Checkerboard Quadrilateral Mosaics

Aaron Kreiner
Dept. of Mathematics
Oberlin College
173 W Lorain St, Oberlin, OH
akreiner@oberlin.edu

Abstract

Checkerboard quadrilateral mosaics were introduced by Robert Bosch. They are formed from a black and white checkerboard. A quadrilateral is positioned in each square; white quadrilaterals in black squares, and black quadrilaterals in white squares. Each quadrilateral has a vertex on each of the edges of the square. Quadrilaterals in neighboring squares share vertices. In this highly constrained environment, it is possible to create figurative mosaics that resemble user supplied images. Designing these mosaics requires sophisticated algorithmic design because the objective function for the tiling system is non-convex, non-linear, and contains a large number of variables. Downsampling is used to resize a target picture to a desired mosaic size. The downsampled image is split into 5×5 grids. To calculate the coordinates of each of the vertices in the 5×5 grids, sequential least squares programming is implemented with the appropriate constraints. Finally, each of the 5×5 grids are stitched together to create the final mosaic.

Introduction

The checkerboard pattern was invented sometime between 3000-1500 BC [1]. Since then, artists and gamemakers alike have used these quintessential patterns in their works. One of the first examples of this was Truchet tiles. A Truchet tile is a square divided along the diagonal and colored half black and half white. Bosch and Colley introduced the "flexible" Truchet tiling which allows the diagonal separating the black and white regions to bend. In doing so, the tiles could now capture varying levels of brightness [2]. The checkerboard quadrilateral system is a more constrained version of the "flexible" Truchet Tiles.

The idea of a checkerboard quadrilateral mosaic was created by Robert Bosch in his book *Opt Art: From Mathematical Optimization to Visual Design*. His book outlines a linear optimization framework to create these mosaics. The framework presented depends on limiting the sliders to discrete positions on the checkerboard tile and has a relatively large run-time for bigger images [3, pp. 25-26]. This paper outlines a non-linear optimization framework which allows infinite slider positions on the tile in addition to run-times of less than a minute for larger mosaics. Further, the methods presented here produce mosaics with significantly lower errors with respect to the target image. Given this paper's contribution to the opt art community, it is helpful to define such a mosaic. A *checkerboard quadrilateral tile* is a unit square tile that has a quadrilateral inscribed in it [3, pp. 145–165]. Each quadrilateral has 4 sliders: top, $bottom$, $left$, and $right$. From now on, they will be referenced as t, b, l, and r respectively. For example, the b slider refers to the slider on the southern edge of the square. These sliders range from 0.1 to 0.9. This therefore imposes a box constraint on the sliders. Each tile has its own coordinate system. The bottom left corner of the square is labeled $(0, 0)$. Similarly, the top right corner of the square is labeled $(1, 1)$. There are two types of tiles: black and white. Black tiles have a black inscribed quadrilateral with the remainder of the square being white. White tiles have a white inscribed quadrilateral with the remainder of the square being black. Figure 1 shows a visual representation of the tiles.

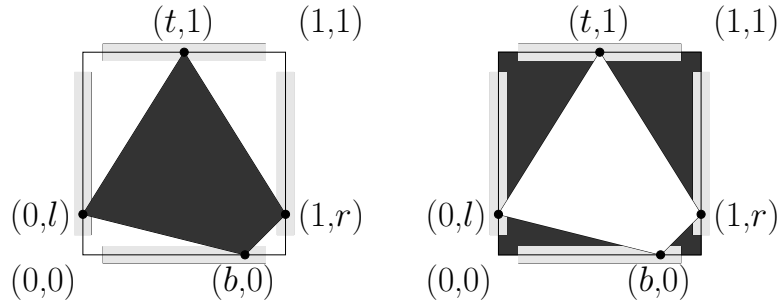

Figure 1: A black tile (left) and white tile (right) with labeled coordinate system. Each of the italicized letters refer to the value of their respective sliders. The light grey rectangles refer to the feasible locations of each of the sliders. It is important to note that the sliders can only move along the edges of the square.

A brightness value for a tile ranges from 0 to 1 (0 being completely black, 1 being completely white). Intermediate brightness levels are created by adjusting the sliders of the tiles. Within the target image, intermediate brightness levels are created by blending black and white channels in fixed proportions. For example, a brightness value of 0.8 blends 80 percent white and 20 percent black . Checkerboard tiles do not blend black and white, instead they contain black and white regions within each tile. When pieced together, the black and white tiles alternate like a checkerboard. Precisely, if the row number plus the column number is even, then a black tile is placed. Otherwise, a white tile is placed. This therefore creates a checkerboard pattern of black and white tiles. However, there are constraints imposed on the sliders that take two forms. First are top-bottom constraints. The b slider must touch the t slider of the tile below. Second are left-right constraints. The r slider must touch the l slider of the tile to the right. Sliders that are on the edge of the mosaic are free to range from 0.1 to 0.9. Figure 2 depicts two instances of the checkerboard quadrilateral tiling.

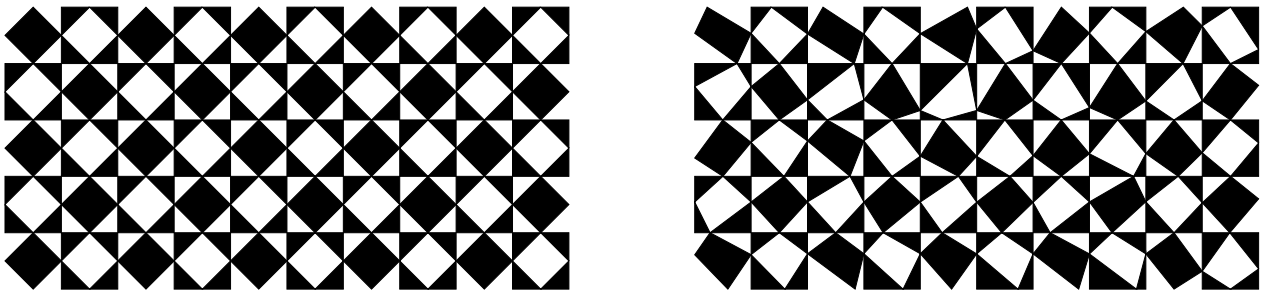

Figure 2: The left figure denotes a 10×5 checkerboard quadrilateral mosaic with all the tiles having brightness 0.5. In addition, each slider t, b, l, and r are all set to 0.5. The right figure denotes another 10×5 instance with all 0.5 brightness, but the sliders are not constrained to be 0.5.

Figure 2 shows tiles with moderate brightness. It is important to note the tiles can model very high or very low brightness values as well. These extreme cases of brightness show up in most of the target images. Figure 3 demonstrates very low and very high brightness tiles.

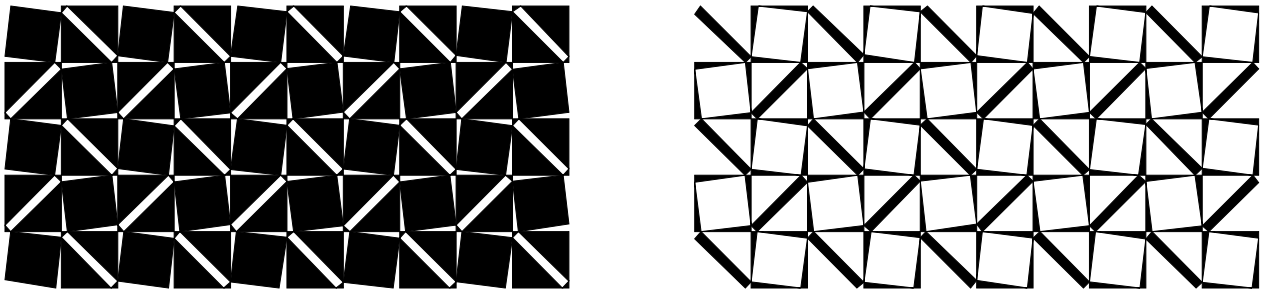

Figure 3: The left figure denotes very dark (or low brightness tiles). The brightness is set to 0.2. The right figure shows very bright tiles with a brightness 0.8.

However, the checkerboard quadrilateral mosaic system can capture tiles of varying brightness. Figure 4 depicts a gradient with low, moderate, and high levels of brightness.

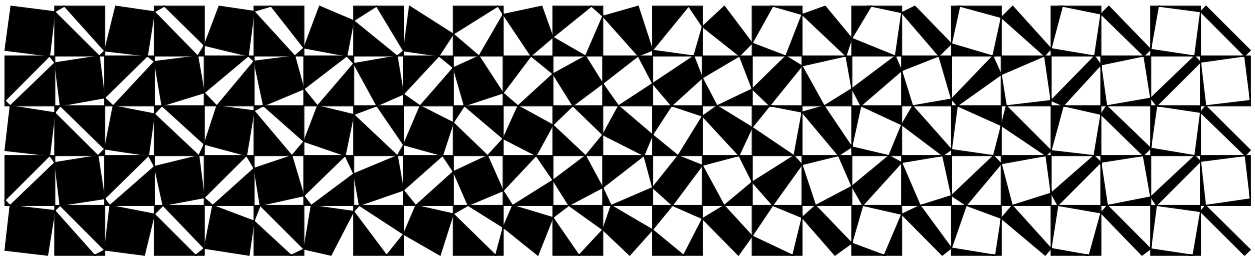

Figure 4: A 25×5 gradient. The leftmost tiles have brightness 0.2 and the rightmost tiles have brightness 0.8.

The checkerboard quadrilateral system relies on a concept known as halftoning to generate mosaics with tiles of appropriate brightness. Halftoning is a process that simulates shades of grey by varying the size of small dots arranged in a specific pattern. When viewed at a distance, these dots will appear the desired shade of gray even though the dots themselves are completely black. However, the process of halftoning is not completely reserved to dots. Other shapes can be used in this process, as long as the appropriate proportion of white in the shapes corresponds to the correct level of gray [2]. Now, let's express this mathematically. Suppose a pixel has brightness B, where brightness is defined earlier in the introduction. Let A be the area of the white region surrounding the black shape. In order to create a halftone representation, the proportion of white area in the mosaic region must match the pixel's brightness. In most cases, it is impossible to exactly match the brightness of a pixel by a halftone representation. This concept introduces the idea of the design problem. Suppose one wants to create a halftone representation of an image. Then the designer faces the following optimization problem:

$$min \sum_{i,j}(A_{i,j} - B_{i,j})^2$$

The indexes i, j represent the row and column of a specific region in the target picture to be created with a halftone representation. The objective function minimizes the sum of squared difference of the white area in region i, j and appropriate level of brightness in region i, j. The design problem can be used to create checkerboard quadrilateral mosaics, which will be described in the next section.

Method

First, a target image is selected and brightness values are collected. The image is re-sized to match the desired checkerboard mosaic size. Black tiles have inverted white regions with respect to white tiles. Therefore, where each black tile is placed, the brightness must be inverted. To understand this further, consider the following example. Suppose a picture has constant brightness D. Black tiles and white tiles will try to capture a white area of D.

Solving the optimization problem for the whole image is not feasible. This is because the quadrilateral area formula is non-linear and non-convex with respect to the slider variables. The target picture, therefore, is grouped into 5×5 grids of pixels. At any given point, the largest problem the optimization solver will tackle is the 5×5. To represent the sliders within the 5×5, each square tile was labeled i, j where $i, j \in \mathbb{Z}$ and $0 \leq i \leq 4, 0 \leq j \leq 4$. The i index is the row and the j index is the column. The top left square is labeled $0, 0$ and the bottom right square is labelled $4, 4$. To represent the sliders themselves, we create 60 variables that range from 0.1 to 0.9. Figure 5 shows each of the 60 sliders and their location within the 5×5.

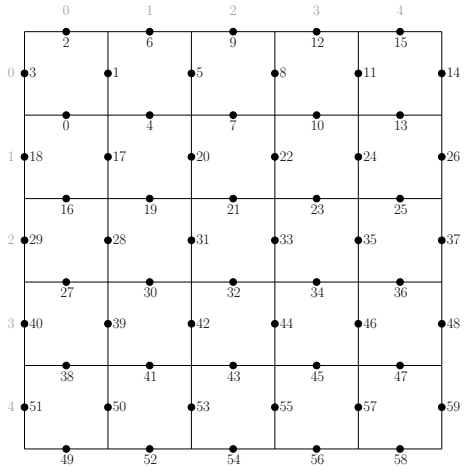

Figure 5: This figure shows the location for each of the numbered sliders. The grey numbers indicate the square tile's row and column index. For example, the tile containing sliders 11 and 13 is contained in square $0, 4$.

Given the representation, the objective function naturally follows. We loop over each square tile in the 5×5 starting with the top left square. The objective function is the squared sum of the brightness values subtracted from the areas of the quadrilateral tiles. This is the error of the area of the white region compared to the actual brightness in the target picture. Using the coordinate system defined in Figure 1, the shoelace method can find the area of the quadrilateral inscribed in the tile. The shoelace method is a well-known formula that inputs coordinates of a quadrilateral and outputs the area [4]. Using this method, it can be shown that $Area_{i,j}$, the area of the quadrilateral inscribed in the square located at row i column j, is as follows. The slider names are abbreviated for easy reading:

$$Area_{i,j}=1/2 * (1 + b_{i,j} * r_{i,j} + t_{i,j} * l_{i,j} - t_{i,j} * r_{i,j} - b_{i,j} * l_{i,j})$$

From this, the objective function is the minimum squared difference of $Brightness_{i,j}$, the correct brightness of the tile, from $Area_{i,j}$ summed across all i, j. This is shown below:

$$min \sum_{i=0}^{4} \sum_{j=0}^{4} (Brightness_{i,j} - Area_{i,j})^2$$

However, there are some constraints that need to be added to the optimization problem. By construction of the variables for the sliders, the right-left, top-bottom constraints are satisfied because neighboring quadrilaterals vertices share 1 slider. The second constraint imposed is an orientation constraint. This constraint is defined as follows. On even rows of the mosaic, $b \leq 0.5$ and on odd rows $b \geq 0.5$, where b is the bottom slider. This defines one orientation. The constraint imposed is that the picture must have tiles with constant orientation. The reason for this is described later on. While the above was the orientation used for the project, one could define a second orientation by letting $b \leq 0.5$ on odd rows and $b \geq 0.5$ on even rows. These two orientations are flipped about the vertical. Figure 6 and Figure 7 visualize the constraint and how it appears in the mosaics.

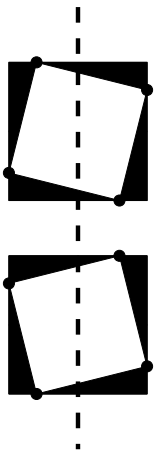

Figure 6: This figure shows two white 0.7 brightness tiles which are mirrors of each other in vertical bisector. This is shown by the dashed line. The tile on the top of the bisector would be feasible on odd rows and the tile on the bottom of the bisector be feasible on even rows based on the location of the bottom slider.

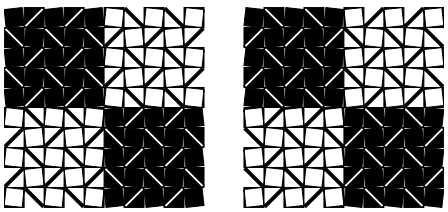

Figure 7: Two 10×10 instances of the tiling problem that have a target image of a checkerboard. Both of these patterns would be infeasible under the orientation constraint. This is due to the fact that in each picture, the very dark tiles carry a different orientation to the very light tiles. On any given row, the bottom slider's relative coordinate is both above and below 0.5. The checkerboard on the right is identical to the one on the left, except each tile's orientation is flipped.

Given the definition of the orientation constraint, this paper now describes the necessity of the constraint in the optimization program. Within very dark or very light regions, multiple orientations create ugly

artifacts. To demonstrate this concept, it is helpful to consider Figure 8.

Figure 8: Both mosaics depict a mosaic with constant brightness of 0.2. The left figure has no orientation constraint and the right figure has an orientation constraint.

As shown by the mosaics, the orientation constraint attempts to eliminate etching artifacts that are a result of conflicting orientations. These etching patterns only occur in very dark or very light tiles because there are very few ways to generate extreme brightness tiles.

Next, this paper describes the optimization process in more detail. Because the optimization problem is non-convex and non-linear, sequential least squares programming algorithm is used. This is a type of gradient descent. The solver requires an objective function, constraints, bounds for the variables, and a seed. A seed is the initial point the solver considers, which iterates to an optimal value. The seed is randomly generated number from 0.1 to 0.9, one for each slider in the 5×5. The solver can then find the coordinates of the 5×5 with the minimal objective value.

In order to piece all the 5×5's together, we start with the top left 5×5 and move from left to right. Depending on the location of the 5×5, the solver passes in values for the vertices that were fixed by neighboring 5×5's. These are to satisfy the top-bottom and left-right constraints. There are four cases that we need to discuss. The first is when the 5×5 is located in the top-left. This 5×5 is unbounded with respect to neighbor constraints, and the solver can solve it as described in the previous section. The second case occurs after the first 5×5 on the first row. The solver can then take the r value of the 5×5's left neighbor, and set the left values of the current 5×5 equal to it. The third case is if the 5×5 is in the first column, then the solver can set the 5 uppermost top values of the current 5×5, and set them equal to the 5 $bottom$ values of the 5×5 above it. Finally, if the current 5×5 is not located in the first row or column then the solver can pass in the $right$ and $bottom$ values of its neighbors to the objective function and create the constraints mentioned above. Passing in values to the objective function is more efficient than creating constraints by design of sequential least squares programming. This process aids the solver in finding the best feasible solution.

Conclusion

The non-convex nature of the objective function for checkerboard quadrilaterals means a high level of computational complexity. The heuristic of 5×5 sub-optimality was introduced to simplify the problem by decreasing the computational complexity of the optimization program. Checkerboard mosaics were then created by piecing together each 5×5 grid in the correct position. Further research can be done to investigate other non-linear and non-convex tiling systems. In the following pages are samples of my work.

Figure 9: 70×90 mosaic representations of da Vinci's 'Mona Lisa' [5] and of Vincent van Gogh [6].

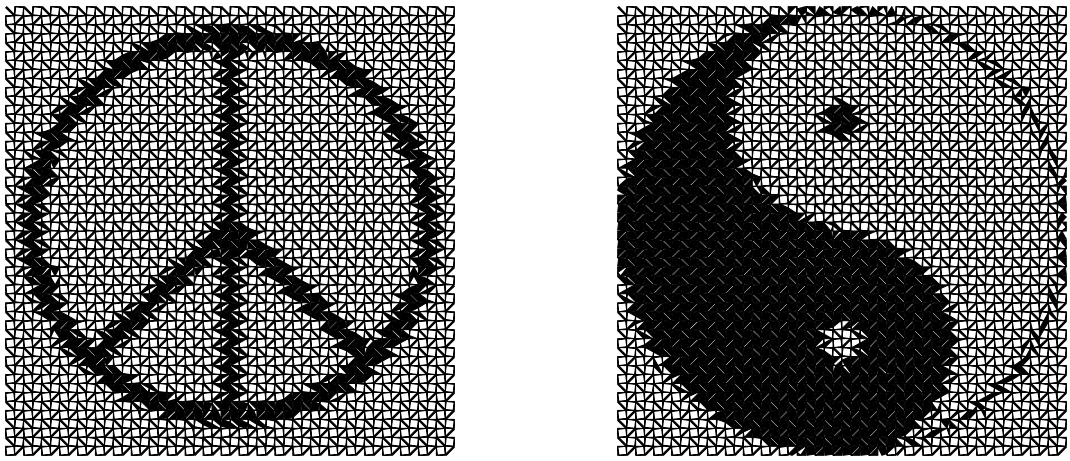

Figure 10: 50×50 mosaic representations of the peace symbol [7] and the yin and yang symbol [8].

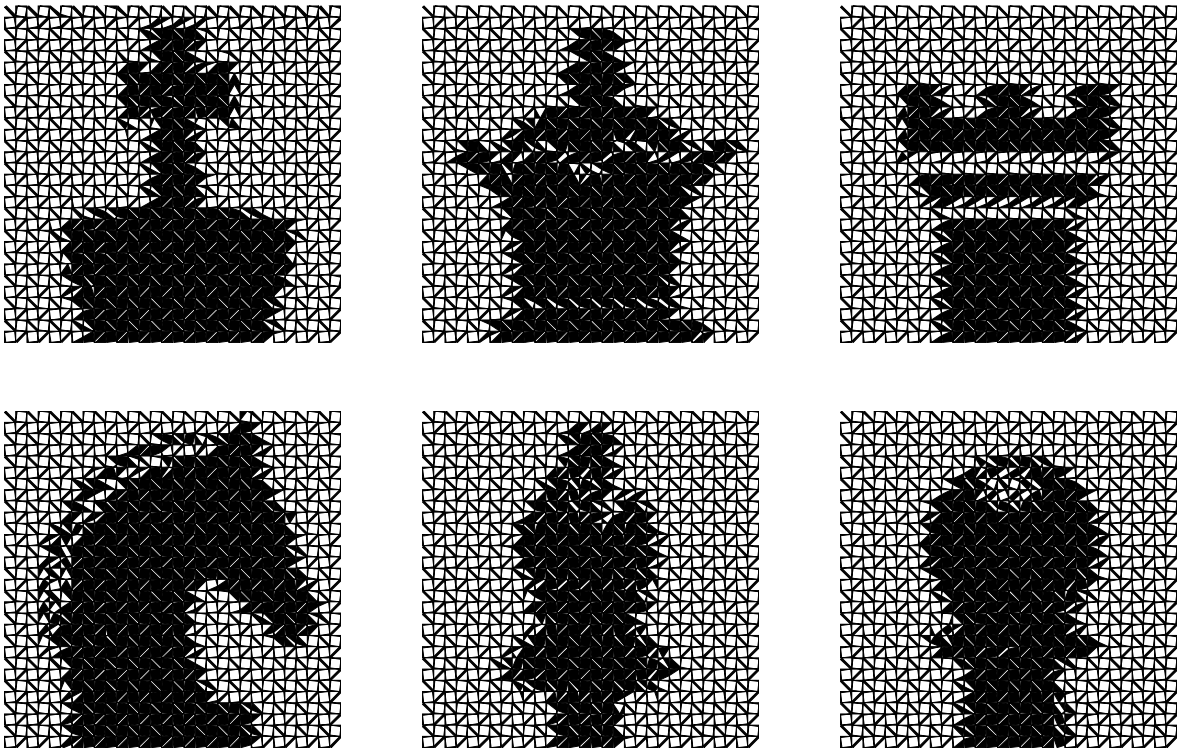

Figure 11: Six 30×30 mosaic representations of black chess pieces: king, queen, rook, knight, bishop, and pawn [9]. These are heavily modified versions of the original images.

References

[1] G. Westerveld. "The Origin of the Checkers and Modern Chess Game. Volume I." *Academia de Estudios Humanísticos de Blanca*, Mar. 1, 2016, pp. 9–25.

[2] R. Bosch and U. Colley. "Figurative mosaics from flexible Truchet tiles." *Journal of Mathematics and the Arts*, vol. 7, no. 3-4, 2013, pp. 1–17.

[3] R. Bosch. *Opt Art: From Mathematical Optimization to Visual Design*, Princeton University Press, 2019.

[4] Y. Lee and W. Kim. "Shoelace Formula: Connecting the Area of a Polygon with Vector Cross Product." *National Council of Teachers of Mathematics*, vol. 110, no. 8, 2017, pp. 631–636.

[5] Wikipedia (Public Domain) , `https://web.archive.org/web/20200428161428/https://en.wikipedia.org/wiki/File:Mona_Lisa.jpg` , [Online; accessed 2018-03-12].

[6] M. Woodie, Artist Network (Public Domain) , `https://www.artistsnetwork.com/art-inspiration/7-painting-techniques-youll-want-try/` , [Online; accessed 2018-03-12].

[7] Wikipedia (Public Domain) , `https://web.archive.org/web/20200428162611/https://en.wikipedia.org/wiki/Yin_and_yang`, [Online; accessed 2018-03-12].

[8] Wikipedia (Public Domain) , `https://web.archive.org/web/20200428162729/https://en.wikipedia.org/wiki/Peace_symbols`, [Online; accessed 2018-04-28].

[9] Wikipedia (Public Domain) , `https://web.archive.org/web/20200425113111/https://en.wikipedia.org/wiki/Chess_piece`, [Online; accessed 2018-04-28].