CSE585/EE555: Digital Image Processing II

Computer Project # 5

Fractal Generation Using Iterated Function Systems

Mandar Parab, Amogh S. Adishesha, Lyuzhou Zhuang

Date: 4/21/2017

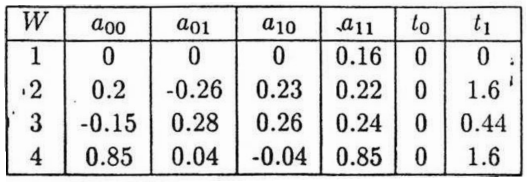
A. Objectives

Learn to generate a fractal image using iterated function systems (IFS). Observe how the fern image is generated through iterations and how changing probabilities affects the result.

B. Methods

In this project, we are to implement IFS (Iterated Function System) in matlab and generate a fractal image. At every iteration we are going to apply one of the 4 affine transformations (chosen at a probability, formula and parameter values are given below) to the result point from last iteration and generate a new point which will then be marked on the matrix. We are going to try 2 sets of probabilities associated with the transformation matrices W_i which determines the chance a transformation matrix be chosen at any iteration. The first set of probabilities (probability set #1, given in the problem) associated with [W_1, W_2, W_3, W_4] is [0.2, 0.35, 0.35, 0.1]. We then test our code with different probability sets (probability set #2) which is [0.4, 0.1, 0.1, 0.4] and (probability set #3) which is [0.2, 0.2, 0.1, 0.5]

$$w(x,y) = w\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_0 \\ t_1 \end{bmatrix}$$



W	a_{00}	a_{01}	a_{10}	a_{11}	t_0	t_1
1	0	0	0	0.16	0	0
2	0.2	-0.26	0.23	0.22	0	1.6
3	-0.15	0.28	0.26	0.24	0	0.44
4	0.85	0.04	-0.04	0.85	0	1.6

Below is the MATLAB program flow (Run main.m)

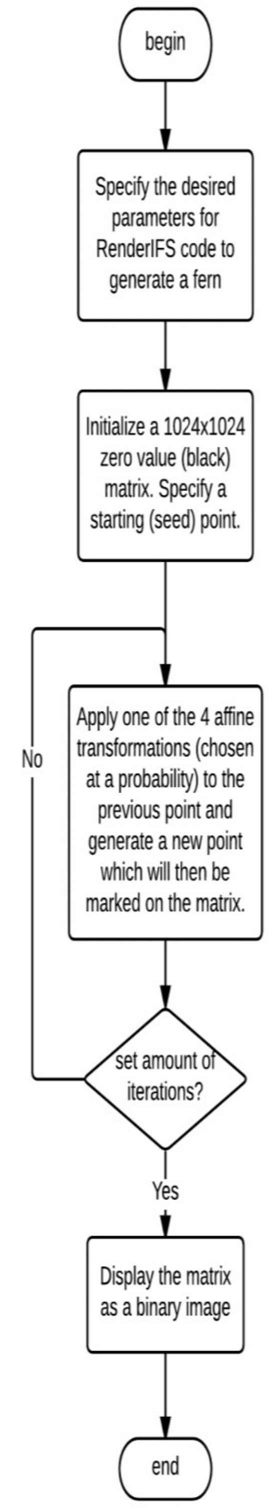

begin
Specify the desired parameters for RenderIFS code to generate a fern
Initialize a 1024x1024 zero value (black) matrix. Specify a starting (seed) point.
Apply one of the 4 affine transformations (chosen at a probability) to the previous point and generate a new point which will then be marked on the matrix.
No
set amount of iterations?
Yes
Display the matrix as a binary image
end

Figure 2.1 project flowchart

1. Specify the desired parameter values for rendering the IFS code, including affine transformation matrices, number of iterations and probability set (array).
2. Initialize a 512*512 zero-value matrix. Initialize a starting (seed) point (x,y). Specify save points for saving images after set amount of iterations.
3. At each iteration:
   1) Produce a random number in the interval [0,Arand].
   2) Determine which transformation matrix W_i is to be use at this iteration
   3) Apply the chosen transformation matrix W_i to the seed point vector (x,y)
   4) Check if the X and Y values are within the max and min range.
   5) Calculate which pixel corresponds to this iteration.
   6) Set pixel (x,y) on the matrix (image) to be 1.
   7) The updated vector (x,y) will be the seed point for next iteration.
4. Display the final matrix as a binary image.

C. Results

Figures 3.1.1- 3.1.8 provide the results for various iterations in the increasing order. It can be seen that with increasing number of points, a better fractal is achieved. All results are of size 512x512 . The probability set corresponding to these results is **[0.2, 0.35, 0.35, 0.1]**

Figures 3.2.1- 3.2.8 provide the results for various iterations in the increasing order. All results are of size 512x512 . The probability set corresponding to these results is **[0.4,0.1, 0.1, 0.4]**

Figures 3.3.1- 3.3.8 provide the results for various iterations in the increasing order. All results are of size 512x512 . The probability set corresponding to these results is **[0.2, 0.2, 0.1, 0.5]**

1. Results of applying IFS with probability set #1

Figure 3.1.1 Result after applying IFS with probability set #1 for 100 iterations

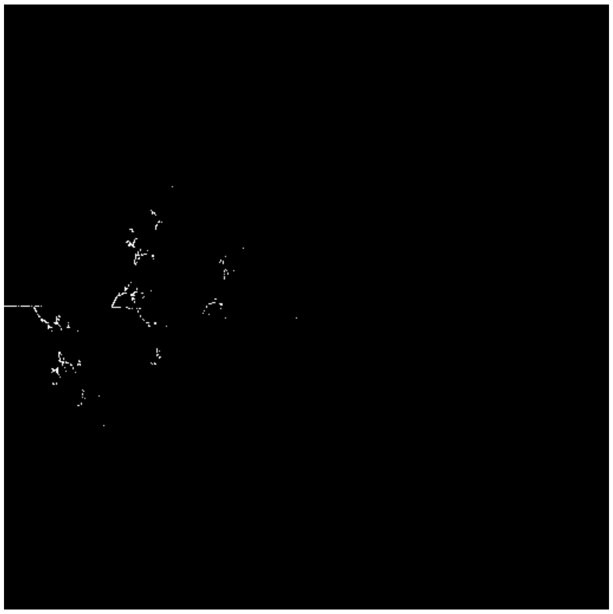


Figure 3.1.2 Result after applying IFS with probability set #1 for 5000 iterations

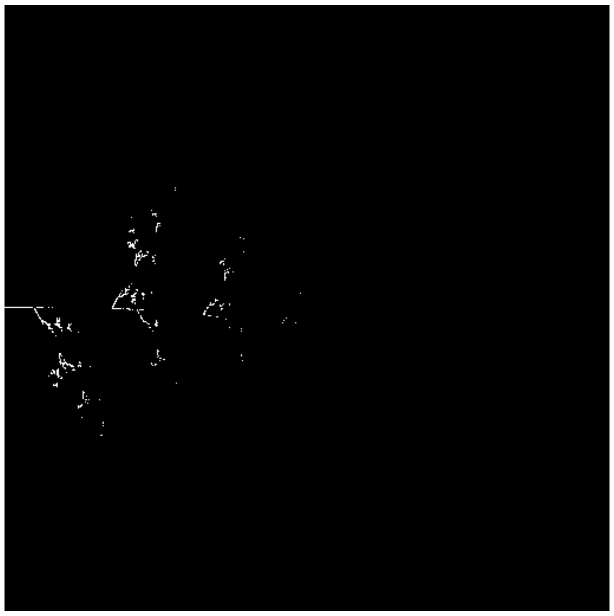


Figure 3.1.3 Result after applying IFS with probability set #1 for 10,000 iterations

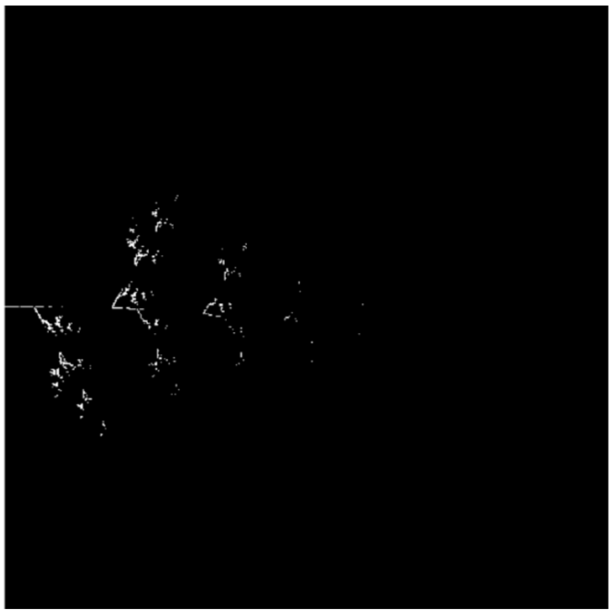


Figure 3.1.4 Result after applying IFS with probability set #1 for 20,000 iterations

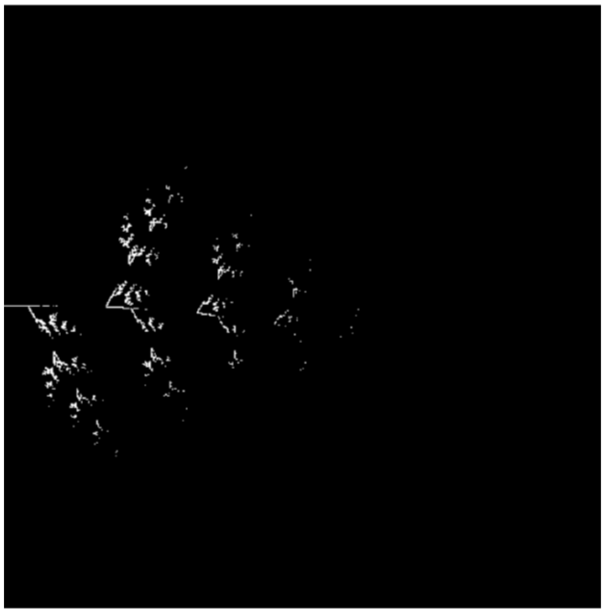


Figure 3.1.5 Result after applying IFS with probability set #1 for 100,000 iterations

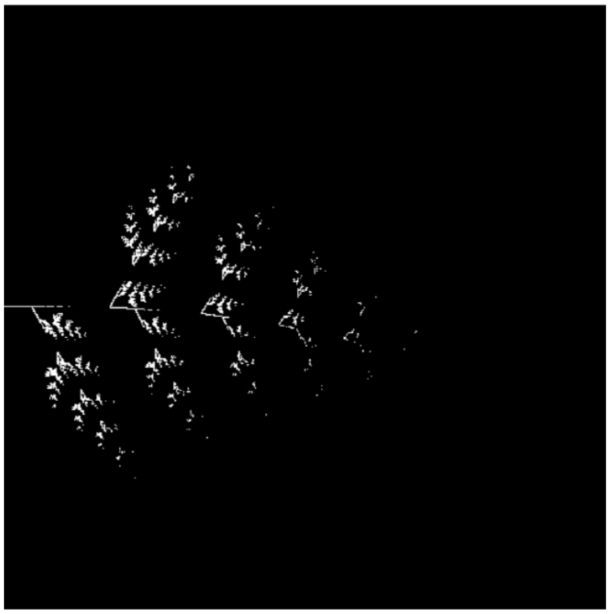


Figure 3.1.6 Result after applying IFS with probability set #1 for 500,000 iterations

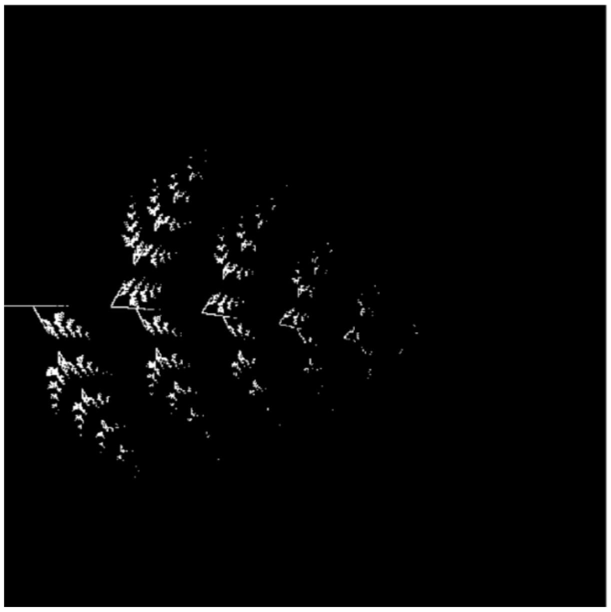


Figure 3.1.7 Result after applying IFS with probability set #1 for 1,000,000 iterations

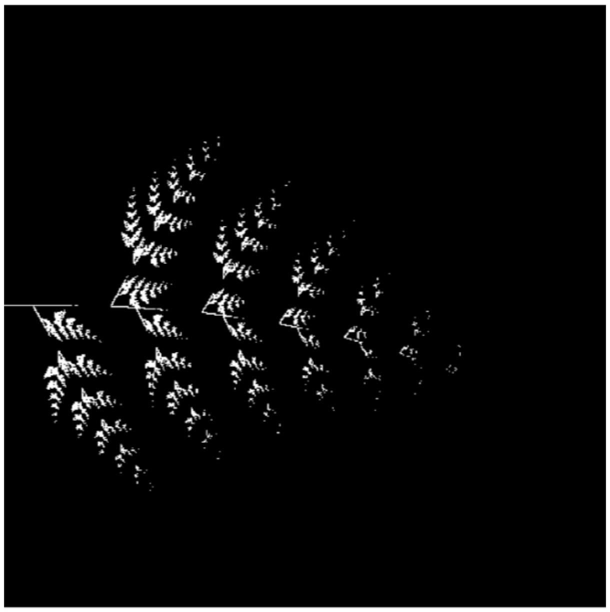


Figure 3.1.8 Result after applying IFS with probability set #1 for 10,000,000 iterations

2. Results of applying IFS with probability set #2

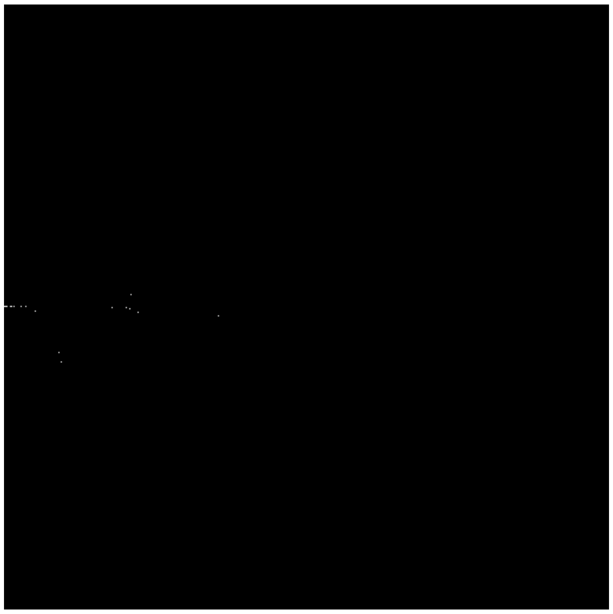
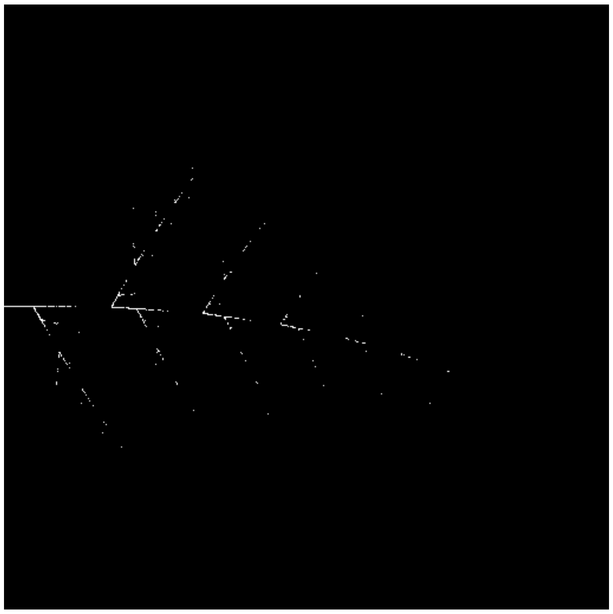



Figure 3.2.1, Figure 3.2.2 Result after applying IFS with probability set #2 for 100 and 5000 iterations

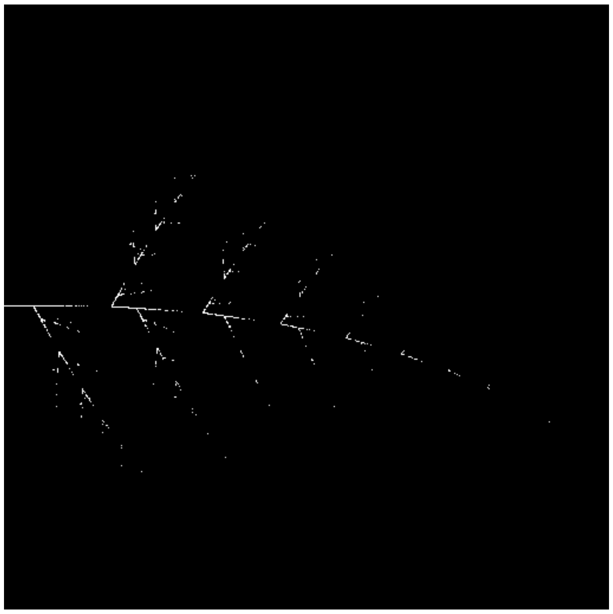
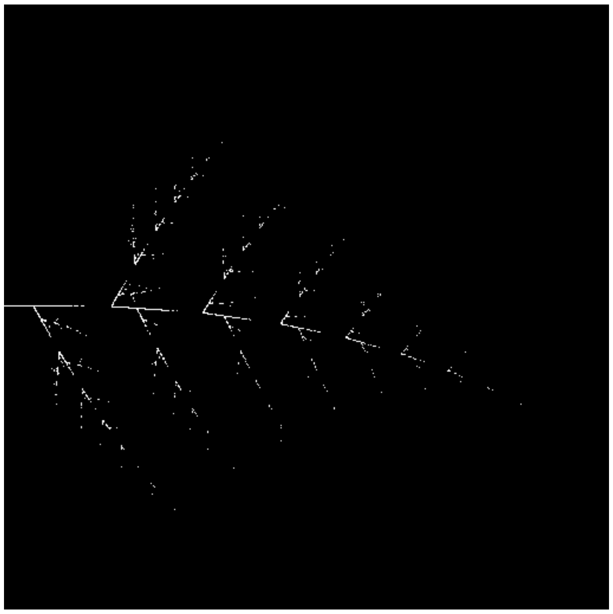



Figure 3.2.3, Figure 3.2.4 Result after applying IFS with probability set #2 for 10,000 and 20,000 iterations

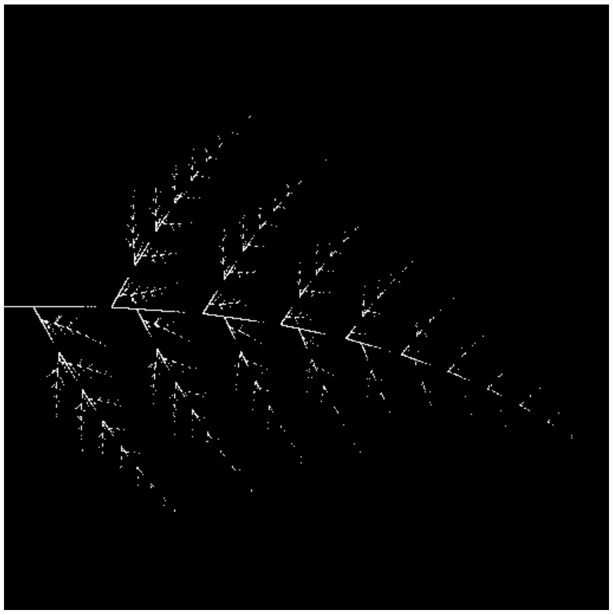
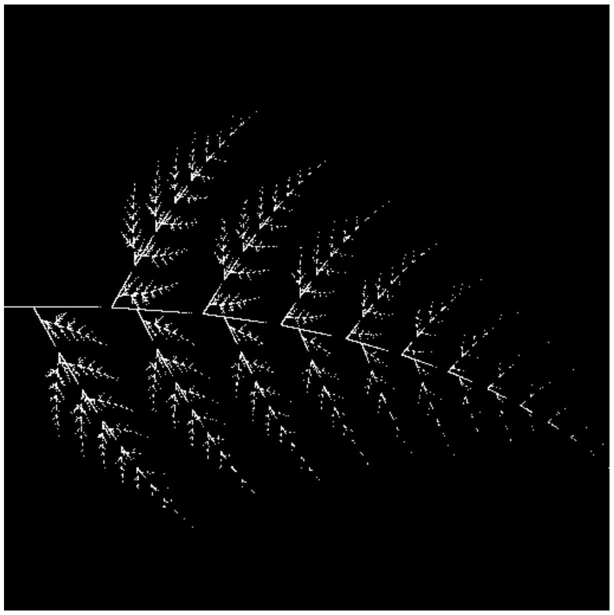



Figure 3.2.5, Figure 3.2.6 Result after applying IFS with probability set #2 for 100,000 and 500,000 iterations

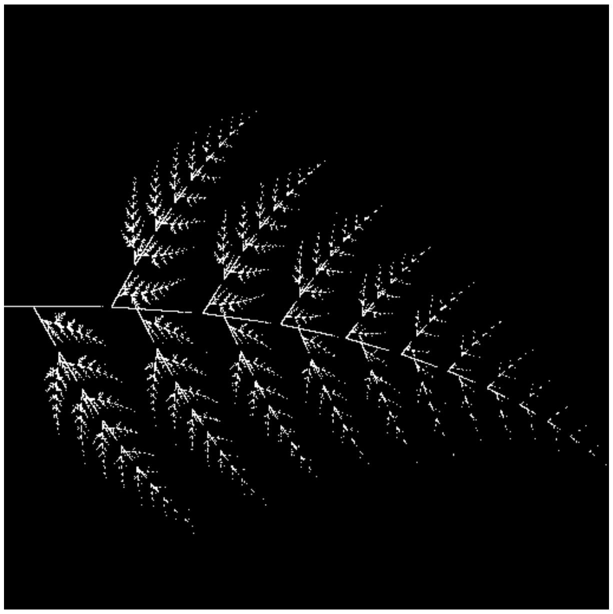
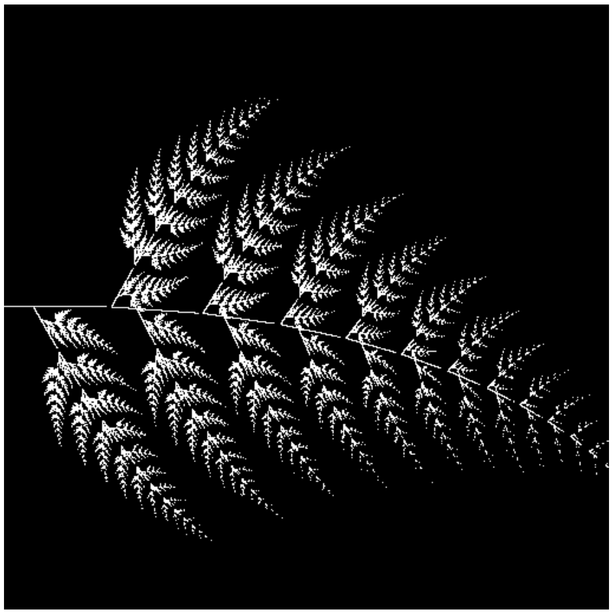



Figure 3.2.7, Figure 3.2.8 Result after applying IFS with probability set #2 for 1,000,000 and 10,000,000 iterations

3. Results of applying IFS with probability set #3

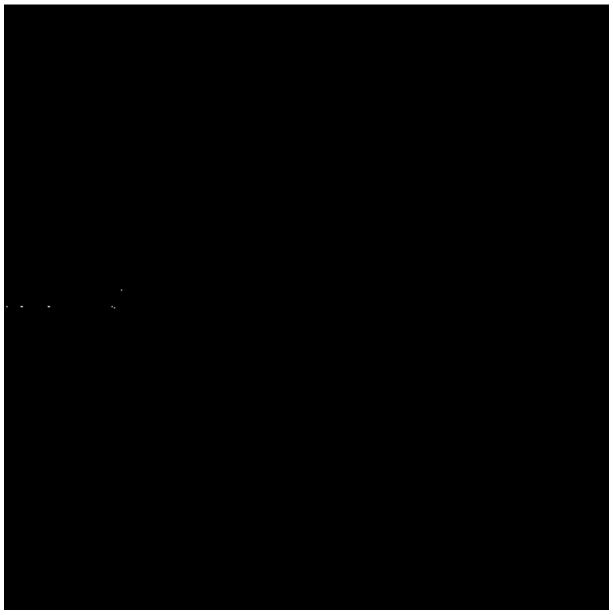
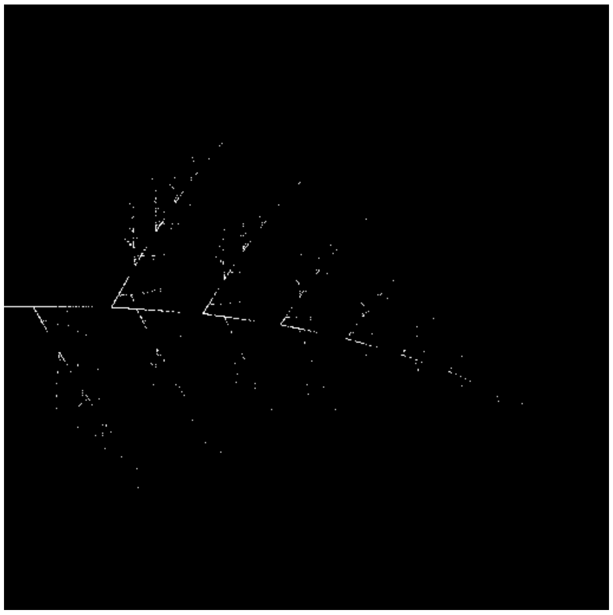



Figure 3.3.1, Figure 3.3.2 Result after applying IFS with probability set #3 for 100 and 5000 iterations

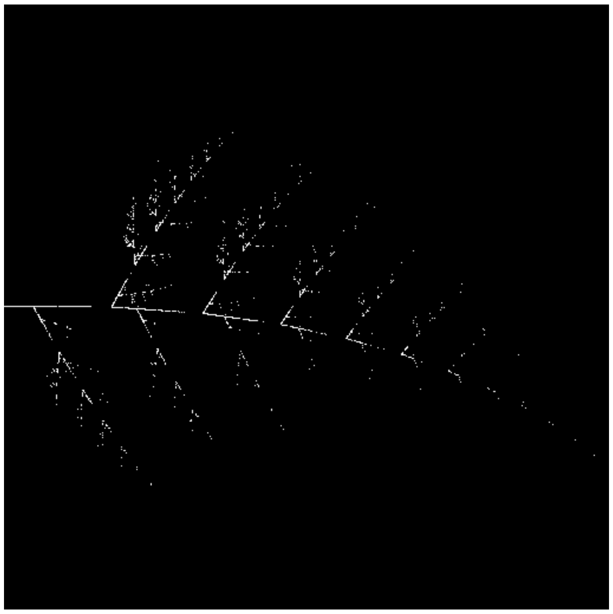
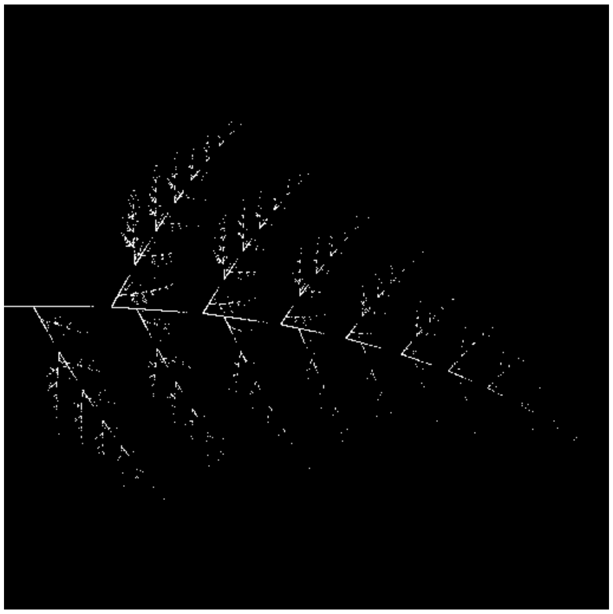



Figure 3.3.3, Figure 3.3.4 Result after applying IFS with probability set #3 for 10,000 and 20,000 iterations

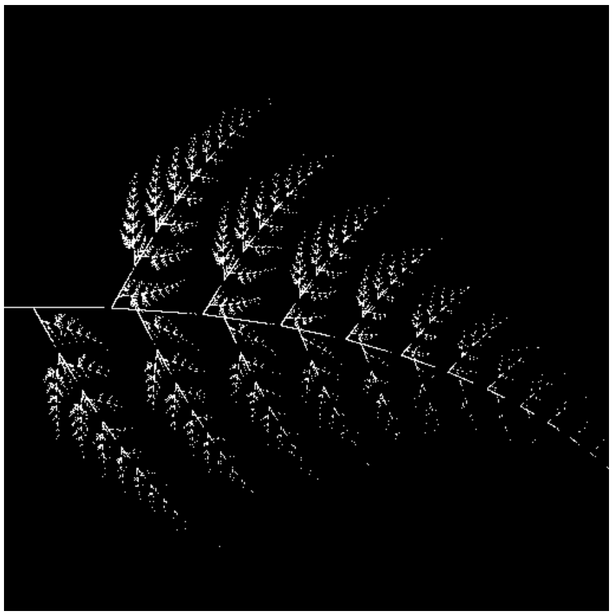
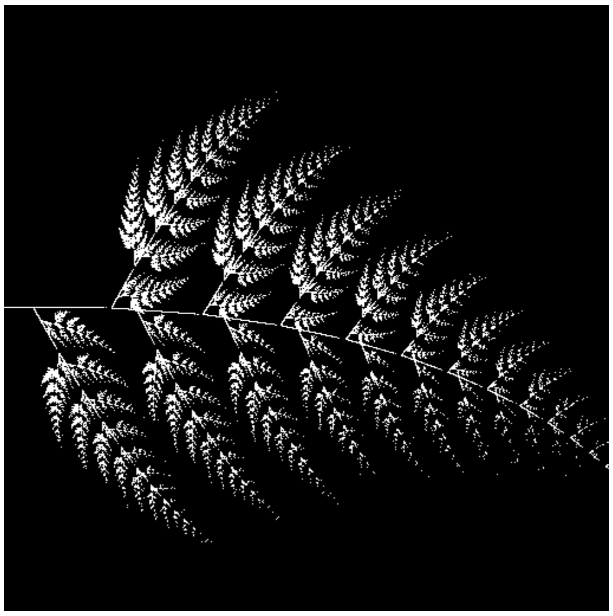



Figure 3.3.5, Figure 3.3.6 Result after applying IFS with probability set #3 for 100,000 and 500,000 iterations

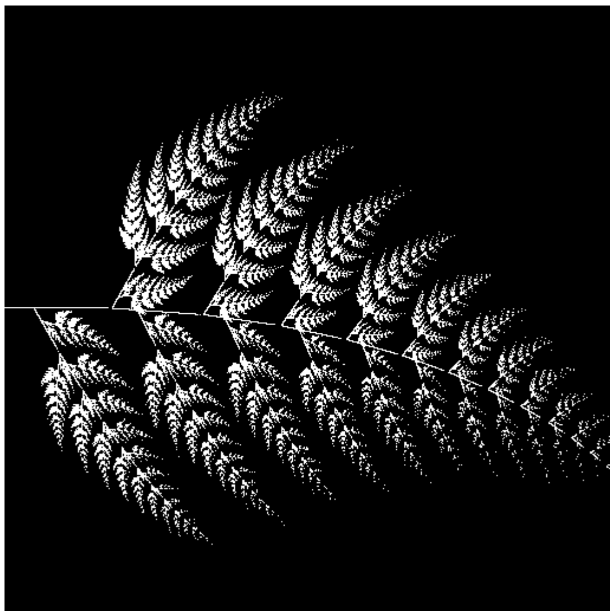
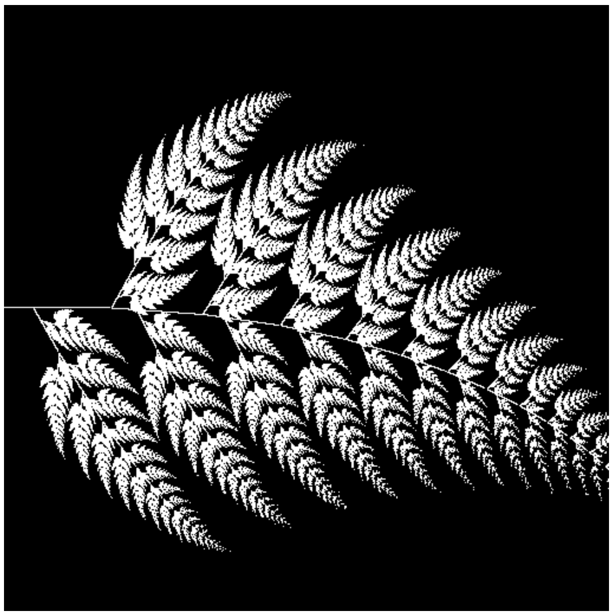



Figure 3.3.7, Figure 3.3.8 Result after applying IFS with probability set #3 for 1,000,000 and 10,000,000 iterations

From the above results, we see that both sets of images were developed towards the same fractal shape (fern), regardless of probability set values. As long as the transformation matrices are the same, the very end result shapes are the same. The probabilities determine the direction of spread at every iteration. Equal probabilities (0.25) would provide equal density spread in all directions at each iteration. With the number of iterations being high, the probability does not play a dominating role. Also, though the program requires very little information to generate the fern, it takes a lot of time to generate it, compared to that of storing all pixels of the fern image which could be read and displayed immediately.

D. Conclusions

IFS is useful for storing/generating fractal images, like ferns. It's a space-efficient (but time-consuming) algorithm to represent (store) fractal-like objects/shapes. The shape of the fractal is determined by the affine transformations matrices, regardless of probability set. However, changing probabilities changes the amount of iterations it needs to generate the fractal. They should be carefully chosen to minimize computational load, thus program running time.