Creating Variation When Building Image Generation Datasets

Jhovani Gallardo Moreno[1], Omar Khan[2], and Michael Wehar[3]

Computer Science Department, Swarthmore College, Swarthmore, PA, USA
[1]jgallar1@swarthmore.edu, [2]okhan1@swarthmore.edu, [3]mwehar1@swarthmore.edu

Abstract

We investigate how an algorithmic artist can parameterize drawing algorithms to create a variety of resulting images. We explore how this variation leads to an image dataset that defines a visual style. Furthermore, having a dataset of hundreds of distinctive images rather than a single output image adds more value and enables a variety of applications.

Motivation

Drawing algorithms need not be one-time use only. They can be reusable generating different, but related visuals each time. A key question that we explore is as follows. How can we increase the reusability of a drawing algorithm to generate hundreds (or thousands) of different yet related images?

It is not uncommon to reuse a past drawing algorithm for another application. We propose to take this concept of reuse a step further by using a drawing algorithm to create a dataset that contains a variety of images. Having a variety of images enables a user to more easily select and customize images. In particular, the authors have found this to be especially helpful when creating digital clothing designs and print media. However, it could be applied to many other visual design use cases as well. Furthermore, having a dataset of varied, yet related images helps to explore and define the visual style captured by a drawing algorithm.

Drawing Algorithms and Our Platform

The kinds of drawing algorithms that we consider take the form of computer programs that place lines and basic shapes step-by-step onto a 2D canvas resulting in 2D images. The concepts presented in this work should still apply to most other forms of drawing algorithms as well.

The third author has been working with students to develop a web-based platform for loading and running drawing algorithms of this form called AlgoArt [1]. The drawing algorithms are written in JavaScript (JS) and follow a simple framework. Each drawing algorithm has its own JS file along with a companion parameters JS file. A drawing algorithm is treated as its own JS object that must implement initialize, start, pause, reset, drawOneStep, and randomize functions (or methods).

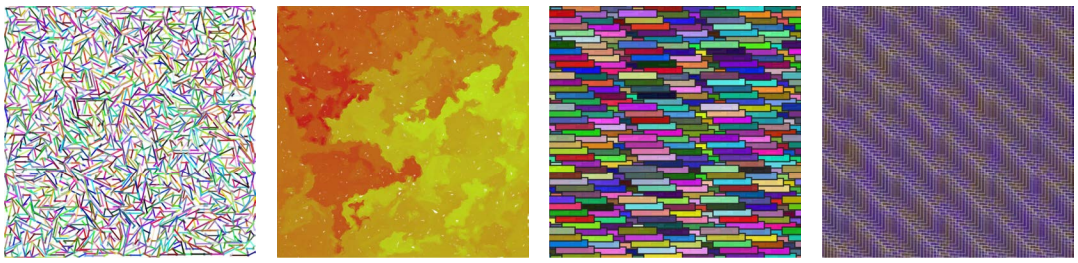
Drawing algorithms are selected and loaded in AlgoArt's open source Creator Studio [2]. This is where users can watch the algorithm draw. The step-by-step drawing process takes the form of a real-time animation generating the image. The user can also select parameters within the user interface to their liking to customize the image. The available parameters differ for each algorithm as determined by the algorithm creators.

Customizing Parameters to Create Image Variation

Designing drawing algorithms with parameters that can be changed allows the user of the algorithm to customize how the image is drawn. This results in differing images depending on the parameter selection. For example, if a drawing algorithm simply draws a rectangle, then parameters could be added to set the

rectangle's border color, fill color, width, height, and position. In the following, we consider general ways to add parameters to drawing algorithms to create image variation. To demonstrate this, we select example images from the AlgoArt image dataset that appear to have substantial variation even when the images are generated by the same algorithm. We then highlight parameter differences between how the images were generated. Additional visual variation could result from an algorithm incorporating randomness. Many of the selected images and their parameters are viewable within AlgoArt's online gallery [1].

Colors, Palettes, and Gradients



(a) **(b)** **(c)** **(d)**

Figure 1: *The left image pair was generated using Geometric Patterns by M. Wehar. The right image pair was generated using Overlapping Tiles by M. Wehar.*

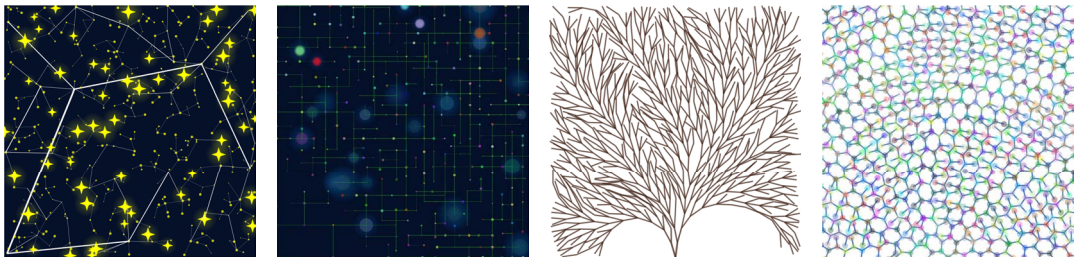
Color is an important way to create patterns, introduce segmentation, and, more generally, add information to an image. To visually separate an image's individual parts, random colors could be used. To create a sense of pattern or texture, a palette with a fixed list of colors could be used. Alternatively, one could just sequentially cycle through the colors to add a predictable texture to the image. Gradients with a continuous spectrum of color are a powerful way to add a sense of lighting, time, and / or macro separation between an image's parts.

For our drawing algorithms, we added parameters to (1) specify or randomize background color, (2) specify or randomize a palette for stroke or fill color, and (3) specify or randomize a linear gradient for border or fill color. When we specified palettes we would sometimes select from classic palettes, like how Piet Mondrian's classic works used primary colors [3], or randomly generate palettes. Furthermore, we created linear gradients by specifying or randomly selecting two distinct colors and making standard affine combinations (or weighted averages) between the two colors. We often would start with one color and gradually adjust the weighting to get closer and closer to the other color with each drawing step. This would visualize a sense of drawing time similar to how watercolors gradually dry. For future work, we would like to incorporate more complex gradients such as those created by parametric curves as illustrated in [4].

To illustrate these parameter options and the variations that they created see Figure 1. Images (a) and (b) differ in that (a) used random stroke colors while (b) used a linear gradient for stroke and fill colors to gradually change the color with each drawing step. Images (c) and (d) differ in that (c) used a random fill color while (d) sequentially chose from a color palette creating a sense of repeated texture.

Angles and Uniformity

For images that create a visual pattern out of connected lines, the angles between the lines can have a significant effect. It can make the difference between something looking natural versus artificial, unorganized versus structured, or irregular versus uniform. For our drawing algorithms, we added parameters to (1) use completely random angles and (2) use angles from a fixed list. For option (2), the angles that make up the list can be manually set, determined by a fixed offset, or randomly chosen. Typically, randomly chosen angles, as described in option (1), lead to unorganized and irregular patterns. However, with option (2), having a fixed list of angles leads to more structure and uniformity even if the fixed list contains random angles.



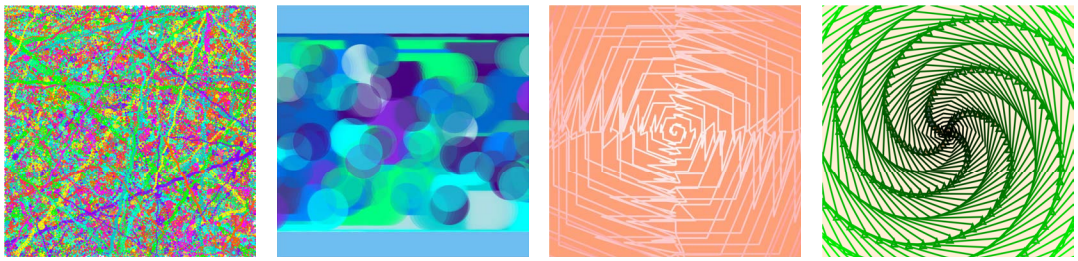
(a) (b) (c) (d)

Figure 2: *The left image pair was generated using Constellations by J. Gallardo Moreno. The right image pair was generated using Trees by M. Wehar.*

Furthermore, once the algorithm starts drawing lines, the drawing algorithm could further narrow its angle selection based on predefined criteria such as to avoid intersecting lines or incentivize a desired behavior.

To illustrate these parameter options and the variations that they created see Figure 2. Images (a) and (b) differ in that (a) used random angles while (b) used only 90 degree angles. Images (c) and (d) differ in that (c) used random angles while (d) only selects from two predetermined angles.

Paths and Movement



(a) (b) (c) (d)

Figure 3: *The left image pair was generated using Collisions by O. Khan. The right image pair was generated using Spirals by L. Suresh.*

A drawing algorithm can create paths by sequentially drawing connected lines, shapes, or curves. When doing so, the image can vary greatly by simply changing the path's sequential form or function. For our drawing algorithms, we have added parameters to customize the path's function and smoothness. In addition, random noise was sometimes added to paths as well. We illustrate these parameter options in Figure 3 where images (a) and (b) differ in that (a) used linear paths with varying slopes while (b) used paths with slope 0 resulting in horizontal lines. Images (c) and (d) differ in that (c) follows a discretized jagged spiral path while (d) more precisely follows a smooth classical spiral path.

Sizes and Repetitions

Drawing the same shape at different sizes can create an entirely different aesthetic and feeling, especially if the shape is drawn repeatedly. For our drawing algorithms, we added parameters to customize the size and dimensions of shapes. In addition, we had parameters to repeat shapes as separate copies or nest shapes within each other. We illustrate these parameter options in Figure 4 where images (a) and (b) differ in that (a) draws many separate copies of the same shape while (b) draws one large shape with smaller copies nested within it. Images (c) and (d) differ in that (c) draws many copies of differing nested shapes while (d) draws one large form and repeatedly nests differing smaller shapes within it.

(a) (b) (c) (d)

Figure 4: *These four images were generated using Stickers by M. Wehar.*

Building Images Datasets

To create our image dataset, all of our drawing algorithms implement a randomize function to randomly sample (reasonable) values for the parameters. Then, we use a program that we developed called the dataset builder (DSB). Our DSB runs the selected drawing algorithm hundreds of times with varying parameterizations to create a variety of images. Once we have our dataset, we feed the images and their metadata into a reviewing web application where our team can further label and review the images. This allows us to rank the images and discover the many differing kinds of results that can come from our drawing algorithms. Many of the images from our current image dataset are available for public viewing in our online gallery [1].

Conclusion

In summary, we explored several ways that a drawing algorithm can be varied to create an image dataset of hundreds or thousands of varying images. In particular, we looked at how colors, angles, paths, sizes, and repetitions can create image variation. Ultimately, having a larger degree of image variation is an important way to make drawing algorithms more easily reusable which adds value to an algorithm creator's works. In our exploration of image variation, we required the algorithm creators to design and implement the parameters. As future work, we would like to design a system that can modify a drawing process and customize the resulting image without having to predefine the parameters within the drawing algorithm itself.

Acknowledgements

We are thankful for all of those who contributed to the AlgoArt Project including E. Brickner, X. Dong, X. Li, C. Liu, J. Mancini, M. Newman-Toker, R. Oet, V. Sumano, L. Suresh, P. Tone, and A. Zhang. We acknowledge support for this work from Swarthmore College's Research Fund and Academic Division Fund.

References

[1] AlgoArt Home Page. Feb. 2025. https://algoart.org.

[2] Algorithmically-Generated-Artwork/Drawing-Program - GitHub. Feb. 2025. https://github.com/Algorithmically-Generated-Artwork/Drawing-Program.

[3] L. Feijs. "Analyzing the Structure of Mondrian's 1920-1940 Compositions." arXiv. 2020. https://arxiv.org/abs/2011.00843.

[4] J. E. S. Sánchez. "Parametric Curves in Color Space." *Bridges Conference Proceedings*, Halifax, Nova Scotia, Canada, Jul. 27–31, 2023, pp. 181–186. http://archive.bridgesmathart.org/2023/bridges2023-181.html.