Curve Stitching Density Plots

John Nicholson

Austin Peay State University, Clarksville, TN, USA; nicholsonja@apsu.edu

Abstract

Curve stitching is a mathematical art style in which envelopes of straight lines create emergent curves. With little effort, curve stitching can be implemented in computer programs that allow for easy manipulation of parameters. The flexibility and power of computer programs reveal color gradations that are interesting in their own right. This paper discusses an algorithm that does not render curve stitching patterns as collections of line segments, but as density plots. These density plots allow for the exploration of the color gradations and introduce a new approach to rendering curve stitching images.

Basic Curve Stitching

Curve stitching was created by Mary Everest Boole in the 1800s [2] when she took cards meant for painting and, instead of painting them, perforated the edges of the images with sewing needles, drawing threads through the holes to create lines. She eventually discovered patterns that allowed the intersecting threads to create "a symmetrical curve made up of a tiny bit of each of my straight silk lines" [1]. People still create curve stitching works as part of math education programs, as a hobby, and as art. The works are created using Mary Boole's original method, as well as hand-drawn and computer generated images.

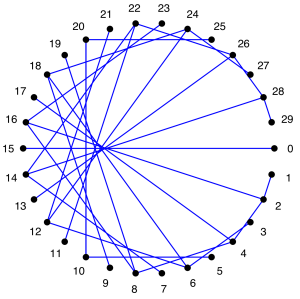


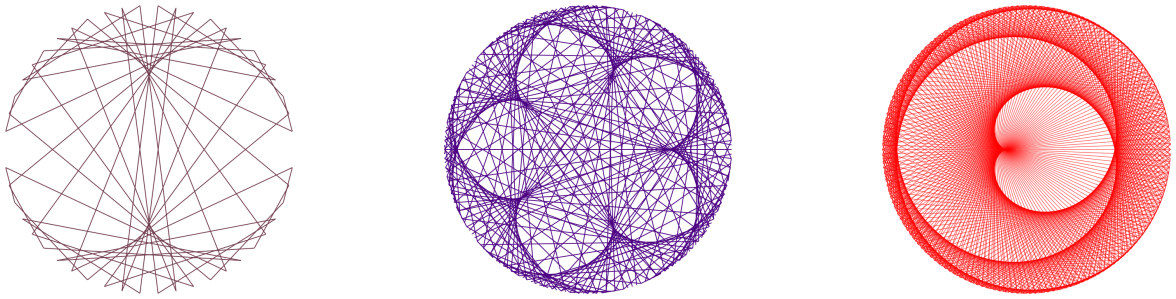
Figure 1: $N = 30$, $k = 2$, *with numbered points.*

1: **for** $i \in [0..N-1]$ **do**
2: $\quad\theta \leftarrow i * \frac{2\pi}{N}$
3: $\quad p_A \leftarrow g(\theta)$
4: $\quad p_B \leftarrow g(f(\theta))$
5: $\quad$ Draw line segment with end points $\{p_A, p_B\}$
6: **end for**

Figure 2: *Basic curve stitching algorithm*

The curve stitching process can generally be thought of as defining a set of points P along a set of one or more paths. A mapping function, $f(i)$, is then used to map every point $p_i \in P$ to $p_{f(i)}$, where $p_{f(i)} \in P$ as well. Each pair of points, $\{p_i, p_{f(i)}\}$, then acts as end points on a line segment. When the line segments are drawn, their envelopes create emergent curves and patterns. In this paper, the paths are limited to closed curves, for example, circles, rose curves, epicycloids, etc.

A common definition for the mapping function is $f(i) = (k * i) \bmod N$, where N is the number of points in P, and k is some integer value such as 2, 3, etc. Using this definition, Figure 1 illustrates the result for a circle when $N = 30$ and $k = 2$. When implementing curve stitching algorithms, it is convenient to use the curve's parametric equations, $g(\theta)$, which can calculate point p_i from angle θ_i. In that case, $f(i)$ is transformed into $f(\theta) = k * \theta$. Changing the approach to parametric equations does not change the final image, and so most programs implement a variation of the basic curve stitching algorithm shown in Figure 2.



(a) $N = 50,\ k = 3$ **(b)** $N = 100,\ k = 3.5$ **(c)** $N = 150,\ k = .75$

Figure 3: *Effects of N and k when using the basic algorithm.*

When implemented in a computer program, the basic algorithm enables quick and easy manipulation of N and k. It easily allows values of k, which tend to be low integer values when curve stitching works are handmade, to be any real number, integer or floating point. Figure 3 illustrates three of the many variations that are possible when the basic algorithm is applied to a circle.

Density Plots

As the basic algorithm is given increasingly higher values for N as in Figure 3(c), the different densities of the lines throughout the image begin to cause color gradations. In areas where there are high numbers of line intersections and the line segments are close, the overall color of those areas is darker and more intense. In the areas where fewer intersections occur and the lines are further apart, the overall color of those areas is lighter and less intense. As I explored progressively more complex images with the basic algorithm, I became more interested in the potential of the color gradations and then worked to develop an algorithm that emphasized the gradations without displaying any visible lines.

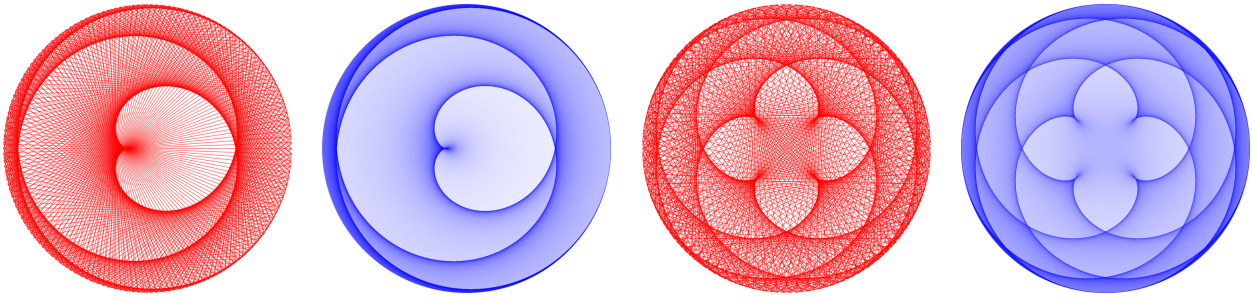
1:	Create matrix $M[W, H]$	▷ Same dimensions, width(W) x height(H), as final image
2:	Initialize all entries in M to 0	
3:	**for** as many samples, S, as desired **do**	
4:	Randomly choose θ	
5:	$p_A \leftarrow g(\theta)$	
6:	$p_B \leftarrow g(f(\theta))$	
7:	Randomly choose p_C on line segment $\{p_A, p_B\}$	▷ p_c's coordinates, (x_c, y_c), are real values
8:	$i \leftarrow \lfloor x_c \rfloor$, $j \leftarrow \lfloor y_c \rfloor$	▷ (i, j) are integer indices of M
9:	Increment $M[i, j]$ by 1	
10:	**end for**	
11:	Convert counts in M to pixel colors in final image	

Figure 4: *Curve stitching density plot algorithm*

The resulting curve stitching density plot algorithm, which is outlined in Figure 4, is a sampling algorithm. Unlike the basic algorithm, which draws a fixed number of N line segments, the density plot algorithm computes S random line segments from among the infinitely many possible line segments that generate the emergent curves. From each of these samples, it chooses a random point, whose location is used to increment a density count in the matrix M. After generating the desired number of samples, the matrix

will contain counts that represent the densities of lines at each location in the final image.

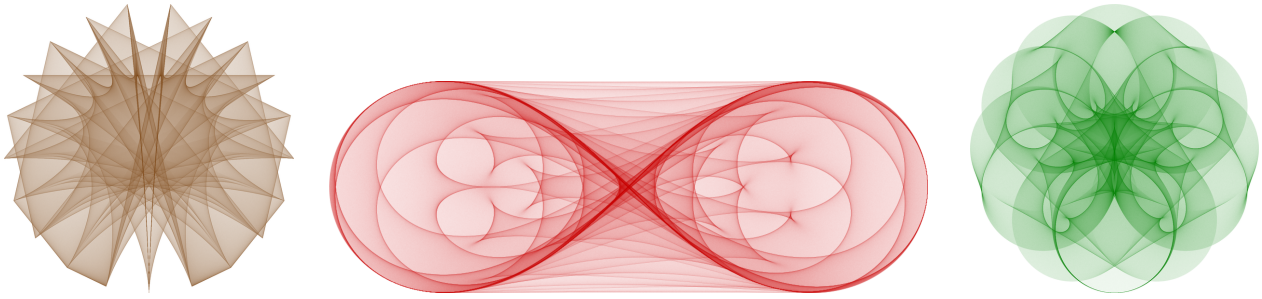
The algorithm's runtime is linear and is dependent on the number of randomly chosen line segments, S. Lower values of S result in grainier images, while higher values of S result in smoother gradients but take longer to render. Larger images require higher values of S than smaller images in order to produce equivalent results. The images in this paper were originally created on a MacBook Pro at 1200x1200 dpi with S = 400,000,000, taking approximately 45 seconds to render each image. The algorithm is easily parallelizable, and improved implementations of the algorithm now render the images in 15 or fewer seconds.



(a) $N = 150,\ k = .75$ **(b)** $k = .75$ **(c)** $N = 150,\ k = 1.8$ **(d)** $k = 1.8$

Figure 5: *Algorithm comparison. Figures (a) and (c) created with original curve stitching algorithm. Figures (b) and (d) created with the new density plot algorithm.*

Figure 5 compares the results of the two algorithms. Even with fine lines, the basic algorithm's line segments are a visible part of the image, especially when viewed up close at high resolution. In the images created with the density plot algorithm, the same emergent curves are still present, however, all traces of the individual line segments have been replaced with smooth areas of colors and gradients. Figure 6 shows additional example renderings created using the density plot algorithm. As the examples show, the algorithm, and curve stitching in general, is not restricted to generating images solely from circles. The examples illustrate how the results can have delicate, wispy textures that would not be as apparent if the line segments had been drawn. The images also resemble the caustic created in a coffee cup by rays of light [3].

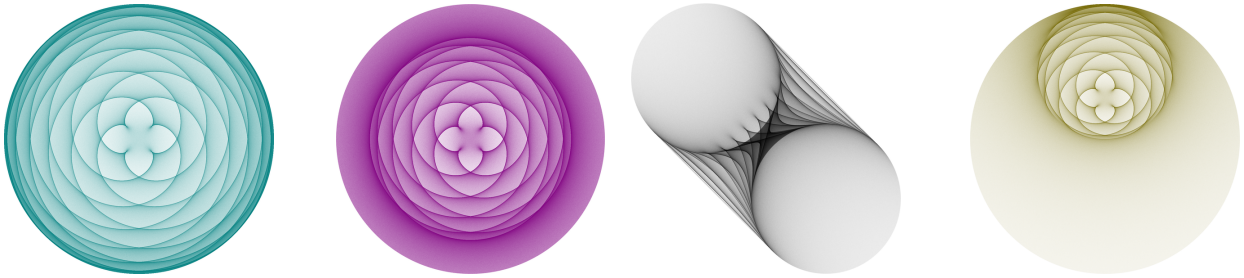


(a) *Hypocycloid* **(b)** *Lemniscate* **(c)** *Rose curve*

Figure 6: *Example results for curves other than circles.*

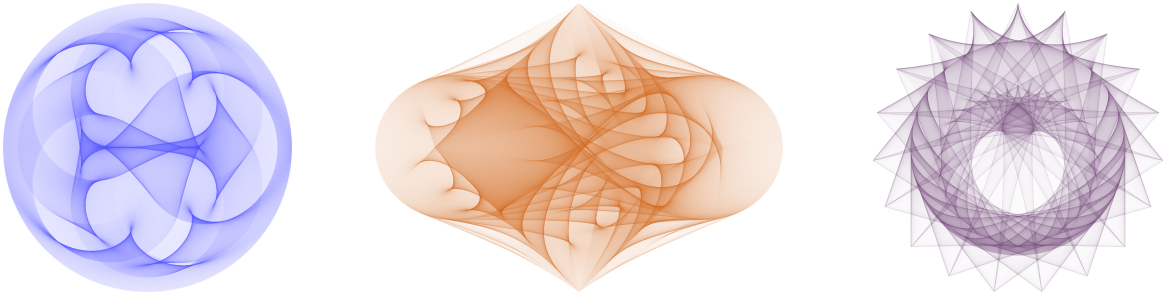
In the previous examples, the algorithm works on the set of points associated with one curve, and all line segments start and end on the same curve. For example, Figure 7(a) shows the results of rendering one circle. A simple extension is to connect points on two independent curves. With two independent curves, we can now resize and reposition the two curves allowing for additional variations. For example, the radii and center points of two independent circles can be manipulated as in Figures 7(b), 7(c), and 7(d).

As the examples show, modifying the algorithm to support two independent curves of the same type



(a) *One circle* **(b)** *Two circles, different radii* **(c)** *Two circles, different centers* **(d)** *Two circles, different radii and centers*

Figure 7: *Comparison of one circle to two circle size and placement variations. For each example,* $k = \frac{17}{13}$.



(a) *Rose curve and circle* **(b)** *Square and lemniscate* **(c)** *Hypocycloid and circle.*

Figure 8: *Combinations of two curves.*

increases the possibilities. However, it is easy to see that the algorithm is not truly restricted to rendering two identical curves. An additional modification renders the interactions of two different types of curves, for example, a circle and a rose curve, as long as both curves can be rendered from parametric equations based on θ. Figure 8 illustrates some of the possibilities that can result from the combinations of two different curves.

Summary and Future Work

It is clear that there are still unexplored possibilities for the density plots. Areas for exploration include using more complex or parameterized color palettes, animating parameter changes, and rendering the density plots in 3D. Other possibilities include combining and layering more than two curves in a single image. Mary Boole's curve stitching is capable of creating works that are visually appealing. Curve stitching density plots create a similar, yet different, type of image that have a related aesthetic. One approach focuses on the line segments, while the other focuses on the effects of density of the line segments. Mary Boole's simple idea, discovered over 100 years ago, still has plenty of room for exploration.

References

[1] M. E. Boole. *The Preparation of the Child for Science*. 1904. Clarendon Press. Available at https://archive.org/details/preparationofchi00boolrich. Last accessed 02/06/2019.

[2] S. Innes. "Mary Boole and curve stitching: a look into heaven." *Endeavour*, vol. 28, no. 1, 2004, pp. 36–38.

[3] B. J. Loe and N. Beagley. "The Coffee Cup Caustic for Calculus Students." *The College Mathematics Journal*, vol. 28, no. 4, 1997, pp. 277–284.