An Introduction to Leaping Iterated Function Systems

Mingjang Chen
Center for General Education
National Chiao Tung University
1001 University Rd.
Hsinchu, Taiwan
E-mail: mjchen@mail.nctu.edu.tw

Abstract

The concept of "Leaping Iterated Function Systems (LIFS in short)" is a variation of Iterated Function Systems (IFS), that originated from "self-similarity", i.e., the whole has the same shape as one or more of the parts. The methodology of Leaping IFS is first to construct a structure of the whole by parts, then to convert it to an image, and then to replace each part with this image. Repeat the procedures until the outcome is visually satisfied. One of the key features of Leaping IFS is that computer resource consumption will be under control during the iterations because the number of parts in the whole structure is fixed.

Introduction

Iterated Function Systems (IFS) is an interactive method of constructing fractals. First proposed by John Hutchinson [2] in 1981, Michael Barnsley [1] then applied this method to imitate self-similar patterns in the real-world. Translation, rotation, scaling, reflection, and shearing are included in the function system of linear fractals; their relative positions, orientations and scaling of geometry elements are used to define 2D geometrical transformation visually. In particular, the concept of the Multiple Reduction Copy Machine algorithm (MRCM) was used [6] to introduce IFS.

An interactive IFS Fractal generator based on "the Collage Theorem" allows users to sketch first an approximate outline of the desired fractal, and then cover it with deformed images of itself to achieve the collage and then render the attractor. However, the number of objects grows exponentially when iterating, so it always exhausts computing resources before the necessary iterations are done. The chaos game [1, 2], sometimes called the random iteration algorithm, is one of several algorithms that can be used to treat the resource consumption problem. One among a few simple rules (functions) is selected in the chaos game at random, and applied it to a point to yield a new point. This process of random selection is repeated over and over again to produce an "attractor".

	Line-based Cloning	Frame-based Cloning	Point-based Cloning	Angle-based Cloning
Base	line segment	rectangle	any object	any object
Translation	√	√	√	√
Rotation	√	√		√
Scaling	isotropic	anisotropic		
Reflection	√*	√*		

*: reflection is excuted in the generator, not in the input data.

Table 1: *Various combinations of geometry transformations for Structural Cloning Method (SCM)*

In this paper, Leaping Iterated Function Systems (LIFS) is introduced as an algorithm that improves Iterated Function Systems (IFS) within Structural Cloning Method. A generator of LIFS consists of a base and a set of visual elements (Table 1); the base can be a line segment, a rectangle, or any object; and visual elements, called a pattern, can be any geometric figures or pictures. Originally, Structural Cloning Method (SCM) is a visual interface used for designing instructional materials used for performing geometry transformations (Table1) represented by a generator, and so to design linear fractals (Figure 2). However, LIFS takes only constant computing resources [3], instead of exponentially growing loading while iterating.

For the generator shown in Figure 1, the base is the dash line, and one stem and 4 line segments form the pattern. There are 5 geometric transformations in this function system associated with the relative positions, orientations and scaling between the base and each element in the pattern. Since there are five line segments and one stem in $W(x)$, the number of line segments in $W^k(I)$ is 5^k and the number of stems in $W^k(I)$ is $5^0+5^1+\ldots+5^{k-1}$, which will grow exponentially as a function of k.

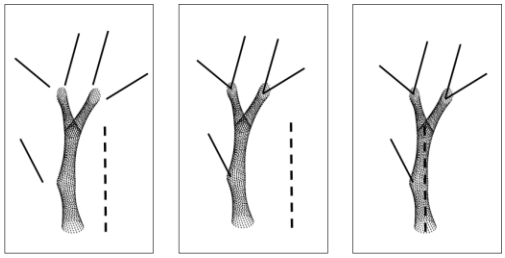

Figure 1: *The profile of W(x)*

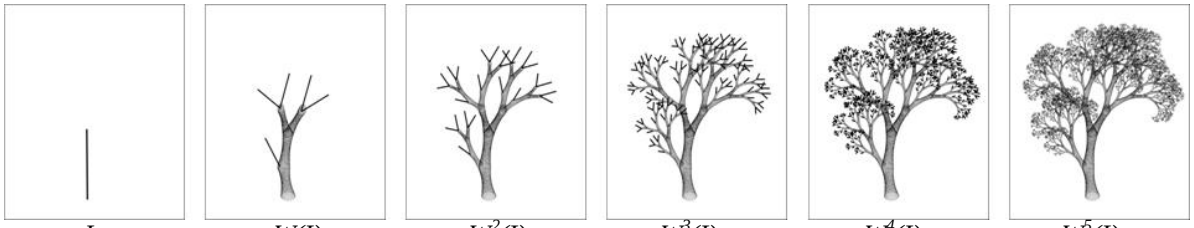

Figure 2: *Five iterations of the designed generator W(x).*

Leaping Iterated Function Systems

Usually, we generate a linear fractal by applying the function system on an object recursively. Suppose A_0 is the initial object, and $w_1, w_2, ..., w_n$ are n transformations applying on A_0, then the output $A_1 = W(A_0)$ is as follows:

$$A_1 = W(A_0) = \cup_{i_1=1}^{n} w_{i_1}(A_0).$$

In the next iteration, A_1 becomes an input data for $w_1, w_2, ..., w_n$ respectively, and we then have

$$A_2 = W(A_1) = \cup_{i_2=1}^{n} w_{i_2}(A_1) = \cup_{i_2=1}^{n} \cup_{i_1=1}^{n} w_{i_2} w_{i_1}(A_0).$$

There are n^2 objects found in A_2. Similarly, the output of *k*-th iteration is the input data for the (*k+1*)-th iteration, we have $A_{k+1} = W(A_k), k = 0,1,2,...$, that is,

$$A_{k+1} = W(A_k) = \cup_{i_1=1}^{n} w_{i_1}(A_0) = \cup_{i_k=1}^{n} \cup_{i_{k-1}=1}^{n} \cdots \cup_{i_1=1}^{n} w_{i_k} w_{i_{k-1}} \cdots w_{i_1}(A_0).$$

The number of objects in $\cup_{i_1=1}^{n} w_{i_1}(A_k) = \cup_{i_k=1}^{n} \cup_{i_{k-1}=1}^{n} \cdots \cup_{i_1=1}^{n} w_{i_k} w_{i_{k-1}} \cdots w_{i_1}(A_0)$ is $O(n^k)$ which grows exponentially at each iteration, and the computing consumption also grows exponentially.

Leaping Iterated Function Systems (LIFS) is proposed to reduce resource consumption during the iterations. For a given function system *W*(*x*), we first iterate *p* times to derive the outcome A_p as described above, convert the outcome into an image, and we then derive $A_{2p}, A_{3p}, ...$ recursively by consuming $O(n^p)$ resources only where *n* is the number of objects in the generator, *i.e*. the number of functions defined in the generator. Usually it is good enough if *p* = 2 or 3. As a summary, the algorithm of the Leaping Iterated Function System is given below.

The algorithm of LIFS:

1. Let *W*(*x*) be the function system represented by a generator.
2. Get $A_p = W(W^{p-1}(I))$, where *I* is an initiator (and *p* = 2 or 3).
3. Convert A_p into an image, and define a new generator $\overline{A_p}(x)$ by using the image of A_p as the only element in the pattern and the initiator *I* as the base.
4. Repeat Step 3, convert $\overline{A_p}(A_p)$ to an image, and define a new generator $\overline{A_{2p}}(x)$ by using this image as pattern and the initiator *I* as the base, then get the result $\overline{A_{2p}}(A_p)$.
5. Repeat the procedures

   $$\overline{A_p}(A_p), \overline{A_{2p}}(A_p), ..., \overline{A_{kp}}(A_p), ...$$

   Similar visual effects correspond to $A_{2p}, A_{3p}, ..., A_{(k+1)p}, ...$ respectively.
   Stop the iteration until the outcome is visually satisfied.

Two examples based on various structural cloning methods (SCM) are given Figure 3 and 4 below:

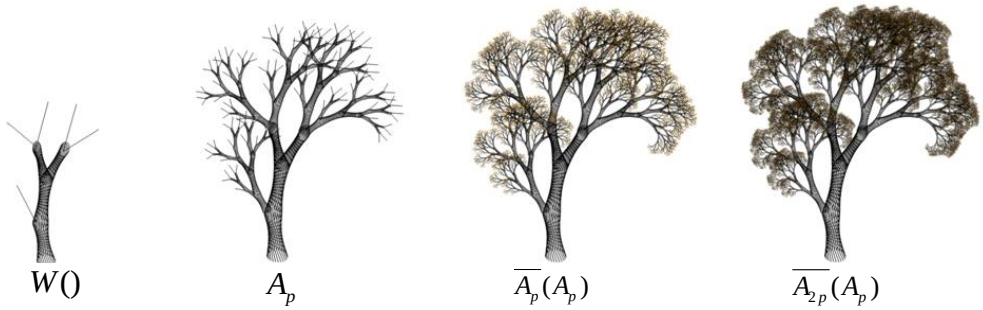

Figure 3: *A tree generated by using line-based cloning with p* = 3.

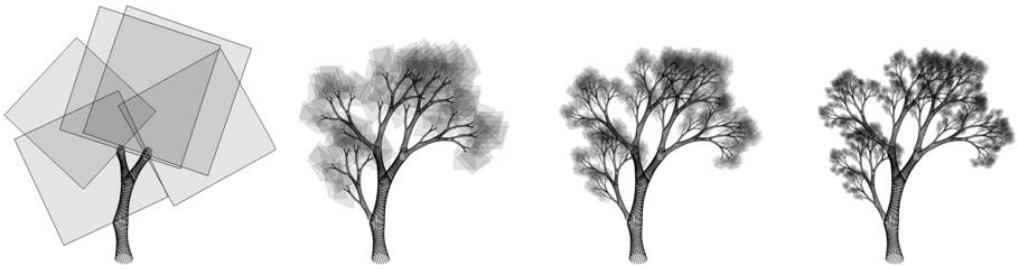

Figure 4: *A tree generated by using frame-based cloning with* $p = 3$

Conclusion

From the viewpoint of visual design, a bridge between mathematics and aesthetics is provided by combining the methodology of Structural Cloning Methods (SCM) and Leaping Iterated Function Systems (LIFS) together, that will make fractals more tractable [3]. In particular, Structural Cloning Methods (SCM) is a core function of the system AMA (Activate Mind and Attention) [4], a tool for instructional material design, an add-in for PowerPoint, and therefore LIFS works well on popular software. Mathematics is behind the mechanism, it is much more of a challenge for conveying properly natural feelings in this methodology.

Figure 5: *The landscape painting is designed on PowerPoint by using SCM and LIFS*

References

[1] M. F. Barnsley, *Fractals Everywhere*, 2nd ed, Academic Press, MA. 1993.
[2] M. F. Barnsley, J. Hutchinson, and Ö. Stenflo: *A fractal valued random iteration algorithm and fractal hierarchy*, Fractals, Vol. 13, No. 2. pp. 111-146. 2005.
[3] M. Chen, *Exploring Chaotic Patterns in Chinese Landscape Paintings by Structural Cloning*, 2010 Joint Mathematics Meetings, San Francisco, January 15.
[4] M. Chen, *An instructional Oriented Environment for Digital Material Design and Presentation*, National Education Quarterly, Vol. 48, No. 6. pp. 57-63. 2008. (in Chinese)
[5] G. W. Flake, *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex System and Adaption*, The MIT Press, Cambridge, 1998.
[6] B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman and Company, New York, 1982.
[7] H. O. Peitgen, H. Jürgens, D. Saupe, *Chaos and Fractals: New Frontiers of Science*, Springer-Verlag Inc., New York. 1992.
[8] P. Prusinkiewicz , A. Lindenmayer, *The algorithmic beauty of plants*, Springer-Verlag Inc., New York, 1990.