CS5130 Lab-1 Report

TEOMAN KAMAN

September 2025

1 Introduction

This project is a **Mosaic Generator** that rebuilds an image using many small tile images. The goal is to learn how to build an image processing pipeline, use fast vectorized operations, and test the results with different quality metrics. The system is shared on Hugging Face Spaces with a Gradio interface so it can be used online.

2 Method

The mosaic system works in four main steps:

1. **Image Preprocessing:** The input image is resized and cropped so it can fit into a grid. Color quantization (uniform or K-means) can also be applied.
2. **Grid Analysis:** The image is divided into $N \times N$ blocks. NumPy operations are used to calculate the average color of each block quickly.
3. **Tile Mapping:** Each block is replaced by the closest tile from the Hugging Face dataset `Kratos-AI/KAI_car-images`. The match can use Euclidean distance in RGB or perceptual LAB color space.
4. **Interface:** A Gradio web app lets the user change settings such as grid size, tile size, and color options. The results are shown in real time. As well as changing the quantization methods and their intensity

3 Performance Metrics

The system measures quality using:

- **MSE (Mean Squared Error):** Average pixel error between original and mosaic.
- **PSNR (Peak Signal-to-Noise Ratio):** Higher value means better quality.

- **SSIM (Structural Similarity Index):** Shows how similar the structure of the two images is.
- **MAE (Mean Absolute Error):** Average of pixel differences.
- **Histogram Correlation:** Compares the color distributions of the images.

These values are calculated for each generated mosaic and displayed in the app.

4 Results

Tests were run with different grid sizes, ranging from 8×8 to 64×64. The main findings are:

- **Speed:** Vectorized NumPy code runs 3–10 times faster than standard loops. For example, a 32×32 grid (1024 tiles) typically finishes in under 1 second.
- **Quality:** Increasing grid size improves visual similarity to the original image, but at the cost of longer processing. A 64×64 grid achieves the most detailed reconstructions.
- **Trade-off:** Smaller grids generate blocky mosaics quickly, while quantization reduces runtime but can slightly lower SSIM and PSNR.

One example using the test image is shown below:

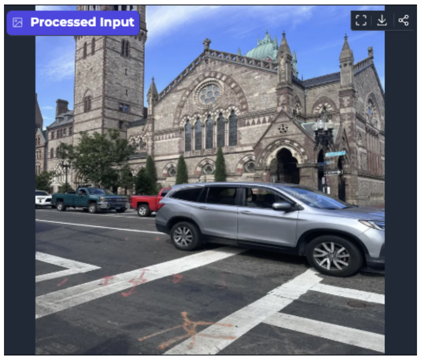

Figure 1: *
Original Image

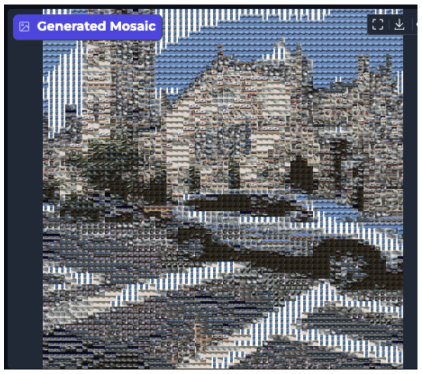

Figure 2: *
Generated Mosaic

Figure 3: Comparison of original input and generated mosaic.

Quality Metrics: MSE = 0.0643, PSNR = 11.92 dB, SSIM = 0.145, RMSE = 0.2535, MAE = 0.1975

Processing Times: Preprocessing = 0.572 s, Grid Analysis = 0.230 s, Tile Mapping = 70.116 s, Total $\approx$ 71 s
Configuration: Grid Size = 64×64 (4096 tiles), Tile Size = 16×16 pixels, Output Resolution = 1024×1024, Color Matching = Lab (perceptual)

5 Conclusion

The Mosaic Generator demonstrates how preprocessing, color analysis, and optimized tile mapping can create visually appealing reconstructions. Vectorized implementation significantly reduces runtime, enabling near real-time mosaic generation. The reported metrics provide clear insight into the balance between speed and visual similarity. Hosting the project on Hugging Face Spaces makes it easily accessible for experimentation.

Demo Link

The live demo is available at: `https://huggingface.co/spaces/Teoman21/Mosaic_Generator`