Creating concave hull for IFS fractals using DNA-based computing

AMM Sharif Ullah*
Department of Mechanical Engineering, Kitami Institute of Technology, 165 Koen-cho, Kitami, Hokkaido 090-8507, Japan

Abstract

Fractal geometry can be used to create CAD models of complex shapes observed in the living organisms (cell, tissue, lung, blood vessels, brain structure, and alike) and in the natural world (tree, leaf, flower, landscape, coastline, cloud). If one considers making a physical model of a fractal-geometry-generated CAD model, it is important to perform some topological transformations (e.g., concave/convex hull generation) for making the CAD model meaningful to the manufacturing devices. As a contribution in this area, this study describes a simple but effective procedure that can be used to generate concave hulls for fractal shapes generated by a random walk called Iterated Function System (IFS). One of the constituents of the proposed procedure is an in silico DNA-Based Computing. To demonstrate how the proposed concave hull generating procedure works, a case study has been performed, and using the information of the concave hull generated, a physical model of the fractal has been produced with the aid of additive manufacturing (3D printer).

Introduction

Fractal geometry [1-3] can be used to create CAD models of complex shapes observed in the living organisms (cell, tissue, lung, blood vessels, brain structure, and alike) and in the natural world (tree, leaf, flower, landscape, coastline, cloud). If one considers making a physical model of a fractal-geometry-generated CAD model, it is important to perform some topological transformations (e.g., concave/convex hull generation) for making the CAD model meaningful to the manufacturing devices. As a contribution in this area, this study describes a simple but effective procedure that can be used to generate concave hulls for fractal shapes generated by a random walk called Iterated Function System (IFS). One of the constituents of the proposed procedure is an in silico DNA-Based Computing. To demonstrate how the proposed concave hull generating procedure works, a case study has been performed, and using the information of the concave hull generated, a physical model of the fractal has been produced with the aid of additive manufacturing (3D printer).
hull generation procedure. Section 4 describes a case study showing the effectiveness of the proposed procedure. Section 5 provides the concluding remarks of this study.

**DNA-based computing (DBC)**

A nature inspired computing methodology called DBC has been developed that takes the inspirations from the central dogma [25] of molecular biology. Central dogma of molecular biology simply means that once the sequential information of DNA/RNA has passed into protein (a sequence of amino acids) it cannot get out again [25]. A comprehensive description of the central dogma based DBC can be found in [21,22]. The remainder of this section briefly describes the DBC employed in this article for generating a concave hull.

Figure 2 schematically describes the form of DBC used in this article. In general, the DBC first maps a given binary array into DNA array. Finally, it maps the generated DNA array to a protein array (i.e., to a sequence of amino acids). The binary array must be a piece of information underlying the given problem (the point-cloud of the fractal shape created by a certain IFS). The protein array must help solve the given problem (in this case the concave hull creation problem).

As it is observed in Figure 2, DBC first maps the given binary array \( b_i \in \{0,1\} \) to an DNA array, \( \forall \mathbf{DNA}_i \in \{A, C, G, T\} \). In doing so, two consecutive elements \( b_{i+1}, b_{i+2} \) are mapped into one of the elements taken from \( \{A, C, G, T\} \).

This process undergoes four different types of reading-frame: continuous/discrete raw-/column-wise reading-frames. The case shown in Figure 2 corresponds to continuous column-wise reading-frame where binary array is read in the manner of \( b_i, b_{i+1}, b_{i+2} \), while creating each elements of DNA array, i.e., \( \mathbf{DNA}_i \). Since \( \forall \mathbf{DNA}_i \in \{A, C, G, T\} \), three consecutive elements of DNA array \( \mathbf{DNA}_i \) or \( \mathbf{DNA}_i \) are \( \{A, A, A\}, \{A, A, C\}, \{A, A, G\}, \{A, A, T\}, \{A, C, A\}, \{A, C, C\}, \{A, C, C\}, \{A, C, C\} \).

As a result, a protein array having \( \forall \mathbf{Protein}_i \in \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, X, Y\} \) forms. The case shown in figure 2 corresponds to a continuous column-wise reading-frame, i.e., \( \forall \mathbf{DNA}_i \) or \( \forall \mathbf{DNA}_i \). As understood from the arbitrary case shown in Figure 2, a few-element piece of information (i.e., the binary or DNA array) transforms to a many-element piece of information (i.e., protein array) due to DBC. This characteristic of DBC has been found effective in solving pattern recognition problems of complex shapes [21-23]. In the case of creating an external concave hull, DBC can also be used. In this case, the protein must help distinguish the internal segment of a point-cloud from the external one. To describe the potentiality of DBC being a concave-hull-generator, consider the schematic diagram shown in Figure 3. The protein array shown in Figure 3 (right-hand-side) clearly distinguishes the outer and internal boundary fences. In this particular case, outer and inner boundary fences can be created following the closed loops.

---

doi: 10.15761/FGNAMB.1000122
Volume 2(1): 129-133

Figure 2. The DBC.
Table 1. The genetic code.

<table>
<thead>
<tr>
<th>No</th>
<th>Amino Acid (single-letter symbol)</th>
<th>Codon in term of DNA base-pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Isoleucine (I)</td>
<td>ATT, ATC, ATA</td>
</tr>
<tr>
<td>2</td>
<td>Leucine (L)</td>
<td>CTT, CTC, CTA, CTG, TTA, TTG</td>
</tr>
<tr>
<td>3</td>
<td>Valine (V)</td>
<td>GTT, GTC, GTA, GTG</td>
</tr>
<tr>
<td>4</td>
<td>Phenylalanine (F)</td>
<td>TT, TTC</td>
</tr>
<tr>
<td>5</td>
<td>Methionine (M)</td>
<td>ATG</td>
</tr>
<tr>
<td>6</td>
<td>Cysteine (C)</td>
<td>TGT, TGC</td>
</tr>
<tr>
<td>7</td>
<td>Alanine (A)</td>
<td>GCT, GCC, GCA, GCG</td>
</tr>
<tr>
<td>8</td>
<td>Glycine (G)</td>
<td>GGT, GGC, GGA, GGG</td>
</tr>
<tr>
<td>9</td>
<td>Proline (P)</td>
<td>CCT, CCC, CCA, CCG</td>
</tr>
<tr>
<td>10</td>
<td>Threonine (T)</td>
<td>ACT, ACC, ACA, AC</td>
</tr>
<tr>
<td>11</td>
<td>Serine (S)</td>
<td>TCT, TCC, TCA, TCG, AGT, AGC</td>
</tr>
<tr>
<td>12</td>
<td>Tyrosine (Y)</td>
<td>TAT, TAC</td>
</tr>
<tr>
<td>13</td>
<td>Tryptophan (W)</td>
<td>TGG</td>
</tr>
<tr>
<td>14</td>
<td>Glutamine (Q)</td>
<td>CAA, CAG</td>
</tr>
<tr>
<td>15</td>
<td>Asparagine (N)</td>
<td>AAT, AAC</td>
</tr>
<tr>
<td>16</td>
<td>Histidine (H)</td>
<td>CAT, CAC</td>
</tr>
<tr>
<td>17</td>
<td>Glutamic acid (E)</td>
<td>GAA, GAG</td>
</tr>
<tr>
<td>18</td>
<td>Aspartic acid (D)</td>
<td>GAT, GAC</td>
</tr>
<tr>
<td>19</td>
<td>Lysine (K)</td>
<td>AAA, AAG</td>
</tr>
<tr>
<td>20</td>
<td>Arginine (R)</td>
<td>CGT, CGC, CGA, CGG, AGA, AGG</td>
</tr>
</tbody>
</table>

Step 2−Creating a binary array

An IFS Algorithm has three segments, namely, Input, Calculation, and Iteration. In the Input segment four inputs items are set by the user, namely, $(x_0, y_0)$, Number of Iterations $(N)$, Mapping Parameters $(a, b, c, d, e, f, j)$, $j = 1, \ldots, n$, and Probabilities $(p_1, \ldots, p_n)$. The Calculation segment calculates the relative weights $w_j = \{0, 1, \ldots\}$, $w_j = \{0, 1, \ldots\}$, where $cp_j = p_1 + \ldots + p_n = 1$. The Iteration segment creates the points $(x_i, y_i)$, $i = 1, \ldots, N$, from the inner boundary (given by the digit 0). The outer concave hull is obtained by traversing the outer boundary fence. The procedure of generating the outer concave hull is guided by an element denoted as $K$ of the protein array. It may help simplify the concave hull generation process.

Proposed concave hull generating procedure

This section describes the proposed concave hull generation procedure. The procedure underlies four steps, as follows:

Step 1−creating an IFS fractal

To create an IFS fractal in the form of a point-cloud consisting of $N + 1$ points, $\{(x_i, y_i)\}$, $i = 1, \ldots, N$, an algorithm called IFS Algorithm can be used [12,13,17,18,22], as follows:

IFS Algorithm:

1. **Input**
   - Seed $(x_0, y_0)$
   - Number of Iterations $N$
   - Mapping Parameters $(a, b, c, d, e, f, j)$, $j = 1, \ldots, n$
   - Probabilities $(p_1, \ldots, p_n)$

2. **Calculation**
   - Generate a Random Number: $r \in [0, 1]$.
   - If $r = w_j$, then $x_i = a x_{i-1} + b y_{i-1} + e x_{i-1} + d y_{i-1} + f_j$.
   - For $i = 1, \ldots, N$

3. **Iteration**
   - Stop $(x)$

Protein Array

Outer Concave Hull

Figure 3. The concept of outer concave hull from the view point of protein array

Step 3−Creating a protein array

This step, the binary array $\mathbf{B} = (\ldots b_{ij} \ldots)$, $\forall b_{ij} \in \{0, 1\}$, is created from $(x, y)$, $i = 0, \ldots, N$, $j = 1, \ldots, N$, i.e., from the point-cloud that models a fractal. To do this, a 2-dimensional grid is considered. To define the grid, the $x_i$ and $y_i$ range from $0$ to $N$. The outer boundary fence is described as a binary array clearly distinguishes the outer boundary (given by the digit 1) and the inner boundary (given by the digit 0).

$$\mathbf{b}_{ij} = \begin{cases} 1, & (x_i, y_i) \in \mathbf{P}, \mathbf{P} \ni (x, y); \\ 0, & \text{otherwise} \end{cases}$$

Step 4−Recreating binary array

In this step, the protein array $\mathbf{P}$ is transformed into a protein array using the DBC described in Section 2. The protein array is denoted as $\mathbf{P} = (\ldots p_{ij} \ldots)$.


Step 5−Determination of Boundary Fence

This is the last step where the boundary fence array $\mathbf{BF} = (\ldots b_{ij} \ldots)$ is determined by transforming the $\mathbf{P} = (\ldots PB_{ij} \ldots)$. The goal is to find out the coordinates of the outer boundary fence. The procedure of getting $\mathbf{PB} = (\ldots PB_{ij} \ldots)$ is described as follows.

Let $CM = [-1, 0, 1] \times [-1, 0, 1]$ be a set and $(p, q)$ be a member of it, i.e., $(p, q) \in CM$. Let $BF_{(p, q)}$ be a binary digit, i.e., $BF_{(p, q)} \in \{0, 1\}$, as
defined by the equation (2).

\[
BF_{ij}(p, q) = \begin{cases} 
PB_{ij} \left( PB_{ij} > 0 \right) - \left( PB_{ij} > 0 \right) & \text{if } PB_{ij} > 0 \\
0, & \text{otherwise}
\end{cases}
\]  

(2)

All possible values of \(BF_{ij}(p, q)\) can be added to determine the elements of PB. This yields equation (3), as follows:

The coordinates of the elements corresponding to \(BF_{ij} > 0\) represent the outer boundary fence or the outer concave hull. Figure 5 shows the BF = (...,BF \(_{ij} \),...) that has been determined from the PB = (...,PB \(_{ij} \),...) shown in Figure 4 using the procedure described above. One can observe from figure 5 that \(BF_{ij} > 0\) clearly defines the required concave hull.

Case study

This section describes a case study where an IFS fractal modeling a snow crystal. The mapping parameters and the probabilities are shown in Table 2. As listed in Table 2, seven affine mappings are used to create the model of the snow crystal in terms of a point-cloud. The results obtained applying the Steps 1-5, as described in Section 3, are shown in Figure 6.

In particular, Figure 6a shows the point-cloud (the result of Step 1), Figure 6b shows the binary array (the result of Step 2), Figure 6c shows the protein array (the result of Step 3), Figure 6d shows the binary array called PB (the result of Step 4), Figure 6e shows the boundary fence array (the result of Step 5). As observed in Figures 6a-e, the outer and inner segments become distinguishable due to the application of the Steps 1-5 in a successive manner. A physical model has also been manufactured by an additive manufacturing equipment (a 3D printer), as shown in Figure 6f. In doing so, the outer concave hull shown in

![Figure 6](image)

Figure 6. A case study of concave hull generation and physical model building.

Figure 6e has been used to generate the STL data. The STL data generating process can be found in [16].

This case study clearly demonstrates that the presented concave hull generation procedure is useful means to handle complex shapes for the sake of manufacturing their physical models.

Concluding remarks

Fractal geometry has extensively been used to quantify the complexity and normality/abnormality of the shapes observed in the living organisms (e.g., cell, tissue, lung, blood vasals, brain structures). These shapes are primarily represented by point-clouds having internal and external boundary fences (concave hulls). The extraction of these boundary fences is not an easy task. This study sheds some lights on this issue by proposing a simple but effective concave hull generating procedure where an in silico DNA-Based Computing plays an important role. It is demonstrated that the proposed concave hull

![Table 2](image)

Table 2. Settings of IFS.
generating procedure is able to create the outer boundary fence of a fractal (i.e., point-cloud created by an IFS) in a lucid manner. Further study can be carried out to generate the inner concave hulls of IFS fractals. Nevertheless, to get benefited from the capability of additive manufacturing in producing complex shapes having biomedical significance, the studies similar to this one must be continue in the years to come.

References


Copyright: ©2016 Humeau-Heurtier A, Bianciardi G. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.