Serial Polar Transformation Motifs Revisited

Gary R. Greenfield
Department of Mathematics & Computer Science
University of Richmond
Richmond, VA 23173, U.S.A.
ggreenfi@richmond.edu

Abstract

Elliot and Bleicher considered using compositions of polar transformations — functions from the plane to the plane viewed as transformations from polar to cartesian coordinates — as a method for generating computer images. In this paper we revisit this technique for creating what we call *motifs* in order to more carefully consider: (1) how these transformations are defined, (2) implementation and resolution issues, (3) motif coloring, and (4) the automated evolution of motifs.

1. Introduction

In [1], Bleicher presents an interesting gallery of images that were computer generated using a technique he calls "serial transformations of squares." The formal model for this technique requires a finite set of functions $\{f_1, \ldots, f_u\}$ where each function f maps the unit square $\{(\theta, r) : 0 \leq r, \theta \leq 1\}$ to the plane. This means that each f can be written in the form $(\theta, r) \longrightarrow (f_x(\theta, r), f_y(\theta, r))$. Suppose that the point (θ, r) is colored $c_{(\theta,r)}$. If we have a composition sequence g of the form $g = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_v}$, then we obtain an image, or *motif*, from g by coloring the point $g(\theta, r)$ using the color $c_{(\theta,r)}$. Because Bleicher uses his model to search for aesthetic motifs in the *space* of serial transformations, his image generation method is distantly related to the evolving expressions method of Sims [6]. And although the image generation method is completely different, it should be noted that many of the images in Bleicher's gallery bear a striking resemblance to the spirolaterals of Krawczyk [4]. With respect to Bleicher's formal model, we note that compositions of transformations of the type Bleicher considers formed an integral part of a functional programming *design* language developed by Elliot [2] for image processing. Elliot's principal interests were *pattern* design and comparing his language to earlier design languages introduced by Maeda [5], Hudak [3], and others. In this paper we revisit the work of Bleicher and make further contributions by considering additional pattern, color, and genetic algorithm techniques.

2. The Polar Transformations

Following [1], the points (θ, r) of the unit square are viewed as parameterizing lines and circles in the plane by sending horizontal lines (fixed r) to circles, and vertical lines (fixed θ), to lines through the origin. In order to obtain such a map from the unit square to the unit square we must use translation of axes, so we define the polar transformation P by

$$P : (\theta, r) \longrightarrow ((1 + r\sin(2\pi\theta))/2, (1 + r\cos(2\pi\theta)/2)).$$

Note that the image of the vertical line segment in the domain from $(0, 0)$ to $(0, 1)$ is the vertical line segment in the range from $(1/2, 1/2)$ to $(1/2, 1)$. To invert this transformation, we use the polar (sic) transformation I determined from the equations

$$\begin{aligned} \theta &= \arctan((2x-1)/(2y-1))/2\pi, \\ r &= \sqrt{(2x-1)^2+(2y-1)^2}. \end{aligned}$$

We also consider the inversion though a circle transformation

$$C:(\theta,r)\longrightarrow(k^2\theta'/R'+1/2,k^2r'/R'+1/2),$$

where $\theta'=\theta-1/2$, $r'=r-1/2$, $R'=\sqrt{(\theta')^2+(r')^2}$, and k is constant, together with the "displacement" transformation (the one suggested in [1] has a misprint) given by

$$D:(\theta,r)\longrightarrow(\theta'+a\sin(r'/b)+1/2,r'+a\sin(\theta'/b)+1/2),$$

where θ' and r' are as above, and a and b are constants chosen for their aesthetic effect. Finally, for future reference, we consider the polar arc transformation

$$A:(\theta,r)\longrightarrow((1+r\sin(2\pi((1-2\omega)\theta+\omega))/2,(1+r\cos(2\pi((1-2\omega)\theta+\omega)/2)),$$

where the constant ω is chosen to lie between zero and one-half.

3. The Resolution Problem

Even though our polar transformations are continuous almost everywhere, in order to visualize the effect of a polar transformation on an $n\times n$ pixel output display device, one must either consider inverse mappings to ensure that each pixel in the range is the image of some pixel in the domain, or forward mappings using the technique of over-sampling in the domain so that the transformation is treated as a mapping from an $m\times m$ array to an $n\times n$ array where $m\gg n$. We chose the latter. Moreover, the choices for the constants necessary to define some of our transformations, or the use of rotations (see below), may send some points of the unit square under the polar transformation mapping to points that lie outside the unit square, in which case clipping must be invoked. To emphasize these complications we show in Figure 1 the effect of applying the transformations P, I, D, C, and $P\circ P$ to a unit square that was colored using a black and white checkerboard pattern. The over-sampled resolution mapping is 800×800 to 200×200, and the "missed" pixels are shown in grey. This figure should be compared with Figure 1 of [1]. To generate what we are referring to here as motifs using serial transformations of the square, Bleicher interleaves the *polar arc* transformation with a sequence of transformations chosen from the set consisting of $R_1,\ldots,R_{11}$, where R_i is a counterclockwise rotation of $\pi i/6$ radians, together with the horizontal reflection H which, for our purposes, needs to be defined by setting $H(\theta,r)=(\theta,1-r)$. Adopting the abusive notation $R_{12}=H$, any finite numerical sequence of indices can be used to represent a serial polar transformation. For example, the sequence 2-7-12-5 specifies the serial transformation

$$A\circ R_5\circ A\circ R_{12}\circ A\circ R_7\circ A\circ R_2\circ A.$$

In our implementation we chose the constant ω in A, which governs how pixels are distributed along the arc determined by θ as θ ranges from zero to one, to be 0.05. To see why this method successfully generates motifs, refer to Figure 2 which shows the iterates A^0, A^1, A^2, A^3, A^6, and A^9 applied to a solid patterned square where, once more, missing pixels are shown in grey. This figure should be compared to Figure 2 of [1]. Presumably, discrepancies between our first two figures and those in [1] arise from the use of forward mappings and/or the fact that transformations are not given *explicitly* in [1].

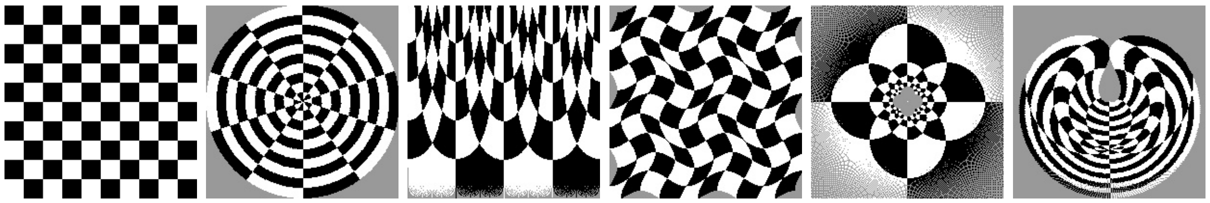

Figure 1: *From left to right, the polar transformations denoted J (identity), P (polar), I (polar inverse), C (inversion through the circle using $k = 0.2$), D (displacement using $a = 0.04$ and $b = 0.06$), and $P \circ P$ (double polar) applied to a unit square with a checkerboard pattern.*

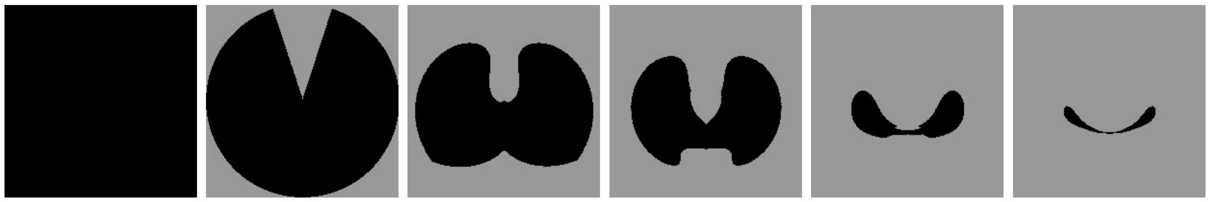

Figure 2: *From left to right, the motifs from the polar transformations $A^0 = J$, $A^1 = A$, A^2, A^3, A^6, A^9, where A is the polar arc transformation with constant ω set to 0.05 and J is the identity transformation, applied to a solid colored unit square.*

4. Randomly Generated Motifs

Since there are only $12 \times 12 \times 12 = 1728$ motifs that can be formed using interleaved sequences derived from *sequence identifiers* of the form i_1-i_2-i_3, it is conceivable that one could exhaustively generate and inspect them all. We chose to use a sampling approach instead by randomly generating 300 of these length three motifs in groups of twenty-five at a time. Like Bleicher, we observed wispy motifs such as those shown in Figure 3. We also obtained a cache of more compact motifs that we found equally intriguing such as those shown in Figure 4. Surprisingly, we found an extraordinary number of *mutant* motifs such as those shown in Figure 5. Mutant motifs were characterized as having hair-like appendages or cascades of isolated pixels. In fact, more than half of our randomly generated motifs were mutant!

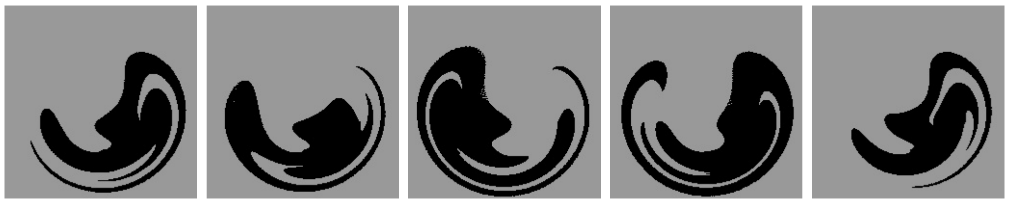

Figure 3: *Randomly generated wispy interleaved length three motifs with sequence identifiers 9-10-3, 3-1-8, 2-4-5, 3-3-12, 9-10-8.*

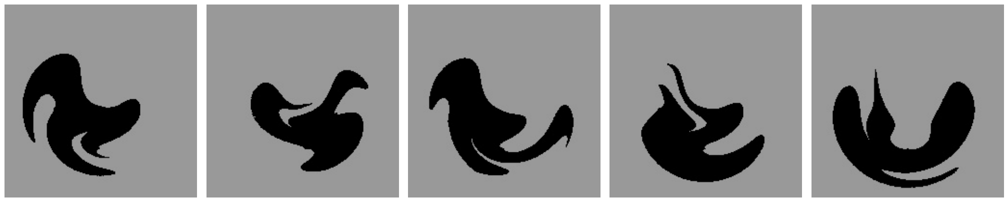

Figure 4: *Randomly generated compact interleaved length three motifs with sequence identifiers 10-2-3, 10-10-10, 12-3-9, 3-1-3, 2-1-6.*

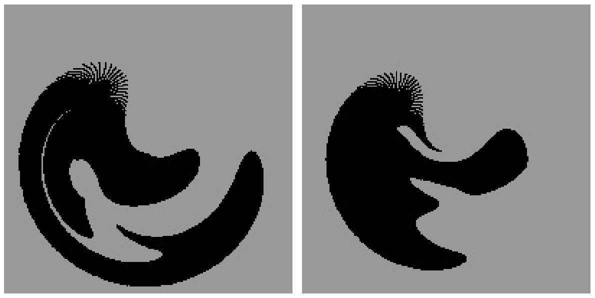

Figure 5: *Randomly generated mutant interleaved length three motifs with sequence identifiers 11-7-1 and 3-6-3.*

5. Motif Colorings and a Simple Variation

In order to better understand how motifs are formed, we replaced the solid background pattern with a vertical gradient pattern consisting of nine vertical stripes in nine different shades of the same hue. To maintain the visual integrity of the motifs, when working with a fixed hue in HSV color space, we chose very narrow spreads in value and saturation. Figure 6 shows examples of such "shaded" motifs.

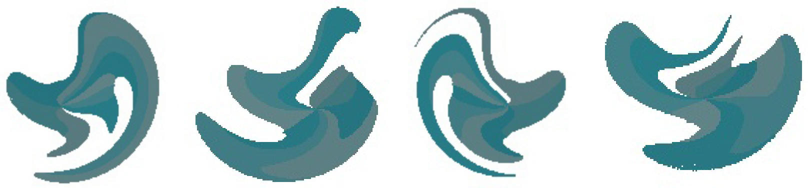

Figure 6: *Shaded, randomly generated interleaved length three motifs with sequence identifiers 3-9-9, 8-11-9, 2-3-3, 9-11-10.*

As an experiment, we also considered altering the *initial* polar transformation before reverting to interleaving the (length three) sequence with the polar arc transformation. Figure 7 shows two images obtained when the initial transformation was the displacement transformation D, while Figure 8 shows five examples obtained when the initial transformation was the polar inversion transformation I.

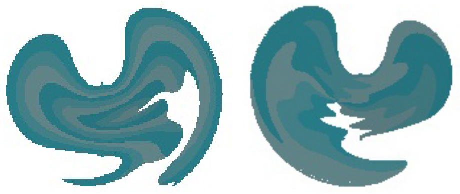

Figure 7: *Interleaved length three motifs where the displacement transformation replaces the initial polar arc transformation. Sequence identifiers are 3-9-11 and 6-6-1.*

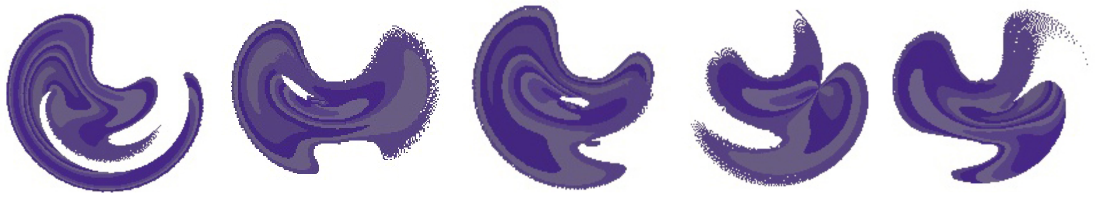

Figure 8: *Interleaved length three motifs where the polar inversion transformation replaces the initial polar arc transformation. Sequence identifiers are 3-3-3, 8-1-1, 8-10-1, 12-11-10, 3-10-11.*

6. The Genetic Algorithm

To consider longer lengths for interleaved sequences, we kept the polar arc transformation as the initial transformation, and made use of the Genetic Algorithm. This method had been suggested by Bleicher but his results were inconclusive (private communication). The sequence identifier now represents the *genome* of an individual motif in the *population* of motifs. We used a population of size twelve, and at the conclusion of each fitness evaluation cycle, we used the four most fit genomes to repopulate by preserving their genomes into the next generation and applying a "point mutation" operator to copies of their genomes to obtain the rest of the individuals needed to fill out the population. To ensure diversity, we mutated the clones sufficiently until all members of the population were distinct. For our fitness function we chose to tally the number of *missed* pixels in the output image whose 3×3 pixel neighborhood did *not* consist entirely of missed pixels. The rationale behind this calculation was to measure the "boundary" of the motif. During most our runs we minimized this fitness function. For such runs the expectation was that we would evolve compact motifs and avoid mutants. The Genetic Algorithm was allowed to run for either five or ten generations. Figure 9 shows three images from our test run using length three genomes. There were no mutant images in the population but, surprisingly, wispy images did appear. Figure 10 shows motifs that were evolved in a run using length four genomes.

Due to space considerations, for our final example, we consider length five genomes, but now during each run we determine fitness by *maximizing* the tally of boundary pixels of the motif. Our objective, of course, is to evolve wispy motifs. The top row of Figure 11 shows the wispy motifs that were evolved in a run lasting five generations. The bottom row of Figure 11 shows wispy motifs from a different run that lasted ten generations. Mutants within such populations were present, but this was easily explained by the high mutation rates we used because, when mutants appeared, they rarely survived for more than one generation.

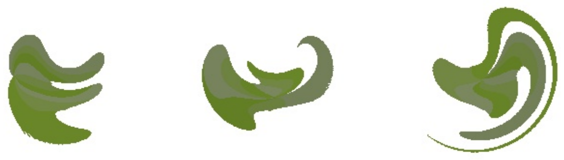

Figure 9: *The Genetic Algorithm applied to a small population of length three genomes. The fitness goal was to* minimize *the number of boundary pixels. Left to right the most fit, the average fit, and the least fit motifs present after five generations. Their sequence identifiers are 11-9-1, 2-1-1, 10-8-9.*

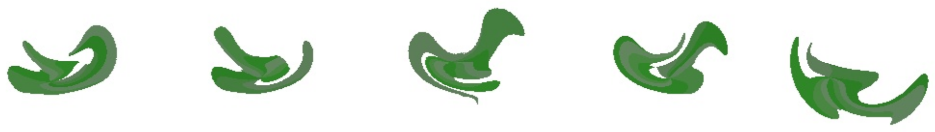

Figure 10: *The Genetic Algorithm applied to length four genomes. Sequence identifiers are 7-2-3-1, 4-1-3-1, 2-4-10-10, 6-10-8-10, 2-10-10-3.*

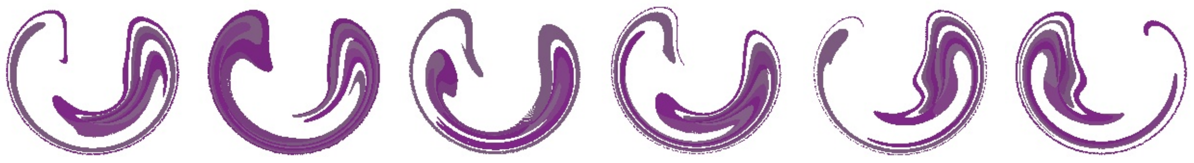

Figure 11: *The Genetic Algorithm applied to a small population of length five genomes. The fitness goal was to* maximize *the number of boundary pixels. The three on the left with sequence identifiers 10-8-7-8-5, 2-10-12-8-5, 2-12-10-5-5 were from a run lasting five generations and the three at the right with sequence identifiers 10-9-6-8-4, 10-9-6-8-6, 9-10-6-8-12 are from a run lasting ten generations.*

References

[1] Bleicher, L., Serial polar transformations of simple geometries, *ISAMA—CTI 2004 Proceedings* (ed. S. Luecking), 2004, 65–68.

[2] Elliot, C., Functional image synthesis, *2001 Bridges Conference Proceedings* (ed. R. Sarhangi and S. Jablan), 2001, 139–158.

[3] Hudak, P., *The Haskell School of Expression — Learning Functional Programming through Multimedia*, Cambridge University Press, New York, 2000.

[4] Krawczyk, R., The art of spirolaterals, *The Millennial Open Symposium on the Arts and Interdisciplinary Computing* (eds. D. Salesin and C. Sequin), 2000, 127–136.

[5] Maeda, J., *Design by Numbers*, MIT Press, Cambridge, MA, 1999.

[6] Sims, K., Artificial evolution for computer graphics, *Computer Graphics*, **25** (1991) 319–328.