Interactive Image Mosaic Generator: Performance Analysis and Quality Assessment

Abstract

This report presents a comprehensive analysis of a custom color- and texture-based mapping strategy for mosaic image generation. The system uses vectorized computation for efficient tile matching across multiple grid sizes and leverages precomputed tile caches for fast reuse. We evaluate quality with PSNR and several SSIM variants. Results show near-linear performance scaling and substantial speedups over a naive loop implementation, while the metrics highlight the trade-off between global visual fidelity and local detail preservation inherent to mosaic reconstruction.

1. Introduction and Methodology

1.1 Algorithm Design

The mosaic generation system reconstructs input images by replacing uniform grid cells with optimally matched tiles from a preprocessed database. The core innovation lies in the hybrid feature representation that captures both perceptual color information and textural characteristics:

Color Features: We take the average color of each cell using the CIELAB (Lab) color space. Lab has three numbers: L* (lightness/brightness), a* (green ↔ red), and b* (blue ↔ yellow). Lab is designed to match how people see color, so differences in Lab line up better with what looks “closer” to us than RGB does. That makes color matching more reliable.

Texture Features: We measure how “edgy” each cell is using the Sobel filter on the L* (lightness) channel. We take the average edge strength and store it as one number. Bigger = more edges/texture. This gives us a quick sense of local texture without doing anything fancy.

Tile Preprocessing: Before matching, we make every tile in all orientations so we don’t have to rotate/flip during runtime:

- 4 rotations: 0°, 90°, 180°, 270°
- 2 flips each way: horizontal (on/off) and vertical (on/off)
- That’s 16 versions per tile.

For each version, we save the pixels and the features (color + texture) into a compressed NumPy file. At runtime, we just look up these precomputed versions, which makes matching much faster.

1.2 Distance Metric and Optimization

Tile selection employs a weighted distance function combining normalized color and texture components:

For image block *b* with features (c_b, t_b) and tile variant *t* with features (c_t, t_t):

- Color distance: $d_c = ||c_b - c_t||_2$
- Texture distance: $d_t = (t_b - t_t)^2$
- Combined distance: $D = \alpha \cdot d_c + (1-\alpha) \cdot d_t$

Distance normalization to [0,1] ensures balanced weighting between color and texture components. The weighting parameter $\alpha = 0.7$ provides 70% emphasis on color matching with 30% texture contribution.

Vectorization Strategy: The matching process leverages NumPy broadcasting to compute the full B×T distance matrix (B blocks, T tile variants) in vectorized operations, eliminating Python loop overhead and utilizing optimized BLAS kernels for substantial performance gains.

1.3 Experimental Configuration

Fixed Parameters: 16×16 pixel tile size with single cache containing all geometric variants. No color quantization applied (k=0). Single natural photograph used across all experiments.

Variable Parameters: Grid dimensions spanning 16×16 (256 blocks) to 128×128 (16,384 blocks), providing 64× variation in computational complexity.

Comparison Baselines: Vectorized implementation benchmarked against naive double-loop Python implementation to quantify optimization benefits.

2. Results and Analysis

2.1 Performance Scaling

The vectorized matching stage demonstrates predictable scaling behavior across grid sizes:

Grid Size	Blocks (B)	Matching Time (s)	Total Pipeline (s)
16×16	256	0.46	0.50
32×32	1,024	1.61	1.73
64×64	4,096	5.98	6.48
128×128	16,384	23.95	25.71

Scaling Analysis: Power law fitting reveals T(B) ≈ 0.00225 · B^0.95. That exponent (~0.95) is very close to 1, so the time grows almost linearly as you add more cells. The matching stage dominates total pipeline time (>90%), while preprocessing and reconstruction contribute minimal overhead, validating the focus on matching optimization.

2.2 Vectorization Efficiency

Performance comparison between vectorized NumPy operations and equivalent Python loops reveals substantial computational advantages:

Grid Size	Vectorized (s)	Loop-based (s)	Speedup Factor
16×16	0.35	81.72	232×
32×32	1.88	328.80	175×
64×64	6.14	1,323.79	215×

The 170-230× performance improvement demonstrates the critical importance of vectorized computation in high-dimensional matching problems. Loop-based implementations involves a lot of tiny computations that are performed one after another, while NumPy executes equivalent operations in optimized C code with efficient memory access patterns.

2.3 Quality Assessment

Multi-metric evaluation reveals the complex trade-offs in mosaic reconstruction quality:

Grid Size	MSE ↓	PSNR (dB) ↑	SSIM (pixel) ↑	SSIM (block) ↑	SSIM (global) ↑	MS-SSIM ↑
16×16	1819.1	15.53	0.119	0.069	0.831	0.855
32×32	1291.5	17.02	0.133	0.086	0.885	0.902
64×64	1044.9	17.94	0.140	0.102	0.913	0.926
128×128	898.1	18.60	0.151	0.115	0.929	0.939

Pixel-Level Metrics: MSE decreases and PSNR increases consistently with finer grids, reflecting improved local color approximation. However, absolute values remain limited due to fundamental texture replacement inherent in mosaic generation.

Structural Metrics: SSIM variants reveal a nuanced quality landscape:

- **Local SSIM** (pixel and block-wise): Modest improvement with grid refinement, but absolute values remain low (0.069-0.115) due to poor texture of the generated pattern replacement destroying local structural coherence.
- **Global SSIM**: High baseline values (0.831-0.929) with continued improvement, indicating preservation of overall luminance distribution and large-scale structural elements.
- **Multi-Scale SSIM**: Consistently highest scores (0.855-0.939) confirm that mosaics maintain visual coherence across multiple spatial frequencies.

3. Discussion and Implications

3.1 Metric Interpretation

The quality evaluation reveals the fundamental characteristics of mosaic reconstruction:

Energy-Based Metrics (MSE/PSNR) provide straightforward fidelity assessment but lack perceptual relevance. The steady improvement with grid refinement reflects better local color approximation.

Local Structural Metrics (SSIM pixel/block) capture the similarity in the very low level. This would cause the value to drop of drastically even if there are little differences in the generated output.

Global Structural Metrics (global SSIM, MS-SSIM) align with human perception of mosaic quality as they evaluate the generated output as a whole, similar to what we humans do. High scores indicate successful preservation of overall image structure and tonal relationships, which dominate visual impression at typical viewing distances.

4. Conclusions

This analysis demonstrates the effectiveness of hybrid color-texture matching for mosaic generation, with several key findings:

1. **Performance Predictability**: Near-linear scaling enables accurate resource planning and interactive system design within computational constraints.
2. **Optimization Criticality**: Vectorized implementations provide order-of-magnitude performance improvements when compared to loop-based implementations.
3. **Perceptual Alignment**: High global and multi-scale SSIM scores correlate with subjective mosaic quality, supporting the use of these metrics for automated quality assessment and parameter optimization.

The system successfully balances computational efficiency with reconstruction quality, providing a robust foundation for both research applications and interactive mosaic generation tools. Future work could explore adaptive grid sizing, hierarchical tile matching, and perceptually-weighted distance metrics to further improve the balance between computational cost and visual fidelity. It.