Fractals with Variable Scaling Factors using IFS

B Dinesh Rao
Associate Professor
SOIS, Manipal
Manipal University

ShridharNayak
Assistant Professor (Sel grade)
SOIS, Manipal
Manipal University

Sathyendranath Malli
Assistant Professor (Sel grade)
SOIS, Manipal
Manipal University

ABSTRACT
Fractals are self-similar images. There are many techniques for generating fractals. IFS [1] are one of them. IFS use a set of linear transformations for generation of fractals. IFS have been modified to difference based IFS[2] to use differences in distance between the points to figure out the new point. We propose a method of modifying IFS such that variable scaling factors can be used in transforms for generating pleasant looking fractals. Fractals with variable scaling factors have been developed as a function of distance from the fixed points.

Keywords
Difference, Variable, Fractal, IFS, Scaling

1. INTRODUCTION
The use of a system of affine transforms to define a fractal object has been described by Barnsley [3]. A system of transforms W_i can be written as

$$W_i: Z \rightarrow T_iZ + V_i$$

Where

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad V = \begin{bmatrix} e \\ f \end{bmatrix}$$

and

$$Z = \begin{bmatrix} x \\ y \end{bmatrix}$$

That is if $Z = \begin{bmatrix} x \\ y \end{bmatrix}$

then $x = ax + by + e$

$y = cx + dy + f$

The set of transforms need to be contractive and there exists a unique attractor set containing infinitely many points Z. For graphical purposes we say that there is a fixed set of pixels which approximate this attractor [4][5]..

Our proposed technique is intended to extend the transforms to have variable scaling factors. We wish to demonstrate that a set of transforms with variable scaling factors also generate a unique attractor set comprising of deterministic number of points.

2. MATERIALS AND METHODS

2.1 Iterated Function Systems
We will first look at a method for generating IFS. Consider the following three transformations.

1. x= 0.5 x + 0
   y = 0.5y + 0
2. x =0.5x + 0
   y = 0.5y + 150
3. x = 0.5x + 150
   Y= 0.5y +0 (Equation 1)

point x', y'. If the above procedure is repeated on the point x', y', we get a new point x", y". This is repeated for fixed number of times

```
for(i=0;i<95000;i++){

ran=rand()%3;

switch(ran)

    {

        case 0:

                x =  0.5 * x + 0;

                y =  0.5 * y + 0;

                break;

        case 1:

                x =  0.5 * x + 0;

                y =  0.5 * y + 150;

                break;

        case 2:

        x =  0.5 * x + 150;

                y =  0.5 * y + 0;

                break;

        }

        putpixel(x,y,1)
```

}

C code for simple IFS

The result is the following fractal depicted in figure 1. It is called the Sierpensky's triangle.

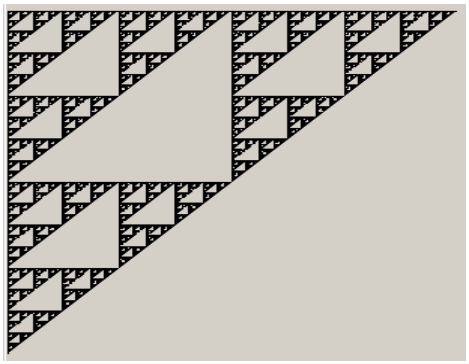

Figure 1: Sierpensky's triangle

The affine transformations are of the form

$$x' = ax + by + c$$

and $$y' = dx + ey + f$$

We use six numbers to represent a transformation {a,b,c,d,e,f}[2]. Hence, the above example would have {0.5,0,0,0,0.5,0},{0.5,0,0,0,0.5,150},{0.5,0,150,0,0.5,0} representing the three transforms.

2.2 Difference based Iterated function system

We modify IFS methodology slightly to get meaningful images in a more intuitive way. We will call this difference based IFS. In IFS, each function is scaled by a constant between 0 and 1. It is generally difficult to predict the result of application of this scaling factor. In difference based IFS,we are going closer to a fixed point, by a measure; this measure is either half the distance or $3/4^{th}$ of the distance, etc which gives us some idea as to what is happening in the image. In difference based IFS, we are always going closer to a fixed point by a measure. If we move away from the fixed point, the image will not converge and a random set of points are generated. The points also move outside the screen.

To convert an IFS to difference based IFS, we obtain an invariable point for every transformation of IFS i.e., next point x' = x and y' = y. Consider the above three transformations in equation 2. We notice that the invariant points are (300,300), (0,300) and (300, 0) for the three transforms respectively. If we apply transform to these points, they do not change. Hence, they are called invariants or fixed points.

Given an invariant point, we develop IFS like equations as follows:

x = x + (invariant (x) –x)*a + b

y = y+ (invariant (y) – y)*c + d (Equation 2)

Here, we are computing the new point as a linear function of distance between the current point and the invariant point. A constant value can be added resulting in translation.

The new set of transforms for equation 2 will be

1. x = x + (300 – x) * 0.5 + 0
   y = y + (300 - x) * 0.5 + 0
2. x = x + (0 – x) * 0.5 + 0
   y = y + (300 – x) * 0.5 + 0
3. x = x + (300 – x) * 0.5 + 0
   y = y + (0 – x) * 0.5 + 0

We use the same method as in IFS to generate the set of points in the image. We select an arbitrary point (x, y) and select one of the three affine transformations in random, apply it to the point (x, y) we get a new point (x', y'). The above procedure is repeated on the point (x', y) and, we get a new point (x", y") and so on. These points are plotted for some fixed number of iterations, say 100000. We get figure 1 as a result of the equations shown above. The two techniques i.e., IFS and difference based IFS are equivalent and are inter-convertible.

Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

2.3 IFS variant with variable scaling factors

In IFS or IFS variant, the scaling factors are fixed. Given a transformation, the scaling factors are constant even though they have to be between 0 and 1. The scaling factor in the Sierpensky's triangle has been changed to be a variable one which is dependent on the distance from the fixed point. The pseudo code is described below.

1. Array points[3][2] ={{400,50},{50,50},{50,400}}
2. x= 100, y=75;
3. Loop 100000 times the following code till line 11.
4. Choose a random integer number between 0 and 2 into variable ran.
5. Variable p is assigned difference of points [ran] [0] and x. All three transforms are similar.
6. j= p*p/775.0;
7. x = x+ j
8. Variable q is assigned difference of points [ran] [1] and y. All three transforms are similar.
9. j= q*q/775.0
10. y= y+ j
11. Plot the point (x,y)

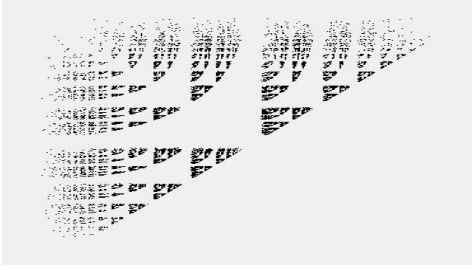

Figure 2 Variable Scaling

The scaling factors can vary as a proportion to the distance from the fixed point. Further the distance, smaller the scaling factor.

In the next experiment we inverted the scaling factor and achieved interesting results. The shorter the distance from the fixed point, the larger was the scaling factor.

1. Array points[3][2] ={{400,50},{50,50},{50,400}}
2. x= 100, y=75;
3. Loop 100000 times the following code till line 11.
4. Choose a random integer number between 0 and 2 into variable ran.
5. Variable p is assigned difference of points [ran] [0] and x. All three transforms are similar.
6. j= (p*(1 - p)/ 600
7. x = x+ j
8. Variable q is assigned difference of points [ran] [1] and y. All three transforms are similar.
9. j= q*(1 - q)/600
10. y= y+ j
11. Plot the point (x,y)

Pseudo codefor inverted variable scaling

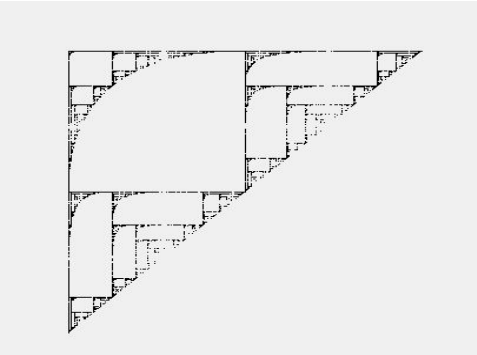

Figure 3: Inverted Variable scaling

3. APPLICATIONS

We can see the generation of the figure 4 with variable scaling factor for only one of the three transforms. Here the scaling factor is larger when the moving point is closer to the fixed point. Figure 5 is the result of variable scaling factors to two of the three transforms. We see variable nature of the curvature also. Figure 6 is twig with transforms. Figure 7 is the result of applying variable scaling factor to one transform corresponding to the left branch. A more realistic image has been obtained when compared to the one with only linear transforms.

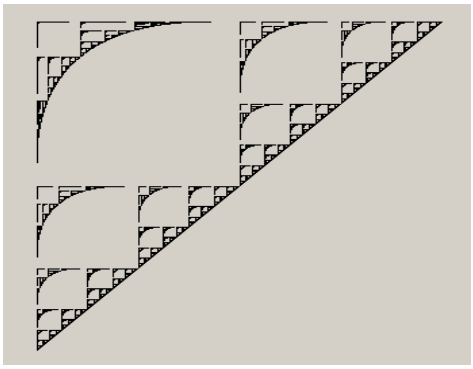

Figure 4:Inverted variable scaling with one transform altered

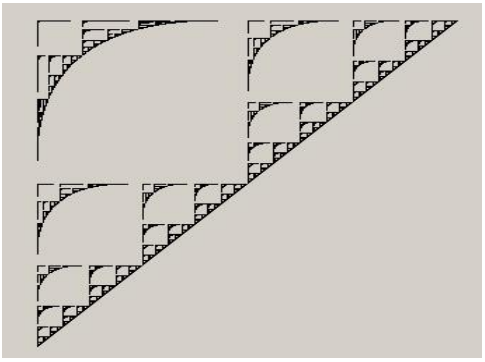

Figure 5:Inverted variable scaling with two transforms altered

Figure 6: Linear twig

Figure 7: Variable scaling applied to twig

4. RESULTS AND DISCUSSION

We see that good looking fractals can be generated with variable scaling factors. We find some blurring towards scaling factors close to 0 and close to 1. We can avoid that region in our transforms. Scaling factors can vary as a distance from a line, angle formed with a line and also distance from a triangle.

We find interesting figures with respect to scaling factors which are greater if the distance from the fixed point is smaller from the fixed point. Inverted scaling can be used to generate complex fractals. Variable scaling factors give realistic nature to fractal images.

5. CONCLUSIONS

Traditionally, an IFS has a fixed scaling factor for x and y quantities. These have to be in the range between 0 and 1 for convergence. By varying the scaling factors as a measure of distance from the fixed point, good looking fractals can be generated. Scaling factors can vary as a distance from a line, angle formed with a line and also distance from a triangle etc. Blurring occurs towards scaling factors close to 0 and close to 1. These regions can be avoided in the transforms. Interesting figures are generated when scaling factors are greater if the distance from the fixed point is smaller.

6. REFERENCES

[1] Frederic Raynal, Evelyne, Lutton, Pierre Collet, Manipulation of Non-Linear IFS Attractors Using Genetic Programming (1999), Proceedings of the Congress on Evolutionary Computation

[2] Dinesh B Rao, Deepak Rao B and U C Niranjan. Article: Difference based Non-Linear Fractals using IFS. International Journal of Computer Applications 80(13):38-42, October 2013. Published by Foundation of Computer Science, New York, USA

[3] Barnsley, Fractals everywhere, Academic Press 1988.

[4] Stephen Demko, Laurie Hodges and Bruce Naylor , Construction of Fractal Objects with Iterated Function Systems, ACM SIGGRAPH Computer Graphics, 1985.

[5] Lu Ning, Fractal Imaging, academic Press 1997.