gradient descriptor**: Sobel magnitude profile (depth 5–6 px) per side to capture edge texture transitions.
3. **Edge region patch**: A small strip/patch (depth 10–14 px) per side for photometric similarity (SSIM/MSE style signals).

3.2 To increase robustness, pieces are normalized before descriptor extraction using:

- Grayscale conversion + Gaussian blur
- CLAHE (clipLimit=2.0, tileGrid=8×8) Descriptors are extracted from the normalized image, while reconstruction uses the original BGR piece image for correct visual output.

4. Assembly Methods (by puzzle size)

Because 2×2, 4×4, and 8×8 have very different search complexity, we used different solvers while keeping the same "edge similarity drives adjacency" principle.

4.1- [2×2] Beam Search Assembly with Mutual Boosting

Goal: maximize total compatibility across placed neighbors without brute force.

Edge-to-edge match score combines:

- Sobel "complement" consistency (dominant signal)
- SSIM on edge region patches (fine alignment)
- Intensity MSE term (fallback and stabilization)
- Normalized cross-correlation (NCC) on edge color profiles (supporting cue)

Beam search: places pieces left-to-right, top-to-bottom, keeping top-K partial solutions.

Mutual-strength bonus: encourages pairs that agree strongly in both directions.

Why logical for 2×2: search space is small; beam search is reliable and fast, and tends to produce exact solutions.

4.2 [**4×4] Global Frontier Growing with LAB Seam Cost + Refinement**

Goal: produce a globally consistent grid using a cost-minimization view of seams.

- **Convert piece images to LAB to compute perceptual seam differences.**
- **Compute pairwise costs:**
  - Horizontal seam cost (right edge of A vs left edge of B)
  - Vertical seam cost (bottom edge of A vs top edge of B)
  - Includes both pixel difference and a gradient consistency penalty along the seam.
- **Frontier growing:**
  - Start from a seed at (0,0)
  - For each frontier cell, pick best candidate based on already placed neighbors
  - Use a "ratio test" (best/second-best) to prefer confident placements
  - Try all seeds; keep grid with minimum total energy
- **Local refinement**:
  - Adjacent swaps (right/bottom swaps) if they reduce global energy.

Why logical for 4×4: full pairwise cost matrix is feasible; global energy provides a stable objective; local swaps often correct early mistakes.

4.3 [**8×8] Tuned Hybrid Cost + Improved Frontier + Smart Refinement**

Goal: handle the much larger combinatorial complexity with a fast greedy strategy plus strong cost modeling.

- **Hybrid cost between two candidate neighbors combines:**
  1. LAB seam pixel difference
  2. LAB seam gradient consistency
  3. Sobel-profile complement error (scaled)
- **Improved frontier**:
  - Evaluate multiple promising seeds (e.g., 24 seeds)
  - Greedily place best candidates on the frontier using a confidence ratio
  - Keep best grid by total energy
- **Smart refinement**:
  - Adjacent swaps pass
  - 2×2 block swaps (coarser correction mechanism) to escape local minima

Why logical for 8×8: exhaustive/beam approaches become expensive; the frontier approach provides a practical compute-time path while still optimizing a meaningful objective.

---

5. Match Visualization and Evaluation

5.1 Implemented visualization (current)

For each puzzle:

- Left panel: assembled image
- Middle panel: ground truth
- Right panel: ground-truth tiles bordered:
  - **Green** if SSIM(assembled_tile, GT_tile) ≥ threshold (e.g., 0.60)
  - **Red** otherwise
- Reported metric: "visual correctness/accuracy" = % of green tiles
  This provides a clear, human-interpretable result and an objective proxy metric per cell.

5.2 Alignment with the spec's "visualization of matches"

The project spec explicitly asks to **display matching results visually** (e.g., overlay matched edges or connect pieces with lines). My current visualization primarily shows final placement correctness vs ground truth, not explicit candidate edge-to-edge match lines.

6. Demonstration Plan (what to show)

Milestone 2 requires a demo (video or notebook) , the Colab notebook is included in the github link , with visualizations

7. Conclusion:

This project successfully demonstrates that classical computer vision techniques can effectively solve jigsaw puzzle assembly problems without relying on machine learning or deep learning approaches. Through careful design of preprocessing pipelines, edge descriptor extraction, and optimization-based assembly algorithms, we achieved strong performance across multiple puzzle complexities.

8.References:

1. **Cho, T. S., Avidan, S., & Freeman, W. T.** (2010). "The Patch Transform and Its Applications to Image Editing." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1064-1071.
2. **Pomeranz, D., Shemesh, M., & Ben-Shahar, O.** (2011). "A Fully Automated Greedy Square Jigsaw Puzzle Solver." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9-16.
3. **Gallagher, A. C.** (2012). "Jigsaw Puzzles with Pieces of Unknown Orientation." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 382-389.