Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 8** : 1–17.
ISSN-0037-8712
doi:10.5269/bspm.83936

# Fractal Design Pattern Generation from Non-Affine IFS and Machine Learning-Driven Clustering-Based Segmentation

Sana Abdulla and K. Mahipal Reddy

ABSTRACT: In this paper, we present a novel application of fractals using different Iterated Function Systems (IFS), which employ non-linear transformations for design patterns for garments or for wall or floor interiors. Here we provide a template for the generation of fractal pattern designs using both contractive IFS and non-contractive IFS, relying on the condition that non-contractive IFS produce self-similar attractors under certain circumstances. To segment the intricate patterns generated by IFS into distinct styles, we employ certain clustering techniques in machine learning, such as K-means clustering and self-organizing map (SOM) clustering. Furthermore, to quantitatively measure the pattern's complexity and self-similarity, the fractal dimension of the fractal design pattern is computed via the box-counting method, allowing for a more precise evaluation of the pattern's intricacy and aesthetic appeal. This method of generating fractal pattern designs and segmenting them using machine learning algorithms enables the fashion industry to create innovative designs with minimal programming expertise, even without traditional fashion or artistic skills.

Keywords: Fractals, Iterated Function System, K-means clustering algorithm, Self-Organizing Maps, fractal dimension.

## Contents



## 1. Introduction

Fractal geometry, introduced by Mandelbrot [13, 14, 15], has revolutionized our understanding of natural forms by highlighting self-similar patterns across scales. Hutchinson's study [8] on fractals and self-similarity further explores these mathematical constructs, demonstrating their relevance in diverse fields such as mathematics and art. Barnsley's findings [3, 4] consolidate these ideas, providing a comprehensive framework for understanding and applying fractal concepts. The scaling and shaping of fractals

2020 *Mathematics Subject Classification:* 35B40, 35L70.
Submitted May 02, 2026. Published June 05, 2026.

are crucial in generating self-similar structures, where affine transformations control the geometric properties of fractal curves [20]. These transformations help preserve the shape while modifying the size and orientation, making them essential for fractal-like Bézier curve construction. The application of Iterated Function Systems (IFS) in real-time image synthesis is discussed by Nikiel [18], illustrating the practical utility of fractal geometry in digital media. Many classic as well as recent researches delve into the construction of fractal objects using IFS, emphasizing computational methods to generate intricate patterns [6, 12]. Nikiel [17] extends this discussion to true color images, showcasing the versatility of IFS in visual representation. The authors present methods for determining the spatial boundaries of IFS attractors with speed and accuracy, crucial for applications in computer graphics [7]. Sherman and Hart [21] explore direct manipulation techniques for recurrent models, enhancing interactive capabilities in fractal design. Machine learning algorithms have become essential for solving complex predictive modeling problems across various scientific and engineering domains [5]. K-means clustering is widely recognized for its simplicity and efficiency in segmenting large datasets into meaningful clusters, making it a popular choice in various applications [22]. This method enables the extraction of significant patterns from data, which is crucial for tasks such as customer segmentation and behavior analysis. Similarly, Self-Organizing Maps (SOM) are instrumental in visualizing high-dimensional data and uncovering intrinsic patterns through unsupervised learning [19]. The ability of SOM to perform topological mapping and clustering simultaneously makes it a powerful tool for data analysis and pattern recognition.

Garment design has traditionally relied on artistic intuition and manual craftsmanship. With advancements in computational methods, there is growing interest in utilizing mathematical techniques to innovate fashion patterns [9, 11, 16, 23]. In their paper, [24] Wang et al. explore the application of fractal graphics in garment pattern design, utilizing both complex dynamic systems and L-systems to generate intricate artistic patterns. They demonstrate the practical use of these fractal patterns in clothing design through digital printing technology, highlighting the feasibility and creative potential of this approach in fashion design. Affine transformations have been utilized in fractal pattern generation, as demonstrated by [1].

In this paper, we rely on the same theme of generating fractal garment patterns, but through a non-affine Iterated Function System. IFS, originally developed in fractal geometry, offers a systematic approach to generating intricate patterns through repeated transformations.

Generating fractal patterns using IFS is often more advantageous than L-systems and other methods because IFS provides a more systematic and flexible framework for creating highly detailed and complex patterns through repeated geometric transformations, leading to greater control over the transformations, scaling of patterns, and final design, as well as the ability to model a broader range of intricate patterns. Furthermore, our work explores how these patterns can be further analyzed and segmented using K-means clustering and SOM clustering, providing a structured framework for exploring design variability and customer preferences. In this study, we build upon the exploration of fractal patterns as described in [2], where MATLAB was used for implementation. However, we employ R programming for our analysis and generation of fractal patterns, leveraging its robust statistical and graphical capabilities.

The outline of this article is as follows: Section 2 introduces the uniform random iterative algorithm for generating fractal design patterns. This section is divided into two subsections, where section 2.1 explains the uniform random iterative algorithm for generating fractal design patterns for a contractive non-affine IFS. Section 2.2 then explains the uniform random iterative algorithm for generating fractal design patterns for a non-contractive non-affine IFS. Section 3 provides examples using both algorithms from the previous section, with section 3.1 presenting an example of generating a unique and beautiful fractal pattern using the algorithm explained in section 2.1. Section 3.2 presents an example of generating another captivating design using the algorithm explained in section 2.2. Section 4 briefly explains two clustering algorithms used in this study for segmenting the generated fractal garment design: the K-means clustering algorithm and the self-organizing maps (SOM) algorithm. Section 5 presents the fractal dimension of the generated images and explains the box-counting methodology used for determining the fractal dimension. Section 6 discusses the results and includes diagrams from the clustering process, along with summary results in both tabular and diagrammatic forms. Finally, the conclusions are presented in Section 7.

## 2. Uniform Random Iterative Algorithm for Generating the Fractal Design Pattern

Designing patterns from IFS is a novel idea, but comparatively an easy and quick way of designing. Here we make use of the random iterative algorithm other than the deterministic algorithm, wherein we have a set of transformations in hand but at each iteration only a single transformation is applied to the points. Also here we do not associate any probability for the selection and hence we rely on the assumption that each transformation from the set $T_i$ has uniform or equal probability of being selected at each iteration and hence the iterative algorithm is a uniform random iterative algorithm.

### 2.1. A Uniform Random Iterative Algorithm to Compute Attractor of a Contractive IFS

**Input:** Number of iterations $N$ and Transformations $\{T_i\}_{i=1}^{n}$, where $n$ is the number of transformations.
**Output:** Set of points forming the non-affine fractal pattern
Initialize point $(x, y)$ at $(0, 0)$
Choose transformations $T_i$ with parameters $(a_i, b_i, c_i, d_i, e_i, f_i)$ for $i = 1, \ldots, n$
Create an empty set of points: $Points \leftarrow \emptyset$
**for** $i \leftarrow 1$ to $N$ **do**
    Randomly select a transformation $T_j$ from the set $\{T_i\}_{i=1}^{n}$
    Apply the selected transformation $T_j$ using nonlinear equations:

$$(x', y') = \Big(a_j x^2 + b_j y^2 + e_j, \; c_j x^2 + d_j y^2 + f_j\Big) \tag{2.1}$$

    Update the current point: $(x, y) \leftarrow (x', y')$
    Add the current point to the set of points: $Points \leftarrow Points \cup \{(x, y)\}$
**end for**
**return** *Points*

### 2.2. A Uniform Random Iterative Algorithm for Computing the Attractor of a Non-Contractive IFS

**Input:** Number of iterations $N$, number of transformations $n$, and a set of transformations $\{T_i\}_{i=1}^{n}$.
**Output:** Grid of points forming the non-affine fractal attractor from the non contractive IFS
Initialize point $(x, y)$ at $(1, 0)$
Select transformations $T_i$ with parameters $(a_i, b_i, c_i, d_i, e_i, f_i)$ for $i = 1, \ldots, n$
Create an empty set of points: $Points \leftarrow \emptyset$
**for** $i \leftarrow 1$ to $N$ **do**
    Randomly choose a transformation $T_j$ from the set $\{T_i\}_{i=1}^{n}$
    Apply the selected transformation $T_j$ using the following nonlinear equations:

$$(x', y') = \Big(a_j \sin(x) + b_j \cos(y) + c_j, \; d_j \cos(x + y) + e_j \sin(x - y) + f_j\Big) \tag{2.2}$$

    Update the current point: $(x, y) \leftarrow (x', y')$
    Add the updated point to the set of points: $Points \leftarrow Points \cup \{(x, y)\}$
**end for**
**return** *Points*

### 2.3. A uniform random iterative algorithm to compute spiral shaped attractor of a non-contractive IFS

**Input:** Number of iterations $N$ and transformations $\{T_i\}_{i=1}^{n}$, where $n$ is the number of transformations.
**Output:** Point set forming the spiral-shaped fractal pattern
Initialize point $(x, y)$ at the origin $(0, 0)$
Define transformations $T_i$ with parameters $(a_i, b_i, c_i, d_i, e_i, f_i)$ for $i = 1, \ldots, n$
Initialize an empty set for storing points: $Points \leftarrow \emptyset$
**for** $i \leftarrow 1$ to $N$ **do**
    Randomly select a transformation $T_j$ from the set $\{T_i\}_{i=1}^{n}$ or choose a specific one
    Apply the transformation with the following procedure:

$$\mathbf{r} = \sqrt{(a_j x)^2 + (d_j y)^2}$$

$$\theta = \text{atan2}(d_j y,\ a_j x),$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \mathbf{r} + e_j \sin(f_j \theta) \\ \mathbf{r} + e_j \sin(f_j \theta) \end{pmatrix} \odot \begin{pmatrix} \cos(\theta + b_j) \\ \sin(\theta + b_j) \end{pmatrix} + \begin{pmatrix} c_j \\ c_j \end{pmatrix}$$

    The function $\text{atan2}(y, x)$ computes the angle between the positive $x$-axis and the point $(x, y)$, with the result in the range $(-\pi, \pi]$. The operator $\odot$ represents the Hadamard product, which performs element-wise multiplication of two vectors or matrices.
    Update the point: $(x, y) \leftarrow (x', y')$
    Add the updated point to the set: $Points \leftarrow Points \cup \{(x, y)\}$
**end for**
**return** $Points$

## 3. Examples

In this section, we elaborate on two examples of pattern generation, where the first one is through contractive IFS transformation, and the second example is by using non-contractive IFS transformation.

### 3.1. Fractal pattern generation using contractive IFS:

Here we use the Algorithm in Section 2.1 to generate our garment pattern, and it involves a series of non-linear transformations applied iteratively to a starting set of points. Here we start with the origin $(x, y) = (0, 0)$. Each transformation is defined by the following nonlinear equations:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x^2 \\ y^2 \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix},$$

where $i = 1, \ldots, n$, $n$ being the number of transformations. Here, $(x, y)$ are the coordinates of a point in the original grid, and $(x', y')$ are the coordinates after applying the nonlinear transformation. The coefficients $(a_i, b_i, c_i, d_i, e_i, f_i)$ define each specific transformation. Three distinct nonlinear transformations $\{T_i\}_{i=1}^{3}$ are considered and among them one is randomly selected and applied to the initial points, and this process is iteratively applied. The transformations are:

$$T_1 = \begin{cases} x' = 0.5x^2 + 0.2y^2 + 0.1 \\ y' = -0.4x^2 + 0.6y^2 + 0.3, \end{cases}$$

$$T_2 = \begin{cases} x' = -0.6x^2 - 0.3y^2 - 0.2 \\ y' = 0.7x^2 - 0.5y^2 + 0.1, \end{cases}$$

and

$$T_3 = \begin{cases} x' = 0.4x^2 + 0.5y^2 + 0.3 \\ y' = -0.2x^2 + 0.3y^2 - 0.4. \end{cases}$$

The transformations are iteratively applied for a specified number of iterations to increase the complexity and detail of the pattern. The transformations are iteratively applied for a specified number of iterations to increase the complexity and detail of the pattern. The resulting points are scaled down using a scale factor to reduce the overall size of the pattern (here we used scale factor as 0.3). The resulting points are scaled down using a scale factor to reduce the overall size of the pattern (here we used scale factor as 0.3).

The basic pattern as shown in Figure 1(a) is generated by applying the above delineated affine transformations in 1x1 grid with 5000 iterations. The Fabric pattern (see Figure 1(b)) generation process uses a 5x5 grid with 5000 iterations ($N = 5000$) and a scale factor of 0.3 as in Figure 1(b). Similarly Figure 1(c) and Figure 1(d) depicts the printed version of our fabric pattern on material 1 and material 2. The resulting pattern exhibits self-similar characteristics due to the recursive application of affine transformations. This method combines mathematical precision with artistic creativity, providing a robust framework for generating intricate and visually appealing fractal patterns, as illustrated in Figure 1.

### 3.2. Fractal pattern generation using non contractive IFS

Here we use the Algorithm in section 2.2 to generate our fractal garment pattern. As discussed in section 3.2 this implementation also undergoes certain non-linear transformations, but here non-contractive transformations are used, and these transformations have the form as in equation 2.2. These are the transformations used:

$$T_1 = \begin{cases} x' = 1.2\sin(x) + \frac{\pi}{4}\cos(y) - 0.5, \\ y' = \cos(x+y) + 0.3\sin(x-y) + 2. \end{cases}$$

$$T_2 = \begin{cases} x' = 0.8\sin(x) - \frac{\pi}{3}\cos(y) + 0.2, \\ y' = 0.9\cos(x+y) + 0.4\sin(x-y) + 3. \end{cases}$$

$$T_3 = \begin{cases} x' = 0.6\sin(x) + \frac{\pi}{6}\cos(y) - 0.3, \\ y' = 0.7\cos(x+y) + 0.5 \cdot \sin(x-y) + 4. \end{cases}$$

and

$$T_4 = \begin{cases} x' = 1.0\sin(x) - \frac{\pi}{2}\cos(y) + 0.4, \\ y' = 1.1\cos(x+y) + 0.3\sin(x-y) + 5. \end{cases}$$

One transformation is randomly chosen among these and is used to generate the new points. This step is continued for 5000 iterations, where in every iteration a transformation from the above $\{T_i\}_{i=1}^4$ is used. And these are again laid on a $5x5$ grid so that every iteration will produce 25 points instead of a single point. Even though we used non-contractive transformations, it is still presumed that self-similar attractors can be generated, as periodic functions can lead to local contractions [10]. Figure 2(a) shows the basic pattern, and Figure 2(b) outlines the fabric pattern which is obtained by placing the basic pattern over a $5x5$ grid. Also, Figures 2(c) and 2(d) represent the printed design on two different styling templates. Overall, Figure 2 represents the design pattern generated by using the Algorithm in section 2.2 and the non-contractive transformations $\{T_i\}_{i=1}^4$ described above.

### 3.3. Spiral shaped Fractal pattern generation using Non-contractive IFS

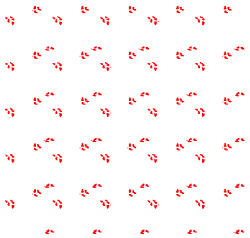
The pattern generation using non-affine IFS involves a series of transformations applied iteratively to a starting set of points. Each transformation is defined by non-affine equations of the form:

$$\mathbf{r} = \sqrt{(a_j x)^2 + (d_j \cdot y)^2} \quad , \ \theta = \operatorname{atan2}(d_j y, a_j x),$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \mathbf{r} + e_j \sin(f_j \theta) \\ \mathbf{r} + e_j \sin(f_j \theta) \end{pmatrix} \odot \begin{pmatrix} \cos(\theta + b_j) \\ \sin(\theta + b_j) \end{pmatrix} + \begin{pmatrix} c_j \\ c_j \end{pmatrix},$$

where $i = 1, \ldots, n$, $n$ is the number of transformations. $(x, y)$ are the coordinates of a point in the original grid, $(x', y')$ are the coordinates after transformation, and $(a_i, b_i, c_i, d_i, e_i, f_i)$ are the constants defining the non-affine transformation. The termination criteria are satisfied after completing $N$ iterations, resulting in the final attractor, which represents our fractal. To generate the pattern:

(a) Basic pattern

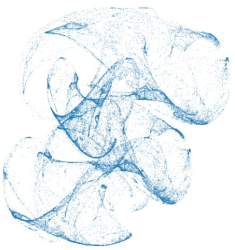


(b) Fabric pattern

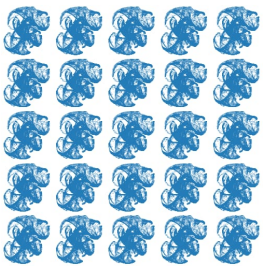
(c) Printed design on material 1

(d) Printed design on material 2

Figure 1: Contractive IFS-generated patterns for fabric design.



(a) Basic pattern

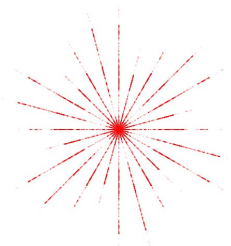


(b) Fabric pattern

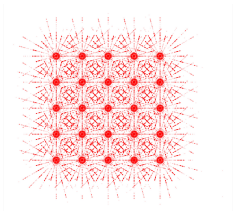
(c) Printed design on material 1

(d) Printed design on material 2

Figure 2: Non-Contractive IFS-generated patterns for fabric design.



(a) Basic pattern



(b) Fabric pattern

(c) Printed design on material 1

(d) Printed design on material 2

Figure 3: Non-Contractive IFS-generated patterns for floor and wall design.

1. **Define Initial Grid**: A grid of initial points $(x, y)$ spanning the range from 0 to 1 in both dimensions is defined.
2. **Apply non-affine Transformations**: A set of predefined non-affine transformations is applied to each point in the grid. Each transformation is randomly selected and alters the position and scale of the points.
3. **Iterative Transformation**: The transformations are iteratively applied for a specified number of iterations to increase the complexity and detail of the pattern.
4. **Scale Down Points**: The resulting points are scaled down using a scale factor to reduce the overall size of the pattern.

Four distinct non-affine transformations are iteratively applied to the initial grid points as follows:

Table 1: Parameters

| **Parameter** | **j** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **a** | 0.85 | 0.15 | 0.15 | 0.85 |
| **b** | $\frac{\pi}{6}$ | $-\frac{\pi}{6}$ | $\frac{\pi}{4}$ | $-\frac{\pi}{4}$ |
| **c** | 0 | 0 | 0 | 0 |
| **d** | 0.85 | 0.15 | 0.15 | 0.85 |
| **e** | 0.2 | 0.2 | 0.2 | 0.2 |
| **f** | 6 | 6 | 6 | 6 |

The basic pattern as shown in 3(a) is generated from the single iteration of the above delineated affine transformations. The Fabric pattern generation process uses a 5x5 grid with 5000 iterations ($N = 5000$) and a scale factor of 0.3 as in Figure 3(b). Similiarly Figure 3(c) and Figure 3(d) depicts the printed version of our fabric pattern on material 1 and material 2. The resulting pattern exhibits self-similar characteristics due to the recursive application of affine transformations. This method combines mathematical precision with artistic creativity, providing a robust framework for generating intricate and visually appealing fractal patterns, as illustrated in Figure 3.

## 4. Segmenting IFS-Generated Patterns Using Clustering Techniques

Clustering algorithms, such as K-means or Self-Organizing Maps (SOM), help in identifying and grouping similar regions within intricate fractal patterns, facilitating different styles of the pattern. The algorithm for clustering of the IFS generated pattern via K-means and SOM is given in Algorithms 4.1 and 4.2 respectively.

### 4.1. Segmentation of IFS-Generated Pattern Using K-Means Clustering

**Input:** Set of generated points, number of clusters $k$
**Output:** Clustered points with K-means labels
**Step 1:** *Initialize Cluster Centers*
Randomly select $k$ initial points from the dataset as initial centroids for the clusters.
**Step 2:** *Assign Points to Clusters*
For each data point, calculate the distance to each cluster center using a distance metric (e.g., Euclidean distance):

$$\text{Distance}_{i,j} = \|\mathbf{x}_i - \mathbf{c}_j\| \quad i \in \{1, 2, \ldots, N\}, j \in \{1, 2, \ldots, k\}$$

where $\mathbf{x}_i$ is the position vector of point $i$ and $\mathbf{c}_j$ is the vector of cluster center $j$.
Assign each point to the nearest cluster center:

$$\text{Cluster}_i = \arg\min_j (\text{Distance}_{i,j}) \quad j \in \{1, 2, \ldots, k\}$$

**Step 3:** *Update Cluster Centers*
Recalculate each cluster center based on the current assignment of points:

$$\mathbf{c}_j = \frac{1}{N_j} \sum_{i \in \text{Cluster}_j} \mathbf{x}_i \quad j \in \{1, 2, \ldots, k\}$$

where $N_j$ is the number of points assigned to cluster $j$, and $\mathbf{x}_i$ are the position vectors of these points.
**Step 4:** *Repeat Assignment and Update*
Repeat Steps 2 and 3 until convergence, i.e., until the cluster centers change by less than a predefined threshold or a set number of iterations is reached.
**Step 5:** *Assign Final Labels and Visualize Results*
Update the dataset with the final cluster labels and create a visualization of clustered points, color-coded by K-means labels.

### 4.2. Segmentation of IFS-Generated Pattern using Self-Organizing Maps (SOM)

**Input:** Set of generated points, number of neurons in the SOM grid, learning rate, and number of iterations
**Output:** Segmented data points with cluster labels from SOM
**Step 1:** *Preprocess the Data*
Convert the IFS-generated data points into a suitable feature representation (e.g., position vectors of the points).
**Step 2:** *Initialize SOM*
Create a grid of neurons with random weight vectors. The grid dimensions are determined by the number of neurons specified.
**Step 3:** *Train SOM*
**for** each iteration $t = 1, 2, \ldots, T$ **do**
**Select a Random Sample:** Choose a random feature vector $\mathbf{x}_i$ from the dataset of generated points.
**Find Best Matching Unit (BMU):** Identify the neuron with weights closest to $\mathbf{x}_i$ using a distance metric (e.g., Euclidean distance):

$$\text{BMU}_j = \arg\min_j \left( \|\mathbf{x}_i - \mathbf{w}_j\| \right)$$

where $\mathbf{w}_j$ is the weight vector of neuron $j$.
**Update Weights:** Update the weights of the BMU and its neighbors using:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \Gamma(t) \cdot \mathrm{h}_j(t) \cdot (\mathbf{x}_i - \mathbf{w}_j(t))$$

where $\Gamma(t)$ is the learning rate at iteration $t$, and $\mathrm{h}_j(t)$ is the neighborhood function that decreases with distance from the BMU.
**end for**
**Step 4:** *Assign Points to Clusters*
For each data point $\mathbf{x}_i$, assign it to the cluster of the neuron with the closest weight vector:

$$\text{Cluster}_i = \arg\min_j \left( \|\mathbf{x}_i - \mathbf{w}_j\| \right),$$

$$i \in \{1, 2, \ldots, N\}, \quad j \in \{1, 2, \ldots, k\}$$

**Step 5:** *Visualize Results*
Create a visualization of the segmented data points, with each point labeled according to the cluster it belongs to, based on the final SOM clustering labels.

## 5. Box Counting Dimension for the Fractal Garment Pattern

In this section, we present the analysis of the box counting dimension, also known as the Minkowski–Bouligand dimension, for the fractal generated using the Iterated Function System (IFS).

The box counting method involves covering the fractal with a grid of boxes of varying sizes and counting the number of boxes $N(r)$ that contain a part of the fractal. The relationship between $N(r)$ and the box size $r$ is given by:

$$N(r) \sim r^{-D}.$$

where $D$ is the box counting dimension. To compute $D$, we take the logarithm of both sides, resulting in:

$$\log(N(r)) = \log(C) - D\log(r).$$

which can be rewritten as:

$$\log(N(r)) = \log(C) + D\log\left(\frac{1}{r}\right).$$

To determine $D$, we performed a linear regression between $\log N(r)$ and $\log \frac{1}{r}$(See Figure 4). The slope gives the fractal dimension. The computed fractal dimension for the contractive IFS-generated fractal is:

$$D \approx 1.0148.$$

The high coefficient of determination ($R^2$) indicates the percentage of the dependent variable's total variation that the linear regression model explains. It is frequently employed to evaluate the model's goodness-of-fit.) value of 0.9799 shows that the linear model fits the data quite well, demonstrating the computed dimension's reliability. This yielded the following regression line equation:

$$\log(N(r)) = 6.5206 - 1.0148\log\left(\frac{1}{r}\right).$$

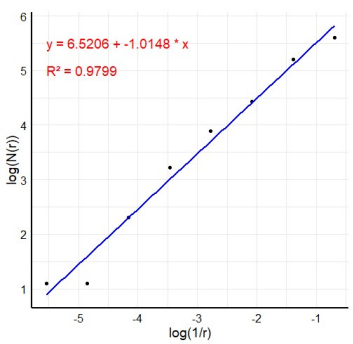



Figure 4: Log-Log Plot of $N(r)$ versus $\frac{1}{r}$ for the contractive IFS-generated fractal Basic pattern in figure 1 (a).

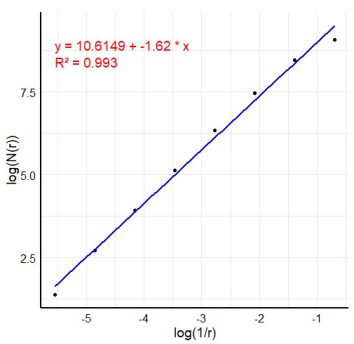



Figure 5: Log-Log Plot of $N(r)$ versus $\frac{1}{r}$ for the non-contractive IFS-generated fractal Basic pattern in figure 2 (a).

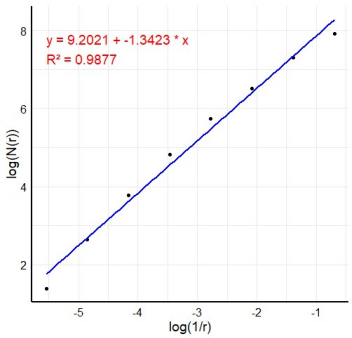



Figure 6: Log-Log Plot of $N(r)$ versus $\frac{1}{r}$ for the non-contractive IFS-generated fractal Basic pattern in 3 (a).

Similarly the computed fractal dimension for the non contractive IFS pattern (see Figure 5) is

$$D \approx 1.62.$$

Also the regression line equation can be written as:

$$\log(N(r)) = 10.6149 - 1.62 \log\left(\frac{1}{r}\right).$$

($R^2$) value is 0.993 which outlines the model's good fit also it may indicate the self similarity of the fractal structure. Non contractive IFS may produce self similar attractors([10]). Similarly the fractal dimension for the pattern generated via non contractive IFS for floor and wall design is illustrated in Figure 6. The estimated fractal dimension is

$$D \approx 1.3423.$$

The corresponding regression equation is given by:

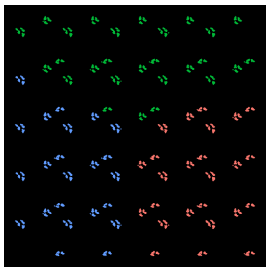
$$\log(N(r)) = 9.2021 - 1.3771 \log\left(\frac{1}{r}\right).$$

The coefficient of determination ($R^2$) is 0.9877, indicating a strong model fit and suggesting the presence of self-similarity in the fractal structure.

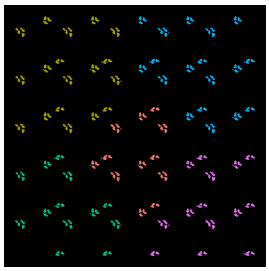
These dimensions suggest that the fractal pattern possesses a level of complexity that is suitable for use in garment design, providing a balance between intricate detail and manufacturability. The self-similar nature of the fractal, characterized by the box counting dimension, can contribute to creating visually appealing and unique textile patterns.

## 6. Results and Discussion

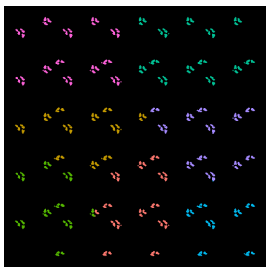
K-means clustering is employed to the IFS fabric pattern in 7(b). These were clustered into $k = 3$ (see Figure 7(a)), $K = 5$ (see Figure 7(b)), $K = 7$ (see Figure 7(c)), $K = 9$ (see Figure 7(d)) clusters using the K-means algorithm in 4.1. The resulting clusters were then visualized, highlighting the segmentation of the fractal pattern into distinct regions.



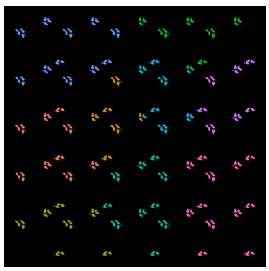
(a) Clustering with center 3



(b) Clustering with center 5

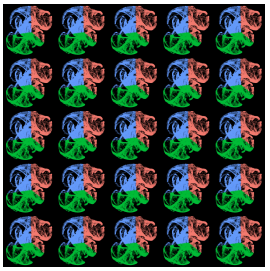


(c) Clustering with center 7



(d) Clustering with center 9

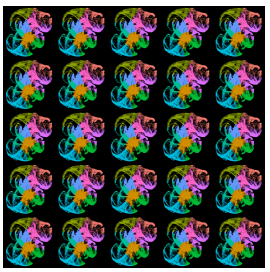
Figure 7: Image segmentation of garment pattern via k-means clustering.



(a) Clustering with center 3

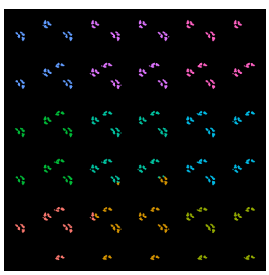
(b) Clustering with center 5

(c) Clustering with center 7

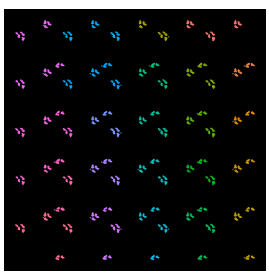


(d) Clustering with center 9

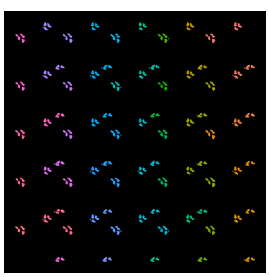
Figure 8: Image segmentation of non contractive IFS garment pattern via K-means clustering.



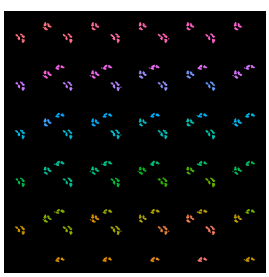
(a) SOM Clustering with x and y dimensions 3



(b) SOM Clustering with x and y dimensions 5

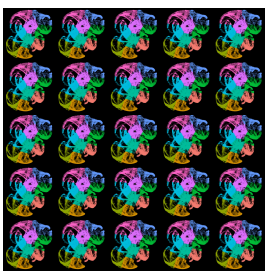


(c) SOM Clustering with x and y dimensions 7

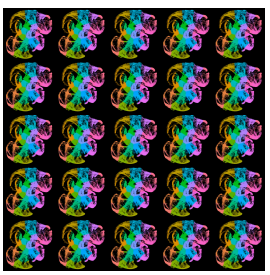


(d) SOM with x and y dimensions 9

Figure 9: Image segmentation of garment pattern via som clustering with hexagonal topology.



(a) SOM Clustering with x and y dimensions 3



(b) SOM Clustering with x and y dimensions 5

(c) SOM Clustering with x and y dimensions 7

(d) SOM with x and y dimensions 9

Figure 10: Image segmentation of non contractive IFS garment pattern via som clustering with hexagonal topology.

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 41624 | 0.845 | 0.267 | 3744 |
| **2** | 45083 | 0.531 | 0.899 | 5905 |
| **3** | 38293 | 0.175 | 0.297 | 3437 |

Table 2: Cluster summary for the pattern generation using K-means clustering(centre 3)

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 19978 | 0.526 | 0.380 | 808 |
| **2** | 29929 | 0.171 | 0.813 | 1877 |
| **3** | 21699 | 0.158 | 0.125 | 1052 |
| **4** | 30043 | 0.848 | 0.840 | 1948 |
| **5** | 23351 | 0.910 | 0.137 | 1323 |

Table 3: Cluster Summary for pattern generation using K-means clustering (centre 5)

| Cluster | Size | Mean(x) | Mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 18903 | 0.523 | 0.136 | 767 |
| **2** | 18348 | 0.242 | 0.524 | 759 |
| **3** | 14420 | 0.0796 | 0.133 | 467 |
| **4** | 21795 | 0.810 | 0.931 | 1048 |
| **5** | 11659 | 0.958 | 0.0363 | 291 |
| **6** | 19916 | 0.898 | 0.491 | 816 |
| **7** | 19959 | 0.181 | 0.925 | 874 |

Table 4: Cluster Summary for the pattern generation using K-means clustering (centre 7)

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 14936 | 0.0881 | 0.438 | 446 |
| **2** | 9530 | 0.392 | 0.588 | 191 |
| **3** | 11676 | 0.146 | 0.0224 | 332 |
| **4** | 11043 | 0.805 | 1.02 | 356 |
| **5** | 16668 | 0.546 | 0.163 | 570 |
| **6** | 11246 | 0.654 | 0.715 | 269 |
| **7** | 18254 | 0.159 | 0.938 | 732 |
| **8** | 11604 | 1.00 | 0.703 | 289 |
| **9** | 20043 | 0.955 | 0.167 | 894 |

Table 5: Cluster Summary for the pattern generation using K-means clustering (centre 9)

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 8047 | -1.1706947 | 1.404072 | 7202.711 |
| **2** | 25565 | -0.1261857 | 3.148536 | 34261.265 |
| **3** | 16388 | 0.9659465 | 4.648318 | 30746.005 |

Table 6: K-means cluster Summary (centre=3) for the pattern generation using non contractive IFS

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 9976 | 1.6577 | 4.2822 | 9225.91 |
| **2** | 7034 | -1.3355 | 1.4030 | 4630.65 |
| **3** | 6733 | -0.2261 | 5.2339 | 5979.79 |
| **4** | 12367 | 0.5970 | 2.3941 | 8005.28 |
| **5** | 13890 | -0.7070 | 3.6380 | 6892.39 |

Table 7: K-means cluster Summary (centre=5) for the pattern generation using non contractive IFS

| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 7222 | 1.9252 | 4.3109 | 6291.77 |
| **2** | 9329 | 0.0902 | 2.7536 | 2723.08 |
| **3** | 3126 | -0.6214 | 5.7508 | 2081.86 |
| **4** | 5455 | 1.1766 | 2.0211 | 2761.09 |
| **5** | 10576 | -0.9145 | 3.7048 | 4211.69 |
| **6** | 6766 | -1.3598 | 1.3766 | 4203.92 |
| **7** | 7526 | 0.3765 | 4.4710 | 2996.65 |

Table 8: K-means cluster Summary (centre=7) for the pattern generation using non contractive IFS

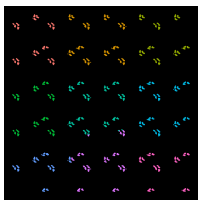
| Cluster | Size | mean(x) | mean(y) | WCSS |
|---|---|---|---|---|
| **1** | 2230 | 2.1575 | 5.5488 | 1076.43 |
| **2** | 8924 | 0.1032 | 2.7339 | 2340.49 |
| **3** | 2817 | -0.7392 | 5.7618 | 1508.90 |
| **4** | 4936 | 1.1008 | 1.9023 | 2130.03 |
| **5** | 7509 | -1.0786 | 3.4721 | 2478.66 |
| **6** | 6566 | -1.3678 | 1.3573 | 3827.69 |
| **7** | 6955 | -0.3114 | 4.1928 | 1577.87 |
| **8** | 5362 | 1.8111 | 3.6868 | 1562.16 |
| **9** | 4701 | 0.7731 | 4.6078 | 1407.11 |

Table 9: K-means cluster Summary (centre=9) for the pattern generation using non contractive IFS

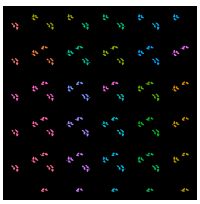
The output of Algorithm 4.2 is a segmented image using SOM clustering with hexagonal topology in which the fractal patterns are divided into distinct regions, as illustrated in Figure 9. Here Figure 9(a) represents the SOM clustering with dimensions $X_{dim} \times Y_{dim}$ as $3 \times 3$, also Figure 9(b) depicts the one with $5 \times 5$ as the $X_{dim} \times Y_{dim}$ dimensions. Similarly Figures 9(c) and 9(d) delineate the SOM clustering with dimensions $X_{dim} \times Y_{dim}$ as $7 \times 7$ and $9 \times 9$ respectively. Likewise Figure 11(a) represents the segmented garment pattern using SOM clustering with rectangular topology and with x and y dimensions as 3. Figure 11(b) denotes the segmented garment pattern using SOM clustering with rectangular topology and with x and y dimensions as 5. Also Figure 11(c) delineates the segmented garment fractal pattern using SOM clustering with rectangular topology and x and y dimensions set to 7. Finally Figure 11(d) shows the segmented garment fractal pattern from SOM clustering with rectangular topology and x and y dimensions 9.

Table 2 displays the summary statistics from k-means clustering for generating patterns using contractive IFS with a center of 3. Similarly, Table 3 illustrates the summary statistics from k-means clustering for generating patterns using contractive IFS with a center of 5. Table 4 details the summary statistics from K-means clustering for generating patterns using contractive IFS with a center of 7, while Table 5 presents the summary statistics from K-means clustering for generating patterns using contractive IFS with a center of 9. Similarly, Table 6 represents the summary statistics obtained from K-means clustering of the non-contractive IFS-generated design pattern with a center of 3. Likewise, Tables 7,

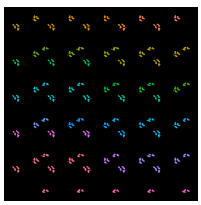
8, and 9 present the summary statistics obtained from k-means clustering of the non-contractive IFS-generated design patterns with centers of 5, 7, and 9, respectively. Additionally, Figure 17 showcases the codebook vectors from SOM clustering for the garment pattern generated using contractive IFS with a hexagonal topology. Codebook vectors (or weight vectors) represent the centroids or prototypes of the input data mapped to each neuron, summarizing the features of the data clustered around it. Similarly, Figure 18 delineates the codebook vectors from SOM clustering for the garment pattern generated using non-contractive IFS with a rectangular topology.



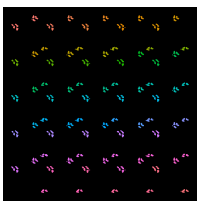
(a) SOM Clustering with x and y dimensions 3



(b) SOM Clustering with x and y dimensions 5

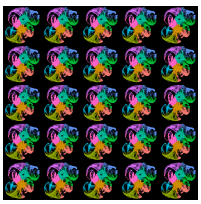


(c) SOM Clustering with x and y dimensions 7

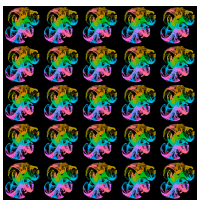


(d) SOM Clustering with x and y dimensions 9

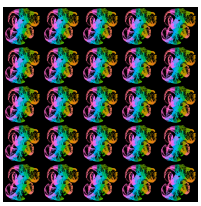
Figure 11: Image segmentation of garment pattern via som clustering with rectangular topology.



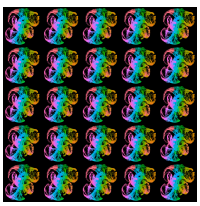
(a) SOM Clustering with x and y dimensions 3



(b) SOM Clustering with x and y dimensions 5

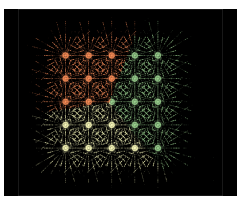


(c) SOM Clustering with x and y dimensions 7

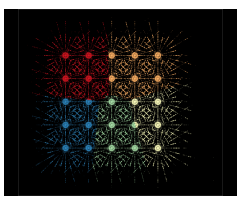


(d) SOM Clustering with x and y dimensions 9

Figure 12: Image segmentation of non contractive IFS garment pattern via som clustering with rectangular topology.



(a) K-Means Clustering with centre=3



(b) K-Means Clustering with centre=5

(c) K-Means clustering with centre=7

(d) K-Means Clustering with centre=9

Figure 13: Image segmentation of Spiral shaped fractal pattern via K-means clustering.

(a) SOM Clustering with x and y dimensions=3

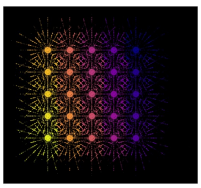
(b) SOM Clustering with x and y dimensions=5

(c) SOM clustering with x and y dimensions= 7

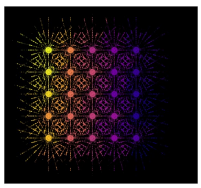
(d) SOM Clustering with x and y dimensions =9

Figure 14: Image segmentation of Spiral shaped fractal pattern via SOM clustering.

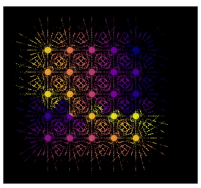
(a) Spiral pattern with rectangular topology and x and y dimensions 3



(b) Spiral pattern with rectangular topology and x and y dimensions 5

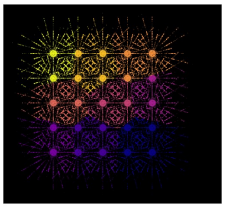


(c) Spiral pattern with rectangular topology and x and y dimensions 7

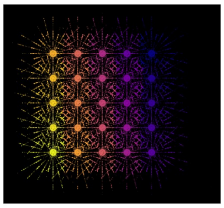


(d) Spiral pattern with rectangular topology and x and y dimensions 9

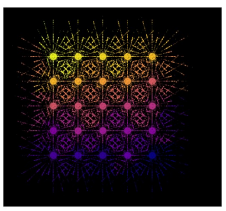
Figure 15: Spiral pattern with rectangular topology



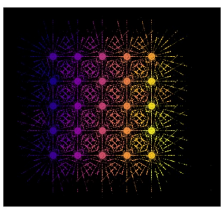
(a) Spiral pattern with hexagonal topology and x and y dimensions 3



(b) Spiral pattern with hexagonal topology and x and y dimensions 5

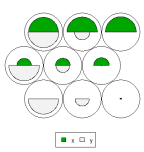


(c) Spiral pattern with hexagonal topology and x and y dimensions 7

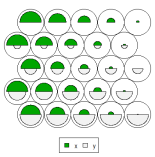


(d) Spiral pattern with hexagonal topology and x and y dimensions 9

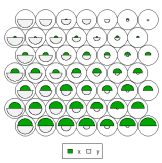
Figure 16: Spiral pattern with hexagonal topology



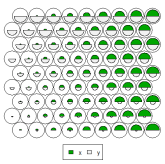
(a) Codebook vectors of the SOM Clustering with x and y dimensions 3



(b) Codebook vectors of the SOM Clustering with x and y dimensions 5

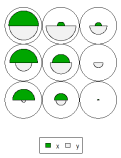


(c) Codebook vectors of the SOM Clustering with x and y dimensions 7

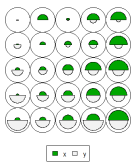


(d) Codebook vectors of the SOM Clustering with x and y dimensions 9

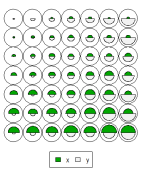
Figure 17: Codebook vectors from the SOM clustering of the contractive IFS pattern with hexagonal topology



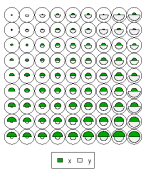
(a) Codebook vectors of the SOM Clustering with x and y dimensions 3



(b) Codebook vectors of the SOM Clustering with x and y dimensions 5

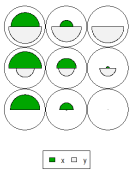


(c) Codebook vectors of the SOM Clustering with x and y dimensions 7

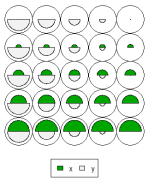


(d) Codebook vectors of the SOM Clustering with x and y dimensions 9

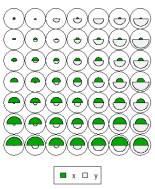
Figure 18: Codebook vectors from the SOM clustering of the non-contractive IFS pattern with rectangular topology



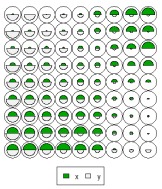
(a) Spiral codebook with x and y dimensions 3



(b) Spiral codebook with x and y dimensions 5

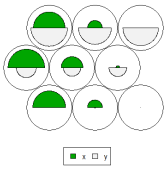


(c) Spiral codebook with x and y dimensions 7

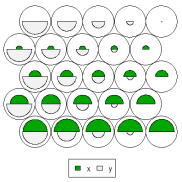


(d) Spiral codebook with x and y dimensions 9

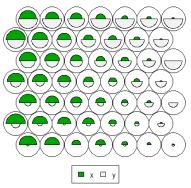
Figure 19: Codebook vectors from the SOM clustering of the spiral pattern with rectangular topology



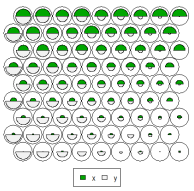
(a) Spiral codebook with hexagonal topology and x and y dimensions 3



(b) Spiral codebook with hexagonal topology and x and y dimensions 5



(c) Spiral codebook with hexagonal topology and x and y dimensions 7



(d) Spiral codebook with hexagonal topology and x and y dimensions 9

Figure 20: Codebook vectors from the SOM clustering of the spiral pattern with hexagonal topology

## 7. Conclusion

This article offers a novel approach to garment pattern design that reduces the need for artistic talent. Rather, it makes use of basic programming and fractal generating expertise. By employing the Iterated Function System (IFS) method, we generated intricate fractal garment patterns using non-affine transformations, which allowed us to create visually engaging and unique designs. Our experimentation with different transformations demonstrated that varied patterns could be produced, enhancing their aesthetic appeal. To further refine these patterns, we applied K-means and Self-Organizing Map (SOM) clustering techniques. These methods effectively segmented the basic patterns, resulting in visually striking designs. Additionally, the computation of the fractal dimension supported the complexity and detail of the generated patterns. This approach not only showcases the potential of fractal-based design in garment creation but also highlights the effectiveness of combining computational techniques with fractal geometry to produce visually appealing and intricate patterns.

## References

[1] Abdulla, S., Reddy, M. K., *Machine Learning Clustering Techniques for Segmenting IFS-Generated Garment Patterns*, IEEE, 1-6, (2024).

[2] Banerjee, S., Gowrisankar, A., Reddy, K. M., *Fractal Patterns with MATLAB*, SpringerBriefs in Complexity, (2023).

[3] Barnsley, M., *Fractals Everywhere*, Academic Press, Boston, (1988).

[4] Barnsley, M. F., Massopust, P. R., *Bilinear fractal interpolation and box dimension*, Journal of Approximation Theory 192, 362–378, (2015).

[5] Barkhordari, M. S., Khoshnazar, S., *Hybrid machine learning algorithms for estimating shear strength of steel-reinforced concrete composite shear walls*, Multiscale and Multidiscip. Model. Exp. and Des. 8, 153, (2025).

[6] Demko, S., Hodges, L., Naylor, B., *Construction of Fractal Objects with Iterated Function Systems*, Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH) 19(3), 271–276, (1985).

[7] He, Y.-X., He, Y., Li, H., *Fast and Accurate Determination of the Spatial Boundary of IFS Attractors*, Computers & Graphics 23(4), 547–553, (1999).

[8] Hutchinson, J. E., *Fractals and Self Similarity*, Indiana University Mathematics Journal 30(5), 713–747, (1981).

[9] Jiang, H. X., Wang, H. F., Liu, J. H., Pan, R. R., *Development of image pattern for textile based on FFT*, International Journal of Clothing Science and Technology 24(5), 295–307, (2012).

[10] Leśniak, K., Snigireva, N., Strobin, F., Vince, A., *Highly non-contractive iterated function systems on Euclidean space can have an attractor*, Journal of Dynamics and Differential Equations, (2024).

[11] Lu, S., Mok, P. Y., Jin, X., *A new design concept: 3D to 2D textile pattern design for garments*, Computer-Aided Design 89, 62–74, (2017).

[12] Lutton, E., *Evolution of fractal shapes for artists and designers*, International Journal of Artificial Intelligence Tools 15(04), 651–672, (2006).

[13] Mandelbrot, B. B., *The Fractal Geometry of Nature*, W. H. Freeman and Company, San Francisco, pp. 460, (1982).

[14] Mandelbrot, B. B., Aizenman, M. J. P. T., *Fractals: form, chance, and dimension*, Physics Today 32(5), 65–66, (1979).

[15] Mandelbrot, B. B., Blumen, A., *Fractal geometry: What is it, and what does it do? [and discussion]*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 423(1864), 3–16, (1989).

[16] Neves, J., Janssens, K., Neves, M., *Fractal geometry – a new tool for textile design development applications in printing*, International Journal of Clothing Science and Technology 6(1), 28–36, (1994).

[17] Nikiel, S. S., *True Color Images and Iterated Function Systems*, Computers & Graphics 22(5), 635–640, (1998).

[18] Nikiel, S. S., *Iterated Function Systems for Real-Time Image Synthesis*, Springer.

[19] Opolon, D., Moutarde, F., *Fast Semi-Automatic Segmentation Algorithm for Self-Organizing Maps*, MINES ParisTech, (2004).

[20] Reddy, K. M., Saravana Kumar, G., Chand, A. K. B., *Family of Shape Preserving Fractal-like Bézier Curves*, Fractals **28**(6), (2020).

[21] Sherman, P., Hart, J. C., *Direct Manipulation of Recurrent Models*, Computers & Graphics 27(2), 143–151, (2003).

[22] Tabianan, K., Velu, S., Ravi, V., *K-Means Clustering Approach for Intelligent Customer Segmentation Using Customer Purchase Behavior Data*, Sustainability (Switzerland) 14(12), (2022).

[23] Wang, S., Yang, X., *Generation of fractal image on complex plane and its application in textiles*, Journal of Silk 54(8), 56–61, (2017).

[24] Wang, W., Zhang, G., Yang, L., Wang, W., *Research on garment pattern design based on fractal graphics*, Eurasip Journal on Image and Video Processing 2019(1), (2019).

*Sana Abdulla,*
*Department of Mathematics,*
*VIT-AP University,*
*India.*
*E-mail address:* sana.22phd7062@vitap.ac.in

*and*

*K. Mahipal Reddy,*
*Department of Mathematics,*
*VIT-AP University,*
*India.*
*E-mail address:* mahipal.reddy@vitap.ac.in