Determining an Iterated Function System Interactively

Mark R. Treuden
University of Wisconsin–Stevens Point
Department of Mathematics
Stevens Point, Wisconsin 54481–3897
e-mail: *m2treude@uwsp.edu*

In applying the Collage theorem to encode an image $\mathcal{A}_0$ as an iterated function system (IFS), contractive affine transformations w_i of the plane are chosen so that the union $\mathcal{A}_1 := \bigcup_{i=1}^{n} w_i(\mathcal{A}_0)$ of transformed images of $\mathcal{A}_0$ forms a collage that covers $\mathcal{A}_0$ as closely as possible. For precise definitions of these terms and related facts see [**1**, **2**, **3**, **5**]. In [**5**, p. 278] there is a description of how an interactive computer program[1] might be used to encode the image of a leaf:

> Using interactive input devices such as a mouse, knobs or even just the keyboard, the user of the program can easily manipulate the six parameters that determine one affine transformation. Simultaneously the computer displays the transformed copy of the initial polygon of the leaf. The goal is to find a transformation such that the copy fits snugly onto a part of the original leaf. Then the procedure is repeated, the user next tries to fit another affine copy onto another part of the leaf that is not yet covered by the first. Continuing in this way the complete leaf will be covered by small and possibly distorted copies of itself.

A vector graphics, computer-aided design (CAD) program capable of making components, such as *Generic CADD* [**4**], can be used to do most of those things described above. In this context a component is a portion of a drawing that can be saved, restored and manipulated as a single entity.

Consider the example of encoding a leaf. First, the leaf outline must be entered into the CAD program. This can be done either by holding a paper cutout of the leaf onto the computer screen and then drawing around it or by loading a digitized[2] leaf boundary. Now draw the zero vector $\mathbf{0} := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and the standard basis vectors $\mathbf{e}_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{e}_2 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Figure 1 shows the outline of a raspberry leaf $\mathcal{A}_0$ and the coordinate system $\mathbf{0}, \mathbf{e}_1, \mathbf{e}_2$.

Next, a component is made from this figure. Components can be scaled independently in the horizontal and vertical directions, and then rotated. Once these values are chosen from within the CAD program, a mouse can be used to visually place the transformed image on top of $\mathcal{A}_0$. This has real advantages over an IFS which is constructed using images of rectangles because of the closer connection with the Collage theorem. Figure 2 shows a typical set $w_i(\mathcal{A}_0)$ and a closeup of the images of $\mathbf{0}, \mathbf{e}_1, \mathbf{e}_2$

[1] See the appendix for a discussion of a search for such a program.

[2] If an image of a leaf is obtained by scanning a picture, then a digitizing program such as WinDig 2.0 (freeware) can be used to digitize the leaf boundary.

under w_i. Figure 3 shows both $\mathcal{A}_0$ and $\mathcal{A}_1$, along with an outline of the leaf stem.

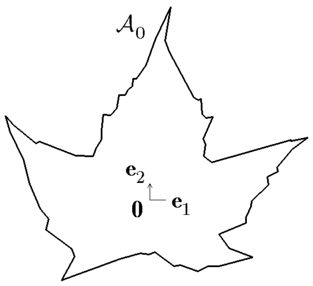

Figure 1.

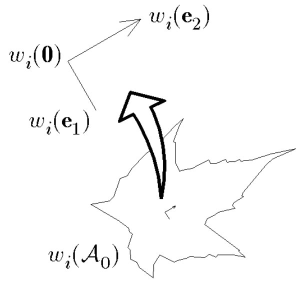

Figure 2.

The images of $\mathbf{0}, \mathbf{e}_1$, and $\mathbf{e}_2$ under any transformation w_i as shown in Figure 2 can now be directly measured in the CAD program, thus determining the parameters of w_i because

$$w_i\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} w_i(\mathbf{e}_1) - w_i(\mathbf{0}) & w_i(\mathbf{e}_2) - w_i(\mathbf{0}) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + w_i(\mathbf{0}).$$

Finally, to render the image of the attractor $\mathcal{A}$ of the IFS, a program such as *Fractint* [**6**] can be used; see Figure 4. The *Fractint* IFS code for this example is shown below[3].

```
Raspberry_Leaf { ; Mark Treuden (m2treude@uwsp.edu)
 0.5        0.0        0.0        0.75       0.515673  3.278964  0.3234152
 0.25       0.649519 -0.433013   0.375      2.817174 -0.697652  0.3234152
 0.286788 -0.614364   0.409576   0.430182 -2.587866 -0.8023     0.3234152
 0.15       0.0        0.0        0.15       0.027476 -2.720842  0.01940491
-0.029886  0.034862 -0.002615 -0.398478   0.174539 -4.869014  0.01034928
 }
```

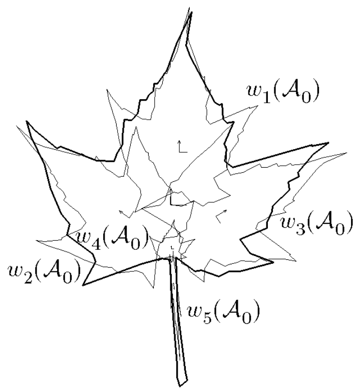

Figure 3.

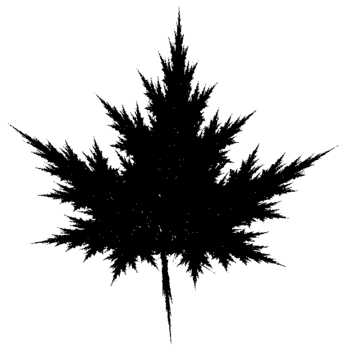

Figure 4.

[3]IFS codes for the attractors and CAD files can be obtained either from the author or from the site where this paper is published.

In some CAD programs, a component may only be scaled vertically and/or horizontally (with negative scale factors allowed) and then rotated. Therefore an IFS obtained by using such components consists of affine transformations of the form

$$w\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} r_1 \cos\theta & -r_2 \sin\theta \\ r_1 \sin\theta & r_2 \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}.$$

Here the scale factors r_1 and r_2 can be negative, in which case reflections are involved. Note that such mappings do not include shear transformations. It may be possible to eliminate this restriction, but it probably depends on the CAD program used.

Using a CAD program to experiment with interactively encoding an image as an IFS can provide a "hands-on" way to gain insight into the Collage theorem. For example, Figures 5 and 7 shows collages which overlay the outline of a juniper branch. The collage in Figure 7 is "closer" to the outline than that in Figure 5 and so the attractor of the corresponding IFS shown in Figure 8 should be "closer" to the figure of the branch than the attractor given in Figure 6.

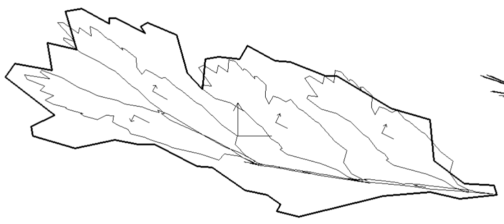
Figure 5.

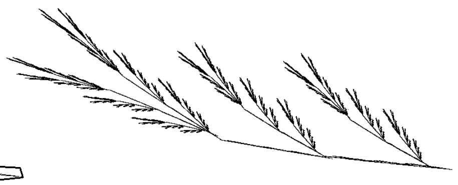
Figure 6.

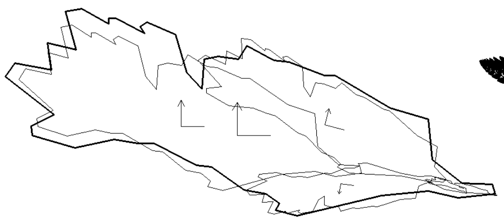
Figure 7.

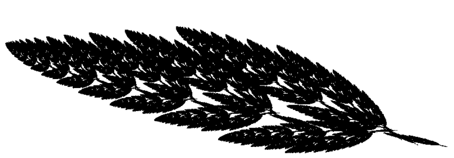
Figure 8.

Appendix

After reading the quoted portion of [5] I thought there must exist some satisfactory, inexpensive (i.e. freeware or shareware) IBM-compatible version of such a program. The most important feature I looked for was the ability to form a collage by interactively placing affine copies of the *original figure*, and not by using affine copies of rectangles.

I searched many internet sites, including the Spanky Fractal Database (`http://spanky.triumf.ca/www/welcome1.html`), but didn't find anything acceptable. The most promising program was *Modified Iterated Function Systems* (*MIFS*) which claims to have the capability to load a scanned image and then apply some kind of "vectorization" process to encode a "real world structure." However, the beta version of *MIFS* wouldn't even load a graphics (BMP) file, and the program and documentation are written in Polish. If the reader knows of a good program with the features described above, please contact the author.

References

1. Michael F. Barnsley, *Fractals Everywhere.* Academic Press, San Diego, 1988.
2. Michael F. Barnsley, Lyman P. Hurd, *Fractal Image Compression.* AK Peters Ltd., Wellesley, MA, 1993.
3. Kenneth Falconer, *Fractal Geometry: Mathematical Foundations and Applications.* Wiley, New York, 1990
4. *Generic CADD*, Version 6.0, Autodesk Retail Products, Bothell, WA, 1992.
5. Heinz-Otto Peitgen, Hartmut Jürgens, Dietmar Saupe, *Chaos and Fractals: New Frontiers of Science.* Springer-Verlag, New York, 1992.
6. Stone Soup Group, *Fractint*, Version 19.5, Spanky Fractal Database, `http://spanky.triumf.ca/www/fractint/fractint.html`, 1996