Drawing with Statistics

Douglas Dwyer

New York City, USA; douglas.dwyer@me.com

Abstract

In this paper, we develop a statistical approach to create a mathematical representation of a line drawing or a silhouette. The representation allows for the imagery to be manipulated across time and space using a mathematical programming language. The approach uses a digital pencil to create a bitmap of a drawing and then uses the statistical programming language called R. We present two applications. The first, uses the representation to make a tessellation in which a silhouette is transformed into the negative space of the silhouette. The second makes a digital representation of an etching by Matisse. The R code is publicly available.

Introduction

We have always been fascinated with line drawings and silhouettes. How could something two-dimensional convey a sense of emotion and personality? Also, a line drawing can convey the illusion of depth. Further, a two-dimensional drawing can be mapped to a 3D surface and when successful it can convey a sense of the shape of the surface—the third dimension. One can also suggest a transformation through time with a sequence of silhouettes (e.g., a bird in flight or a person running).

We introduce an approach to creating a *connect-the-dots* representation of a line drawing or a silhouette. By "connect-the-dots," we are referring to an exercise one may find in a child's coloring book where a set of dots are numbered. When the child draws line segments connecting each consecutive point a recognizable drawing emerges. This approach allows for the drawing to be easily transformed in space and time using a programming language. We will show one possible application of the approach—we will transform a silhouette into the negative space around the silhouette, which results in the silhouette becoming the negative space (see Figure 1). The approach is shown to be useful in making a digital representation of a traditional work of art—a Matisse etching.

Alternative Approaches

There are many approaches to creating a line drawing or a silhouette from a photograph in a digital environment and which approach works better than others depends on what one intends to do with the final result. We enjoy

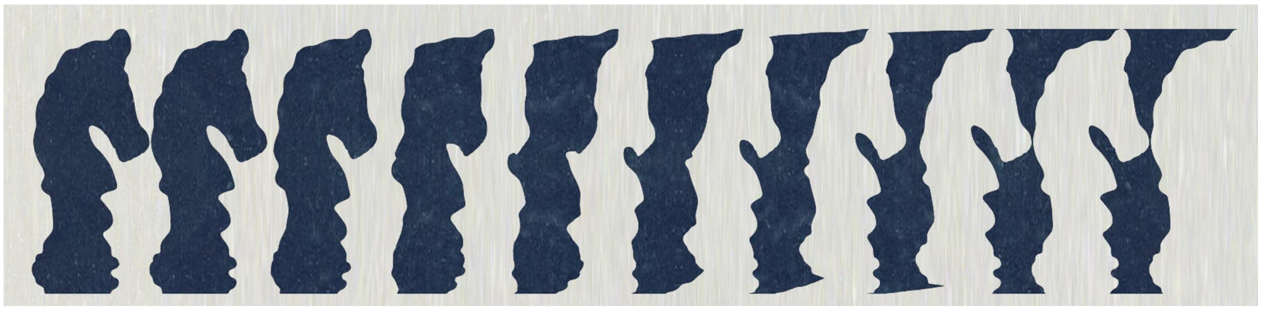

Figure 1: *The silhouette transforms into the negative space of the same silhouette.*

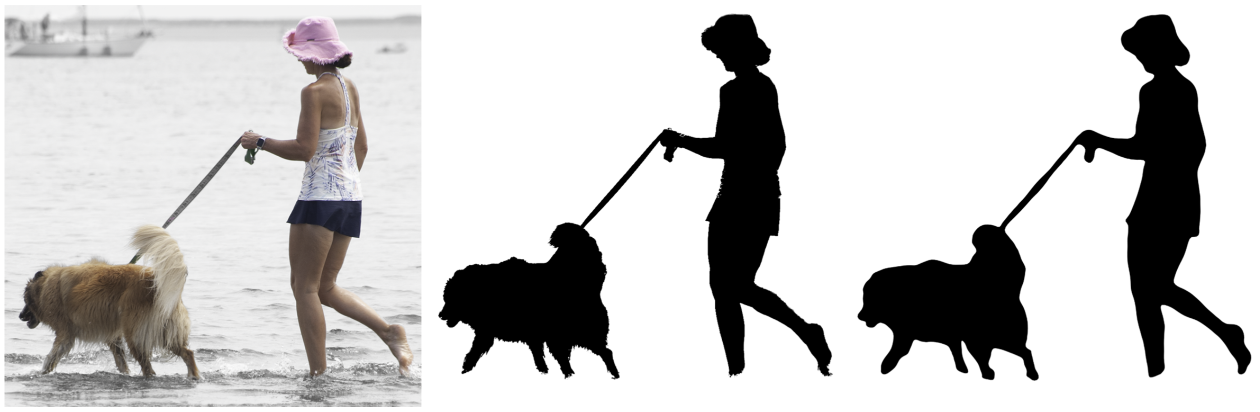

Figure 2: *A photograph and two silhouettes*

drawing with a digital pencil in a raster-based environment because the technology successfully emulates the experience of drawing with pencil and paper without the frustration of erasing an actual pencil mark from actual paper. Further, we like the artistic control it provides. Finally, the result can be transformed into a *connect-the-dots* format which allows one to transform the dots with mathematics using a programming language.

A core task in digital photography is selecting the subject. The selection allows one to adjust the exposure or color of the subject relative to the background and so forth. Overtime, software has gotten much better at the "selecting subject" task. The desired outcome of a "select subject" process depends on what the outcome will be used for. In the photograph on the left-side of Figure 2, we used the select subject tool in Photoshop (v24.4.1 in May 2023) to select both the women and the dog. We inverted the selection and then removed the saturation from the selected pixels. The result is that the background is black and white, and the subject is in color drawing attention to the subject. For this purpose, the Photoshop one-click selection worked very well. We created the center silhouette with an iPhone (iOS16 in May 2023) using the new "Lift Subject" from a photograph feature. This feature identifies the subject and sets the other pixels to transparent. It works very well for creating an image of a subject on a blank or alternative background. We created the silhouette by setting all the selected pixels to black and placing on a white background. We produced the second silhouette on the right using the techniques described in this paper. There are artistic differences. The one produced using "lift subject" has some jagged edges reflecting the fur of the dog and the folds of the women's clothing, whereas we make the artistic choice to smooth over these details. Further, her foot appears more natural in our version. Which approach one prefers is a question of objectives and taste. We like the artistic control that drawing with a digital pencil provides.

One can achieve high quality line drawings or silhouettes using a pen tool in a vector-based environment such as Adobe Illustrator. The illustrations can be saved in Scalable Vector Graphics (.svg) format (cf., [6]). This industry standard format stores an image as a set of mathematical shapes rather than a grid of pixels. Presumably, one could write a program to extract the parameters of each shape from the .svg file to create a *connect-the-dots* representation and then transform these shapes mathematically.

We have created a digital representation of Picasso's line drawing of a dove by *locating* key points on the drawing and then fitting cubic splines through the coordinates of the key points to capture each line [1]. In R, there is a locator function that will locate the coordinates of mouse clicks in the order of the clicks. With cubic splines, one can convert the relatively small number of located points into many points for each line. This approach works provided the points are *located* carefully. We find drawing and erasing with a digital pencil to be more enjoyable and quicker in achieving a pleasing result.

Why R?

The approach utilizes the open-source statistical programming language called R. As R is known for data visualization and we use it professionally, it was a logical choice. R works for this project, because it has a well-developed set of functions for cluster analysis and image processing. Cluster analysis is an unsupervised statistical learning technique with many applications. In marketing, for example, it is often used to group customers into clusters with common characteristics [5]. To do image processing, we are actually using the open-source cross platform software library *ImageMagick*, which has been developed over many years. We use this library to process photographs in R through the R package *magick* [8]. R also has many other packages that are potentially useful. For example, we have used a travelling salesperson problem package called TSP. Further, R also has a locator function that records the location of mouse clicks, which we have used as an alternative approach. Finally, R is well established and the codes developed for this paper should be easy to work with for decades to come. Another language widely used in data science is Python. Python is actively used for image processing and image recognition tasks. Presumably, the tasks accomplished in this paper could be done in Python as well. Python would be another logical alternative.

To produce the drawings, we use an Apple pencil on an iPad using Procreate, which is a raster-based app intended for digital painting.

The Approach

The first step is to start with an image that one could trace with either a graphite pencil using tracing paper or a digital pencil with a digital art program. We use the latter.; we have also applied this approach to an etching done by Matisse. We will start with our photograph of a teapot. The teapot is a nontrivial example of the problem, because the handle produces a *hole* and the spout has significant curvature. The left-hand image of Figure 3 is the photograph, and the center image is our drawing of the photograph.

To work with a line drawing using statistical algorithms, we want a table that lists the pixels that are marked by the pencil and the locations of the pixels (the row and the column). An analogy to the traveling salesperson problem (TSP) is useful. To solve a "TSP" one needs a table of the cities to be visited and their locations (e.g., their latitude and longitude).

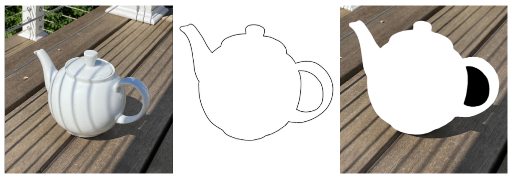
Figure 3: *Three versions of a teapot*

The center image of Figure 3 is a rendering of a bitmap in which each pixel has a location and a color. The colors are represented using an RGB color model in which pure black is a vector of three elements with each being zero, and pure white is represented as a vector of three elements with each is 255. Pure red is achieved by setting the first element to 255 and the other two to zero. This color space is referred to as 24-bit color as it allows 2^{24} distinct colors. In the bitmap at hand, the background is pure white and the pixels that are *drawn* are either close to black or a shade of grey if the pixel is on the edge of the lines. We eliminate the background pixels by dropping all the pixels whose three elements in the RGB vector sum to greater than 300 (pure white would sum to 765). We thereby convert a grid of a few million pixels into a table of 8,101 points with x and y coordinates. We plot each point in the left-hand panel of Figure 4. As the zoom (center panel of Figure 4) reveals, this is not yet a mathematically convenient form to work with as there are too many dots due to the thickness of the line. At this stage, if one *connects-the-dots* one actually recovers a drawing that covers not only the teapot, but also the hole formed by the handle of the teapot and the space between the spout and body of the teapot (right-hand panel in Figure 4). This happens because when we converted the observations from an array to a table the observations became sorted by the row of the pixel first and then the column of the pixel second. Dots sorted in this way result in numerous *horizontal lines* when they are connected.

One approach to reduce the number of points is to use a statistical learning technique called *K-Means Clustering*. For intuition as to what this technique does consider the following. Suppose you a had a list of 8,101 people with the latitude and longitude of their addresses and they each lived in one of 300 cities. If you used cluster analysis to group these people into 300 clusters based on their latitude and longitude, the latitude and longitude of each cluster would be the coordinates of each city. What this approach does is find the coordinates of a specified number of clusters such that: when each observation is assigned to a cluster and the distance to the center of the cluster is computed, the sum of the squares of the distances of the observations from their respective cluster is minimized. We can use this technique to extract the coordinates of the clusters from the procedure and reduce 8,101 observations into 300 points (in this example). The left-hand panel of Figure 5 shows the new set of dots. The right-hand panel shows a *connect-the-dots* version of these points, which results in a jarring configuration of jagged lines. The clusters are not yet in the correct order for a connect-the-dots representation.

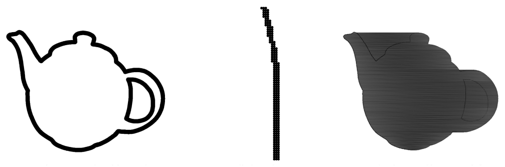
Figure 4: *Conversion of a bitmap into a set of points*

There are many ways to order 300 dots. One way well-studied approach is TSP, where the objective is to find a path that minimizes the distance travelled while visiting every city on a list. Our objective is not identical, but it is closely related. Our objective is to find the ordering that yields a pleasing drawing. Further, for some drawings, we will want multiple lines—the virtual equivalent to where the artist picked up the pencil to make a new line. If 300 dots do not yield a pleasing result, in our context, we can step backward and see if we can improve the result by using a different number of clusters in the previous step. In the TSP problem, in contrast, the number of cities is fixed.

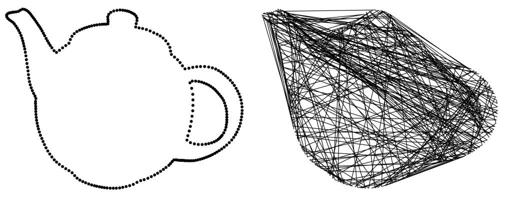
Figure 5: *Clustering to reduce the number of dots*

An example is useful for demonstrating the difference between a TSP solution and an artistic solution. Suppose you have a set of cities (or dots) evenly spread around a circle and another set of cities evenly spread around a slightly smaller concentric circle. The artist may want to loop around the circles twice drawing one circle first and the other circle second (left-side drawing of Figure 6). Depending on the density of the cities and the distance between the inner and outer circle, the solution to TSP may minimize distance traveled by going around the two circles once and jumping back and forth between to the two circles (right-side drawing of Figure 6).

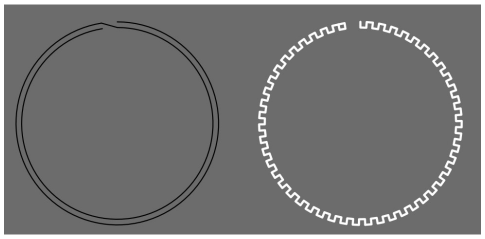
Figure 6: *Artistic solution (left side) and TSP solution (right side)*

Our approach starts by choosing the *next point* such that it is closest to the previous point and has not already been used. We then choose the starting point such that of all the possible starting points (there are 300) the resulting path is the shortest. These steps are the "nearest neighbor algorithm" and the "repetitive nearest neighbor algorithm" that are useful in finding a good solution to the TSP [4].

The left-hand panel in Figure 7 shows the result from choosing a poor starting point (the red dot). When the line returns to the red dot it makes a long jump to the next set of points which form the *hole* inside of the handle. The right-hand image in the Figure 7 is the result from choosing the starting point to minimize the resulting distance traveled given the constraint of always moving to the next closest point. This results in a small jump between the outer shape and the inner shape that is consistent with the size of the handle. This represents a good solution to the TSP problem but it need not be the best from the perspective of minimizing distance traveled. This good solution works well for our purposes.

We break the curves where the distance between two consecutive points is the greatest. This results in the final representation: two sets of points, one for each curve, where the points are ordered to give a pleasing line drawing (left-hand panel of Figure 8). Using two polygons, one that is black for the outer shape and one that is white for the *hole*, we can create a silhouette of the teapot (right-hand panel of Figure 8) using the polygon function in R. In the right-hand image of Figure 4, we drew the silhouette on top of the photograph to see if the approach worked.

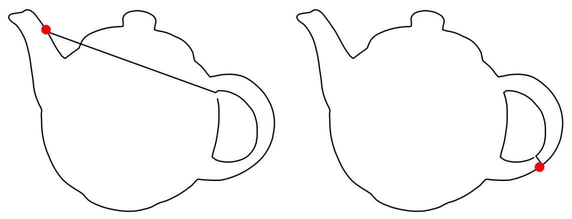
Figure 7: *Connect the dots using the next closest dot*

For future use, we can store this representation in comma-separated values (.csv) file format with three columns—two columns for the *x* and *y* coordinates and one column for a number indicating the line that the point belongs too. In fact, we can create a four-column data file for many drawings, where the fourth column identifies the drawing. The row number can be used to ensure that the points are in the correct order.

We have applied this technique across multiple drawings. We will typically choose the number of clusters to be consistent with the complexity of the drawing. Too many clusters on a simple shape can result in a somewhat jagged line even though the line is smooth in the underlying drawing. With too few clusters, the line can jump from one edge to the next in an undesired fashion. For example, we have one silhouette of a woman walking a dog on a leash. For this silhouette, we used 1,000 clusters to cleanly capture both sides of the leash, as well as the woman and the dog, with a single curve (the right-hand image in Figure 2). We also choose the number of breaks to match the number of lines in the drawing. We find that this approach works across a wide variety of drawings in that it yields a representation that is *what would have been drawn*. Nevertheless, manually reordering of some of the dots is required in complex drawings.

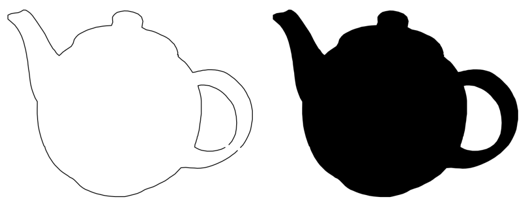
Figure 8: *Final representation as a line drawing and a silhouette*

First Application

As an application, we will transform a silhouette into the negative space surrounding the silhouette as a sequence in which the silhouette emerges as the negative space (e.g., Figure 1). We start by choosing a silhouette that is simple, organic and easily recognizable. We then need to construct the negative space of the silhouette. Finally, we will need to map each point of the negative space of the silhouette to the positive space of the silhouette such that the one can be transformed into the other in a pleasing fashion.

We construct the negative space of silhouette in a few steps. We find the "top", "bottom", "left-most", "right-most" values of the drawing by finding minimum and maximum values of the *x* and *y* coordinates across all the points in the drawing. From these values we can compute the four corners of the rectangle that encloses the drawing. The outline of this rectangle is drawn in light blue in Figure 9. We define the left-hand negative shape as the polygon formed by the left-hand corners of the enclosing rectangle and the left-hand side of the drawing (the points between the top and the bottom on the left-side of the drawing).

Likewise for the right-hand negative shape. We then shift the left-hand negative shape to the right by the width of the enclosing rectangle. The negative space silhouette is the union of the right-hand negative shape and the shifted left-hand negative shape. Care needs to be taken to order the points correctly. We now have a pair of shapes that "tessellate" in the sense that one could fill an infinitely large table with cut outs of both the silhouette and negative space silhouette as seen in the Figure 9. The pair of shapes form a tessellation because we have used translation to exchange the left and right-hand sides of the original silhouette, which is a basic technique for constructing a tessellation (c.f., Chapter 2 of [3] and [9]).

We wish to smoothly transform the black shape into the white shape. We can do this by taking the weighted average of the two shapes and progressively putting more weight on the white shape. Before proceeding, however, we first need a "one-to-one" mapping of the points in the black shape and the points in the white shape. Further, for the transformation to be pleasing, the points need to align well. We accomplish this by creating new versions of each shape in which each point on the corresponding side of each shape aligns horizontally. In Figure 9, the points of each shape are numbered starting in the lower left and moving around the figure counterclockwise. Note that the points 1 to 1,000 align the right-side of the silhouette with the right-side of the negative space silhouette and points 1,000 to 2,000 align the left-side of the silhouette with the left-side of the negative space silhouette. This alignment is accomplished by creating 4 functions that take a value for the *y* coordinate as an input and returns the *x* and *y* coordinates for the left and right-side of each figure (the *y* coordinate that is returned by these functions is the *y* value that was used as an input to the functions.). These functions are constructed by ordering the original set of points correctly and using linear interpolation to compute the values between the points. We then use these functions to map 1,000 evenly spaced values of *y* to each side of each shape to build up new versions of each figure.

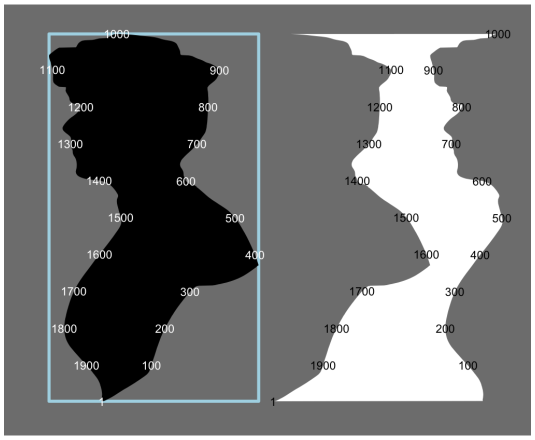

Figure 9: *The positive and negative*

The top row of Figure 10 is the *positive shape* colored white and repeated 10 times and each time shifted to the right by the width of the enclosing rectangle. The bottom row of the figure is the same thing except it uses the *negative shape*. The positive shape emerges from between the negative shapes in dark grey. In the middle row, the *positive shape* is gradually transformed into the *negative shape* and as a result the *positive shape* emerges from the grey background as we move to the right. It aligns with the top row on the left and the bottom row on the right.

This approach does not generalize to all silhouettes. In some silhouettes, when moving counterclockwise around the right-side (left-side) the *y* coordinate may not be monotonically increasing (decreasing). Consequently, other approaches will be needed to align the points. For example, Figure 1 presents a transformation of a chess piece (a knight) into its negative space. For this example, a horizontal line can be placed so that it intersects the outline of the silhouette four times. Therefore, a function that maps the *y* coordinate of right-hand side to the *x* coordinate does not exist for the right-hand side of the chess piece. As an alternative, we created 1,000 points on each side of the silhouette that are evenly spaced according to the Euclidean distance between each point. We do this by computing the total distance traveled by moving up the left-side of the silhouette and using this as the input to a function that takes distance traveled as an input and returns the *x* and *y* coordinate for the left side of the silhouette. The function uses linear interpolation to compute the values that are between points. With such a function, we can compute the exact locations of 1,000 equally spaced points. We repeat for the right-hand side and for both silhouettes (the knight and the negative space

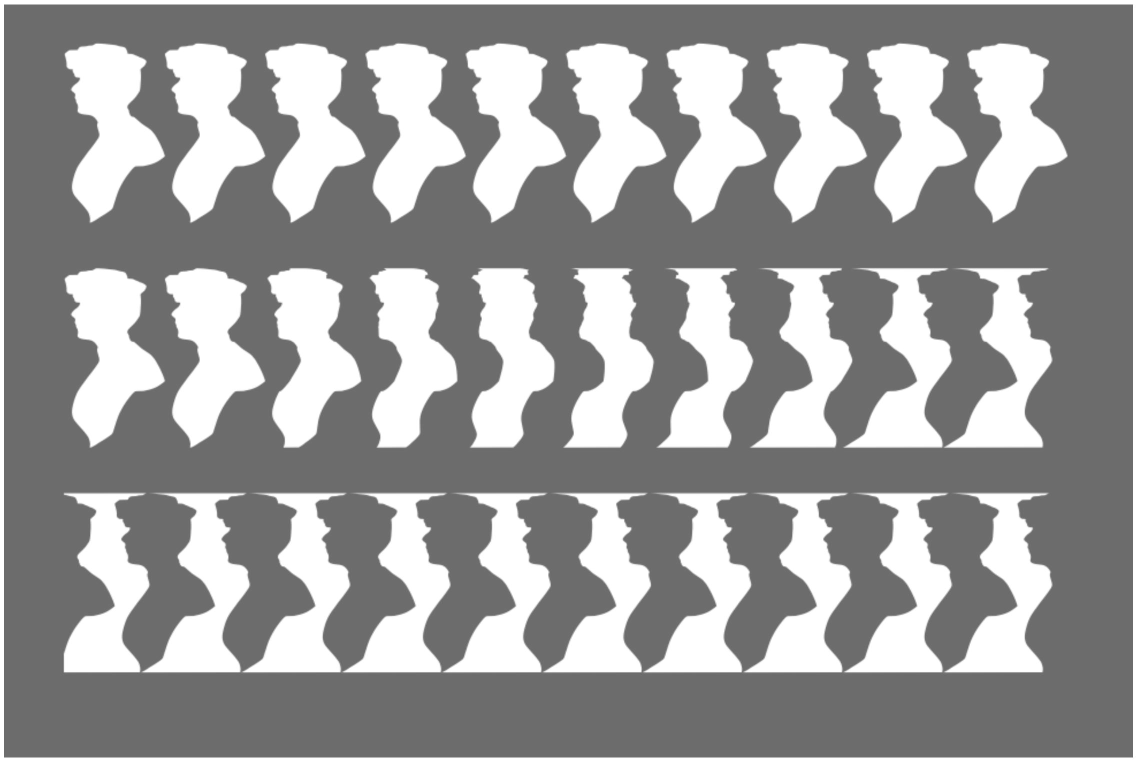

Figure 10: *Positive becomes negative*

of the knight). From these points, we can build up a 1-1 mapping to get the pleasing transformation.

MC Escher would do things very similar to this, but not identical. In *Sky and Water 1* (reproduced in [9]), in the middle there is a tessellation formed by a motif of a bird (in black) and a fish in (in white). As you move up the fish disappears and becomes the negative space of the bird. The bird becomes more detailed and realistic. As you move down, the bird disappears and becomes the negative space of the fish. The fish becomes more detailed and realistic. In Regular Division of the Plane Drawing #80 (reproduced in [3]), he shows a 12 step process that starts with a grid of parallelograms and uses a couple of translations to produce black and white versions of the same bird that tessellates. What I am doing here is inspired by that work, but I am not aware of him (or others) transforming an organic shape into the negative of the shape. I have not seen it.

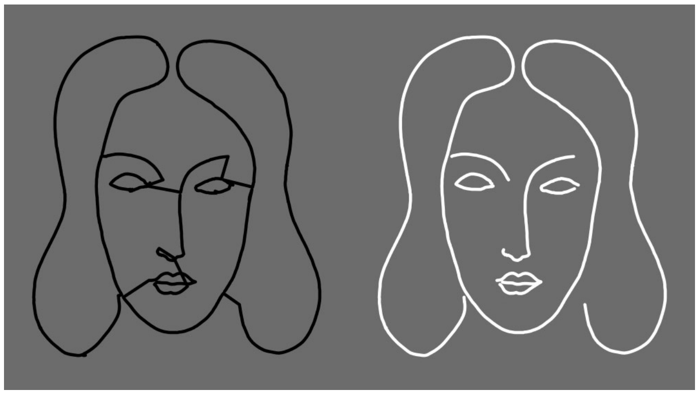

Figure 11: *Derived from a Matisse Etching.*

Second Application

Figure 11 is based on a 1932 Etching by Matisse [7], in which one long fluid curve represents both the outline of the hair of the model and the shape of her face. The left-side image is the direct output of the algorithm with 1,000 clusters and two breaks. We produced the one on the right by "cutting and splicing" the ordering of the points on the left into eight curves. The large outer curve on the right is actually formed by concatenating two "cuts" of the ordering of the points on the left. This figure demonstrates that the method can be applied to line drawings produced using traditional techniques (without the add of a computer), but at the same time the approach is not a fully automated for more complex drawings.

Conclusion

In this paper, we have shown how to create a mathematical representation of a silhouette or a line drawing created either digitally or using a photograph of traditional art. This representation allows one to transform the silhouette in space and in time using mathematical programming. We present a novel application in which a silhouette is transformed into the negative space of the silhouette. We also show the approach can be used to create a connect-dots representation of a historical line drawing executed as an etching. A gallery supplement provides additional examples, with links to animations. These applications include mapping silhouettes onto a sphere, a cylinder and a sculpture as well as transforming one silhouette into another silhouette. There are also animations of silhouettes of rotating sculptures that are done with a similar process and build upon our 2022 Bridges paper [1].

Future work could include moving back into the physical world by creating cutouts using a laser cutter. If we made versions of Figures 1 or 10 where the dark colors were cut out of a light-colored material, we could photograph the board in different light conditions for different effects. If we front light the board, the cutouts would appear as dark holes. Conversely, if we back light the board, the cutouts would become light. We could use the board to create shadows on flat or curved surfaces. We could use 3D printing to wrap these figures around a cylinder that could be illuminated from within and spun around its axes.

The codes used to generate the key figures in this paper are available on a github [2].

References

[1] D. Dwyer. "Spiral Based Point Cloud Representations of Sculptures." *Bridges Conference Proceedings*, Helsinki and Espoo, Finland, Aug. 1-5, 2022, pp. 213–220. http://archive.bridgesmathart.org/2022/bridges2022-213.html.

[2] D. Dwyer. "Bridges 2023 Repo." 2023. https://github.com/dashdotdotdashdotdot/Bridges2023Repo.git.

[3] M.C. Escher. *Escher on Escher: Exploring the Infinite*. Abrams, 1989.

[4] M. Hahsler and K. Hornik. "TSP—Infrastructure for the Traveling Salesperson Problem." *Journal of Statistical Software*, vol. 23. no 2, 2007.

[5] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Springer, 2008.

[6] C. Johnson and I. McCormack. "Computational Making via Bidirectional Parametric Modeling." *Bridges Conference Proceedings,* Online, Aug 1-3, 2021, pp. 359–362. http://archive.bridgesmathart.org/2021/bridges2021-359.html.

[7] H. Matisse, "Tailpiece." *Poésies (Poems) by S. Mallarmé*. p. 67, Museum of Modern Art, New York, NY, 1932. https://www.moma.org/collection/works/28264?association=illustratedbooks&page=1&parent_id=28260&sov_referrer=association

[8] J. Ooms, "Package 'magick': Advanced Graphics and Image-Processing in R." March 9, 2023. https://cran.r-project.org/web/packages/magick/magick.pdf

[9] D. Schattschneider, "The Mathematical Side of M.C. Escher." *Notices of the American Mathematical Society*, vol. 57. no 6, 2010, pp. 706-718.