The Art of Iterated Function Systems With Expanding Functions

Philip Van Loocke
Rectorate
University of Ghent
Sint-Pietersnieuwstraat 25
9000 Gent, Belgium
E-mail:philip.vanloocke@ugent.be

Abstract

Iterated function systems with contracting functions have been widely applied in contexts relating art and science. This paper explores iterated function systems which consist of expanding linear functions. An area in the plane is divided in parts which are defined implicitly by an inverse function technique. With each part, an expanding function is associated. A coloring technique is proposed which yields textures suggestive of sophisticated patterns of depth and light. It is briefly described how a rendering technique for recurrent origami can be obtained as a special case of this method.

Introduction

Iterated function systems defined by contracting functions are a familiar tool among researchers bridging art and science. Such systems yield compact codes for a large variety of images, which can be generated at any desired resolution. This paper describes a fractal image creation technique based on iterated function systems with expanding functions. The method is based on an inverse technique for iterated function systems consisting of two linear holomorphic functions [1]. It is described in this paper in a geometric way. I briefly explain how a rendering technique for recurrent origami can be obtained a special case of this method.

Suppose that an iterated function system $\mathcal{F}$ is defined by n functions f_j (j=1,…,n). In the classical treatment all functions f_j need to be contracting in all directions (see for instance chapter seven in [2] for a formal specification of this constraint). I call a function 'expanding' if its inverse is contracting. Function systems with linear expanding functions have been considered in the context of inverse techniques. In the early nineties Prusinkiewicz formulated such a technique which allows one to color the complement of attractors [3]. I confine myself to an informal description. Suppose that P is a point on the outside of the attractor of $\mathcal{F}$. Let C be a contour which encloses the attractor. Consider compositions of s inverse functions $f_{r_1}^{-1} f_{r_2}^{-1} ... f_{r_s}^{-1}$ with $r_j \in \{1,...,n\}, j = 1,...,s$. Since the inverse functions are expanding, each composition of sufficient length will map P onto a point on the outside of C. Consider the composition for which P travels the longest path inside C before bailing out. The length of this path associates a number with P which can be transformed into one or three color values.

This type of inverse technique has one advantage. The color of a point P is determined by an algorithm that takes this point as argument. In the direct method, one has to wait and see if the chaos game lands on the corresponding pixel until one knows if P is given a colour value (and if so, wait longer until a relative frequency of pixel visitation can be reliably determined). In this sense, an inverse technique acts like the usual method for coloring points of non-linear escape fractals. But there are two drawbacks. First, in contrast with the latter, the inverse technique needs calculation of many trajectories in P. The number of trajectories required increases combinatorially with n, which makes this method in most cases practically

unfeasible if n is larger than 2. Second, it only colors the complement of attractors. There are many situations in which the attractors themselves, and the fractal textures defined on them, are visually interesting objects. In this paper I present a geometric formulation of a different inverse technique, which solves both problems. First I formulate it in a general way. Then I describe two ways in which it can be specified.

Iterated function systems with expanding functions

The basic concepts of the present method are illustrated in Figure 1. Let A be an area in the plane. $S_1,\ldots,$ S_m are disjoint subsets of A. Let B be the set defined by $B = \bigcup_{j=1,\ldots,m} S_j$. Note that $B \subseteq A$. I confine the illustrations to the case in which B is a strict part of A. With each subset S_j, an expanding linear function h_j is associated which maps S_j onto $h_j(S_j)$, which is assumed to be a subset of A as well. Consider the function $g : B \rightarrow A$ which is defined by $g(P)=h_j(P)$ if $P \in S_j$ (j=1,…,m). The algorithm considered in this paper defines for each point $P \in B$ a series of points $P_0,\ldots P_N$. The first point P_0 is initialized by $P_0=P$. Then, P_1 is defined by $P_1=g(P_0)$. If $P_1 \in B$, $P_2=g(P_1)$. Else, if $P_1 \in A \setminus B$, the algorithm terminates. This process is iterated. If $g(P_{k-1}) \in B$, $P_k=g(P_{k-1})$, else the algorithm halts. If the algorithm terminates at step t with $t<N$, a series of N points is constructed by adding $N-t$ times the origin O to $P_0,\ldots P_t$.

In the two last illustrations of next section, the algorithm is made slightly more sophisticated. Suppose that at iteration t a point $P_t \in A \setminus B$ is produced. Then the algorithm continues but P_t is replaced by $h_1^{-1}(P_t)$. For this modification, also points in the complement of B (and hence points in the complement of A) can be processed and lead to a series $P_0,\ldots P_N$. Since h_1^{-1} is contracting, features in the inside of A reappear in magnified version in the complement of A.

The sequence $P_0,\ldots,P_N$ is turned into a color value as follows: q points $Q_1,\ldots,Q_q$ are selected in the plane (these are allowed to be in A but do not have to). For each point P_k in the series $P_0,\ldots P_N$, the distance d_{kj} to all points Q_j is calculated (j=1,…,q). The inverses of the q distances associated with P_k are normalized to one, yielding coefficients c_{kj}. Then, with each point Q_j a quantity ϕ_j is associated in accordance with

$$\phi_j(P) = \sum_{k=1,\ldots N} c_{kj} / k^{\rho}$$

where ρ is a constant which was put to ρ =0.5 in the illustrations which follow. These quantities are linearly combined into $\varphi(P)$:

$$\varphi(P) = \sum_{j=1,\ldots,q} w_j \phi_j(P)$$

where w_j (j=1,…,q) are fixed coefficients. (Exploration of expressions with non-linear combinations, such as combinations of squares of ϕ_j, can be aesthetically rewarding as well). After $\varphi(P)$ has been calculated for each point $P \in B$, it is normalized between zero and one, and next it is turned into three color values by a colormap. In all illustrations in this paper, four points Q_j are used, which are located at the vertices of a square with center at the origin, and which is a scaled version of the unit square. Informally, the points Q_j define neighbourhoods which act as traps. The weight of trap j is a function of the inverse distance

between P_k and the defining point Q_j of the trap (which is expressed by c_{kj}), and different traps contribute in different ways to the quantity onto which the colormap is applied (which is expressed by w_j).

Illustrations for the square

In the first specification of the general concepts, I use an iterated rotation technique to divide B into disjoint subsets S_j. In Figure 1, A is the unit square with center at the origin. B is a smaller square with side $2a$ and four sets S_j are defined by:

$$S_1 = \{(x,y) \mid (x > -b) \,\&\, (x < a) \,\&\, (y > -a) \,\&\, (y < b)\}$$
$$S_2 = \{(x,y) \mid (x > -b) \,\&\, (x < a) \,\&\, (y > b) \,\&\, (y < a)\}$$
$$S_3 = \{(x,y) \mid (x > -a) \,\&\, (x < -b) \,\&\, (y > -b) \,\&\, (y < a)\}$$
$$S_4 = \{(x,y) \mid (x > -a) \,\&\, (x < -b) \,\&\, (y > -a) \,\&\, (y < -b)\}$$

with $b<a$. Similar partitions for the other regular polygons are obtained in a straightforward way (below this is illustrated for the octagon). In case of a square, a scaling is applied onto B with center in the lower right vertex of B, which results in S_1. Then, S_j (j=2,3,4) is the part of B which is not included in $S_1 \bigcup ... \bigcup S_{j-1}$ and which, after rotation around the origin with angle $-(j-1)2\pi/4$, overlaps with S_1. In Figure 4 this procedure is applied to the octagon, but S_j (j=2,…,8) was obtained for rotation of the reduced octagon with different angles. Variation of these rotation angles, and the resulting variation in the definition of S_j, is one way to find aesthetically relevant variations of images.

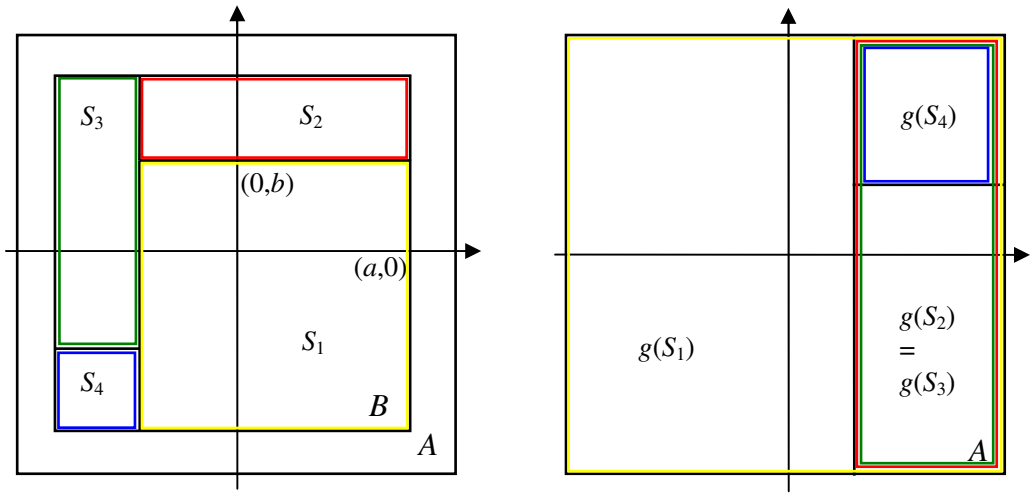


Figure 1: *Left: Partition of subset B of A into four parts S_j. Right: Image of the parts after application of g (the images of S_2 and S_3 are identical and include the image of S_4)*

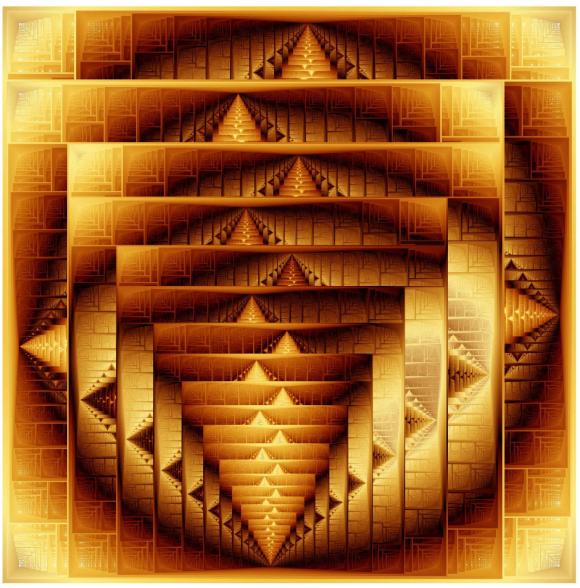
There are different ways to define the functions h_j, but the most natural strategy is to relate h_j with the way in which the partition of B into the sets S_j was obtained. If these sets are obtained by iterated rotation, the simplest strategy is to dissect the action of h_j into three steps. I specify the procedure for the case of a square set A, but the generalization for other regular polygons is straightforward. Consider a point $P \in S_j$. First P is rotated around the origin with angle $-(j-1)2\pi/4$, so that the resulting point, which is denoted $P_{r,j}$, is in S_1 (points in S_1 are left unmodified by this step). The second step is the same for all h_j. Let C denote the center of S_1. As can be verified on basis of Figure 1, C has coordinates $((a-b)/2,$

$(-a+b)/2$). Let $\gamma=2/(a+b)$. Then, $P_{r,j}$ is translated with the vector which connects C with the origin. Subsequently, the resulting point is subject to a scaling with factor γ and center at the origin (the effect of this scaling on the translation of area S_1 is that the image of the latter coincides with A). These two actions define the second step. Third, the point obtained is rotated around the origin with an angle which is a multiple of $2\pi/4$, and/or subject to a reflection. The reflections and rotations in the third step can be chosen different for different j. (The case of identical rotations and absence of reflection corresponds to the situation considered in [1]).

Turning to complex number notation, the equations involved become very simple. For each point $P \in S_j$, $P_{r,j}$ can be written as $P_{r,j} = P \exp(-i\,(j-1)2\pi/4)$ (where i is the imaginary unit). The point onto which $P_{r,j}$ is mapped after the second step is $\gamma(P_{r,j}-C)$, with $C = (a-b)/2+i(-a+b)/2$. Then, the transformations h_j which lead to Figure 2 simply read $h_j(P) = (\gamma(P_{r,j} - C))^*$, where * denotes the complex conjugate (which corresponds to a reflection; this choice of h_j is illustrated in the right part of Figure 1). The transformations used to obtain Figure 3 are:

$$h_1(P) = \gamma(P_{r,1} - C)\exp(i2\pi/4); \qquad h_2(P) = \gamma(P_{r,2} - C)\exp(i6\pi/4)$$
$$h_3(P) = \gamma(P_{r,3} - C)\exp(i6\pi/4); \qquad h_4(P) = \gamma(P_{r,4} - C)\exp(i2\pi/4)$$

The two other illustrations in this section are based on octagonal sets A and B. The dissection of B in eight subsets is shown in Figure 4. With each set S_j (j=1,…,8) a function h_j is associated in accordance with the recipe specified above. The rotations in the third step of the definition of h_j were put equal to each other. The only difference between the algorithms leading to Figures 5 and 6 is that, in case of Figure 5, the rotations in the third step rotate with an angle $4\pi/8$, whereas in Figure 6, this angle is equal to $2\pi/8$. As was mentioned above, the algorithm used for these figures includes the additional step according to which a point not in $A \setminus B$ is contracted toward the center of S_1 with factor δ (which was put equal to 0.8 in the illustrations).



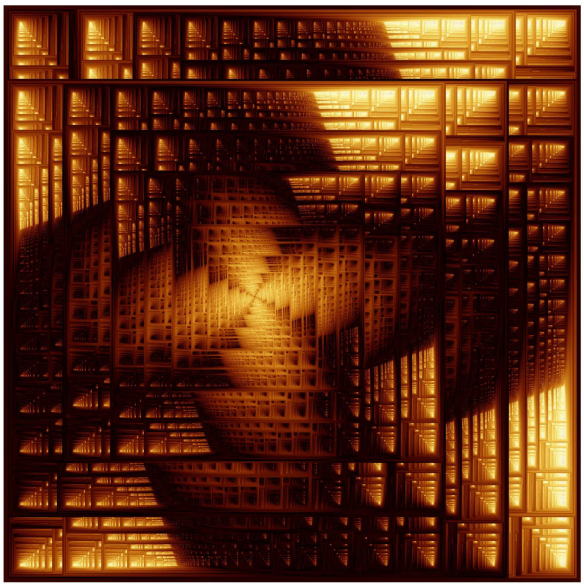
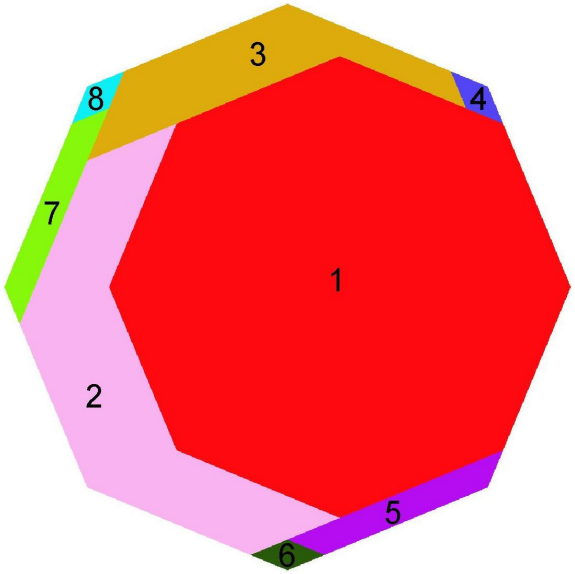


Figure 2: *(left) First illustration for square A, with a=0.9925 and b=0.75 and* **Figure 3***: (right) Second illustration for square A, with same sets S_j and same values of a and b, but for different definition of h_j*



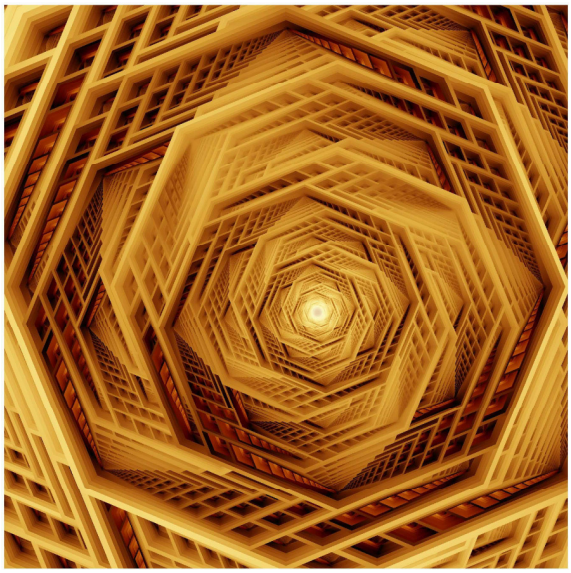


Figure 4:*(left) Partition of octagonal set B into 8 sets* S_j *and* **Figure 5:** *(right) Octagon-based illustration with rotation angle in third step of* h_j *equal to* $4\pi/8$

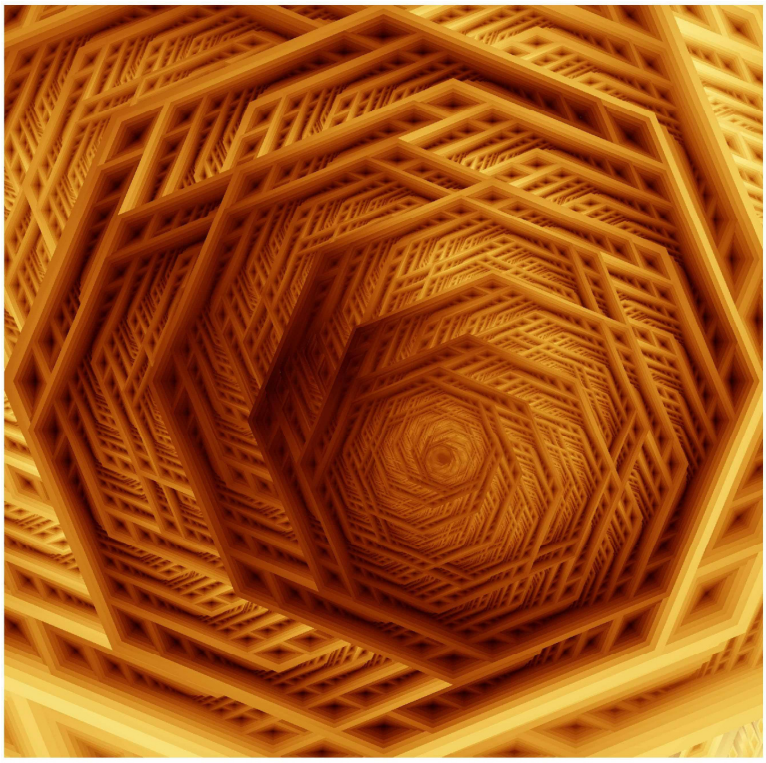


Figure 6*: Octagon-based illustration with rotation angle in third step of* h_j *equal to* $2\pi/8$

Definition of S_j and h_j by origami

In the second specification of the general concepts, S_j and h_j are defined by an origami process. Suppose that a polygon B is folded so that a smaller version of the same polygon is obtained. After unfolding, the sets S_j are specified by the partition generated by the crease pattern. In order to define h_j, the reduced polygon is magnified (and possibly rotated) so that it coincides with A. For each set S_j, this defines an expanding function which maps S_j onto a subset of A. In combination with the algorithm described above, this is basically all it takes to obtain Figure 8 on basis of the folding described in Figure 7 (Figure 7 for simplicity only shows B. A is a hexadecagon slightly larger than the blue hexadecagon B). Note that the process by which the reduced polygon is magnified has an alternative description. Mathematically, this is equivalent with downscaling the crease pattern to the reduced polygon, after which the folding is iterated. Therefore, the concepts described offer a way to create patterns corresponding to recurrent origami (some possibilities for variation, and a more extended discussion is given in [4]).

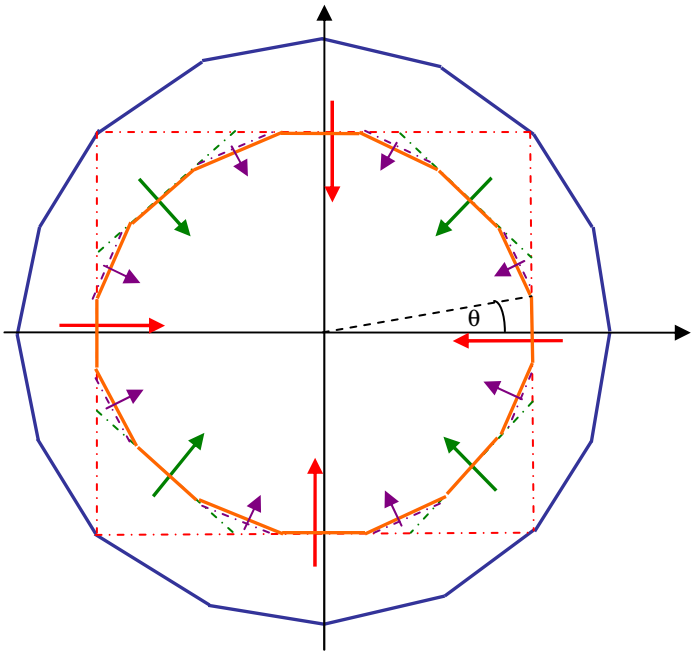


Figure 7: *The blue hexadecagon B is folded inward onto the smaller orange hexadecagon. First the parts between the square and the hexadecagon are folded (see the red arrows). Then the four corners of the square are folded, which results in an octagon (as indicated by the green arrows). Finally eight corners of the octagon are folded (in accordance with the purple arrows), so that the smaller hexadecagon is obtained. In order to define the functions h_j, the latter is rotated first around the origin with angle $\theta=\pi/16$, and next scaled so that it coincides with A.*

Figure 8: *Image obtained by iteration of the origami process in Figure 7*

Discussion

Iterated function systems are classically used with contracting functions. I have illustrated that expanding functions can yield fine results as well. One condition is that the domain on which functions act is divided carefully into parts. I gave two illustrations of how this can be achieved. The first one defines the parts by an iterated rotation method, the second one uses origami. There is little doubt that other partitions can be found which work equally well from an aesthetic point of view. But the latter application has the advantage of relating origami with fractal techniques, which is a domain in which many possibilities remain to be explored.

References

[1] Ph. Van Loocke, *Polygon-based fractals from compressed iterated function systems* (IEEE CG&A, accepted). 2009

[2] M. Field, M. Golubitsky, *Symmetry in chaos. A search for patterns in mathematics, art and nature*, Oxford: Oxford University Press. 1992

[3] P. Prusinkiewicz, M. Hammel, *Escape-time visualization method for language restricted iterated function systems*, Proceedings of Graphics Interface '*92*, Morgan Kaufmann, pp. 213-223. 1992

[4] Ph. Van Loocke, *Combination of basic origami with fractal iteration* (Computers and Graphics, accepted). 2009