A New Tile-Based Method for Constructing Single-Line Drawings

Robert Bosch and Izzy Snyder

Oberlin College, Oberlin, Ohio, USA; rbosch@oberlin.edu, isnyder@oberlin.edu

Abstract

We present a new tile-based method for constructing single-line drawings. Our method involves positioning decorated square tiles on a rectangular board. We use a set of eight tiles. Each one has a light beige background, and seven of the eight are adorned with black line segments that connect one or more midpoints of certain sides of the tile to one or more midpoints of other sides. When we place copies of these tiles on the board, we follow one rule: we must arrange them in such a way that their black line segments join together to form a single line that enters the board on the left and leaves it on the right. Our goal is to construct a single line that can be easily traced by hand or eye when viewed from up close yet also resembles a recognizable target image when viewed from a distance.

Introduction

We present a new tile-based method for constructing single-line drawings. Our method involves positioning decorated square tiles on a rectangular board. We use the set of eight tiles shown in Figure 1.

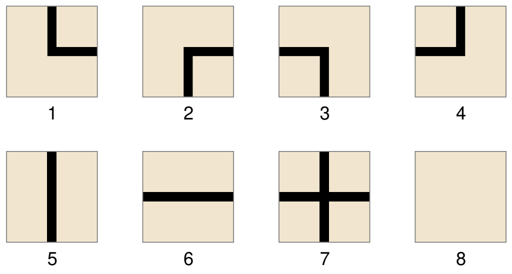

Figure 1: *The eight tiles.*

Each of these tiles has a light beige background, and seven of the eight are adorned with black line segments that connect one or more midpoints of certain sides of the tile to one or more midpoints of other sides. When we place copies of these tiles on the board, we follow one rule: we must arrange them in such a way that their black line segments join together to form a single line that enters the board on the left and leaves it on the right. Our goal is to construct a single line that can be easily traced by hand or eye when viewed from up close yet also resembles a recognizable target image when viewed from a distance.

Figure 2 displays an example. The top left subfigure is a low-resolution (19×19) target image, and the top right subfigure is an easily-traced single-line approximation of this target image. To understand what we mean by "easily traced," it is helpful to think of the single-line drawing as a map of the streets in some neighborhood of some city and to imagine that a school bus driver wants to enter the neighborhood at the midpoint of its western border (the midpoint of the left edge of the map), traverse each street segment in order to drop off the children who live there, and ultimately exit the neighborhood from the midpoint of its eastern border (the midpoint of the right edge of the map). To minimize distance traveled, the driver wants to traverse each street segment exactly once. And to allow themselves to focus entirely on the safety of the children, the driver wants to be able to drive through the neighborhood without giving any thought to where they'll turn left and where they'll turn right.

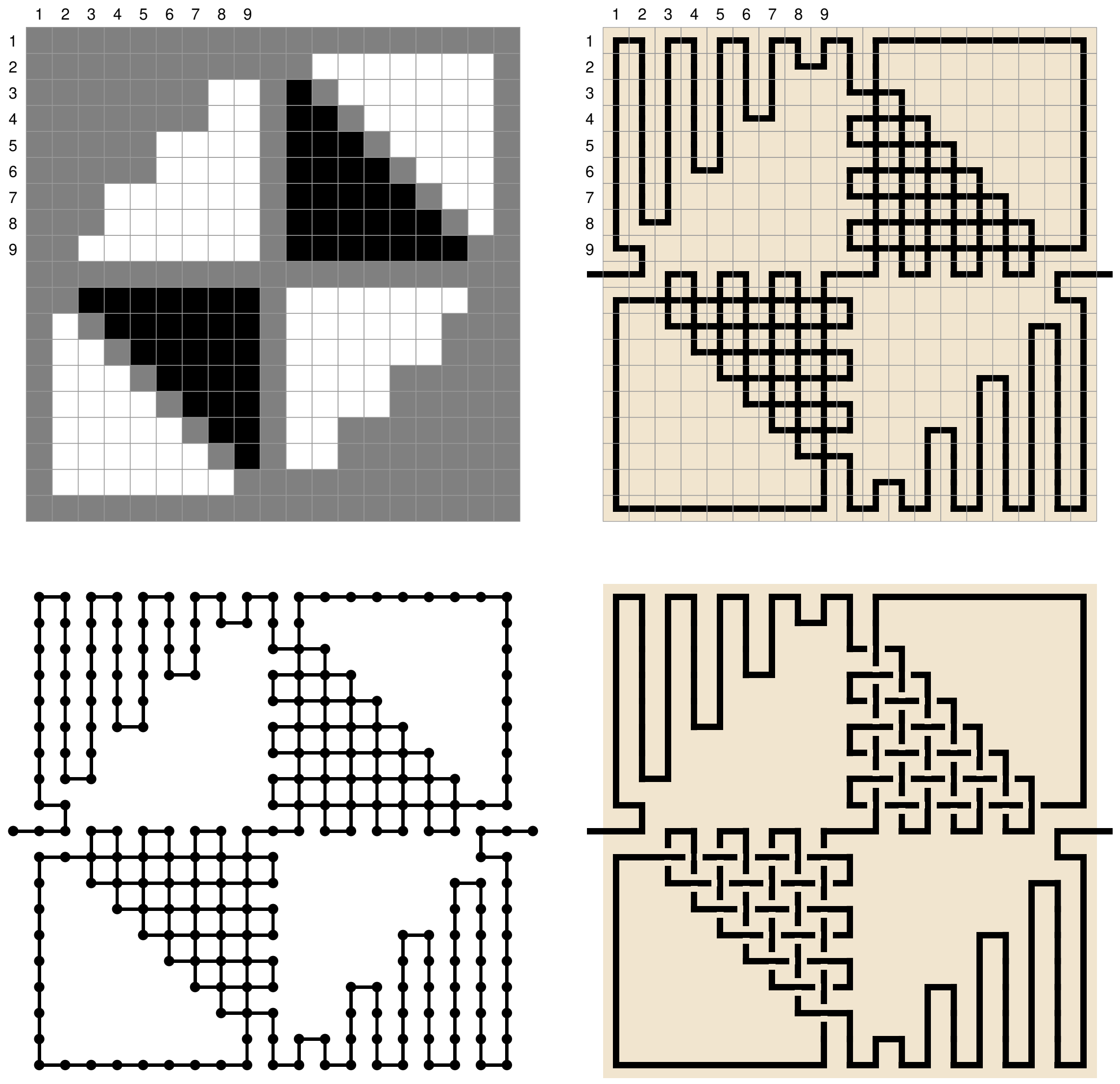

Figure 2: *(top left) A* 19×19 *target image, (top right) an easily-traced single-line rendition, (bottom left) the associated Eulerian graph, and (bottom right) an interlaced rendition of the single-line drawing.*

Consider the graph displayed in the bottom left subfigure of Figure 2. We constructed this graph by replacing each nonblank tile with a vertex and also introducing both a "start vertex" and an "end vertex." The edges connect the vertices in a way that is consistent with the way in which the street segments are connected in the map. A quick glance at this graph reveals that it is Eulerian: it is a connected graph, and all of its vertices—other than its start vertex and end vertex—have even degree. It therefore can be *traced*: one can find a route from the start vertex to the end vertex that uses each edge exactly once. In fact, there are many such routes. One of them is displayed in the bottom right subfigure of Figure 2, in which we drew the line so that it passes under and over itself. This interlaced rendition proves that both the graph and the line drawing can be *easily traced*: the driver need not spend any time worrying about where they'll turn left and where they'll turn right. Each time they come to an intersection (a copy of tile 7), they can and should drive straight through the intersection, provided that it is safe to do so. By following this simple rule, they will make sure that they will traverse each street segment exactly once.

Our inspirations for this project include Bosch's extended-Smith-tile and knot-tile mosaics [1], the "linear idea" designs of Waclaw Szpakowski [4], and the "Right-Angle Doodling Machine" of Clive Thompson [5]. In the next section of this paper, we will present the mathematical optimization models we use to create our easily-traced tile-based single-line drawings. In the final section, we will share some additional drawings.

Optimization Models for Designing Tile-Based Single-Line Drawings

All of our models use the same data, the same variables, and the same constraints. They only differ in the objective functions they use.

Data

Most of our data describes our $m \times n$ grayscale target image. We let $\beta_{i,j}$ denote the brightness value of the image's row-i-column-j pixel, using a 0-to-100 scale in which 0 denotes black, 100 denotes white, and the remaining integer values represent various shades of gray. In the target image displayed in Figure 2, each pixel is black, mid-range gray, or white. Pixel $(1,1)$ is one of the gray pixels and has brightness value $\beta_{1,1} = 50$. Pixel $(3,8)$ is one of the white pixels and has brightness value $\beta_{3,8} = 100$. Pixel $(3,11)$ is one of the black pixels and has brightness value $\beta_{3,11} = 0$.

The remaining data describes our tiles. We let b_t denote the brightness value of tile t, and we set $b_1 = b_2 = \cdots = b_6 = 50$, $b_7 = 0$, and $b_8 = 100$. (Tile 7, the "intersection tile," is the darkest. Tile 8, the blank tile, is the lightest.) In addition, we set $\textsc{t}(t) = 1$ if tile t has a line segment that touches its top edge, and we set $\textsc{t}(t) = 0$ if it does not. We similarly use $\textsc{r}(t)$, $\textsc{l}(t)$, and $\textsc{b}(t)$ to capture the edge information for tile t's right, left, and bottom edges. Note, for example, that $\textsc{t}(1) = 1$, $\textsc{r}(1) = 1$, $\textsc{l}(1) = 0$, and $\textsc{b}(1) = 0$.

Decision Variables

We use binary variables to model the placing of the tiles on an $m \times n$ board. For each position (i,j) and each tile t, we set $x_{i,j,t}$ equal to 1 if we place a copy of tile t in position (i,j) (in place of pixel (i,j)) and 0 if we don't. Here is an example of how this works: In the top right subfigure of Figure 2, we see that $x_{1,1,2} = 1$ and $x_{1,2,3} = 1$. Finally, note that the total number of $x_{i,j,t}$ decision variables is $8mn$ as there are 8 tile possibilities for each position on an $m \times n$ board.

Core Constraints

First, for each position (i,j), where $1 \le i \le m$ and $1 \le j \le n$, we impose the equation

$$\sum_t x_{i,j,t} = 1.$$

We do this to ensure that we assign each position (i,j) exactly one tile. If position (i,j) is an "interior" position (not in the top or bottom rows of the board and not in its left or right columns), then the sum is over all tiles t $(t = 1,\ldots,8)$. If not, the sum is over a reduced set of tiles. For example, for position $(1,1)$, the equation is $x_{1,1,2} + x_{1,1,8} = 1$, as tiles 2 and 8 are the only ones that can be placed in position $(1,1)$ without there being a line segment that leaves the board. Second, for each position (i,j), where $1 \le i \le m$ and $1 \le j \le n-1$, we impose the equation

$$\sum_{t:\mathrm{R}(t)=1} x_{i,j,t} = \sum_{t:\mathrm{L}(t)=1} x_{i,j+1,t}.$$

We do this to ensure that the patterns on horizontally adjacent tiles match. In other words, if we assign position (i,j) a tile that has a line segment that touches the tile's right edge, then we must make sure that we assign position $(i,j+1)$ a tile that has a line segment that touches the tile's left edge (and vice versa). For similar reasons, for each position (i, j), where $1 \leq i \leq m-1$ and $1 \leq j \leq n$, we impose the equation

$$\sum_{t:\mathrm{B}(t)=1} x_{i,j,t} = \sum_{t:\mathrm{T}(t)=1} x_{i+1,j,t}.$$

We do this to ensure that the patterns on vertically adjacent tiles match. Note that the total number of core constraints is $mn + m(n-1) + (m-1)n = 3mn - m - n$. If our target image has two-fold rotational symmetry and we want our line drawing to have this symmetry as well, we can impose additional constraints: $x_{i,j,t} = x_{m+1-i,n+1-j,t'}$, where t' stands for the number of the tile we obtain when we rotate tile t by 180°.

Auxiliary Variables

We express our objective functions in terms of auxiliary variables (variables that depend on the decision variables). For each position (i, j), where $1 \leq i \leq m$ and $1 \leq j \leq n$, we define the 1×1 residual to be

$$r_{i,j} = \beta_{i,j} - \sum_t b_t x_{i,j,t}.$$

Note that $r_{i,j}$ is simply the difference between the brightness value of pixel (i, j) and the brightness value of the tile that we place in position (i, j). We think of $r_{i,j}$ as the "1×1 error term" for pixel/position (i, j). Somewhat similarly, for each position (i, j), where $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$, we define the 2×2 residual to be

$$s_{i,j} = \left(\beta_{i,j} + \beta_{i,j+1} + \beta_{i+1,j} + \beta_{i+1,j+1}\right) - \left(\sum_t b_t x_{i,j,t} + \sum_t b_t x_{i,j+1,t} + \sum_t b_t x_{i+1,j,t} + \sum_t b_t x_{i+1,j+1,t}\right).$$

Note that $s_{i,j}$ is the difference between the sum of the brightness values of a 2×2 block of pixels—pixels (i, j), $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$—and the sum of the brightness values of the 2×2 block of tiles that end up in the corresponding 2×2 block of positions. Accordingly, we think of $s_{i,j}$ as the "2×2 error term" for the 2×2 block whose upper left corner is pixel/position (i, j).

Objective Functions

All of our objective functions take the following form:

$$w_{1\times1} \sum_{i=1}^{m} \sum_{j=1}^{n} r_{i,j}^2 + w_{2\times2} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} s_{i,j}^2,$$

where $w_{1\times1}$ and $w_{2\times2}$ are nonnegative weight parameters. If we want to minimize the sum of the squares of the 1×1 residuals, we set $w_{1\times1} = 1$ and $w_{2\times2} = 0$. If we want to minimize the sum of the squares of the 2×2 residuals, we set $w_{1\times1} = 0$ and $w_{2\times2} = 1$. For most of the artwork displayed in this paper, we used a hybrid approach, setting $w_{1\times1} = 1$ and $w_{2\times2} = 1$.

All of our objective functions are nonlinear, but they can be linearized. When $w_{2\times2} = 0$, the linearization can be done at no cost (i.e., with no additional variables). By using the first core constraint and the fact that squaring a binary variable leaves it unchanged, we obtain

$$\begin{aligned} r_{i,j}^2 &= \left(\beta_{i,j} \cdot 1 - \sum_t b_t x_{i,j,t}\right)^2 \\ &= \left(\beta_{i,j} \sum_t x_{i,j,t} - \sum_t b_t x_{i,j,t}\right)^2 \\ &= \left(\sum_t (\beta_{i,j} - b_t) x_{i,j,t}\right)^2 \\ &= \sum_t \left(\beta_{i,j} - b_t\right)^2 x_{i,j,t}, \end{aligned}$$

which is a linear function of the decision variables.

When $w_{2\times2} > 0$, linearization forces us to include additional binary variables. For each position (i, j), where $1 \le i \le m-1$ and $1 \le j \le n-1$, and each nonnegative integer k that is one of the 9 possible values for the sum of the brightness values of tiles that can be placed in a 2×2 block of positions, we introduce a binary variable $y_{i,j,k}$ that equals 1 if and only if the sum of the brightness values of the tiles placed in positions (i, j), $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ is equal to k. With these variables, we can write the square of each 2×2 residual as

$$s_{i,j}^2 = \sum_k \left(\beta_{i,j} + \beta_{i,j+1} + \beta_{i+1,j} + \beta_{i+1,j+1} - k\right)^2 y_{i,j,k},$$

which is a linear function of the $y_{i,j,k}$ variables. For each position (i, j), where $1 \le i \le m-1$ and $1 \le j \le n-1$, we need to impose the following two linear equality constraints:

$$\sum_k y_{i,j,k} = 1$$

and

$$\sum_k k y_{i,j,k} = \sum_t b_t x_{i,j,t} + \sum_t b_t x_{i,j+1,t} + \sum_t b_t x_{i+1,j,t} + \sum_t b_t x_{i+1,j+1,t}.$$

The first ensures that we assign each position (i, j) one and only one 2×2 block brightness value. The second makes sure that the values of the $y_{i,j,k}$ variables and the $x_{i,j,t}$ variables are consistent with one another. Note that the total number of $y_{i,j,k}$ variables is $9(m-1)(n-1)$, and the total number of additional constraints is $18(m-1)(n-1)$.

Loop Elimination

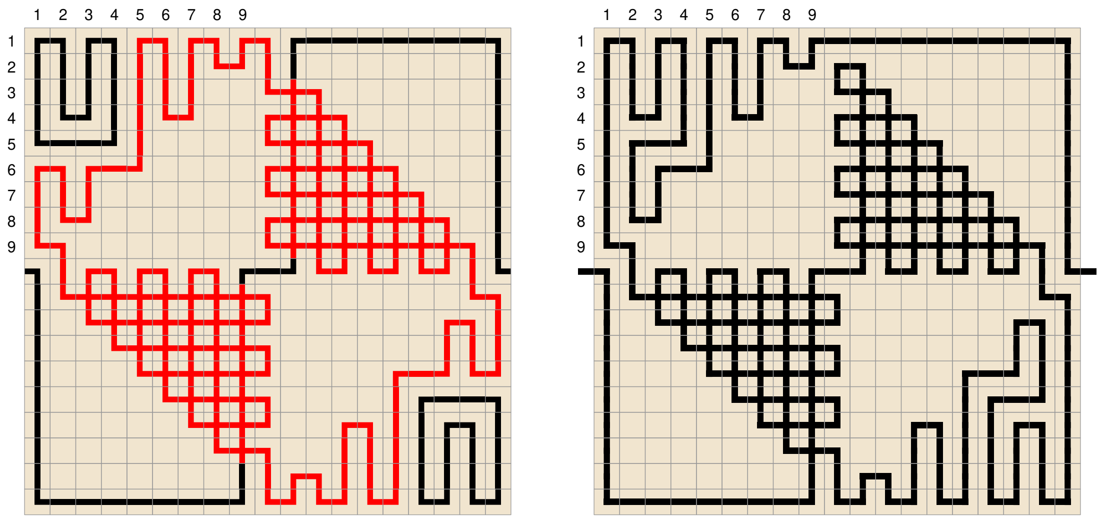

Figure 3: *(left) The solution to the first-stage problem (with just the core constraints and symmetry constraints) and (right) the nonsymmetric solution we obtained when we merged the loops.*

The lefthand subfigure of Figure 3 shows the solution we obtained on the first stage, when we used the Gurobi Optimizer [3] to minimize our objective function (with $w_{1\times1} = w_{2\times2} = 1$) subject to the core constraints and symmetry constraints. The solution is symmetric and does contain, as desired, a route from the start position to the end position, but this route does not traverse each and every street segment. Some of the untraversed segments belong to the U-shaped loop in the top left corner. Others belong to its symmetric counterpart in the bottom right corner. Even more belong to a much larger loop. All of the tiles that form the much larger loop are drawn in red.

On this instance, we can merge the route and the loops into the nonsymmetric solution shown in the righthand subfigure of Figure 3. Alternately, we can eliminate loops by imposing linear inequalities that prohibit their reoccurrence, as is done with TSP subtour elimination constraints [2]. For each loop, we identify the variables that correspond to the tiles that form the loop. Then we add up these variables and force the sum to be strictly less than the number of tiles that form the loop. For the U-shaped loop, the inequality is

$$\begin{aligned} & x_{1,1,2} + x_{1,2,3} + x_{1,3,2} + x_{1,4,3} \\ + & x_{2,1,5} + x_{2,2,5} + x_{2,3,5} + x_{2,4,5} \\ + & x_{3,1,5} + x_{3,2,5} + x_{3,3,5} + x_{3,4,5} \\ + & x_{4,1,5} + x_{4,2,1} + x_{4,3,4} + x_{4,4,5} \\ + & x_{5,1,1} + x_{5,2,6} + x_{5,3,6} + x_{5,4,4} \leq 19. \end{aligned}$$

The top right subfigure of Figure 2 shows the solution we obtained when we used Gurobi to minimize our objective function subject to the core constraints, the symmetry constraints, and the four loop-elimination constraints from the first stage. (We treated the stage-one route from the start position to the end position as a fourth loop, and we handled it in the same way that we dealt with the others.) On each stage, Gurobi required less than 0.1 seconds on a 2019 MacBook Pro to obtain the optimal solution for the stage.

Additional Examples

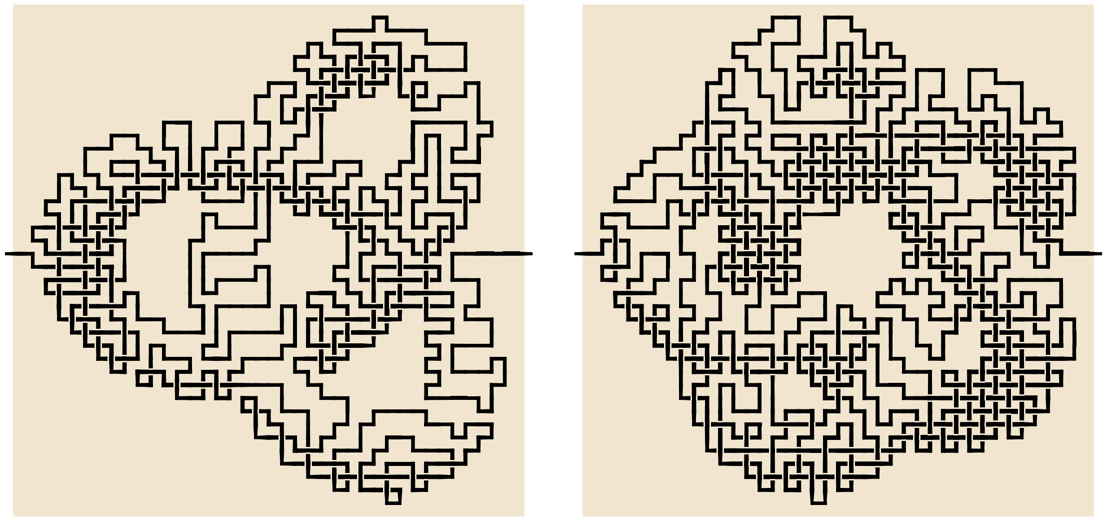

Figure 4: *Two single-line drawings that resemble knots, each drawn in knot-like fashion.*

Figure 4 contains two higher resolution (39 × 39) examples that are based on screenshots of 3D renderings of a trefoil knot and a torus knot. For these drawings (and the rest of those shown in this paper), we gave ourselves permission to vary the thickness of the line in accordance with the target image, making it thinner in brighter regions and thicker in darker regions. We did this to heighten the contrast. For these two pieces, Gurobi required considerably more time (more than a day in each case!) to find the optimal solutions.

Figure 5 displays two 49×49 single-line renditions of a section of Vermeer's *Girl with a Pearl Earring* [6]. The lefthand subfigure shows the optimal solution for the objective function with $w_{1\times 1} = 1$ and $w_{2\times 2} = 0$.

Here, Gurobi was able to obtain the optimal solution in less than a minute. The righthand subfigure shows a solution for the objective function with $w_{1\times1} = w_{2\times2} = 1$. Gurobi required several hours to obtain this solution, which might not be optimal.

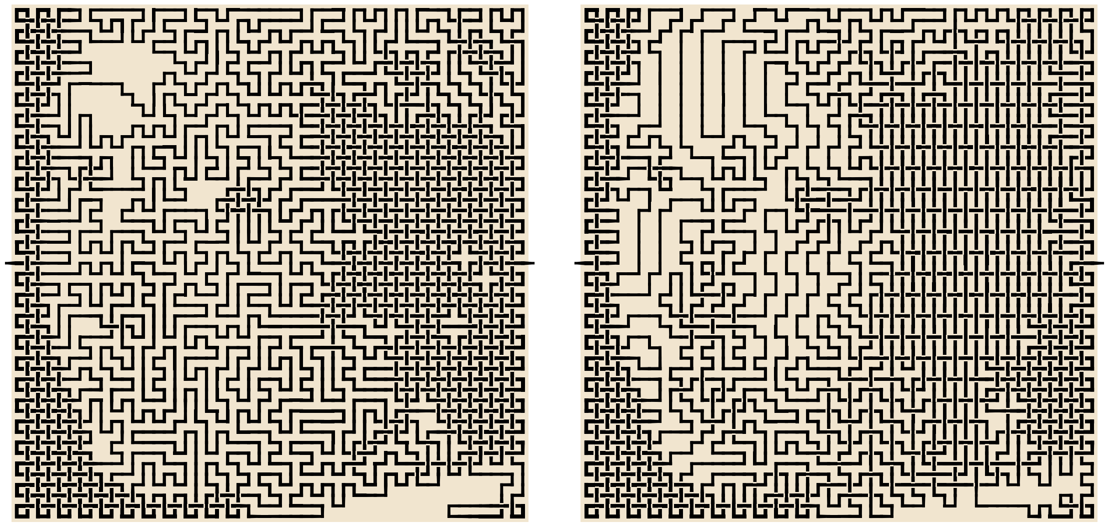

Figure 5: *Single-line renditions of a section of Vermeer's Girl with a Pearl Earring. The lefthand version uses $w_{1\times1} = 1$ and $w_{2\times2} = 0$, while the righthand version uses $w_{1\times1} = w_{2\times2} = 1$.*

The lefthand version looks posterized, as if it were made with only three levels of brightness. The righthand version looks much less posterized and appears to have additional levels of brightness. We believe that by incorporating the 2×2 error terms into the objective function, we achieve an effect that is similar to what could, in theory, be produced by error diffusion. For comparative purposes, we encourage the reader to step back and view both versions from a distance.

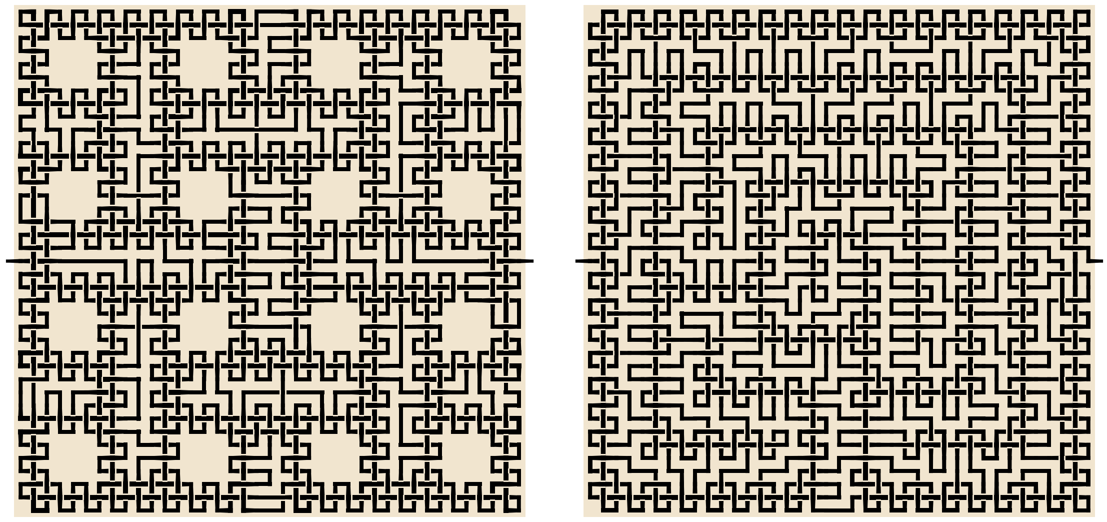

Figure 6: *(left) "Unicursal weave" and (right) "Doubly unicursal labyrinth."*

Figure 6 contains two additional 39×39 examples, and Figure 7 contains an additional 49×49 example.

Each was made in an attempt to pair this medium (tile-based single-line drawing) with a relevant message (a single-thread weave, a unicursal labyrinth, and a computationally intensive visual pun).

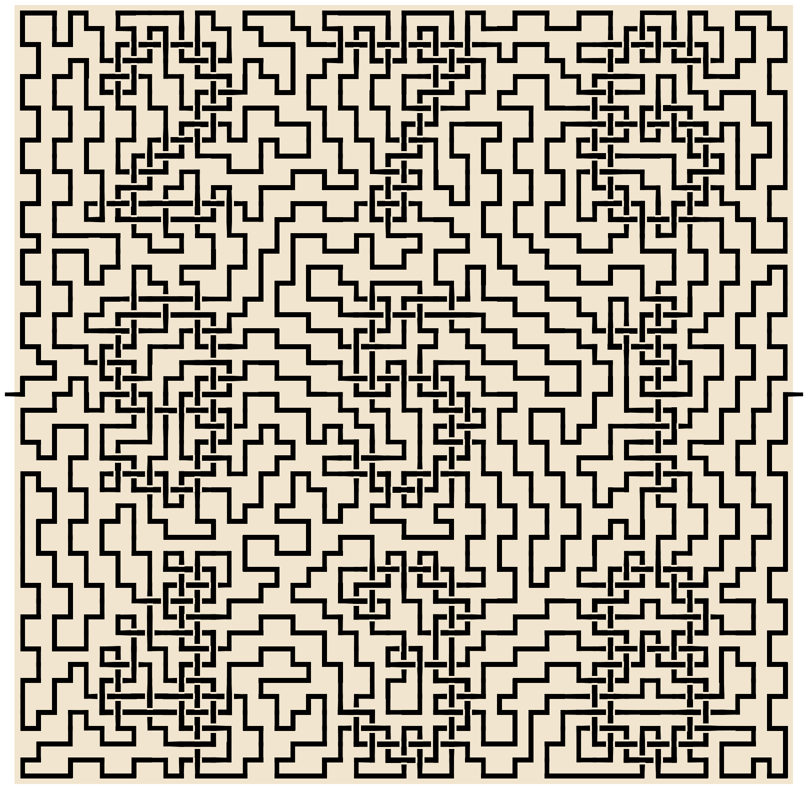

Figure 7: *“Magic square knot."*

References

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

[2] W. J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press, 2012.

[3] Gurobi optimization. http://www.gurobi.com/products/gurobi-optimizer.

[4] E. Lubowicz, ed. *Waclaw Szpakowski,* 1883-1973: *Rhythmical Lines*, Culture and Art Center in Wroclaw, Culture Institution of Lover Silesia Province Government, 2015.

[5] C. Thompson. A machine for helping you doodle. https://betterhumans.pub/a-machine-for-helping-you-doodle-ce1c38529f2b.

[6] J. Vermeer. *Girl with a Pearl Earring*, 1665. Oil on canvas. Mauritshaus, Den Haag.