Map-Colored Mosaics

Robert Bosch
Oberlin College
Oberlin, OH 44074
(bobb@cs.oberlin.edu)

Andrew Pike
Michigan State University
E. Lansing, MI 48824
(pikeand1@msu.edu)

Abstract

We describe how to construct map-colored mosaics. When viewed from afar, they resemble familiar images. When viewed from up close, they look like properly colored maps.

1 Introduction

After a long day of coloring maps, a cartographer might view a grid of squares (Figure 1(a)) as an uncolored map. To color this map, the cartographer would follow the traditional map-coloring protocol and assign colors to countries in such a way that no two countries (squares) that share a border (edge) receive the same color. The Four Color Theorem states that, at least in the plane, four colors suffice; never will the cartographer need more. When coloring a grid of squares, one needs only two colors, as in Figure 1(b). But with four (or more) colors, much more interesting and aesthetically pleasing results can be achieved. For example, with four colors a grid of squares might take on the likeness of the Mona Lisa's right eye, as in Figure 1(c).

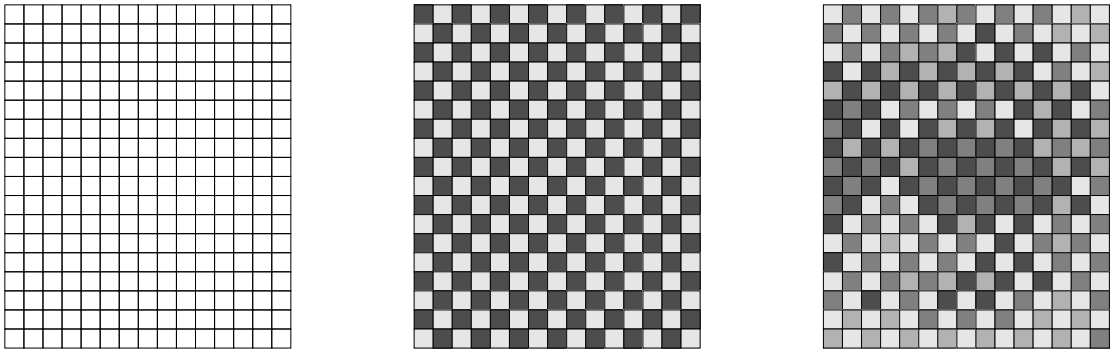

Figure 1: (a) an uncolored map, (b) colored with two shades of gray, (c) colored with four shades of gray

In this paper, we describe how to produce pictures like Figure 1(c). We call them *map-colored mosaics*. When viewed from afar, they resemble familiar images. When viewed from up close, they look like properly colored maps. In a properly colored map, no two countries that share a border can share a color. In a map-colored mosaic, no two tiles that share an edge can share a color.

2 The Map-Colored Mosaic Design Problem

We begin with a user-supplied target image, a map (a tiling of the plane with copies of one or more tiles), and a small set of colors (usually between four and eight). We first assume that our target image is in grayscale, our map is a grid of squares, and all of our colors are different shades of gray.

Furthermore, we assume that our target image consists of m rows and n columns of pixels and that our map is a grid that consists of m rows and n columns of squares. We denote the brightness of the row i, column j pixel—pixel (i, j)—by $\beta_{i,j} \in [0,1]$, where 0 stands for completely black, 1 stands for completely white, and intermediate values stand for intermediate shades of gray.

Finally, we assume that we only have χ colors. (We refer to them as colors even though they are just shades of gray.) We denote the brightness of color $c \in \{1,\ldots,\chi\}$ by b_c, using the same 0-to-1, black-to-white scale we use for the pixel brightnesses.

Our goal is to construct an $m \times n$ mosaic that (1) when viewed from afar, resembles the target image as much as possible, and (2) when viewed from up close, looks like a properly-colored map. To attain this goal, we use integer programming formulations as in [2-4].

3 A Simple Integer Programming Formulation

For each square (i, j) and each color c, we must decide if we will paint square (i, j) with color c. We model this with binary variables, setting $x_{i,j,c}$ equal to 1 if we do indeed paint square (i, j) with color c, and setting $x_{i,j,c}$ to 0 if we don't. We need a total of χmn such variables.

Given that we want our mosaic to resemble our target image as much as possible, we need to be able to measure our mosaic's "goodness of fit" with our target image. Our strategy is to compute a separate error term for each square of the mosaic and then add them all up. If we paint square (i, j) with color c, then it will have brightness b_c, whereas pixel (i, j) has brightness $\beta_{i,j}$. So one possible error term for square (i, j) is

$$\sum_{c=1}^{\chi} \left|b_c - \beta_{i,j}\right| x_{i,j,c},$$

and another is

$$\sum_{c=1}^{\chi} (b_c - \beta_{i,j})^2 x_{i,j,c}.$$

We use the first one, which gives us the goodness-of-fit measure

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{c=1}^{\chi} \left|b_c - \beta_{i,j}\right| x_{i,j,c}$$

for our model's objective function.

To force ourselves to paint square (i, j) with precisely one color c, we incorporate the following equation into our model:

$$\sum_{c=1}^{\chi} x_{i,j,c} = 1.$$

We need a total of mn such equations.

To force ourselves to paint squares (i, j) and $(i, j+1)$ with different colors, we introduce, for each color $c \in \{1,\ldots,\chi\}$, the inequality

$$x_{i,j,c} + x_{i,j+1,c} \leq 1.$$

This inequality prohibits us from painting both square (i, j) and its right neighbor, square $(i, j+1)$, with color c. We need a total of $\chi m(n-1)$ such inequalities.

Similarly, to force ourselves to paint squares (i, j) and $(i+1, j)$ with different colors, we introduce, for each color $c \in \{1,\ldots,\chi\}$, the inequality

$$x_{i,j,c} + x_{i+1,j,c} \leq 1.$$

This inequality prevents us from painting both square (i, j) and its lower neighbor, square $(i+1, j)$, with color c. We need a total of $\chi(m-1)n$ such inequalities.

We end up with the following integer programming formulation:

$$\begin{aligned}
\text{minimize} \quad & z = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{c=1}^{\chi} \left|b_c - \beta_{i,j}\right| x_{i,j,c} \\
\text{subject to} \quad & \sum_{c=1}^{\chi} x_{i,j,c} = 1 && \text{for each } 1 \le i \le m,\ 1 \le j \le n \\
& x_{i,j,c} + x_{i,j+1,c} \le 1 && \text{for each } 1 \le i \le m,\ 1 \le j \le n-1,\ 1 \le c \le \chi \\
& x_{i,j,c} + x_{i+1,j,c} \le 1 && \text{for each } 1 \le i \le m-1,\ 1 \le j \le n,\ 1 \le c \le \chi \\
& x_{i,j,c} \in \{0,1\} && \text{for each } 1 \le i \le m,\ 1 \le j \le n,\ 1 \le c \le \chi.
\end{aligned}$$

This formulation is simple and fast but yields low quality mosaics for small values of χ. Each of the mosaics displayed in Figure 2 was produced in a fraction of a second by CPLEX [8] on a 3.2 GHz Pentium IV PC.

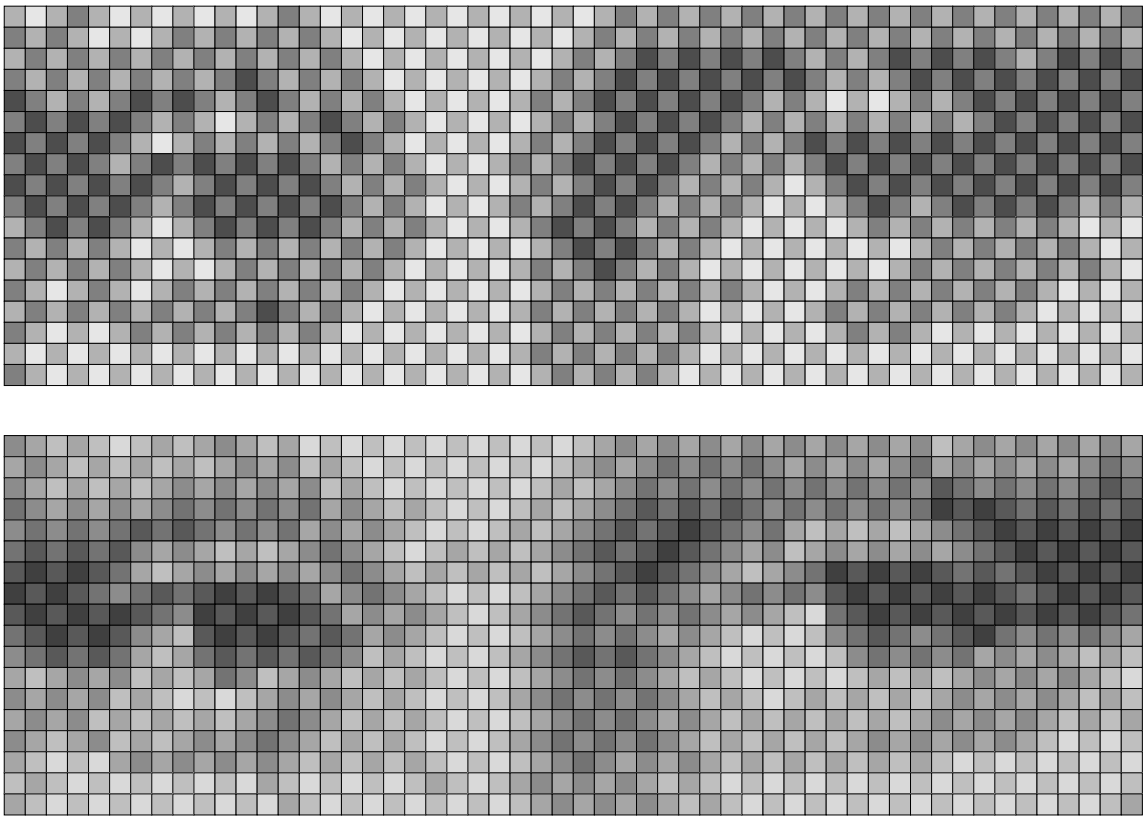

Figure 2: The Mona Lisa's Eyes, simple formulation ($m = 18$, $n = 54$) (a) $\chi = 4$, (b) $\chi = 8$

The four-color mosaic (Figure 2(a)) is recognizable, but crude. The eight-color mosaic (Figure 2(b)) achieves a better likeness. We encourage the reader to view both of these mosaics (and all of the others displayed in this paper) from two vantage points: from up close and from afar (e.g., from across the room).

4 An Alternate (Block) Formulation

The problem with the simple integer programming formulation is that it measures the mosaic's goodness of fit by comparing each individual square with its corresponding pixel. Recent research on matrix rounding and digital halftoning [1,5] suggests another approach: comparing each two-by-two block of squares with its corresponding two-by-two block of pixels.

Consider the $m \times n$ matrix M whose row-i, column-j entry gives the number of the color in the row-i, column-j position of our mosaic. Let $\mathscr{Q}$ equal the set of all 2×2 matrices that *could be* a 2×2 block of M. Note that $Q = \binom{q_{11}\ q_{12}}{q_{21}\ q_{22}} \in \mathscr{Q}$ if and only if each entry of Q belongs to $\{1, \ldots, \chi\}$ and no two adjacent entries are equal (in other words, $q_{11} \neq q_{12}$, $q_{11} \neq q_{21}$, $q_{12} \neq q_{22}$, and $q_{21} \neq q_{22}$).

In this formulation, for each $1 \leq i < m$, each $1 \leq j < n$, and each $Q = \binom{q_{11}\ q_{12}}{q_{21}\ q_{22}} \in \mathscr{Q}$, we set $y_{i,j,Q}$ equal to 1 if we place the upper-left corner of block Q in square (i, j), painting square (i, j) with color q_{11}, square $(i, j+1)$ with color q_{12}, square $(i+1, j)$ with color q_{21}, and square $(i+1, j+1)$ with color q_{22}. So, in this formulation, setting a variable equal to 1 paints an entire two-by-two block of squares. It turns out that we need a total of $[\chi(\chi-1)^2 + \chi(\chi-1)(\chi-2)^2](m-1)(n-1)$ such variables.

Here is our alternate (block) formulation:

$$\min \quad z = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{Q \in \mathscr{Q}} \left| (b_{q_{11}} + b_{q_{12}} + b_{q_{21}} + b_{q_{22}}) - (\beta_{i,j} + \beta_{i,j+1} + \beta_{i+1,j} + \beta_{i,j+1}) \right| y_{i,j,Q} \tag{1}$$

$$\text{s.t.} \quad \sum_{Q \in \mathscr{Q}} y_{i,j,Q} = 1 \qquad \text{for each } 1 \leq i \leq m-1,\ 1 \leq j \leq n-1 \tag{2}$$

$$\sum_{\substack{Q \in \mathscr{Q}: \\ q_{12}=u,\, q_{22}=v}} y_{i,j,Q} = \sum_{\substack{R \in \mathscr{Q}: \\ r_{11}=u,\, r_{21}=v}} y_{i,j+1,R} \qquad \begin{array}{l}\text{for each } 1 \leq i \leq m-1,\ 1 \leq j \leq n-2, \\ \quad \text{and } 1 \leq u, v \leq \chi : u \neq v\end{array} \tag{3}$$

$$\sum_{\substack{Q \in \mathscr{Q}: \\ q_{21}=u,\, q_{22}=v}} y_{i,j,Q} = \sum_{\substack{R \in \mathscr{Q}: \\ r_{11}=u,\, r_{12}=v}} y_{i+1,j,R} \qquad \begin{array}{l}\text{for each } 1 \leq i \leq m-2,\ 1 \leq j \leq n-1, \\ \quad \text{and } 1 \leq u, v \leq \chi : u \neq v\end{array} \tag{4}$$

$$y_{i,j,Q} \in \{0,1\} \qquad \text{for each } 1 \leq i \leq m-1,\ 1 \leq j \leq n-1,\ Q \in \mathscr{Q}.$$

The objective function, (1), measures goodness of fit by comparing the total brightness of each two-by-two block of squares ($b_{q_{11}} + b_{q_{12}} + b_{q_{21}} + b_{q_{22}}$) with the total brightness of the corresponding two-by-two block of pixels ($\beta_{i,j} + \beta_{i,j+1} + \beta_{i+1,j} + \beta_{i,j+1}$). The type (2) equations make sure that we paint each and every two-by-two block of squares, and the type (3) and (4) equations make sure that we paint the two-by-two blocks in such a way that they overlap with each other both horizontally and vertically.

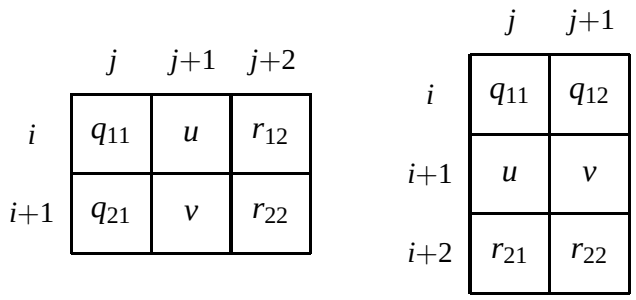

Figure 3: (a) horizontal overlapping, (b) vertical overlapping.

When thinking about the type (3) equations it helps to consider Figure 3(a), and when thinking about the type (4) equations, it helps to consider Figure 3(b). The left-hand side of a type (3) equation counts how many blocks Q of the form $\binom{q_{11}\ u}{q_{21}\ v}$ we place in square (i, j). The right-hand side counts how many blocks R of the form $\binom{u\ r_{12}}{v\ r_{22}}$ we place in square $(i, j+1)$. Note that each side is either 0 or 1 (due to the type (2)

equations and the fact that the variables are binary.) By setting the left-hand and right-hand sides equal, we force the blocks to overlap horizontally.

The block formulation does have its drawbacks. It is certainly harder to understand. It has many more variables and constraints than the simple formulation (on the order of χ^3 times the number of variables, and on the order of χ times the number of constraints). And it is slow. It is so large that to solve it, we had to divide the problem into sections, creating a dozen columns of the mosaic at a time. Still, the advantage of the block formulation is that it produces high quality mosaics even for small values of χ. The four-color mosaic displayed in Figure 4 (which required about a minute for CPLEX) is much nicer than the one in Figure 2(a). In fact, we strongly prefer it to the eight-color mosaic displayed in Figure 2(b). It uses half the number of colors, yet when viewed from a distance, it still achieves a better likeness of the target image.

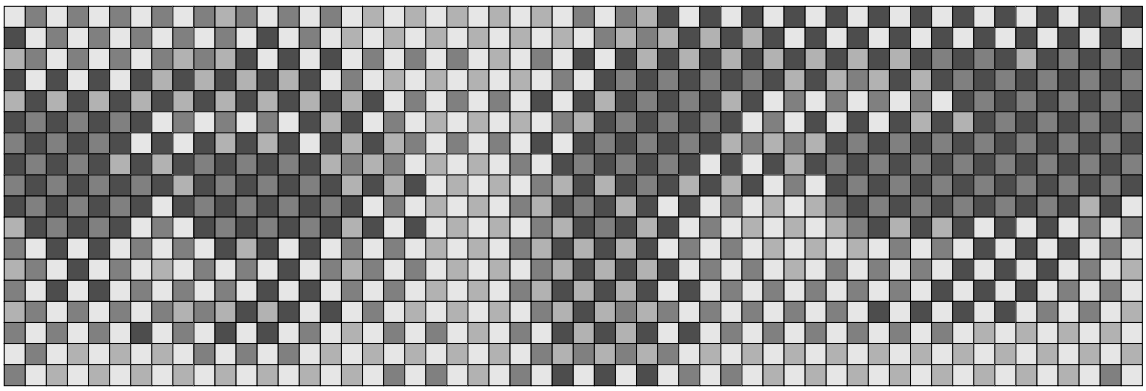

Figure 4: The Mona Lisa's Eyes, block formulation ($m = 18$, $n = 54$, $\chi = 4$)

5 Modifying the Mosaic

Once we have created a map-colored mosaic that pleases us, we can modify it by replacing its square tiles with other tiles that behave like squares. By doing this, we obtain images that are reminiscent of Escher-like tessellations [6,10] or Huff-like parquet deformations [7,9] when viewed from up close, yet still look like familiar images when viewed from a distance. See Figures 5 and 6.

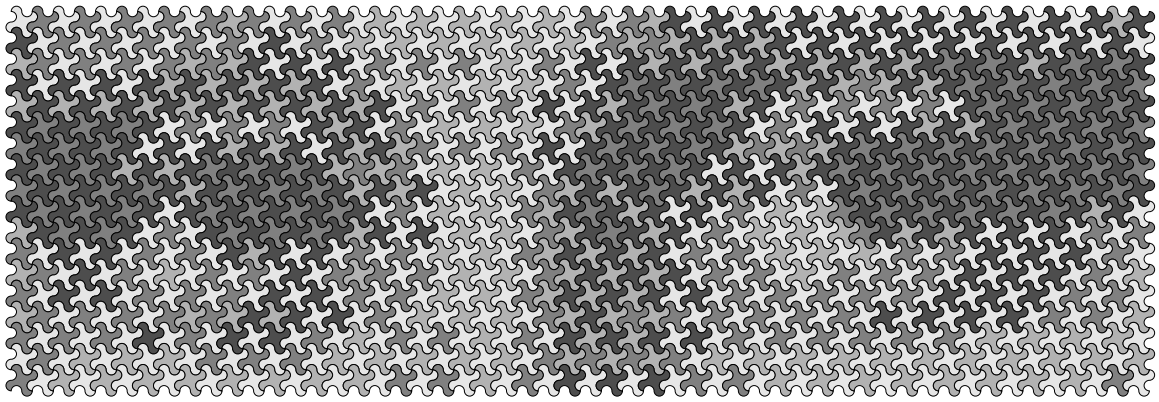

Figure 5: The Mona Lisa's Eyes, block formulation, propeller tiles ($m = 18$, $n = 54$, $\chi = 4$)

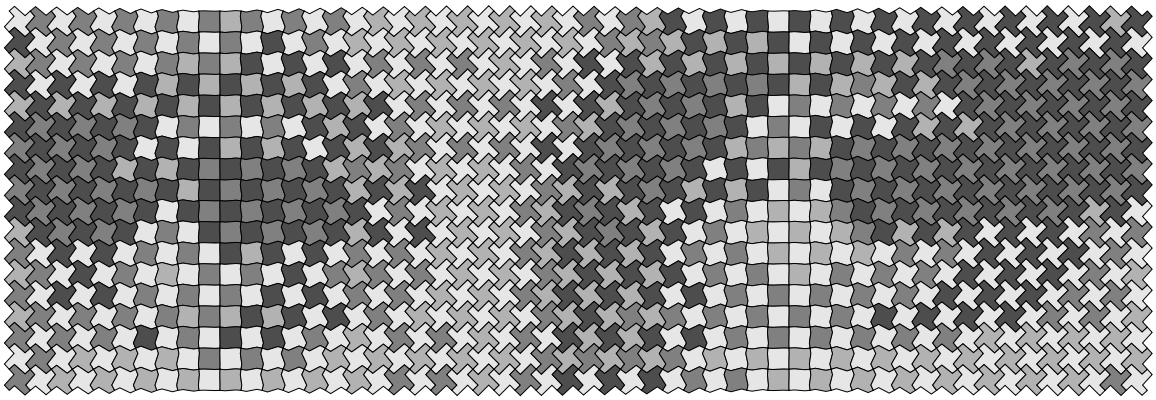

Figure 6: The Mona Lisa's Eyes, block formulation, parquet deformation ($m = 18$, $n = 54$, $\chi = 4$)

6 Other Tilings

One can adapt both the simple and block formulations for non-square tilings based on hexagons, equilateral triangles, rhombuses, etc., but it is usually much easier to adapt the simple formulation. Figure 7 displays a six-color map-colored mosaic, based on a hexagonal tiling, that we created by solving a hexagonal version of the block formulation. Figures 8-10 display what happens when we replace the hexagons with pinwheels, a Mongolian three-pronged arrow design, and Escher-like lizards.

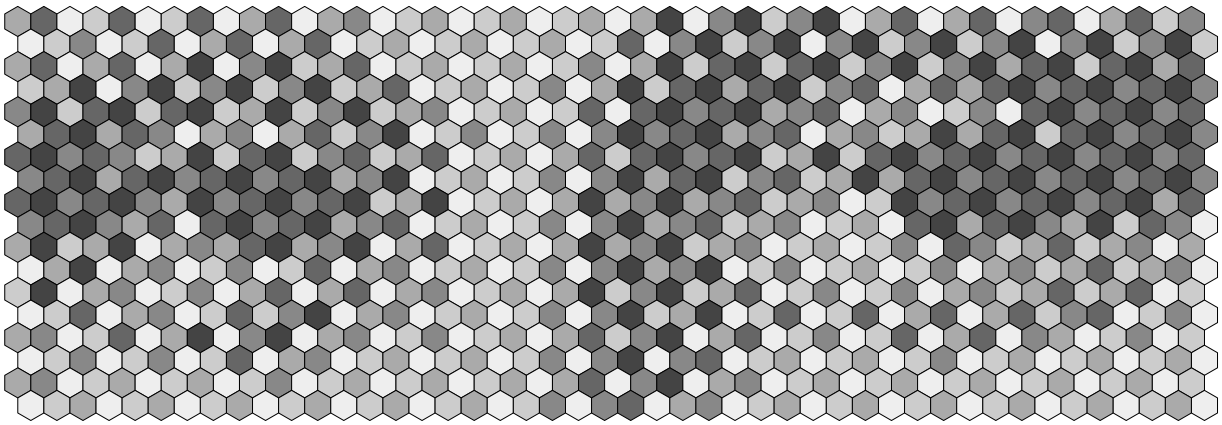

Figure 7: The Mona Lisa's Eyes, block formulation, hexagons ($\chi = 6$)

7 Color

It is also possible to adapt the formulations so that they work with different color palettes. The main difficulty is that the user needs to specify the color palette in advance. A palette that works well for one target image may fail spectacularly for another. See Figure 11 for a successful example.

References

[1] Tetsuo Asano, Naoki Katoh, Koji Obokata, and Takeshi Tokuyama. Matrix rounding under the L_p-discrepancy measure and its application to digital halftoning. *SIAM Journal on Computing*, 32: 1423-1435, 2003.

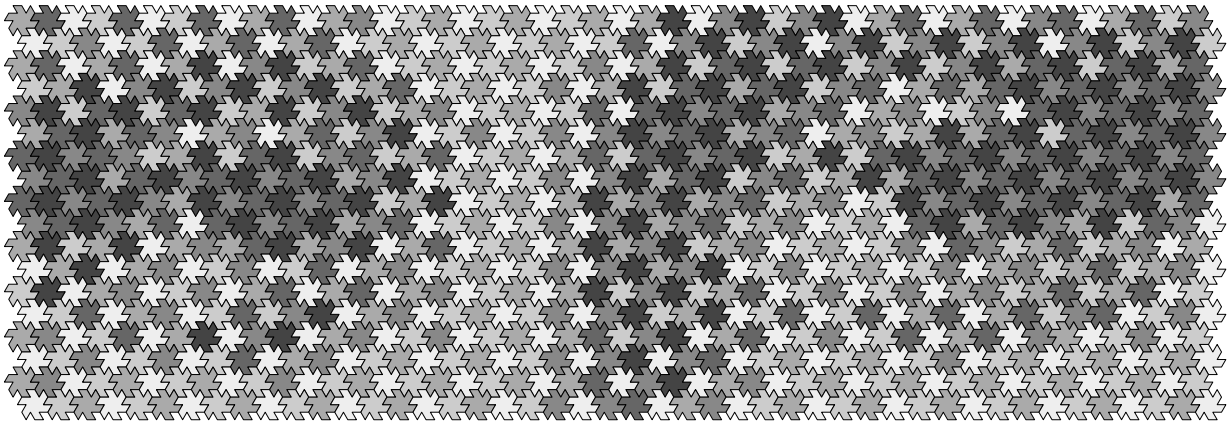

Figure 8: The Mona Lisa's Eyes, block formulation, pinwheels ($\chi = 6$)

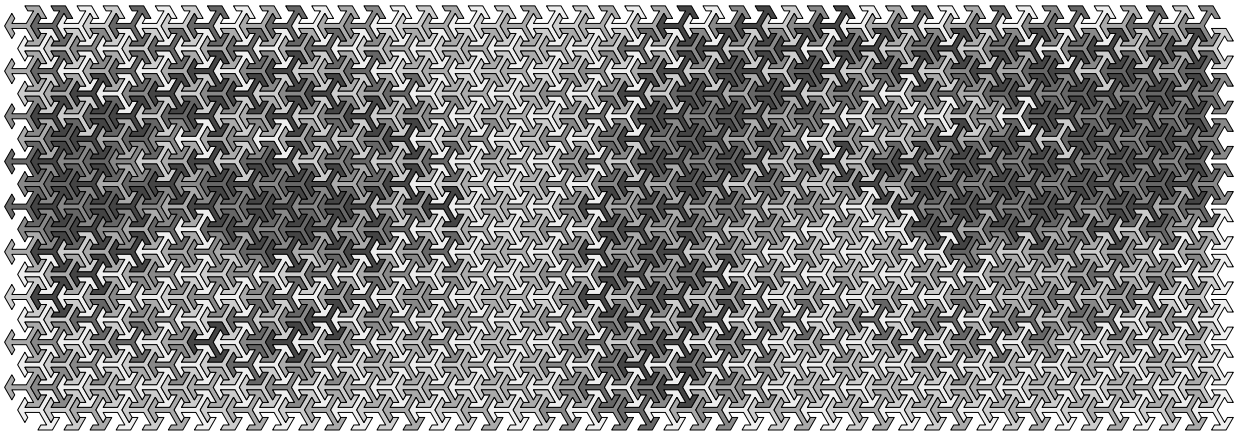

Figure 9: The Mona Lisa's Eyes, block formulation, arrows ($\chi = 6$)

[2] Robert Bosch. Constructing domino portraits. In B. Cipra et al., editor, *Tribute to a Mathemagician*, pages 251-256, A.K. Peters, 2004.

[3] Robert Bosch. Opt Art. *Math Horizons*, pages 6-9, feb 2006.

[4] Robert Bosch. Edge-constrained tile mosaics. In *Bridges Donostia: mathematical connections in art, music, and science*, pages 351-360, 2007.

[5] Benjamin Doerr. Nonindependent randomized rounding and an application to digital halftoning. *SIAM Journal on Computing* 34: 299-317, 2004.

[6] M.C. Escher. *Escher on Escher: Exploring the Infinite.* Henry N. Abrams, 1989.

[7] Douglas R. Hofstadter. *Metamagical Themas: Questing for the Essence of Mind and Pattern.* Basic Books, 1985.

[8] ILOG CPLEX. High-performance software for mathematical programming and optimization. `www.ilog.com/products/cplex/`.

[9] Craig S. Kaplan. Metamorphosis in Escher's art. In *Bridges Leeuwarden: mathematical connections in art, music, and science*, pages 39-46, 2008.

[10] Doris Schattschneider. *M.C. Escher: Visions of Symmetry.* Henry N. Abrams, second edition, 2004.

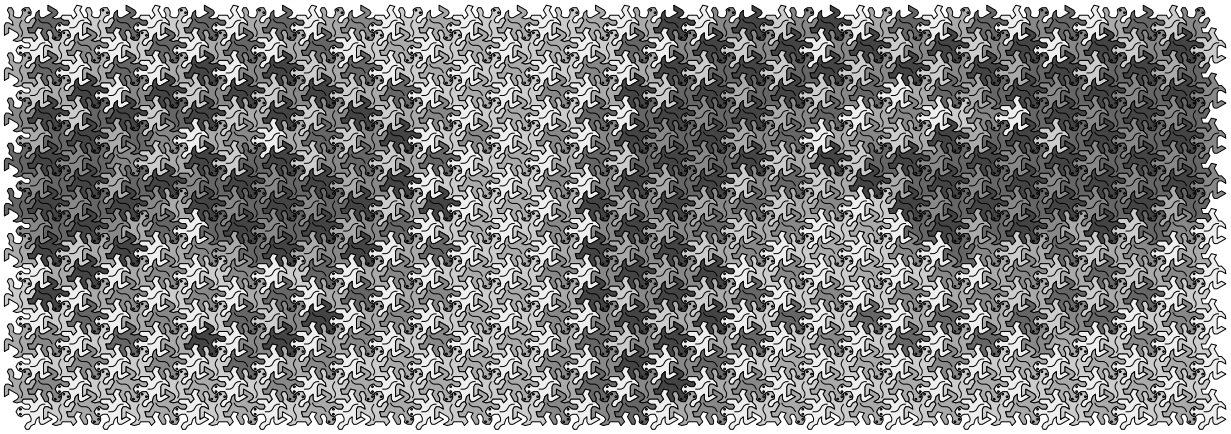

Figure 10: The Mona Lisa's Eyes, block formulation, lizards ($\chi = 6$)

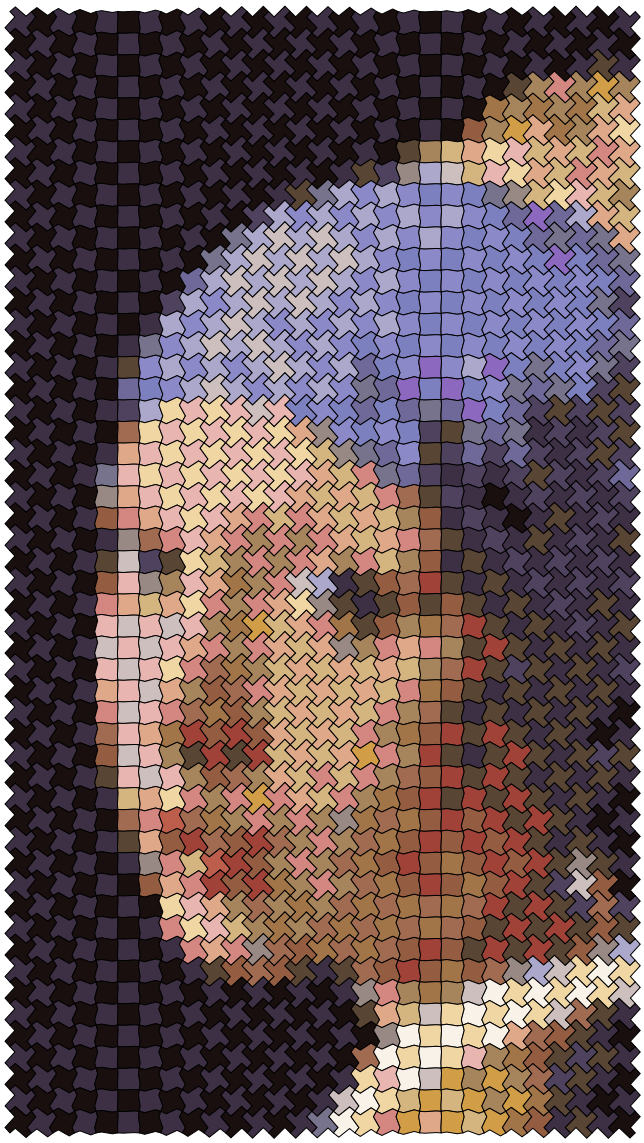

Figure 11: Girl With A Pearl Earring, simple formulation, parquet deformation ($\chi = 25$)