2021

Collaborative Game Mosaics

Xiaoyun Gong
Oberlin College

Collaborative Game Mosaics

Xiaoyun Gong
Honors Advisor: Robert Bosch

March 26, 2021

0 Introduction

Imagine two players are playing a strategy board game. With one player holding black stones and the other player holding white stones, they take turns to place their pieces. After a while, they start to recognize some patterns. With three colors in front of them, black, white, and the color of the game board, they are curious about if they can form a recognizable image. Instead of playing the game to win, they start to collaborate with each other to form a mosaic. Later, they invite more players and provide them their own stones in different shades of gray.

The goal of this project is to use mathematical optimization techniques to design mosaics that simultaneously resemble well-known images and look like the current state of a board game like gomoku or something that could be played on a Go board or chess board. The work described here builds upon techniques devised by Professor Robert Bosch and some of his student[1][2].

The paper will be structured as follows: First, we will discuss the data, variables, objective function, and universal constraints. Then we will explore constraints for two specific games, gomoku and chess. Next we will present several examples, sharing both images and statistics, and we will talk about methods we have devised for improving the results. We will conclude with a discussion of future directions for our research.

1 Data, Variables, Objective Function, and Universal Constraints

1.1 Data

Given a grayscale target image, we first partition it into a grid that consists of r rows and c columns of squares. After calculating the average grayscale value of each cell,

we have an $r \times c$ matrix with grayscale value of each cell in it. If we denote the row-i column-j entry in the matrix as $\beta_{i,j}$, where $i \in \{1, \cdots, r\}$ and $j \in \{1, \cdots, c\}$, then

$$\beta_{i,j} = \text{average grayscale value of cell } (i,j) \ .$$

Here, $\beta_{i,j} \in [0,1]$ and $\beta_{i,j} = 0$ means cell (i,j) is completely black, while $\beta_{i,j} = 1$ means cell (i,j) is completely white. Intermediate values of $\beta_{i,j}$ correspond to shades of gray. Larger $\beta_{i,j}$'s are for cells (i,j) that are brighter.

For images with lower contrast levels, an optional adjustment to the data is to scale each $\beta_{i,j}$ entry so that the maximum and minimum of all the $\beta_{i,j}$'s are 1 and 0 respectively. To update the $\beta_{i,j}$'s, we set $\beta_{\min} = \min_{i,j}\{\beta_{i,j}\}$ and $\beta_{\max} = \max_{i,j}\{\beta_{i,j}\}$ and then we redefine

$$\beta_{i,j} := \frac{\beta_{i,j} - \beta_{\min}}{\beta_{\max} - \beta_{\min}} \ .$$

1.2 Variables

Imagine we are placing stones or game pieces of different shades of gray on a Go board or a Chess board. If we have two players, one player's stones will be black (shade 0) and the other player's stones will be white (shade 2). We reserve shade 1 for the board, which will serve as a background. Note that we assuming that the board is mid-range gray. We will use k to index the shades of gray. For each (i,j) where $i \in \{1, \cdots, r\}$ and $j \in \{1, \cdots, c\}$, we introduce Boolean variables

$$x_{i,j,k} = \begin{cases} 1 & \text{if the cell } (i,j) \text{ is colored in shade } k \ . \\ 0 & \text{if the cell } (i,j) \text{ is not colored in shade } k \ . \end{cases}$$

While these $x_{i,j,k}$ variables enable us to model stone-placement decisions in the classic two-player gomoku game, we want to allow for more than two players. If there are n players where n is even and each player has their own shade of stone, then we assume that player k uses stones of shade k. Here we reserve shade $n/2$ for the board (mid-range gray). In total, there are $r \times c \times (n+1)$ such $x_{i,j,k}$ variables.

1.3 Objective Function

We want to minimize the difference between the shades of gray in our mosaic and the original grayscale values. We define b_k as

$$b_k := \frac{k}{n}.$$

Our goal is to minimize

$$\sum_{i=1}^{r} \sum_{j=1}^{c} \left(\beta_{i,j} - \sum_{k=0}^{n} b_k x_{i,j,k} \right)^2 \ .$$

Let's consider $\big(\beta_{i,j} - \sum_{k=0}^{n} b_k x_{i,j,k}\big)^2$ for each cell (i,j) now. Note that

$$\big(\beta_{i,j} - \sum_{k=0}^{n} b_k x_{i,j,k}\big)^2 = \beta_{i,j}^2 - 2\beta_{i,j} \sum_{k=0}^{n} b_k x_{i,j,k} + \underbrace{\big(\sum_{k=0}^{n} b_k x_{i,j,k}\big)^2}_{①} . \tag{1}$$

Term ① can be rewritten as

$$\begin{aligned}\big(\sum_{k=0}^{n} b_k x_{i,j,k}\big)^2 &= \sum_{k_1=0}^{n}\sum_{k_2=0}^{n} b_{k_1} b_{k_2} x_{i,j,k_1} x_{i,j,k_2} \\ &= \underbrace{\sum_{k=0}^{n} b_k^2 x_{i,j,k}^2}_{②} + \underbrace{\sum_{k_1=0}^{n-1}\sum_{k_2=k_1+1}^{n} 2 b_{k_1} b_{k_2} x_{i,j,k_1} x_{i,j,k_2}}_{③} .\end{aligned}$$

Because the $x_{i,j,k}$ variables are Boolean, we have $x_{i,j,k}^2 = x_{i,j,k}$ in term ②. In addition, if we impose constraints that say that each cell (i,j) can hold at most one stone, then for each (i,j) there will be only one $x_{i,j,k}$ that equals 1. Thus, term ③$= 0$, as each of its product terms equals 0. With these two changes, term ① $= \sum_{k=0}^{n} b_k^2 x_{i,j,k}$. Now, when we substitute the simplified version of term ① into equation (1), we obtain

$$\begin{aligned}\big(\beta_{i,j} - \sum_{k=0}^{n} b_k x_{i,j,k}\big)^2 &= \beta_{i,j}^2 - 2\beta_{i,j} \sum_{k=0}^{n} b_k x_{i,j,k} + \sum_{k=0}^{n} b_k^2 x_{i,j,k} \\ &= \beta_{i,j}^2 + \sum_{k=0}^{n} (b_k^2 - 2 b_k \beta_{i,j}) x_{i,j,k}.\end{aligned}$$

We end up with a simplified objective function,

$$\sum_{i=1}^{r}\sum_{j=1}^{c}\Big(\beta_{i,j}^2 + \sum_{k=0}^{n}(b_k^2 - 2b_k\beta_{i,j})x_{i,j,k}\Big) = \sum_{i=1}^{r}\sum_{j=1}^{c}\beta_{i,j} + \sum_{i=1}^{r}\sum_{j=1}^{c}\sum_{k=0}^{n}(b_k^2 - 2b_k\beta_{i,j})x_{i,j,k} \ . \tag{2}$$

Note that while the original objective function is convex quadratic in the $x_{i,j,k}$'s, the new and simplified version is affine (a linear function of the $x_{i,j,k}$'s plus a constant). This simplification helps optimization solvers to find the optimal solution much faster.

1.4 Universal Constraints

In most games, each location on the game board can hold at most one game piece. We therefore impose constraints that ensure that each cell holds at most one stone. Suppose n players are playing on a $r \times c$ board. For each cell (i,j), we impose

$$\sum_{k=0}^{n} x_{i,j,k} = 1 \ . \qquad \text{(Universal Constraint\#1)}$$

As players take turns to place their stones, the difference between the number of stones in different shades should be 0. For each cell (i, j) and each $k_1, k_2 \in \{0, \cdots, n\}$ with $k_1 \neq k_2$, $k_1 \neq n/2$, and $k_2 \neq n/2$,

$$\sum_{i=1}^{r} \sum_{j=1}^{c} (x_{i,j,k_1} - x_{i,j,k_2}) = 0. \qquad \text{(Universal Constraint\#2)}$$

As these universal constraints apply to all games, in the next section we going to introduce particular games and describe constraints for these games.

2 Gomoku Game

The game of *gomoku* is an abstract strategy board game. It is traditionally played with Go pieces (black and white stones) on a Go board. The gomoku game is also called "Five in a Row." Players alternate turns placing a stone of their color on an empty intersection. The winner is the first player to form an unbroken chain of five stones - horizontally, vertically, or diagonally.

2.1 Gomoku Constraints

To form an unfinished gomoku game on a go board, we require that there are no five consecutive stones in the same shades in a row, column or diagonal.

To make sure that there are not any rows of five consecutive stones of the same shade,for all $i \in \{1, \cdots, r\}$, $j \in \{1, \cdots, c-4\}$, $k \in \{0, \cdots, n\}$ with $k \neq n/2$, and $J \in \{1, \cdots, c-4\}$, we impose

$$\sum_{j=J}^{J+4} x_{i,j,k} \leq 4. \qquad \text{(Gomoku Constraint\#1)}$$

Similarly, to ensure that there are not any five consecutive stones of the same shades, for all $i \in \{1, \cdots, r-4\}$, $j \in \{1, \cdots, c\}$, $k \in \{0, \cdots, n\}$ with $k \neq n/2$, and $I \in \{1, \cdots, r-4\}$, we require that

$$\sum_{i=I}^{I+4} x_{i,j,k} \leq 4. \qquad \text{(Gomoku Constraint\#2)}$$

For diagonals, we need that for all $i \in \{1, \cdots, r\}$, $j \in \{1, \cdots, c\}$, and $k \in \{0, \cdots, n\}$ with $k \neq n/2$,

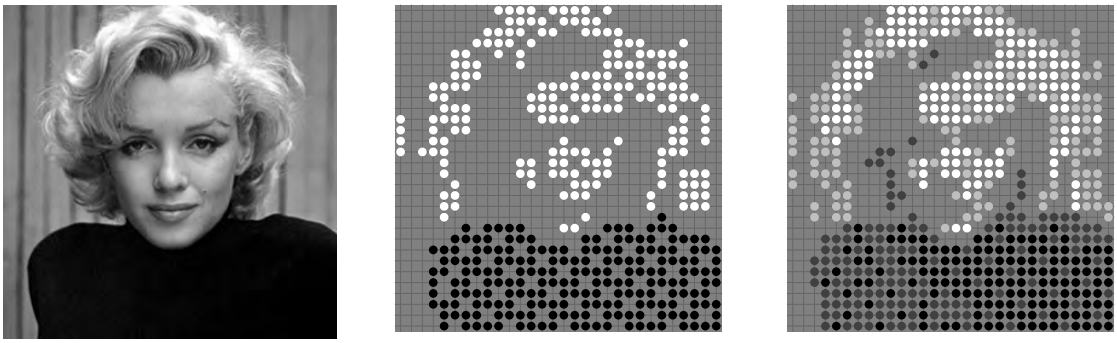
$$x_{i,j,k} + x_{i+1,j+1,k} + x_{i+2,j+2,k} + x_{i+3,j+3,k} + x_{i+4,j+4,k} \leq 4\ , \quad \text{(Gomoku Constraint \# 3)}$$

$$x_{i+4,j,k} + x_{i+3,j+1,k} + x_{i+2,j+2,k} + x_{i+1,j+3,k} + x_{i,j+4,k} \leq 4\ . \quad \text{(Gomoku Constraint \# 4)}$$

2.2 Examples

Based on the objective function we discussed in Section 1.3 and the constraints we discussed in both Section 1.4 and 2.1, we created a C program that can generate a *.lp* file that can be read by an optimization solver *Gurobi*[3]. We then use the solution *Gurobi* provided to draw the image. We generated three groups of images, and in each case we collected statistics on work time (in seconds), the size of the branch-and-bound tree (in nodes), and two measures of quality: total squared error and average squared error. While some of these images are high quality, some are merely passable. We will also discuss what leads to a poor result, and what are some possible solutions.

2.2.1 Example 1: *Photograph of Marilyn Monroe* by Alfred Eisenstaed[4]



(a) Original Image (b) $n = 2$ (c) $n = 4$

(d) $n = 6$ (e) $n = 8$

Figure 1: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Gomoku, $n = 2, 4, 6, 8$.

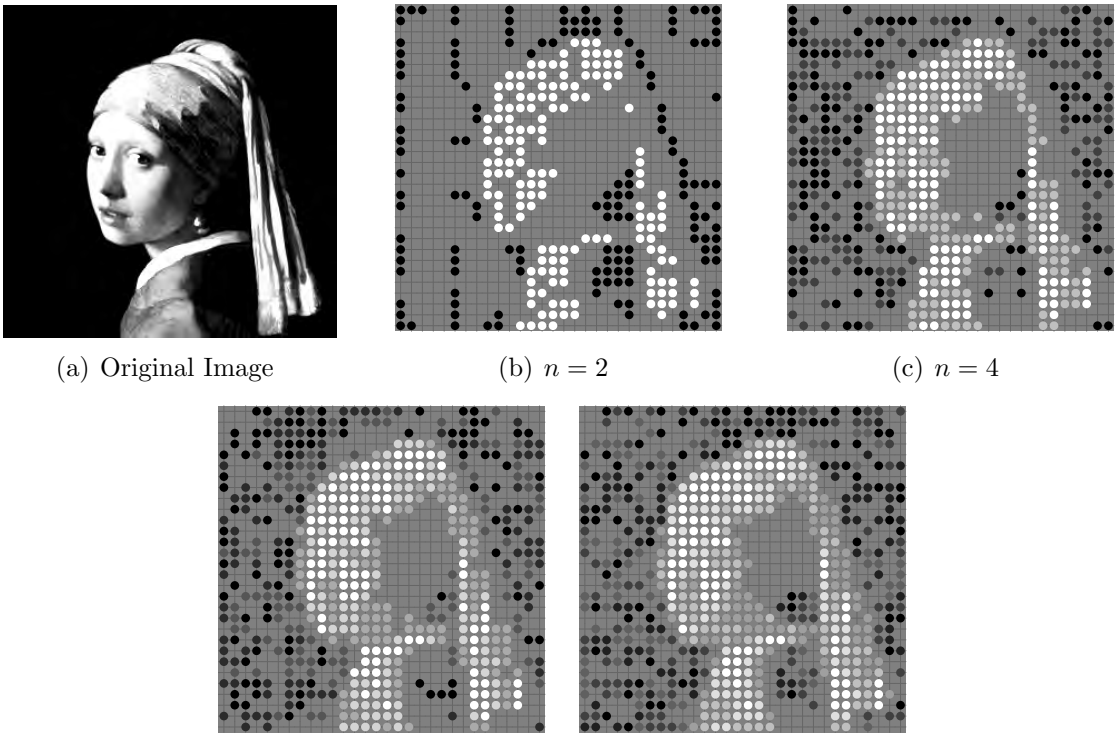
In the table above, for each case, the total square error is the optimal value of the objective function. The average squared error is given by

$$\text{Average Squared Error} = \frac{\text{Total Squared Error}}{r \times c}.$$

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.19	1	43.2034	0.0480
4	0.40	1	24.5684	0.0273
6	1.20	11	20.5178	0.0228
8	0.62	1	18.5184	0.0206

Table 1: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Gomoku, $n = 2, 4, 6, 8$.

2.2.2 Example 2: *Girl with a Pearl Earring* by Johannes Vermeer[5]



(a) Original Image (b) $n = 2$ (c) $n = 4$

(d) $n = 6$ (e) $n = 8$

Figure 2: *Girl with a Pearl Earring* by Johannes Vermeer, Gomoku, $n = 2, 4, 6, 8$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.45	1	121.6094	0.1351
4	0.28	0	105.9544	0.1177
6	0.60	1	102.6205	0.1140
8	0.63	1	101.4719	0.1127

Table 2: *Girl with a Pearl Earring* by Johannes Vermee, Gomoku, $n = 2, 4, 6, 8$.

Compared to the last example (*Photograph of Marilyn Monroe*), keeping the number

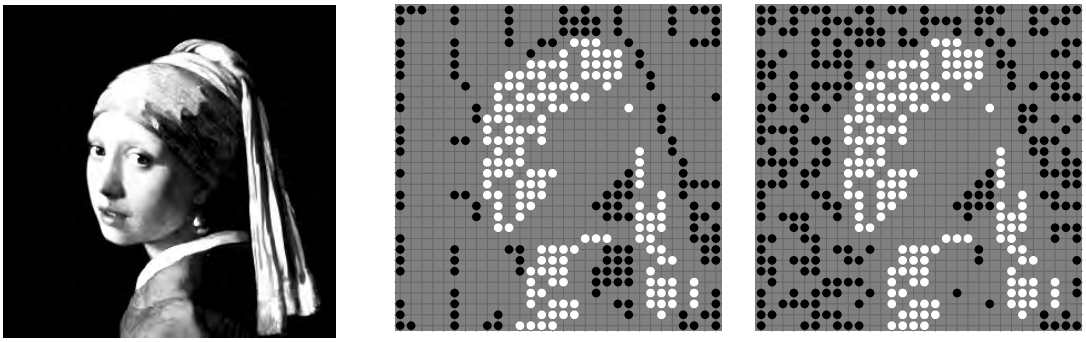
of different shades the same, this group of images has larger total and average errors. Why is this the case?

Notice that *Girl with a Pearl Earring* has a black background while the *Photograph of Marilyn Monroe* has a gray background. As we did not put any "Gomoku constraints" on the shade of middle-range gray, we require each player to play the same number of stones in other shades. In the *Girl with a Pearl Earring* example, we need more stones in lighter shades to balance the dark background.

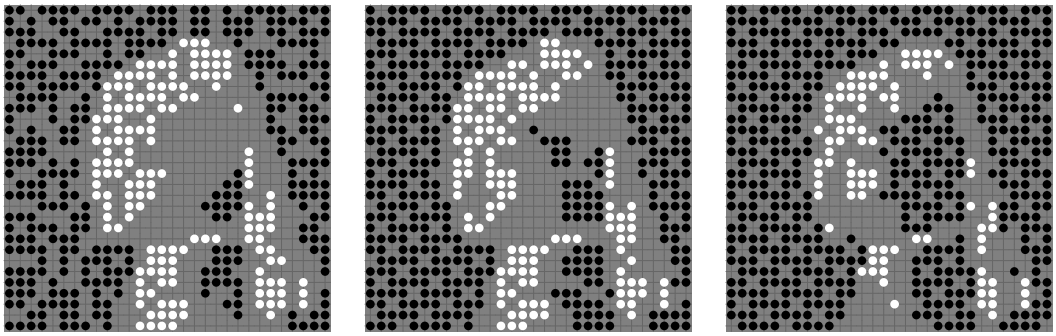
How can we solve this problem? A way to balance highlight and shadow is to alter "Universal Constraint #2." In this case, we need to allow the player holding black stone to play more than the player holding white stone. To generalize this idea, we want to give a *handicap* of h stones to either player. With everything else remaining the same, we alter "Universal Constraint #2." For all $i \in \{1, \cdots, r\}$, $j \in \{1, \cdots, c\}$, $k_1, k_2 \in \{0, \cdots, n\}$ with $k_1 \neq k_2$, $k_1 \neq n/2$, and $k_2 \neq n/2$, we now impose

$$\left| \sum_{i=1}^{r} \sum_{j=1}^{c} (x_{i,j,k_1} - x_{i,j,k_2}) \right| \leq h.$$

The figure and table below shows the resulting images and statistics of altering "Universal Constraint #2" with different h's.



(a) Original Image (b) $h = 0$ (Reference) (c) $h = 100$



(d) $h = 200$ (e) $h = 300$ (f) $h = \max = 418$

Figure 3: *Girl with a Pearl Earring* by Johannes Vermeer, Gomoku with altered Universal Constraint #2.

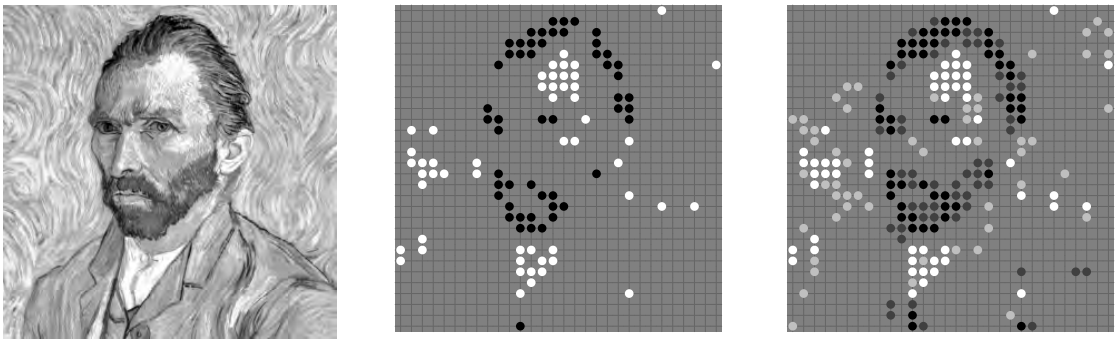
Table 3: Statistics for *Girl with a Pearl Earring* by Johannes Vermeer, Gomoku with altered Universal Constraint #2.

$h =$	work time (seconds)	# of nodes	Total Squared Error	Average Error
0	0.45	1	121.6094	0.1351
100	0.47	0	96.6094	0.1973
200	0.63	1	71.6094	0.0795
300	0.36	1	48.2094	0.0536
$h_{\max} = 418$	0.15	1	36.5894	0.0406

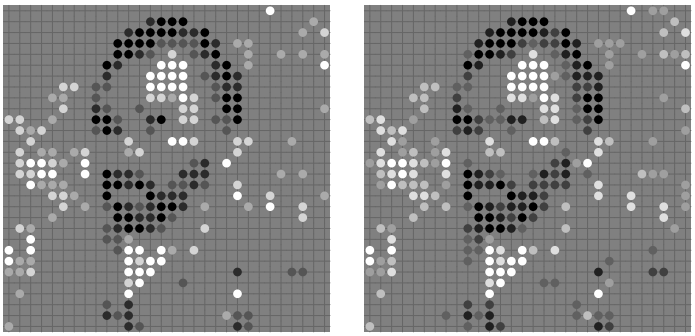
(Note: We obtained $h_{\max}$ by removing "Universal Constraint #2" instead of altering it. In such case, there are 505 black stones, 328 empty cells, and 87 white stones.)

Altering "Universal Constraint #2" helped us reduce the errors significantly. But this might not be the case for all images. We are going to discuss this topic in example 3.

2.2.3 Example 3: *Self-portrait* by Vincent Van Gogh



(a) Original Image (b) $n = 2$ (c) $n = 4$



(d) $n = 6$ (e) $n = 8$

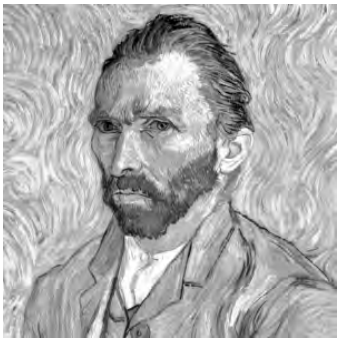
Figure 4: *Self-portrait* by Vincent Van Gogh, Gomoku, $n = 2, 4, 6, 8$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.05	0	29.8272	0.0331
4	0.15	0	24.8572	0.0276
6	0.38	1	23.0305	0.0255
8	0.48	0	22.1447	0.0245

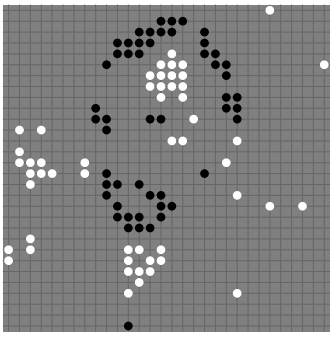
Table 4: *Self-portrait* by Vincent Van Gogh, Gomoku, $n = 2, 4, 6, 8$.

For this group of images, we observed that the Average Errors are about the same as the *Photograph of Marilyn Monroe* group of example, but the resulting images are not very ideal. As mentioned before, we wondered what will happen if we adapt the same alternation as we did in the *Girl with a Pearl Earring* example? The Figure and table below show the result of altering "Universal Constraint # 2."

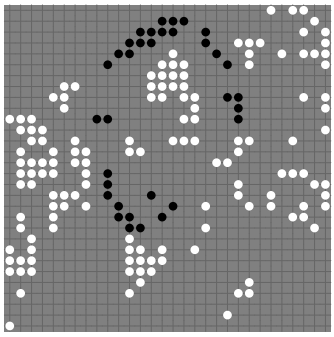
Figure 5: *Self-portrait* by Vincent Van Gogh, Gomoku with $h = 0, 107$.



(a) Original Image



(b) $h = 0$ (Reference)



(c) $h_{\max} = 107$

Table 5: *Self-portrait* by Vincent Van Gogh, Gomoku with $h = 0, 107$.

$h =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
0	0.05	0	29.8272	0.0331
$h_{\max} = 107$	0.01	1	26.7372	0.0297

(Note: When $h = h_{\max} = 107$, there are 34 black stones, 725 empty cells, and 141 white stones.)

Unlike in the example of *Girl with a Pearl Earring*, after altering "universal constraint #2", the errors didn't change as much. It is not hard to arrive at the conclusion that balancing highlight and shadow cannot result in a better image. However, by comparing to the last two examples, we noticed that *Self-portrait* has a very large gray area with little highlight and shadow. How should we handle situations like this? We will discuss this issue in the section "Future Directions."

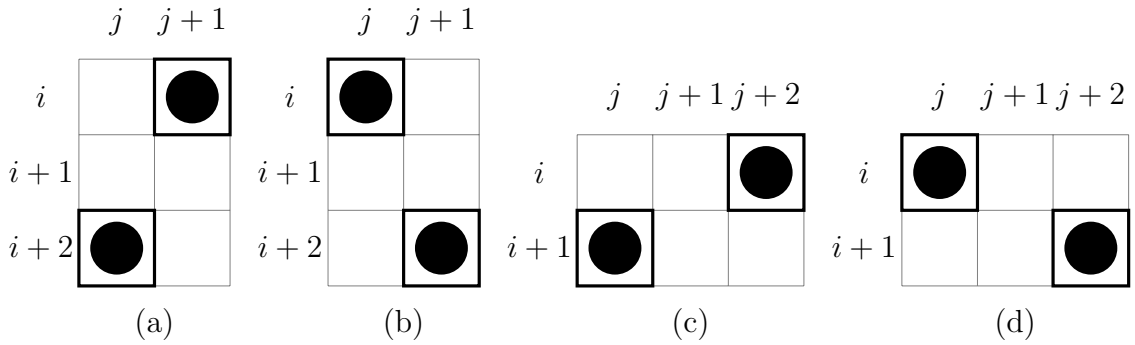
3 Chess Game

We now turn our attention to a different board game, chess. Here, we do not attempt to use all of the pieces. We instead allow ourselves to use knights and nothing else. When playing chess, each of the two players has two knights. Here, we allow each player to use as many knights as they want, provided that each player uses the same number.

3.1 Chess Constraints

We are mostly interested in the constraints because the data, variables, and objective functions are the same as they are in the gomoku game. We can separate knight moves

into two cases as in the figure below.



As before, suppose there are n players facing a $r \times c$ chess board. We still reserve shade $n/2$ for the background.

In case (a) and (b), for all $i \in \{1, \cdots, r-2\}$, $j \in \{1, \cdots, c-1\}$, and $k_1, k_2 \in \{0, \cdots, n\}$ with $k_1 \neq k_2 \neq n/2$, we impose

$$x_{i,j+1,k_1} + x_{i+2,j,k_2} \leq 1 \qquad \text{(Knight Constraint \#1)}$$

$$x_{i,j,k_1} + x_{i+2,j+1,k_2} \leq 1 \qquad \text{(Knight Constraint \#2)}$$

In case (c) and (d), for all $i \in \{1, \cdots, r-1\}$, $j \in \{1, \cdots, j-2\}$, and $k_1, k_2 \in \{0, \cdots, n\}$ with $k_1 \neq k_2 \neq n/2$, we impose

$$x_{i+1,j,k_1} + x_{i,j+2,k_2} \leq 1 \qquad \text{(Knight Constraint \#3)}$$

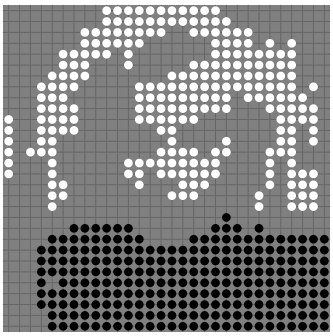
$$x_{i,j,k_1} + x_{i+1,j+2,k_2} \leq 1 \qquad \text{(Knight Constraint \#4)}$$

In addition to these Knight constraints, we also need Universal Constraint #1 and #2 discussed in Section 1.4.

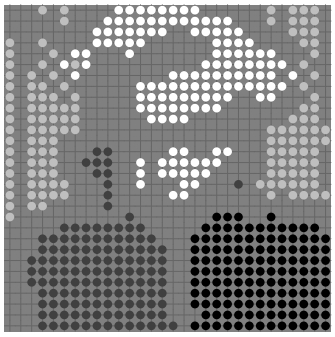
3.2 Examples

3.2.1 Example 1: *Photograph of Marilyn Monroe* by Alfred Eisenstaed

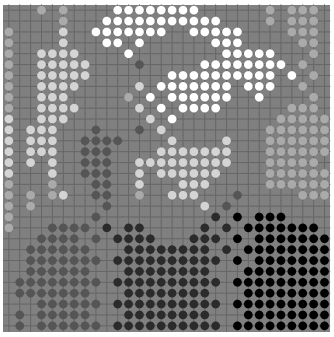
(d) Original Image



(e) $n = 2$



(f) $n = 4$



(g) $n = 6$

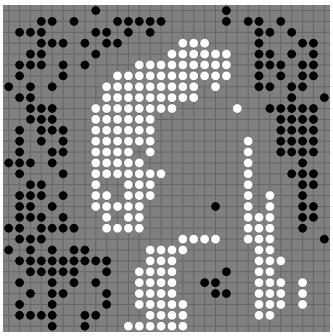
Figure 6: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Chess, $n = 2, 4, 6$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.05	0	36.6134	0.0406
4	7.76	318	27.6734	0.0307
6	10529.82	86474	29.6556	0.0329

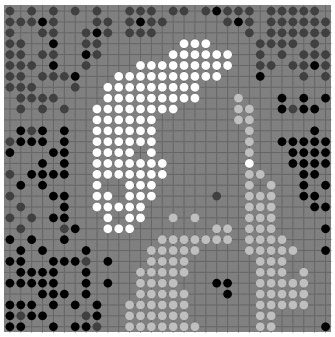
Table 6: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Chess, $n = 2, 4, 6$.

3.2.2 Example 2: *Girl with a Pearl Earring* by Johannes Vermeer

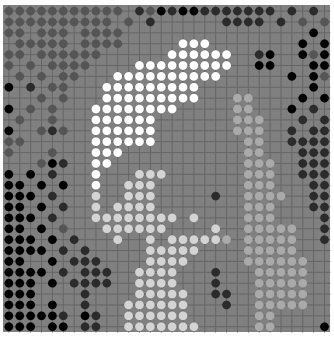
(a) *Girl with a Pearl Earring*



(b) $n = 2$



(c) $n = 4$



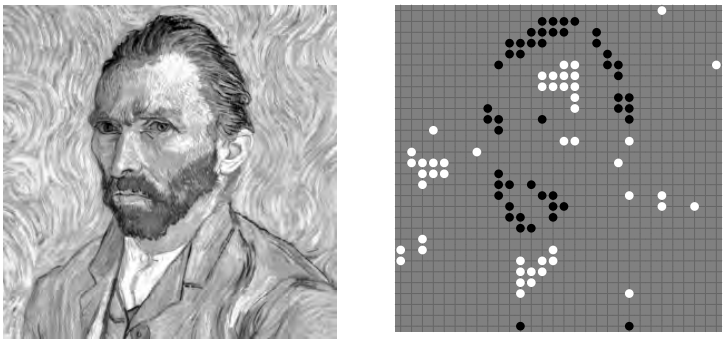
(d) $n = 6$

Figure 7: *Girl with a Pearl Earring* by Johannes Vermeer, Chess, $n = 2, 4, 6$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.07	0	114.1894	0.1268
4	1.71	1	110.7844	0.1230
6	240.95	3667	112.9571	0.1255

Table 7: *Girl with a Pearl Earring* by Johannes Vermeer, Chess, $n = 2, 4, 6$.

3.2.3 Example 3: *Self-portrait* by Vincent Van Gogh



(a) Original Image (b) $n = 2$

(c) $n = 4$ (d) $n = 6$

Figure 8: *Self-portrait* by Vincent Van Gogh, Chess, $n = 2, 4, 6$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.04	0	30.0472	0.0333
4	0.8	4	26.5822	0.0295
6	2.61	1	25.4127	0.0282

Table 8: *Self-portrait* by Vincent Van Gogh, Chess, $n = 2, 4, 6$.

In all three examples we see that the Knight constraints force stones of the same shade to form clusters. Also note that, unlike in the gomoku game, as n increases, work time increases.

How should we break the clusters? This leads us to the discussion in next section.

4 Alter the Knight Constraints

4.1 Knight Constraints (Altered)

In the last section, we wanted to find a way to prevent stones from clustering. What if stones in shade k attack stones in shade k instead of stones in other shades? In light of this idea, we alter the constraints so that for all $i \in \{1, \cdots, r-2\}$, $j \in \{1, \cdots, c-1\}$, and $k \in \{0, \cdots, n\}$ with $k \neq n/2$,

$$x_{i,j+1,k} + x_{i+2,j,k} \leq 1 \qquad \text{(Altered Knight Constraint \#1)}$$

$$x_{i,j,k} + x_{i+2,j+1,k} \leq 1 \qquad \text{(Altered Knight Constraint \#2)}$$

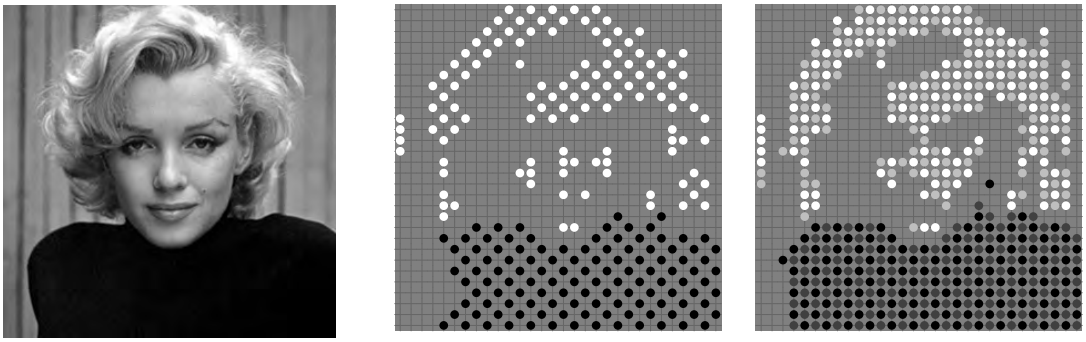
For all $i \in \{1, \cdots, r-1\}$, $j \in \{1, \cdots, c-2\}$, and $k \in \{0, \cdots, n\}$ with $k \neq n/2$,

$$x_{i+1,j,k} + x_{i,j+2,k} \leq 1 \qquad \text{(Altered Knight Constraint \#3)}$$

$$x_{i,j,k} + x_{i+1,j+2,k} \leq 1 \qquad \text{(Altered Knight Constraint \#4)}$$

4.2 Examples

4.2.1 Example 1: *Photograph of Marilyn Monroe* by Alfred Eisenstaed



(a) Original Image (b) $n = 2$ (c) $n = 4$

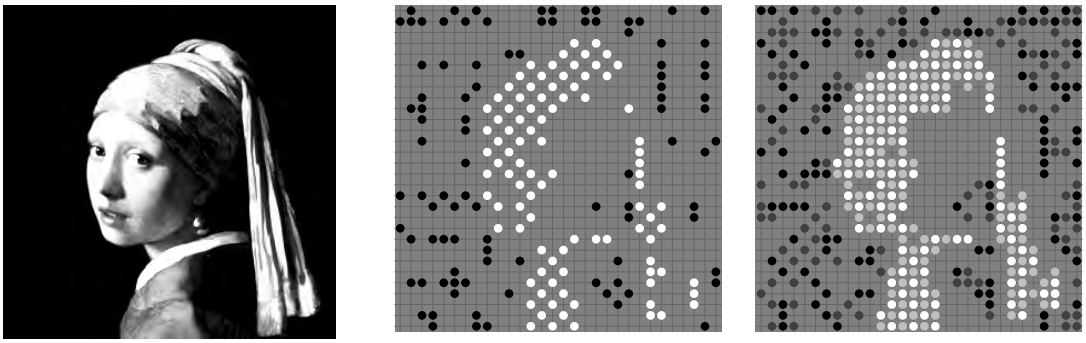
(d) $n = 6$ (e) $n = 8$

Figure 9: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Chess (altered), $n = 2, 4, 6, 8$.

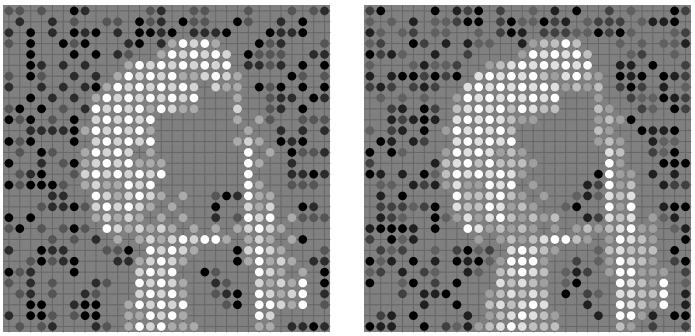
$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.07	0	56.2134	0.0624
4	0.27	0	30.2434	0.0336
6	1.47	1	24.1156	0.0267
8	1.20	1	20.9934	0.0233

Table 9: *Photograph of Marilyn Monroe* by Alfred Eisenstaed, Chess (altered), $n = 2, 4, 6, 8$.

4.2.2 *Example 2: Girl with a Pearl Earring* by Johannes Vermeer



(a) Original Image (b) $n = 2$ (c) $n = 4$



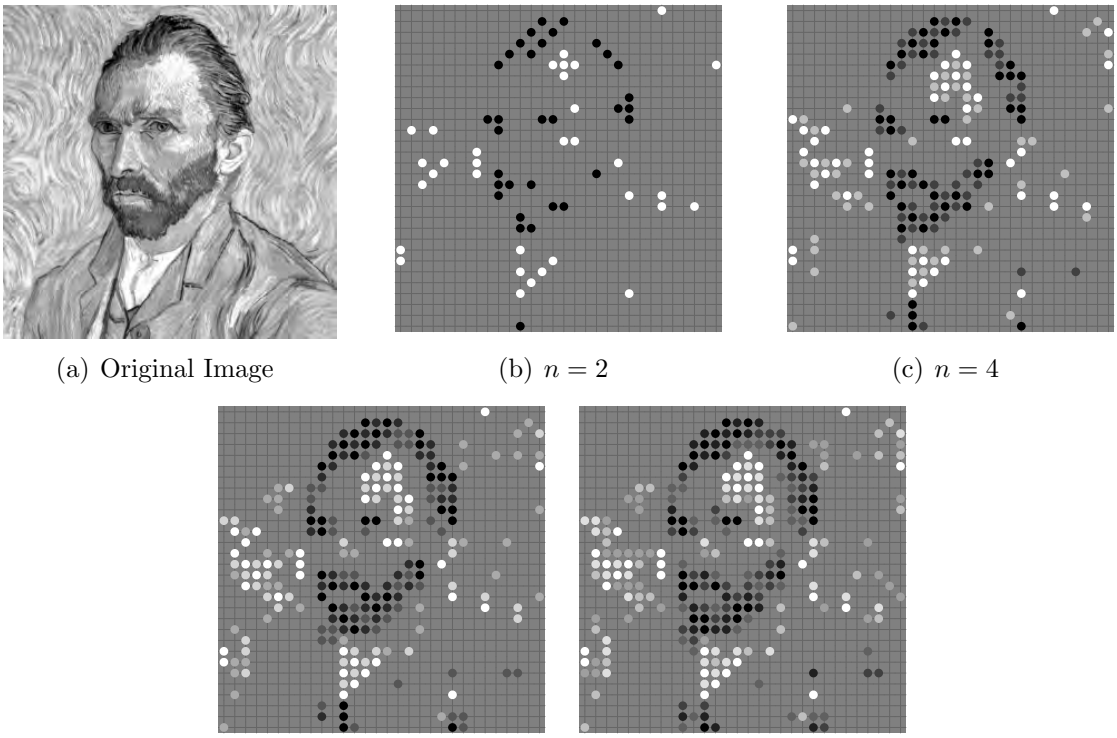
(d) $n = 6$ (e) $n = 8$

Figure 10: *Girl with a Pearl Earring* by Johannes Vermeer, Chess (altered), $n = 2, 4, 6, 8$.

$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.11	0	137.7594	0.1530
4	0.66	0	113.5494	0.1261
6	1.02	0	106.9771	0.1188
8	3.13	1	103.7744	0.1153

Table 10: *Girl with a Pearl Earring* by Johannes Vermeer, Chess (altered), $n = 2, 4, 6, 8$.

4.2.3 Example 3: *Self-portrait* by Vincent Van Gogh



(a) Original Image (b) $n = 2$ (c) $n = 4$

(d) $n = 6$ (e) $n = 8$

Figure 11: *Self-portrait* by Vincent Van Gogh, Chess (altered), $n = 2, 4, 6, 8$.

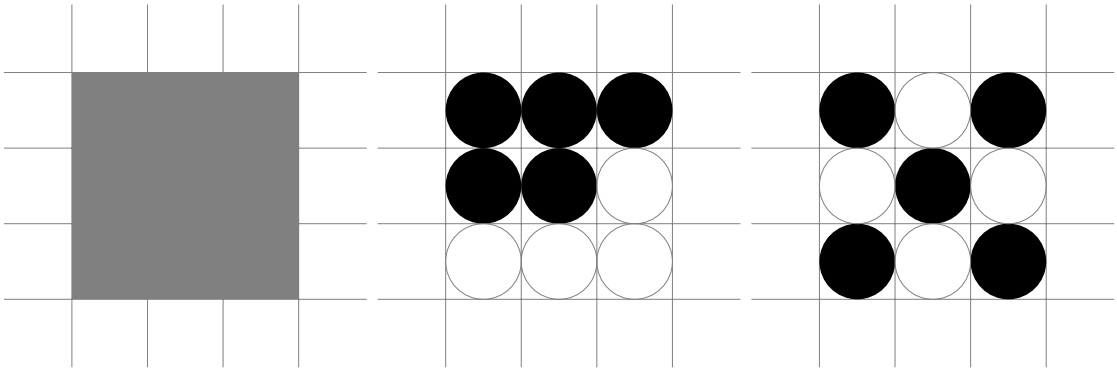
$n =$	work time (seconds)	# of nodes	Total Squared Error	Average Squared Error
2	0.04	0	32.4872	0.0360
4	0.24	1	26.7672	0.0297
6	0.82	1	24.1538	0.0268
8	0.87	0	22.8372	0.0253

Table 11: *Self-portrait* by Vincent Van Gogh, Chess (altered), $n = 2, 4, 6, 8$.

5 Future Directions: Alternative Approach

In section 2.3.3, we left a question on how to handle areas of middle-range gray. Recall that in section 2, we measured the goodness of fit by comparing each individual cell to its corresponding average grayscale value. However, this approach can not handle a big area of middle-range gray. To solve this, we propose the alternative approach.

Let's consider the case in which the image we are fitting is a middle range gray square, as shown in the left graph below. Suppose that we are fitting this image with black and white stones. Here, all the $\beta_{i,j}$'s are 0.5, and 0 represent pure black and 1 represent pure white. The optimization method we discussed in the last section will give us random output of 0's and 1's. One example is shown in the center graph below. Because the error is the same either way, it doesn't matter which color of stones we put. But is this the best solution? The ideal solution should look like the right graph below. Instead of each one by one cell, let's consider each two-by-two square composed of four cells and compare it with the average grayscale value of the corresponding two-by-two square in the original image.



In the center figure, the top left 2×2 square has a total brightness level of 0, while the top right 2×2 square has a total brightness value of 1, the bottom left 2×2 square has a total brightness value of 2, and the bottom right 2×2 square has a total brightness of 3. The total brightness value of all the 2×2 squares vary. However, in the right figure, each of the 2×2 squares has a total brightness value of 2, which is the same as 4×0.5, the ideal value of a middle-range gray target image.

5.1 Variables

Suppose we cut the target image into $r \times c$ squares. We need a variable to represent all the possibilities of color combinations in each two-by-two square. Consider the $r \times c$ matrix M whose row-i-column-j entry is the number of shade of the stone we placed at the row-i-column-j position on the board. Let $\mathcal{M}$ be the set of all possible 2×2 blocks of M.

For all $i \in \{1, \cdots, r-1\}$, $j \in \{1, \cdots, c-1\}$, and $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \in \mathcal{M}$, we denote

$$y_{i,j,M} = \begin{cases} 1, & \text{if cells } (i,j), (i,j+1), (i+1,j), (i+1,j+1) \\ & \text{are colored in shades } m_{11}, m_{12}, m_{21}, m_{22} \text{ respectively.} \\ 0, & \text{otherwise.} \end{cases}$$

If there are n players and $n+1$ shades including the background gray, we have a total of $(n+1)^4(r-1)(c-1)$ such variables.

5.2 Objective Function

We want to minimize the difference between the sum of the fitted color in the two-by-two squares and the sum of the original grayscale value in the two-by-two squares. Therefore, for all $i \in \{1, \cdots, r-1\}$, $j \in \{1, \cdots, c-1\}$, and $M \in \mathcal{M}$, we want to minimize

$$\sum_{i=1}^{r-1} \sum_{j=1}^{c-1} \sum_{M \in \mathcal{M}} [(m_{11} + m_{12} + m_{21} + m_{22}) - (\beta_{i,j} + \beta_{i,j+1} + \beta_{i+1,j} + \beta_{i+1,j+1})]^2 y_{i,j,M} \ .$$

5.3 Constraints

For each two-by-two squares, we can only place one two-by-two matrix M. Therefore, for all $i \in \{1, \cdots, r-1\}$ and $j \in \{1, \cdots, c-1\}$ we impose

$$\sum_{M \in \mathcal{M}} y_{i,j,M} = 1 \ . \qquad \text{(Two by Two Constraint \#1)}$$

We also need to make sure that we color the two-by-two squares in such a way that they overlap with each other both horizontally and vertically. To achieve that, we asked that for all $i \in \{1, \cdots, r-1\}$ and $j \in \{1, \cdots, c-2\}$,

$$\sum_{M \in \mathcal{M}} y_{i,j,M} = \sum_{N \in \mathcal{M}} y_{i,j+1,N} \quad \text{when } m_{12} = n_{11} = u \text{ and } m_{22} = n_{21} = v \ .$$

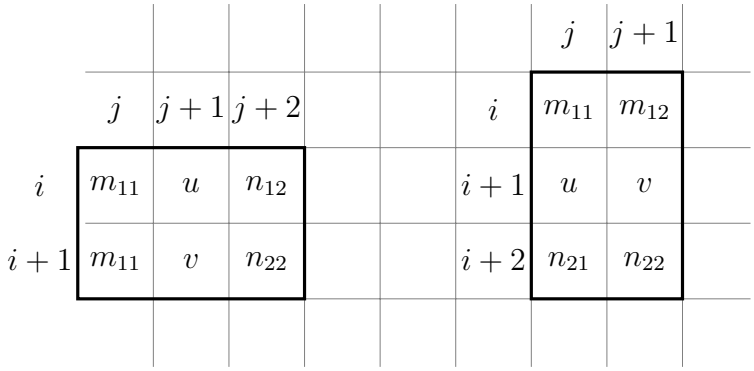
(Two by Two Constraint #2)

and for all $i \in \{1, \cdots, r-2\}$ and $j \in \{1, \cdots, c-1\}$,

$$\sum_{M \in \mathcal{M}} y_{i,j,M} = \sum_{N \in \mathcal{M}} y_{i+1,j,N} \quad \text{when } m_{21} = n_{11} = u \text{ and } m_{22} = n_{12} = v \ .$$

(Two by Two Constraint #3)

The figure below demonstrate the constraints above.



We still need that there are no five consecutive cells in a row, column, or diagonal that are all colored in the same shades of gray, and the difference between cells colored in different shades should be 0. We need the corresponding constraints from the previous sections. Therefore we need constraints that link the $x_{i,j,k}$'s to the $y_{i,j,M}$'s. We need to make sure that $y_{i,j,M} = 1$ if and only if $x_{i,j,m_{11}} = 1$, $x_{i,j+1,m_{12}} = 1$, $x_{i+1,j,m_{21}} = 1$ and $x_{i+1,j+1,m_{22}} = 1$. To achieve this, we impose linear constraints

$$y_{i,j,M} \leq x_{i,j,m_{11}}, \qquad \text{(Two by Two Constraint \#4)}$$

$$y_{i,j,M} \leq x_{i,j,m_{12}}, \qquad \text{(Two by Two Constraint \#5)}$$

$$y_{i,j,M} \leq x_{i,j,m_{21}}, \qquad \text{(Two by Two Constraint \#6)}$$

$$y_{i,j,M} \leq x_{i,j,m_{22}}, \qquad \text{(Two by Two Constraint \#7)}$$

$$x_{i,j,m_{11}} + x_{i,j,m_{12}} + x_{i,j,m_{21}} + x_{i+1,j+1,m_{22}} \leq 3 + y_{i,j,M}. \qquad \text{(Two by Two Constraint \#8)}$$

Although this approach would solve the problem in theory, when we tried to use *Gurobi* to solve it, *Gurobi* wasn't able to find an optimal solution after two full days. The reason why is that the model has a large number of variables. For example, when $r = 30$, $c = 30$, and $n = 2$ we will need $3^4 \times 29 \times 29 = 68121$ $y_{i,j,M}$ variables and $3 \times 30 \times 30 = 2700$ $x_{i,j,k}$ variables.

6 Acknowledgements

I developed and greatly extended this project from the group final project assignment of Nonlinear Optimization class taught by Professor Robert Bosch. I appreciate the efforts that Shuangwei Yu and Carolyn Zhao contributed to this project. Under the supervision of Professor Robert Bosch, I continued this project as my honor thesis. I would like to thank Professor Robert Bosch for providing valuable suggestions on research directions and offering practical assistance on programming and wording. I am also grateful for the endless encouragement from Professor Robert Bosch.

7 References

1 Bosch, R., 2019. Opt Art: From Mathematical Optimization to Visual Design. Princeton University Press.

2 Bosch, R. and Pike, A., 2009. Map-Colored Mosaics. Bridges Banff: mathematical connections in art, music, and science, pp.139-146.

3 "Gurobi Optimizer," http://www.gurobi.com/products/gurobi-optimizer.

4 Pix Inc., 1953. Photograph of Marilyn Monroe. [image] Available at: https://www.biography.com/actor/marilyn-monroe [Accessed 3 October 2020].

5 Vermeer J., 1665. Girl with a Pearl Earring. [oil on canvas].

6 Van Gogh, V., 1889. Self-portrait. [oil on canvas].

7 Vanderbei, R.J., 2014. Linear Programming: Foundations and Extensions, Springer.

8 Yucata.de. 2021. Yucata - Rules for the game 'Gobang & Gomoku'. [online] Available at: https://www.yucata.de/en/Rules/Gomoku [Accessed 18 March 2021].