Rendering Photos with Two-Colored Convex Tiles

Mathewe Banda and Declan Galleher

Abstract

We construct mosaic reproductions of photographs using two-colored convex quadrilateral tiles whose adjacent corners touch, producing recognizable images from a simple geometric vocabulary.

1 Introduction

Robert Bosch's work on edge-constrained truchet tile mosaics [1] showed that expressive images can emerge from highly constrained tiling systems. We wanted to see how far a similar idea could go with convex polygons—specifically, four-sided convex tiles whose corners must touch their neighbors.

This touching-corners constraint makes every tile in the mosaic dependent on the tiles around it, which at first seemed likely to wash out any recognizable imagery. The key insight was a "slider" mechanism (described below) that lets each tile grow or shrink continuously in response to local brightness, while still satisfying the adjacency constraint. Combined with a scheme for optimizing individual tile brightness, the system produces clear photographic reproductions.

2 Tiling Method

Our implementation uses Python 3 to generate `.eps` mosaic images. We begin with an $x \times y$ pixel photograph, convert it to greyscale, and scale the dimensions by a factor ϵ so that the image divides evenly into $n \times n$ mosaic squares. The result is an $\frac{x}{n} \times \frac{y}{n}$ grid of square objects.

Each square stores:

- the average pixel brightness of its $n \times n$ region,
- the x–y coordinates of its top-left corner in the original image,
- an integer $c \in \{0, 1, 2, 3\}$ that determines how the interior polygon is oriented and colored, and
- the coordinates of the four vertices of the polygon it contains.

2.1 Sliders

As shown in Figure 1, we position each polygon's vertices using "sliders" along the shared edges of adjacent squares. A slider runs along an edge of length n, shortened by a buffer δ at each end.

The position of the vertex on a slider is set by the average brightness of the two squares that share that edge: a value near 255 (bright) pushes the vertex toward one end, while a value near 0 (dark) pushes it toward the other.

The integer c determines both the polygon's color (black or white) and the orientation of its slider endpoints. Using Figure 1 as reference: $c = 0$ is the top-right square, $c = 1$ top-left, $c = 2$ bottom-left, and $c = 3$ bottom-right. Odd rows alternate between $c = 1$ and $c = 0$; even rows alternate between $c = 2$ and $c = 3$. Every square starts at its darkest possible configuration, and sliders can only move outward from there.

This arrangement guarantees that every polygon can grow or shrink freely—becoming brighter or darker—while maintaining the touching-corners constraint. The mosaic effect arises naturally: black polygons shrink in bright regions and white polygons expand, and vice versa in dark regions.

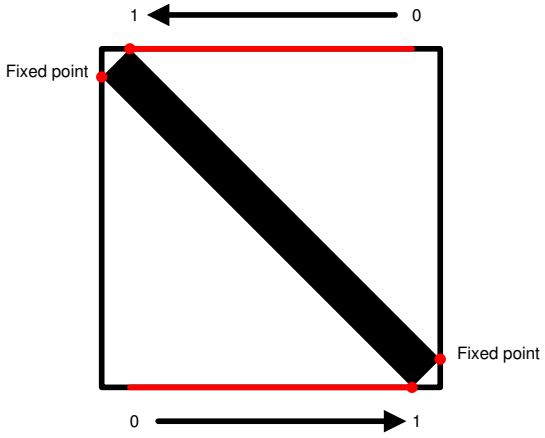

Figure 1: *Close-up of the tiling. Red dots mark slider positions; arrows indicate the direction of increasing brightness.*

3 Tile Variations

Our initial approach—setting each slider position from the average brightness of two adjacent squares—produced recognizable but washed-out mosaics. Several refinements addressed this.

3.1 Edge Contrast

Rather than averaging blindly, we check the brightness difference between the two squares sharing a slider. If the difference exceeds a threshold, the slider snaps to a position based on the *minimum* brightness of the pair. This sharpens edges considerably: dark regions stand out crisply against light ones. We also tried using the maximum brightness instead, but the minimum consistently produced cleaner results.

3.2 Hard Thresholding

Taking the contrast idea further, we partition the full 0–255 range into discrete bands. If the average brightness of two adjacent squares falls within a given band, the slider snaps to the bottom of that band. This quantization splits the mosaic into high-contrast regions that read clearly even at low resolution. Figure 2 shows the Mona Lisa rendered with this method.

Figure 2 : *Mona Lisa rendered using the hard-contrast thresholding method.*

3.3 Fixed Sides

The most effective improvement came from a bug fix. We realized that when sliders move on all four sides of a square, a polygon set to its brightest configuration does not actually expand—it inverts. To correct this, we fixed the vertical sliders in place (Figure 3), leaving only the horizontal sliders free. This gave each tile a true brightness range from fully black to fully white, producing the sharpest and most faithful mosaics.

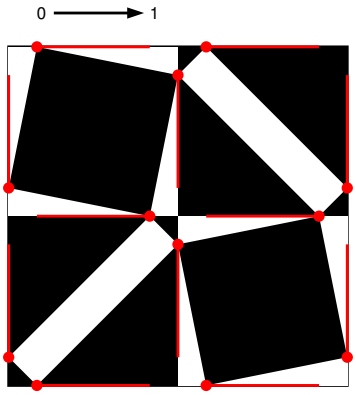

Figure 3: *Fixed-sides configuration. Vertical sliders are locked at the diagonal; only horizontal sliders move.*

4 Future Work

We would like to explore adding color—assigning RGB values to tiles rather than restricting them to black and white—to see whether chromatic variation can bring new life to the mosaics.

We are also interested in expanding the tile vocabulary beyond a single polygon type, and in developing an optimization algorithm that adjusts slider positions globally based on fidelity to the source photograph, rather than relying solely on local brightness averages.

5 Acknowledgements

This project builds on Robert Bosch's research on edge-constrained tile mosaics. We thank him for suggesting the problem and for his guidance throughout.

References

[1] R. A. Bosch, "Edge-Constrained Tile Mosaics," in *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, R. Sarhangi and J. Barrallo, eds., Tarquin Publications, 2007, pp. 351–360.