Georgia Institute of Technology

CSE 6730: Modeling and Simulation

Project Group 13

Differential Growth: An Exploration

Authors
Sasha M. Bakker
Sebastian Gutierrez H.
Jieun Seong
Nathan B. Williams

Link to GitHub Repo

April 29, 2022

Abstract

Differential growth is a process that generates the unique geometries in nature, such as the interesting patterns round in plant leaves, biofilm formation, brains, and fingerprints. It is also one of the methods used in generative art. We simulate differential growth with different types of forces that reflect attraction and repulsion between particles – spring force, planet force, and Lennard-Jones force, – starting with particles on a simple closed curve. We quantitatively analyze the results for different parameters using Hausdorff measure, so that one who wants to generate a particular shape can use this method of analysis as a catalogue in choosing appropriate parameters for their projects.

1 Project Description

The aim of our project is to simulate *differential growth*. This process produces unique geometries and structures found in natural phenomena such as plants, biofilm formation, and flowers. It is also observable within ourselves, for-instance, in our heart, brain, and fingerprints. In order to simulate differential growth, we start with an initial distribution of nodes and connecting edges. New nodes are added by two mechanisms, add points in regions with high curvature and add points when edges are sufficiently large. Additionally the nodes interact through attraction and repulsion forces. As a result of adding new nodes and the interaction forces, the system is observed to have spatially morph, grow and get particular characteristics based on the parameters into a unique structure. The fundamental idea of this algorithm was inspired by Anders Hoff, who included a description of differential line growth in his generative art project, *Inconvergent*. In addition to programming our version of the differential growth algorithm, we use mathematical machinery to study the dependence of the long term behaviour of the structures to some of its parameters.

2 Literature Review

In 2015, members, Jessica Rosenkrantz and Jesse Louis-Rosenberg, of the design studio *Nervous System*, began to explore differential growth in the context of design. Around this time, Anders Hoff published code for his differential line growth generative art project on GitHub. His work became influential in the topic and a common reference for implementation [1]. Following this, similar work can be seen in the 2016 paper "Organic Labyrinths and Mazes", by Hans Pedersen and Karan Singh, which involves curve generation for computer graphics. The idea of differential line growth has since been extrapolated to 3D, and it is being used to model growth processes over manifolds, such as faces or feet, to design apparel [2]. At its core, differential growth has a strong connection with natural phenomena, as previously mentioned. The essence of this relationship is described in the following quote.

> "Some areas grow more than others, and this leads to the formation of macroscopic shape. ... Carefully coordinated subdivision and differentiation [allow] biological systems [to] produce structures with specific, reproducible forms and functions" [3].

This describes that differential growth is the opposite of uniform growth, and thus allows for unique shapes and structures to come to fruition in natural phenomena. Studying differential growth is especially interesting due its intersection of art, nature, and computation. Thus, our literature review will focus on generative art, differential growth in plants and the human body, and algorithms related to differential growth.

2.1 Generative Art

Differential growth can be exemplified in the context of "generative art". The development of the generative art movement started in the late 1950s due to popularization of computers. In 1965, the first generative art exhibition,"Generative Computergraphik", was displayed in Stuttgart. The focus of this exhibition were works by Georg Nees, who is considered a pioneer of the discipline. He majored in Mathematics and Physics at the University of Erlangen, and earned a doctorate of philoshopy in 1965 [4]. The second exhibition of generative art included work of other artists such as Frieder Nake, who

received his Ph.D. in Mathematics from the University of Stuttgart. Since the early stages of generative art, mathematics, and computing have been fundamental tools used in the discipline. Outside the realm of visual art, a pioneer of generative music is Iannis Xenakis, who developed "Stochastic Music", which used stochastic processes to produce scores [5]. An important part of this art form is that art must be generated by a predefined set of rules or constraints. The rules guide the actions, but do not give specific directions. This has created a debate on whether generative art can actually be considered art. The trends in modern generative art involve taking inspiration from biological processes and creation of simulation methods that autonomously generate art which mimics organic growth. Innovators in the study of the relationship between growth and natural forms were Ernest Haeckel, D'Arcy Thompson, Alan Turing, Greg Turk, and Aristid Lindenmayer [6].

2.2 Plants

Morphogenesis, the process of organ growth in biological systems, is driven by differential tissue growth. In plants, this growth occurs as cell expansion in layers of mesophyll cells "in response to environmental stimuli" such as temperature, light, or humidity [3]. Not only the movements, but also the formation of complex 3D shapes in plant leaves and flower petals is dictated by differential growth. The paper by Huang et al. establishes a morphological phase diagram capable of rationalizing four geometric configurations commonly found in plant organs: (1) twisting, (2) helical twisting, (3) saddle bending, and (4) edge waving [7]. They show that the morphology of a growing leaf is determined by both the maximum value and the spatial distribution of growth strain $\epsilon_g = (l - l_0)/l_0$, where l_0 and l are the lengths of the center stem and the strip at a distance y from the stem, respectively. This, along the direction of the width, can be approximated by the power-law,

$$\epsilon_g(y) = \beta \left(\frac{y}{W}\right)^n,$$

where y and W are the distances from the strip to the center and half-width of the leaf, respectively. The growth strain monotonically increases from zero at the center to a maximum value, B, at the edge, when $y = W$. The power-law exponent, n, characterizes the steepness of the differential growth strain profile. Leaves with a twisting configuration feature a parabolic growth strain profile. Leaves with edge-waving configuration feature a steeper increase in growth strain near the marginal region, leading to a higher power. On the other hand, Leaves with saddle-bending configuration feature much smaller maximum growth strain level ($B < 0.1$) [7].

2.3 The Body

There are several body parts that exhibit differential growth patterns, such as the brain, tubes, and fingerprints. The brain has convoluted patterns and folding, especially in the convex gyri and concave gyri, the outer cortex and inner core, respectively. The mechanism of brain patterns may provide useful clues into normal and pathological brain function. Differential growth provides preliminary insight into critical growth ratios for instability and crease formation of the developing brain. Critical growth ratio of the cortex to core for crease formation is approximately 1.68 [8]. Along with this, differential growth impacts the formation of the tubing in both the heart and the foregut [9]. The journal by Hosseini et al. displays its impacts using animations and in-depth analysis. It hypothesizes that differential anisotropic growth between mesoderm and endoderm drives diagonal folding [9]. Another body part that exhibits differential growth is fingerprints. Differential growth leads to "the different ridge flows, patterns and sequences of characteristic. ... Even in identical twins the friction ridge development for each twin is different" [10]. During fetal development in the womb, "the stresses and strains affect the development of the friction ridges ... and individual friction ridge units join together to form ridges" [10]. This is an interesting phenomenon because differential growth causes individual ridges to join rather than diverge.

2.4 Algorithms

2.4.1 Differential Line Growth

For a 2D differential growth, the simulation only needs a few factors: nodes and edges, attraction, repulsion, and subdivision and growth. Nodes (a) maintain a mutual attraction force with "physically

connected" neighbor nodes, (b) maintain a minimum distance and are repulsed from all nearby nodes, and (c) experience an alignment force toward the midpoint [11]. The algorithm can be modified to interact with obstructions or the geometry can begin as multiple systems. A. Hoff's approach is as follows: a set of connected nodes begin in some geometry, and new nodes are randomly introduced between existing pairs. At each iteration the nodes aim to optimize their positions with respect to neighbors and other nodes within a distance [1]. The updated path between points should be smoothed after each sampling iteration. The position of points in space can be used to determine what distance the points should be relaxed to in order to produce asymmetric growth. This can be done by assigning points an individual scaling value for how far they should be pushed apart, or by using noise such as Perlin or Worley noise to drive the scaling attribute [12].

2.4.2 Circle Packing and Lloyd's Algorithm

For Lloyd's algorithm, k points on a surface are initially chosen and the relaxation step is iterated: (a) compute the Voronoi diagram of the k point sites, (b) integrate each cell of the diagram and compute the centroid, and (c) move each point to the centroid of its cell. This method moves points toward the center of gravity of the cells, creating a more even point distribution [13]. Lloyd's method can be modified to move points away from the center of gravity in order to form small sets of clusters [14]. Circle packing is a generalization of Lloyd's algorithm. The points can be considered as spheres with specified radii. If the spheres overlap, they repeatedly are moved to nearby locations that reduce overlap, until there are no overlapping points [15]. 2D splitting and differential curve growth algorithms divide polylines into smaller segments. They use circle packing to fill up a constraining surface while preventing self-intersection [16].

2.4.3 Related Algorithms

There are other algorithms, such as Marchal's Random Growth Algorithm, L-Systems, and binary Tree Derivatives, that can also simulate growth. In Marchal's Random Growth Algorithm, we (a) initialize a tree with a root and a leaf and one edge connecting them, (b) assign a weight to each edge and branch point, (c) choose and edge or branch point using probability proportional to the weights, (d) if an edge was selected, split it at its midpoint and add a vertex and new edge carrying a leaf, (e) if a branch was selected, attach a new edge carrying a leaf, and (f) repeat steps (b-e) [17]. L-systems can be used to generate self-similar fractals. They are recursive and can model natural-looking organic forms such as plants. Many rules are applied simultaneously per iteration, unlike a formal language which applies only one rule per iteration [18]. In Binary Tree Derivatives, we (a) set an area for division, (b) compute the ratio of division, (c) divide the area along an axis, (d) restore the divided results in two nodes of structure, and (e) repeat [19].

3 Conceptual Model

Before discussing the conceptual model, it is helpful to define the parameters that will be used.

3.1 Parameters

- ν = proportion of the maximum length
  - When an edge between nodes becomes longer than ν, we add *one* node to the edge.
- δ_1 = attraction distance
  - If the distances between two nodes are less than δ_2, they attract each other as long as the distances are greater than δ_1.
- δ_2 = repulsion distance
  - If the distances between two nodes are less than δ_1, the nodes repel each other.
- Low Curvature Threshold

  - If the curvature of the edge between the nodes becomes greater than the low curvature threshold, we add *two* nodes to the edge.

- k = initial number of nodes
- r = radius of the circle
- Perturb Percent
  - Each node is perturbed by this percentage from its neighbors.
- Force
  - planet
  - Spring
  - Lennard-Jones
- N_t = number of iterations used in the simulation
- Upper Bound Particles
  - This is the maximum number of particles the system will accumulate.
- Δt = time step

3.2 Toy Problem

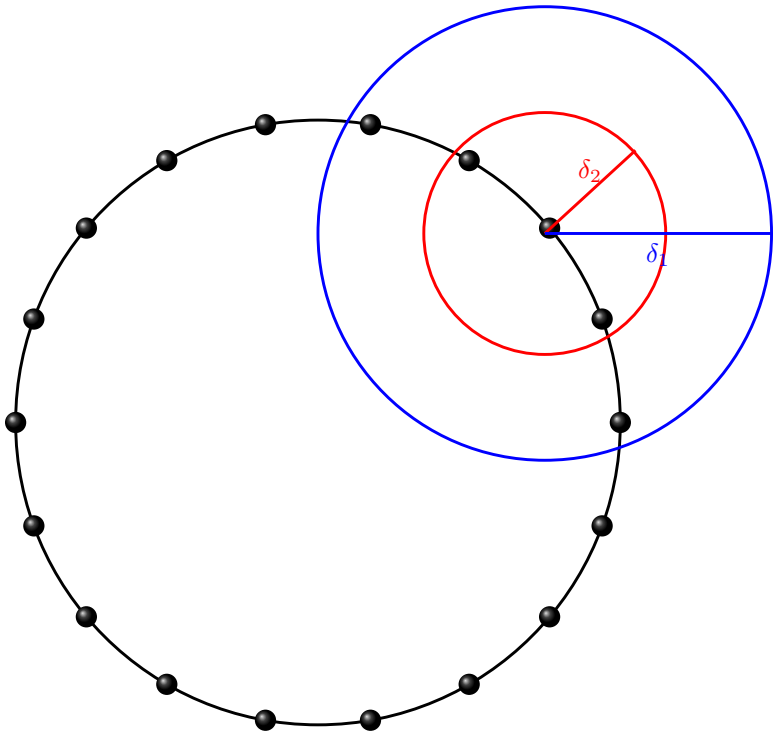


Figure 1: Initial distribution of nodes in a toy problem

In the case of the toy problem represented in Figure 1, there are 18 initial nodes, uniformly distributed, with δ_2, the repulsion distance, being equal to 1.6 and with δ_1, the attraction distance, being equal to 3.

In Figure 1 we display the regions of attraction and repulsion for one of the nodes, one can see that there are 2 other nodes within the repulsion distance, represented by the red inner circle, and a further 2 nodes within the attraction distance, represented by the blue outer circle. The "level" of repulsion and attraction are determined by the force used in the simulation, leading to the positions of the nodes to change. Once the nodes positions have been updated, depending on the distance between 2 nodes, the edge is split and a new node will be added. This will continue until a stopping criteria is met.

3.3 Definitions, Notations, and Assumptions

- G denotes the graph and E_G denotes the set of edges of the graph.
- T denotes the k-d tree. E_T denotes the set of edges of the tree.
- All measured distances, ν, δ_2, and δ_1, are Euclidean distances.
- ν, δ_2 and δ_1 are constants.
- Two nodes, x_1 and x_2, are *edge-connected* or *G-connected*, if there is a graph edge connecting them.
- Two nodes, x_1 and x_2, are *tree-edge-connected* or *T-connected*, if there is a tree edge connecting them.
- We define that two nodes, x_1 and x_2, are *force-connected* if and only if $\delta_2 \leq d(x_1, x_2) \leq \delta_1$ or $d(x_1, x_2) \leq \delta_2$.
- x_1 and x_2 exert *repulsive* force to each other if $d(x_1, x_2) \leq \delta_2$.
- x_1 and x_2 exert *attractive* force to each other if $\delta_2 \leq d(x_1, x_2) \leq \delta_1$.
- The attraction/repulsion process is done before the splitting and growing processes. This is because the point generated by the splitting process will, generally, not be in the curve resulting from the deformation of the attraction/repulsion. The attraction/repulsion process only occurs in force-connected nodes.
- Splitting and growth processes are *neighborhood processes*, meaning that they occur only between the edge-connected neighbors of x_i.
- Attraction and repulsion, in contrast, are *local processes*, meaning that, within δ_1, we consider the curve as a subset of $\mathbb{R}^2$ using the topology induced by the Euclidean distance.
- Constraint topology for curve is in $\mathbb{R}^2$.

3.4 Forces

We decided upon using 3 distinct forces in our model: (1) planet, (2) spring, and (3) Lennard-Jones. These forces will be covered more in depth under the Simulation Model section. However, it is important to state our reasoning behind these 3 forces. First, we wanted to model interesting things found in nature, however, we wanted to look at various time-space scales in nature. Specifically, from the atomic scale upto the planet scale. Hence, we have chosen to focus on the planet force (large scale), the spring force (medium scale), and the Lennard-Jones force (atomic scale). The time scale plays a critical role in the simulation. The size of the time step for the forward algorithm is tightly related to the physical magnitude. In essence, in our conceptual model, the nodes we have created will act as if they are planets, parts of a spring, and even molecules - depending on what force is used in the simulation. We now formally define the forces we are using. Starting with the planet force, its equation is

$$F_{planet} = -\frac{\delta_2 - r_s^2}{r_s^2} \tag{1}$$

where δ_2 is the neighborhood of attraction and r_s is the norm of the differences. In essence, this is a modification of the gravitational force, to obtain an attraction repulsion dynamics. The spring force is given by

$$F_{spring} = -\delta_2 - r_s. \tag{2}$$

Its function also returns force components in both the x and y coordinates. Using this, the nodes act as a spring coil. Lastly, the Lennard-Jones force uses the equation

$$F_{Lennard\text{-}Jones} = -24\sigma^6 r_s^{-7} \left(\frac{2\sigma}{r_s}\right)^5 \tag{3}$$

where $\sigma = 2^{\frac{1}{6}}\delta_2$. Here, the nodes are acting as molecules. All of the mentioned forces can be derived from a potential function V,

$$F = -\nabla V.$$

These forces will be modeled on a problem very similar to the toy problem, however, a reasonable level of randomness will be added to ensure differential growth. This will be added through two main methods. The first, being an initial perturbation of the nodes, and the second being a pseudo-random splitting and subsequent addition of nodes. These two methods will be covered in the Simulation Model section. If the nodes are uniformly distributed along a circle with zero initial velocities, then the net force on every node will also be uniform, leading to a uniform contraction or a uniform expansion. Thus, in order to see more interesting behaviors, we perturbed the shape of the circle and also randomly assigned the initial velocities.

4 Simulation Model

The simulation model is best understood through flowcharts that describe important aspects of the code. These flow charts are the high-level flowchart, the coordinate update flowchart, the split edges flowchart, and the split nodes flowchart. The simulation has the following constant parameters:

- $\nu = 0.95$
- Low Curvature Threshold = 7.5
- $k = 15$
- $r = 2$
- Perturb Percent = 0.45
- $N_t = 610$
- $\Delta t = 0.001$ (Spring Force), 0.0008 (Planet Force), 0.0000001 (Lennard-Jones Force)
- Upper bound on the number of particles = 800 (Spring Force and Planet Force), 80 (Lennard-Jones Force)
- Lower bound on the distance between particles = 0.01

If these parameters are changed, the changes are explicitly mentioned.

5 High-Level Flowchart

The high-level flowchart (Figure 2) summarizes the overall architecture of the code. It begins with the initialization of the system. This consists of the initializing the number of nodes, the radius of the circle, the force used for coordinate updates, and its respective Δt value, the perturbation percent, the number of iterations used in the simulation, the upper bound for particles, a curvature threshold, and the attraction and repulsion distances, δ_1 and δ_2, respectively. It is important to note here that in the simulation of the model, the only parameters varied were the force and its Δt value, and δ_1 and δ_2.

The simulation begins with plotting the initial circle on a graph, taking into consideration the perturbation percent. The nodes are first initialized to be equal distance from one another. The perturbation percent determines the extent of a range of allowed perturbation. This maximum perturbation distance is the length of the arc between two nodes on the circle multiplied by the perturbation percent. For each node, the perturbation distance is randomly picked from a uniform distribution that spans from zero to the maximum allowed perturbation. The perturbation direction is along the arc between the nodes, which is randomly selected from a binary number generator. The coordinates for each node are then updated due to the force. Immediately thereafter, a check occurs. This check occurs at each step to determine whether a specific number of nodes is reached. In our simulation, those numbers were 50, 100, 150, and 250. If any k is any of these numbers, the value of Δt is decreased. For example, $k = 100$, the Δt value is decreased. This is so that the forces can be properly calculated due to the large amount of nodes in the simulation. Then, according to our splitting algorithm, edges are split and new nodes are added every 20 iterations. This continues until the stopping criteria, 610 iterations, is met. After the stopping criteria is met, the code will output an animation of the simulation from iteration 1 to iteration 610.

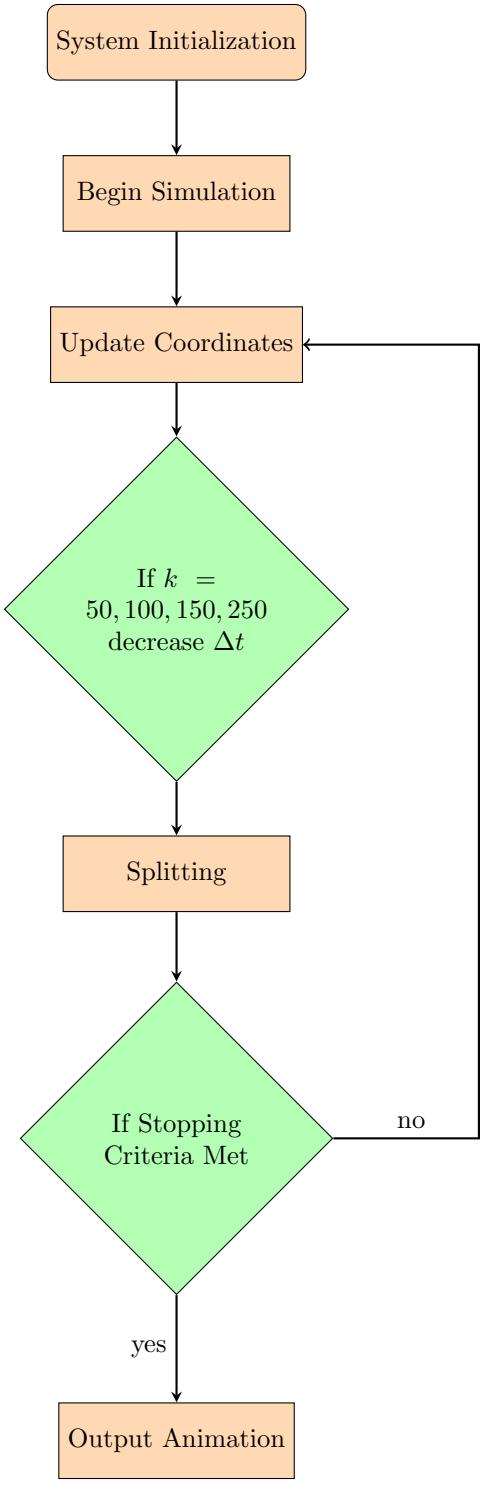


Figure 2: High-Level Flowchart

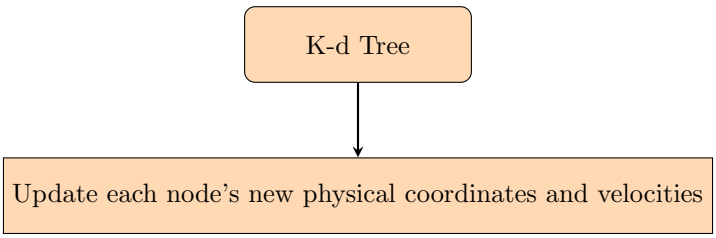


Figure 3: K-d tree flowchart.

5.1 Forces Algorithm

In order to implement attraction and repulsion between nodes, we implemented a force-based algorithm. Three common forces in physics are used as the catalyst for attractive and repulsive movement within the system. These forces are the planet force, spring force, and Lennard-Jones Force. Each of the functions for these forces returns the forces components in both the x and y coordinates. Using Newton's second law, we have that

$$\vec{F} = m\vec{a},$$

assuming that nodes have a unit mass, this becomes

$$\vec{x}'' = \vec{F}.$$

Then, using the Taylor expansion,

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{x}_i'(t-1)\Delta t - \frac{\Delta t^2}{2} \sum_{(i,j)\in E} \nabla V(r_{ij}) \tag{4}$$

$$\vec{x}_i'(t) = \vec{x}_i'(t-1) + \Delta t \sum_{(i,j)\in E} \nabla V(r_{ij}) \tag{5}$$

with r_{ij} being the distance of node i to its physical neighbors j, the coordinates and velocities of each node are updated. Since we are using a second order Taylor expansion, the algorithm is second order for space and first order for the velocities.

Note: The forces were multiplied each by a different constant to modify their intensity, this only with the finality of having an accelerated or decelerated phenomena.

5.2 Coordinate Update Flowchart

The coordinate update flowchart (Figure 3) summarizes the function "coordinate_update()" used in the simulation. In this function, a dictionary of nodes, with respective x and y coorindates is taken as one of inputs. The k-d tree algorithm is then used to organize the nodes into a tree structure that determines, for each node, the nodes that are within the δ_1 neighborhood.

5.3 K-d Tree Algorithm

A k-d tree is generated using the function 'KDTree()' from 'sklearn.neighbors'. The purpose of its implementation is for quick organization of (x,y) points into a tree structure that determines branches based on the proximity of the points to one another. The basic steps of constructing a k-d tree are:

1. Compute the median in the x-coordinate, $\hat{x}$. Its corresponding node becomes the root of the tree.
2. Assign points to branches based on where the x-coordinate is located with respect to the computed median. Points satisfying $x < \hat{x}$ join the left branch of the root and points satisfying $x > \hat{x}$ join the right branch of the root.
3. Compute the median in the y-coordinate for each branch. Then, split the branches into left and right branches using the criteria in (2).

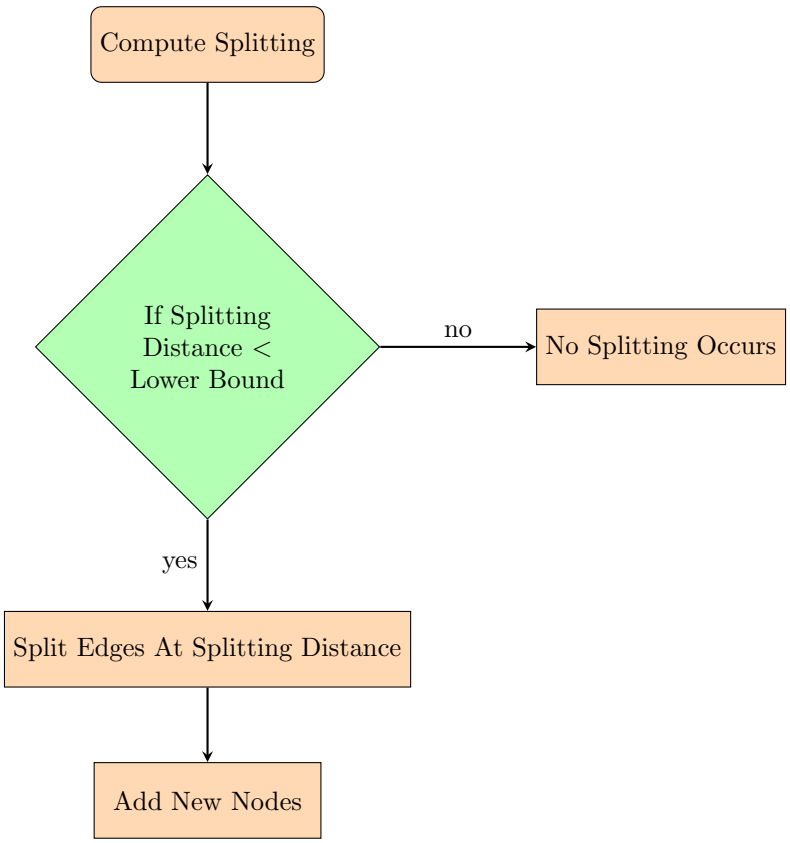


Figure 4: Splitting Method 1 flowchart.

4. Iterate the above steps until all points are used, alternating which coordinate, x or y, is used for the ordering.

Once the k-d tree is generated, the function 'query_radius' is used to find neighbor nodes that are within a radius δ_1 of each node. We tried to compute these neighbors directly by computing and evaluating the matrix of squared distances between each possible pairs of points. This implementation, however, was much more slow and expensive than the k-d algorithm, while producing the same results. Therefore, our choice of determining neighborhood nodes is by the k-d tree algorithm.

5.4 Splitting Algorithm

The splitting algorithm must be designed with randomness in mind, because it is not ideal to have the system grow symmetrically and too quickly. This is because (1) the nodes are initialized to be equal distance from one another and (2) all edges satisfying a length metric are split, meaning, all edges are split. Two methods are defined for the splitting.

Method 1 - Split Edges: The split edges flowchart will explore the first method, a function used in the simulation promptly named "split_edges()". It first computes the splitting distance, which is equal to the distance between two nodes. If the splitting distance is less than the lower bound, the edges are split at the splitting distance and new nodes are added. If the splitting distance is greater than the lower bound, no splitting occurs.
Method 2 - Split Nodes: The split nodes flowchart (Figure 5) also takes a look at the "split_nodes()" function in the simulation. This is the second splitting method implemented. The first part of this function is computing the curvature at each node. If the absolute value of the curvature is less than the threshold, the function moves onto the next node. If the absolute value of the curvature is greater than the threshold, and, if the distance between two neighboring nodes is greater than the lower bound length, a new node will be added to both the "left" and the "right" of the node. If the distance between

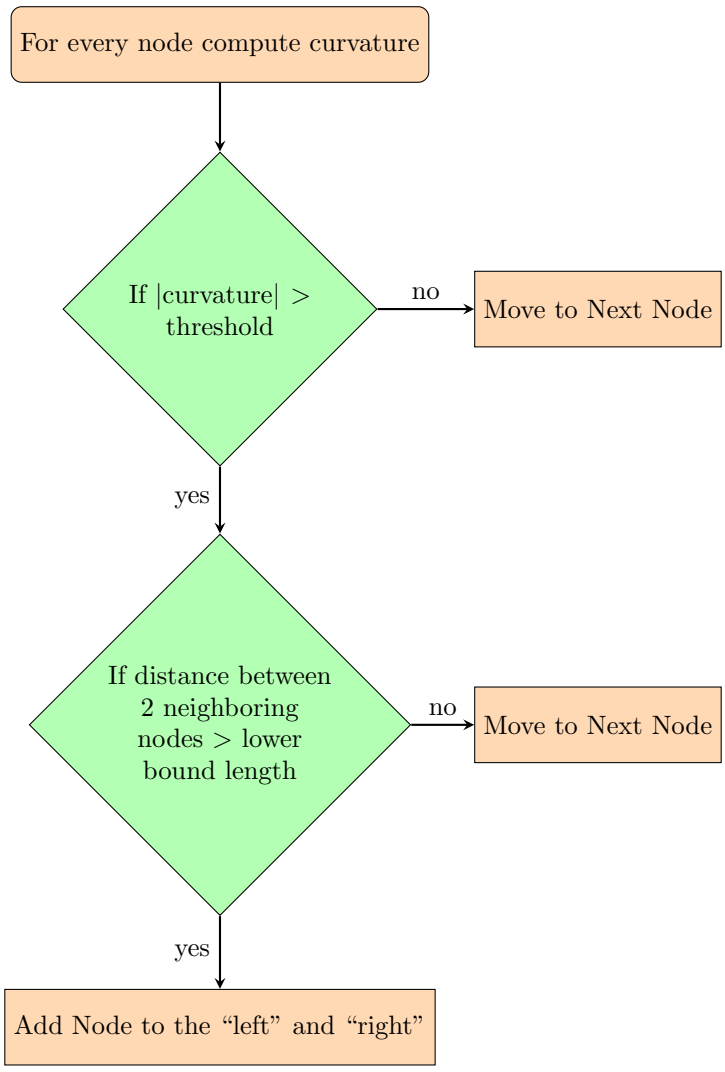


Figure 5: Splitting Method 2 flowchart.

the two neighboring nodes is less than the lower bound length, the function also moves onto the next node.

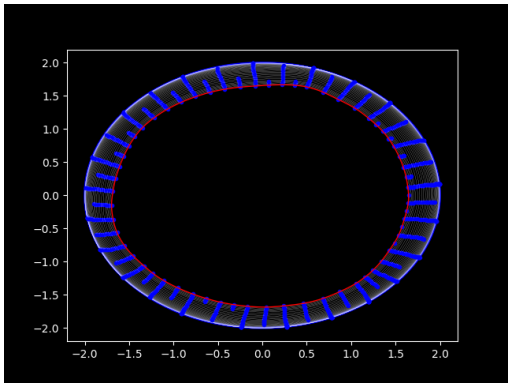
For purposes in our study, both M 1 and Method 2 were implemented. Furthermore, the lower bound for Method 1 is equal to 10^{-2} and the curvature threshold for Method 2 is equal to 7.5.

5.5 Visualization

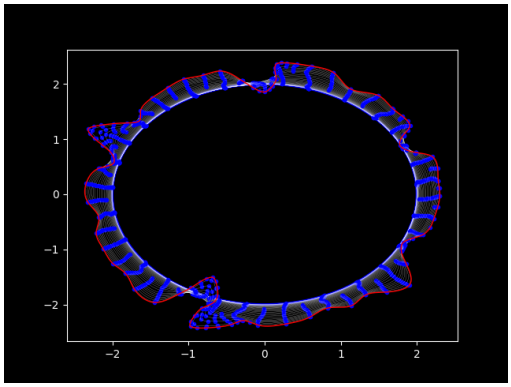
The interpolated curves are added to the plot at every ten iterations of the simulation. The result is a plot of the superimposed curves, which is useful for visualizing the evolution of system's geometry. The starting curve is plotted in white, the evolution in blue, and the end curve in red such as in Figure 6b. There are also figures of the simulation at each time step that are then stacked to generate an animation.

5.6 Verification of forces and splitting schemes

To validate the concept behind our simulation, we consider two experiments: (1) 6a and (2) 6b. Observe that the figures display the evaluation in time from the thick white curve (initialization) to the red curve (position at stopping criteria). Additionally, it is important to note that the force used in the experiments the planet force and that every parameter other than δ_1 and δ_2 were fixed.



(a) Test for attraction and splitting by distance



(b) Test for repulsion and splitting by curvature

Figure 6: Test for forces and splitting schemes.

Furthermore, these results are independent of the results of the main simulation. In Figure 6a, an experiment was run with a large attraction region and negligible repulsion region. Thus, according to our conceptual model, the overall behavior of the points should be to contract. This is evident in Figure 6a. Additionally, as time progresses, the additions of nodes at the midpoint location between existing adjacent nodes is clearly seen. In Figure 6b, an experiment was run with a repulsion radius large enough so that nodes instantaneously start to repel one another. We expect to see a growth of the radius of the circle, an expansion. It is evident in Figure 6b that as time increases, the structure of the nodes expands. In addition, we can see that due to the action of attraction, the line starts to curve. We see that in the regions of high curvature, nodes are added, thus, keeping the line smooth.

6 Validation and Experimental Results

6.1 Input Justification

The perturbed initial configuration we introduced in our simulation in order to generate more interesting output aligns more closely to what would happen in the nature than a perfect circle or a perfectly symmetric geometry. Also, at the development of our model, we introduced new nodes by taking simply taking the average of their coordinates when edges between them become too long, which led to the output shown in Figure 7. This was not consistent with the system we tried to simulate. Instead, for each edge to split, we found the corresponding arc on the interpolated curve between the nodes and added a node at the mid-index of the arc. Also, instead of setting the initial velocities to zero, we took the average velocities between the two nodes to be the velocity of the new node between them. This is more consistent with the behavior seen in the nature, because we would expect nearby points to move at similar speeds.

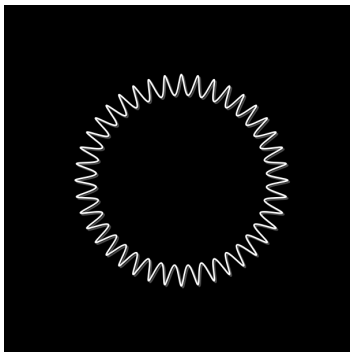
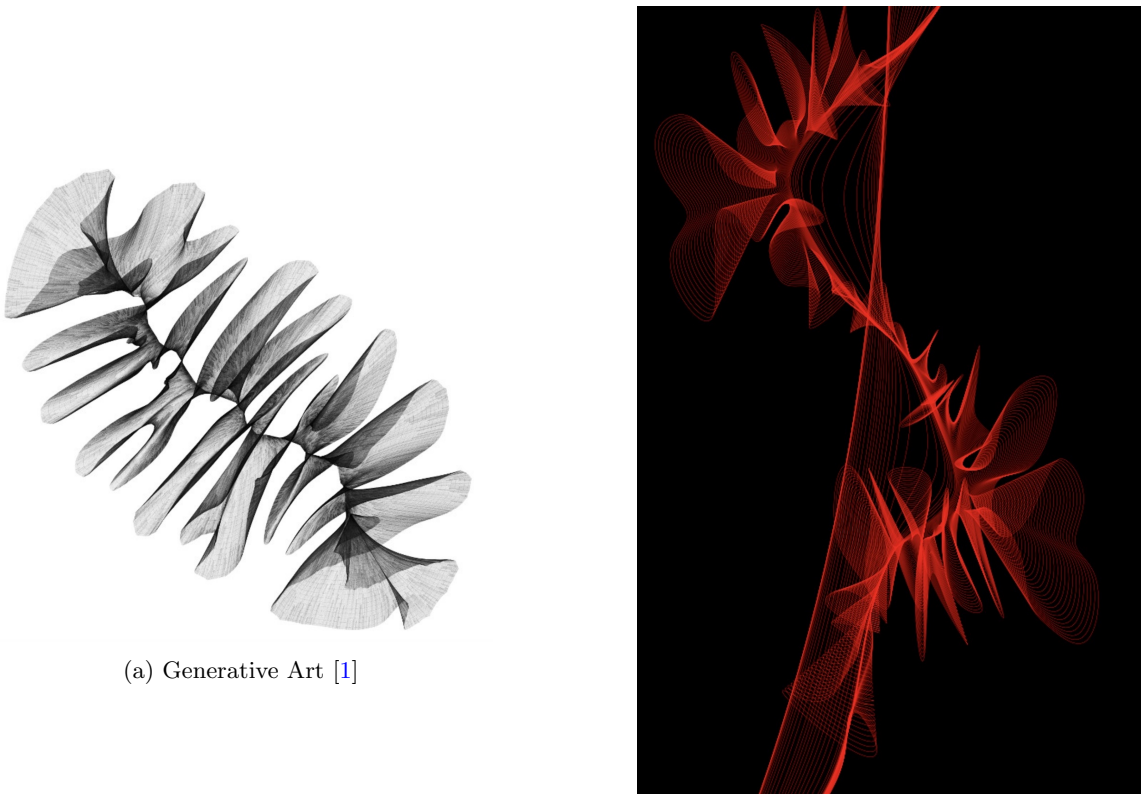


Figure 7: In our initial modeling, we chose to add nodes at the midpoint of a straight line between the nodes, instead of taking an arc from the interpolated curve. This choice led to this uniform, dense wrinkle, which was inconsistent with the system we tried to simulate.



(a) Generative Art [1]

(b) Art Created in Project

Figure 8: Comparison Seen in Art.

6.2 Comparison of the Simulation to Art, and Nature

Throughout our project, differential growth in art, nature, and the body have consistently been taken into consideration. For example, Anders Hoff created a wonderful differential line growth generative art project. With his art project as inspiration, we can compare pieces of our simulation with the art he has curated. See in Figure 8 the comparison of his results to our results. Furthermore, nature can be seen in our simulation as well, see Figure 10. It is important to note that differential growth found in the body, such as in a finger print or the tubing surrounding the heart, was hard to simulation as the required initialization is different than the ones used in the simulation.

6.3 Clustering Analysis

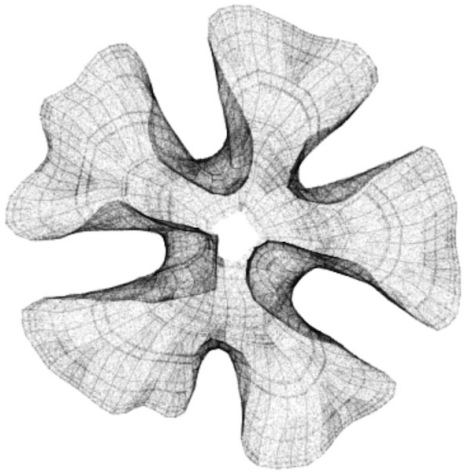
An objective of the project is to classify the long-term behaviour of differential growth with respect to its parameters. To achieve this, we propose using the Hausdorff metric, which was developed for comparing compact subsets of complete metric spaces. The intuition behind it, in the metric space $(\mathbb{R}^d, ||\cdot||_2)$, is that it compares the morphology and spatial location of the compact subsets. In this particular case, the Hausdorff metric is defined over $(\mathbb{R}^2, ||\cdot||_2)$. The formal definition of the Hausdorff metric is for two subsets $A, B \subset \mathbb{R}^2$

$$d_H(A,B) = \max\{\sup_{a\in A} d(a,B), \sup_{b\in B} d(A,b)\},$$

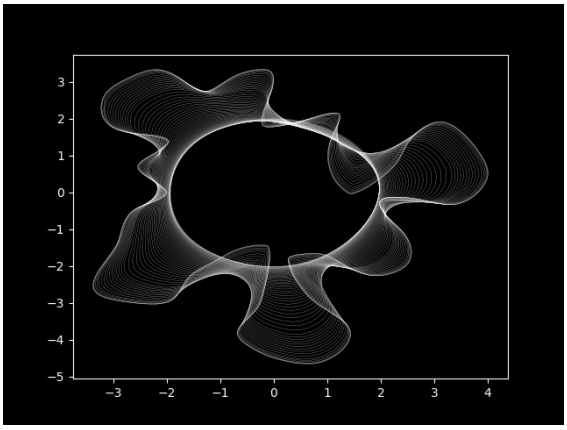
with

$$d(a,B) = \inf_{b\in B} ||a-b||^2,$$

see [22]. To compute the Hausdorff distance we use the function "directed_hausdorff" from SciPy. For the clustering, we consider the parameter space (δ_1, δ_2). We consider the constrain $\delta_1 \in (0, r]$. Since $\delta_2 < \delta_1$, the parameter space is the triangle $\Delta = \{(\delta_1, \delta_2) : 0 < \delta_1 \leq r, 0 < \delta_2 < \delta_1\}$. The parameter space is discretized. Each element of the parameter space is evolved 610 times. The final curves are



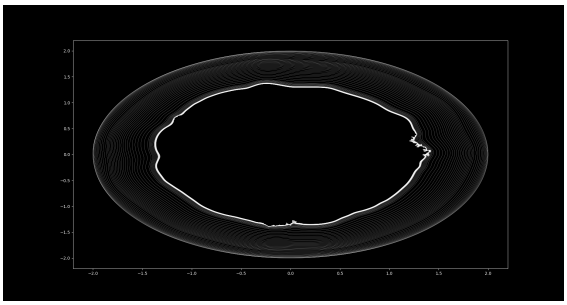
(a) Generative Art [1]



(b) Art Created in Project

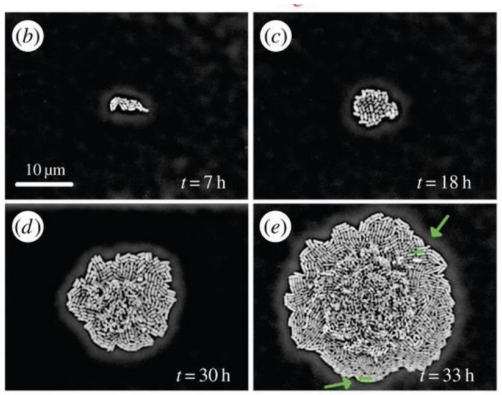
Figure 9: Comparison seen in art.

(a) Nature - Tree rings [20]

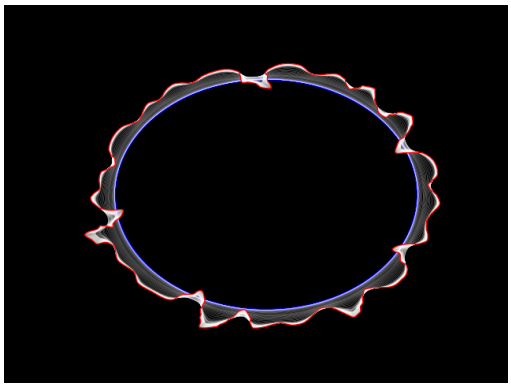


(b) Nature created in project

Figure 10: Comparison seen in nature.



(a) Nature - Biofilm [21]

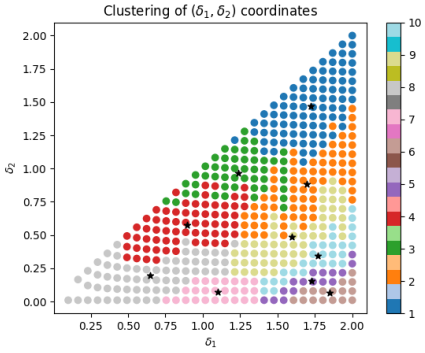


(b) Our simulation

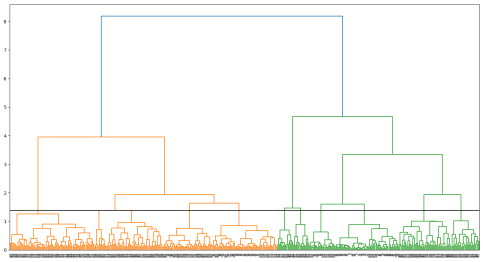
Figure 11: Comparison seen in biology.

stored in a distance matrix in its condensed form, i.e., the upper triangular part of the distance matrix is stored as a vector. We apply a hierarchical clustering algorithm to the condensed matrix. The linkage method is ward, see [23] for more details on the ward linkage. We choose this linkage method after several tests with the linkages detailed in [23].

There is a main theoretical concern about this idea, although the scope of this work does not allow to pursue its clarification. The clustering process as described, has as hypothesis that the curves evolved from parameters close enough in the parameter space, remain close in the Hausdorff space. We have strong reasons to believe this is true. The physical process applied to the points in the curve, is continuous, and the introduction of points through the two splitting mechanism is a local process. We now present the result of the clustering process. For these initial results, we consider a grid over the triangle Δ of 465 elements as shown in Figure 12a and Figure 13a. For the evolution of each of these means, look at the Appendix section at the end of the paper. In Figures 12b and 13b we show the dendograms resulting from the ward linkage. To validate the number of clusters obtained. We computed the mean coordinates $(\hat{\delta}_1, \hat{\delta}_2)$ of each cluster, those are shown in black stars in Figures Figure 12a and Figure 13a. We then evolved the means of each cluster and qualitatively compared them. We choose the number of clusters so that the change in the curves resulting from the means had a significant visual difference.

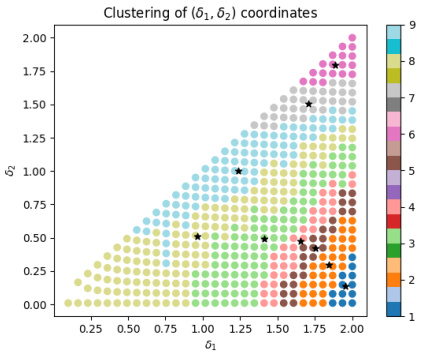


(a) Clusters in the parameter space for the planet force

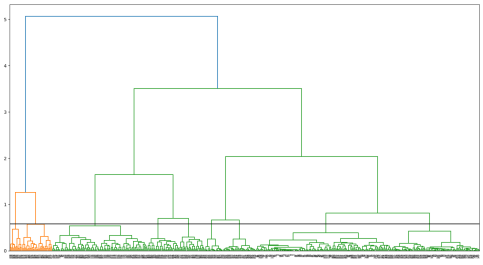


(b) Dendogram planet force

Figure 12: Analysis of parameter space for planet force.



(a) Clusters in the parameter space for the spring force



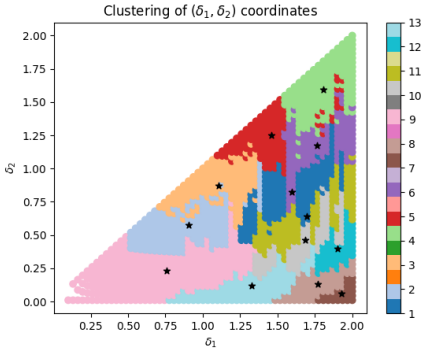
(b) Dendogram spring force

Figure 13: Analysis of parameter space for spring force.

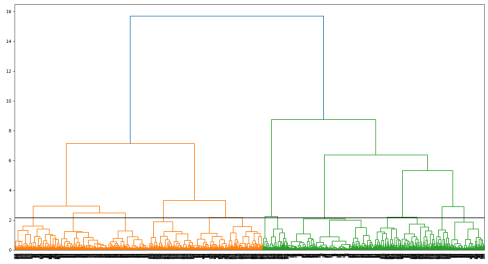
We can see that the geometric properties of the clusters in Figure 12a and 13a are quite different. The clusters in Figure 13a have, to some extend radial symmetry, whereas the clusters in Figure 12a seem to occupy larger regions and to have less defined boundaries.

Because of the irregularities that we found in the boundaries of 12a we decided to explore the effects

of considering a finer partition of the triangle Δ, see Figures 14a and Figures 14b. The introduction of more points to the partition lead to disconnected areas, see cluster 1 and mixture of clusters, see the red area in cluster 4. These results results are definitely unexpected and need to be explored in more detail.



(a) Clusters in the parameter space for the planet force



(b) Dendogram planet force

Figure 14: Analysis of parameter space for planet force.

Towards getting a better understanding on the differences between the behaviours of the forces, we studied how the planet force and the spring force induce the addition of points to the curves.

Note: The cuts for on the dendograms for the clusters are Figure 12a 1.39, FIgure 13a 0.58, Figure 14a 2.14, Figure 19a 0.957, and Figure 19b

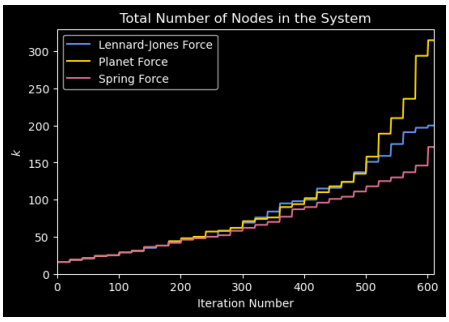
6.4 Number of Nodes in the System

For this experiment, we aim to compare how many nodes are being added to the system as time progresses. There are four measures to quantify this result. These are:

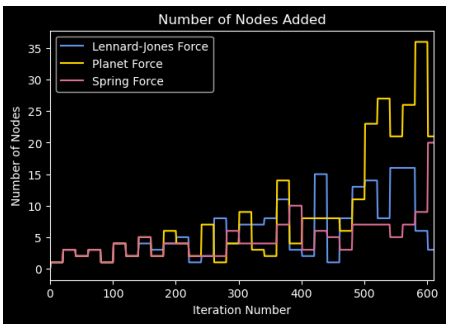
1. The total number of nodes k in the system at the end of each iteration
2. The number of nodes added to the system during an iteration
3. The number of nodes added to the system by the midpoint splitting scheme, during an iteration
4. The number of nodes added to the system by the curvature splitting scheme, during an iteration

The experiment was executed at the attraction distance $\delta_1 = 1.5$ and the repulsion distance $\delta_2 = 0.25$. For the Lennard-Jones force, planet force, and spring force, the time steps Δt used are 1×10^{-6}, 0.0008, and 0.001, respectively. In Figure **??** we observe that the total number of nodes k in the system is increasing as time propagates. This agrees with what we would expect because nodes are being added in each step. In Figure 15c , the number of nodes added due to the midpoint splitting scheme is always below the maximum of allowed nodes to split, which is twenty. For each of the three forces tested, the number of nodes added is lowest in the initial iterations (≤ 5). As time progresses, the number of nodes added has more significant variation. It is clear this makes up most of the behavior seen in Figure 15b , which plots the number of nodes added from both the midpoint and the curvature splitting methods. Since, in Figure 15d , only the planet forces uses curvature splitting, and it is used only in the later iterations. To check whether nodes can be added by the curvature splitting method when implementing the spring force, we check this result for nine δ_1, δ_2 combinations. These combinations correspond to the cluster centers from the hierarchical clustering for the spring (Fig. force 13a). The following tests could not be implemented for the Lennard-Jones force because it cannot handle as large of a range of δ_1, δ_2 combinations.

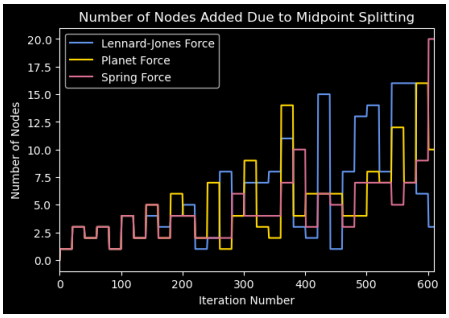
In Figure 16a, it is shown that changing δ_1, δ_2 did not increase the number of nodes added due to curvature splitting. This incentivized us to execute the test again, but for a smaller threshold on the curvature. The result in Figure 16b shows that decreasing the curvature threshold can allow for nodes to be added due to the curvature splitting method at some δ_1, δ_2. It is shown in Figure 17a and Figure 17b that the final state of the system is exactly the same for $\Delta t = 0.001$, regardless of the curvature



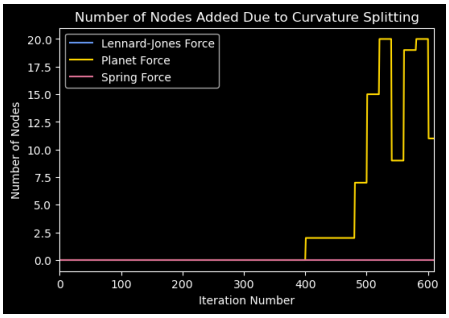
(a) Measure 1.



(b) Measure 2.



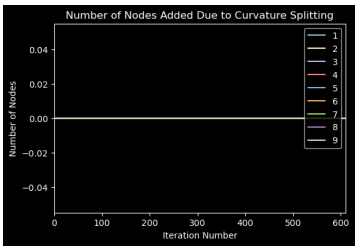
(c) Measure 3.



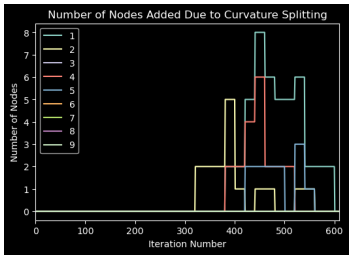
(d) Measure 4.

Figure 15: Plots with the iteration number as the independent variable and measures 1-4 as the dependent variables. The blue, yellow, and red lines correspond to results from the Lennard-Jones force, planet force, and spring force, respectively.

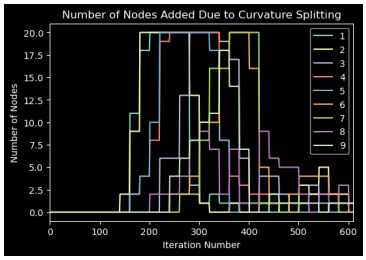
threshold. This signifies that the results in Figure 16b are entirely due to the lowering of the curvature threshold. While keeping the curvature threshold at this new low, we increased Δt to 0.003 so the system can have a more interesting geometry as seen in Figure 17c . The result in Figure 16c shows that the curvature splitting method was implemented for all tested cluster centers. This agrees with what we would expect because a system with more variable protrusions should exhibit splitting due to the curvature.



(a) Curvature Threshold = 7.5, $\Delta t = 0.001$



(b) Curvature Threshold = 3.5, $\Delta t = 0.001$

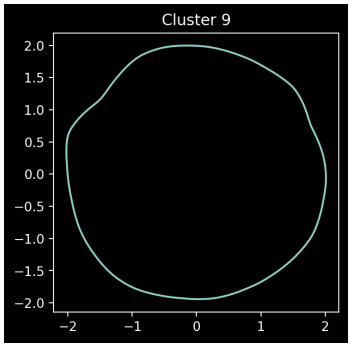


(c) Curvature Threshold = 3.5, $\Delta t = 0.003$

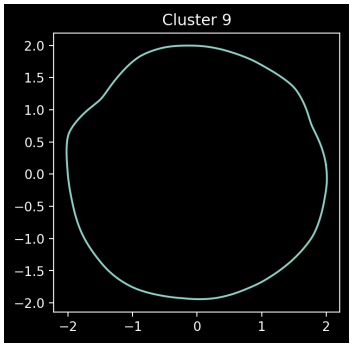
Figure 16: Number of nodes added due to curvature splitting tested at each δ_1, δ_2 corresponding to cluster centers.

6.5 Spring force clustering

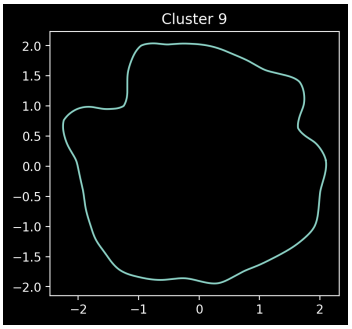
In the previous section, we explored the number of nodes added due to the curvature splitting algorithm while using the spring force. The tested δ_1, δ_2 combinations were the centers of the nine clusters seen



(a) Curvature threshold = 7.5, $\Delta t = 0.001$



(b) Curvature threshold = 3.5, $\Delta t = 0.001$



(c) Curvature threshold = 3.5, $\Delta t = 0.003$

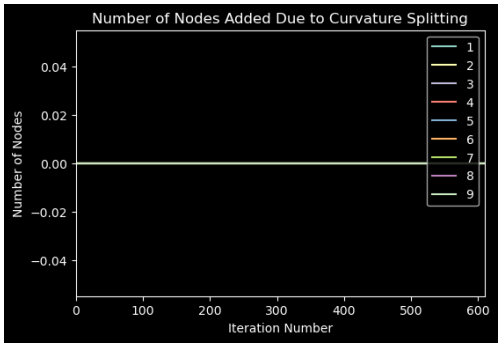
Figure 17: Final state of the system using $\delta_1 = 1.23864$, $\delta_2 = 1.00309$.

in Figure 13a. We wondered if the use of the curvature splitting scheme affects the clustering for the parameter space of the spring force. As seen in 13a, the spring force clusters appear to have a radial pattern. The case in which the clustering was determined was when 7.5 was the lower curvature threshold, such that no splitting occurred due to curvature. This clustering looks different from the one for the planet force in Figure 14a, where splitting due to curvature was exhibited as seen in Figure 15d. We also wondered whether, if splitting due to curvature affects the clustering for the spring force, the new clustering for the spring force will look similar to the planet force clustering. To explore these ideas, the first case we tested with the spring force was when the curvature threshold is 3.5 and $\Delta t = 0.001$, which corresponds to the parameters tested in 16b. The new clustering for this case is seen in Figure 19a. This still exhibits a radial pattern but the edges of the clusters are less smooth. In this case, we know that the curvature splitting is not always executed, and the magnitude of nodes added due to curvature splitting may be less than ten on average. It was executed when the previous cluster centers were tested, but it was not executed when the correct cluster centers were tested, as seen in Figure 18a. The second case we tested was when the curvature threshold is 3.5 and $\Delta t = 0.003$, which corresponds to the parameters tested in 16c. The new clustering for this case is seen in Figure 19b. This does not exhibit the radial pattern. Instead, its patter looks almost exactly like the one for the planet force in 14a. In this case, the curvature splitting appears to always be executed. Since, using the previous cluster centers it was executed for all δ_1, δ_2 combinations (Fig. 16c) and the same was true when using the correct cluster centers (Fig. 18b). In summary, for the spring force, the clusters exhibit a radial pattern (Fig. 13a) when there is no splitting due to curvature, the clusters exhibit a radial pattern with rough edges (Fig. 19a) when there is occasional splitting due to curvature, and the clusters exhibit a pattern similar to the planet force (Fig. 19b) when splitting due to curvature occurs in (likely) all simulations.

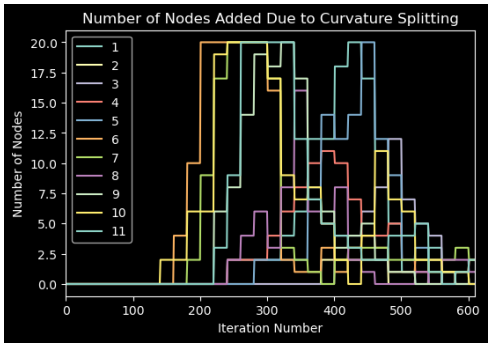
7 Summary and Conclusion

Differential growth is a process that simulates patterns found in nature. A core part of the simulation model is consideration of forces that define the behavior of the evolving structure of a smooth curve. To simulate this, we chose nodes on a smooth perturbed circle and let the nodes interact with each other through interaction forces found in nature – spring force, planet force, and the Lennard-Jones force. There were difficulties when simulating with the Lennard-Jones force; these be discussed in the next section, discussion and future works.

The simulation results showed that the output shapes depend on the input parameters, such as the forces, δ_1, δ_2, etc.. We coined a method to quantify how the output shapes depend on the parameters, in particular δ_1 and δ_2, which uses the Hausdorff measure. For different pairs of parameter values, we ran simulations and computed the Hausdorff measure of the final output, and ran clustering on these values. The set of pairs of parameter values in a same cluster, therefore, generates similar outputs in the sense of Hausdorff.

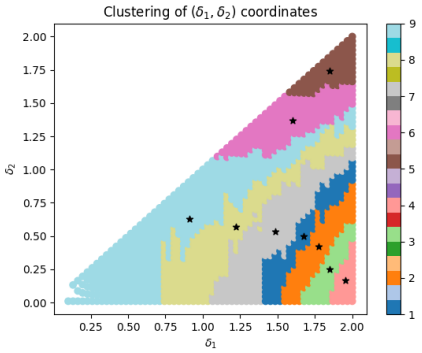


(a) Curvature threshold = 3.5, $\Delta t = 0.001$

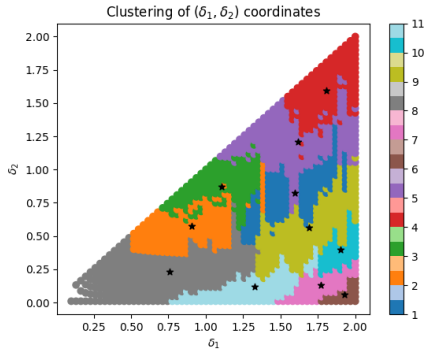


(b) Curvature threshold = 3.5, $\Delta t = 0.003$

Figure 18: Number of nodes added due to curvature splitting tested at each δ_1, δ_2 corresponding to the correct cluster centers.



(a) Clusters in the parameter space for the spring force $\Delta t = 0.001$



(b) Clusters in the parameter space for the spring force $\Delta t = 0.003$

Figure 19: Analysis of parameter space for the spring force for curvature threshold 3.5.

The goal of the project was to design an algorithm that exhibits differential growth. The main tool we leveraged for analysis was clustering using the Hausdorff measure. We hypothesized that the parameter space will vary in different regions (δ_1,δ_2 combinations) such that the results could be clustered. We successfully implemented a differential growth scheme that provides similar results to the artistic references and produces patterns appearing in nature. Of the proposed forces, the Lennard-Jones force was problematic. This may attribute to the fact that the denominator has such a high-degree polynomial makes the need of a time step really low. Thus obtaining interesting simulations from it, requires a lot of time steps.

The clustering procedure yielded valid outputs. However, it gave unexpected results such as unconnected clusters and getting the same clusters for two different forces, the planet and spring forces, by changing the lower bound on the curvature and increasing Δt. This leads us to think curvature might be the most relevant parameter when it comes to the clustered result. As part of the exploration of the clustering, we introduced four measures to determine the influence of the forces in the splitting schemes. These measures lead to the conclusion that the planet force is acts faster than the spring force. Meaning, it takes less total time for the planet force to produce interesting geometries that require the curvature splitting method than it would while implementing the spring force.

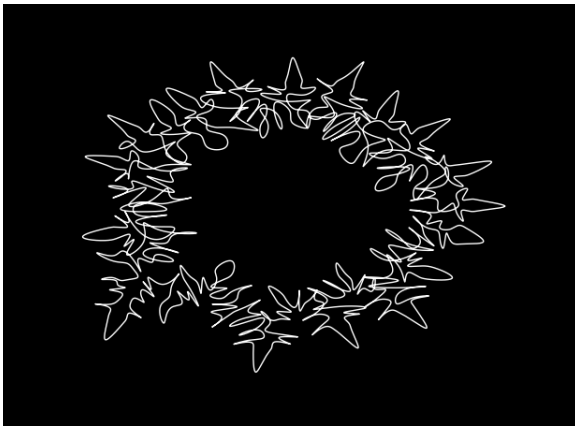


Figure 20: An example of a self-intersecting curve.

8 Discussion and Future Works

8.1 Intersecting Curve

A main difficulty in our simulations was that the curve would self-intersect after long iterations, as seen in Figure 20. Initially, we did not expect this behavior because the repulsion between nodes is magnitudes higher than the attraction (especially for planet force and Lennard Jones). However, we observed that if there are enough number of neighbors in the attractive range, the sum of the attractions can neutralize the repulsion. Also, considering the attraction radius is higher than the repulsion distance, there's a high chance that there are magnitudes higher number of attractive nodes than repulsive nodes.

Another source of the problem was that the nodes are not dense enough. Because they are not dense, there is enough space between two adjacent nodes for another node to pass through. However, there's a limit to computational resource. Continuing to increase the number of nodes to fill up all the gaps is not a practical solution. Also, we can observe that, because of the repulsion between pairs of close nodes, we tremendous number of new nodes would have to be added to fill up the fresh gaps after each iteration.

The third reason could be that the simulation is carried out in a discrete time domain rather than continuous. In our scheme, in each iteration, we compute the net force on every node based on our current state and then update the position of the node. Thus, while the nodes are moving from its current state to the next, the interactions among nodes are not taken into account, hence, letting the crossing happening.

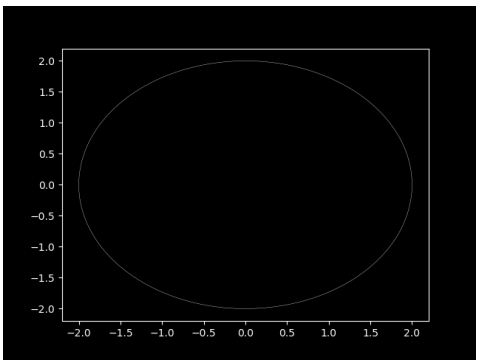
One solution may be to adapt the optimization scheme from *Inconvergent*. In this scheme, the nodes will try to optimize their positions in every iteration in such a way that they get close but not too close to their adjacent nodes. Simultaneously, the nodes are finding positions such that they are as far away as possible from other nodes within a certain distance [1]. It can be implemented in the future to see if this scheme effectively prevents intersections. If we can resolve the intersection problem, we can run the simulations longer to see more interesting geometry.

8.2 Lennard-Jones

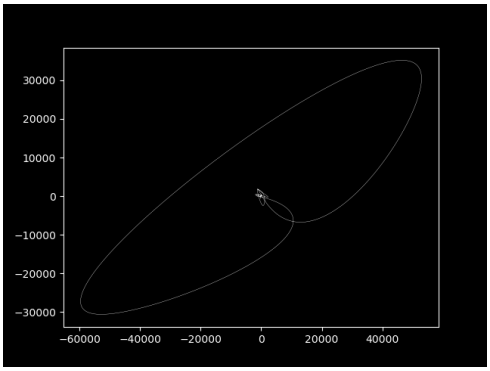
The Lennard-Jones force given by:

$$F_{Lennard\text{-}Jones} = -24\sigma^6 r_s^{-7} \left(\frac{2\sigma}{r_s}\right)^5 \tag{6}$$

This force was causing an issue in our simulation, which can be attributed to the high power of the distance between two nodes, r_s, in the denominator. Because of this, the denominator sometimes becomes zero, causing a division by zero error, or the repulsive force becomes too large that in the next iteration we see nodes exploding far beyond our scale as shown in Figure 21.



(a) Initial configuration



(b) After one iteration with Lennard-Jones

Figure 21: Observe the change in the scale of the system. When applying Lennard-Jones, if the time step is not chosen small enough, the system explodes after just one iteration.

One solution we tested was to significantly reduce the time step so that, even when there is a high repulsive force, the coordinates of the nodes are updated slowly, thus preventing the system from exploding. This was an effective solution for preventing the explosion, but the time step was too small to see any meaningful results within the same number of iterations we used for the spring and the planet forces.

Thus, for future work, we can try working with higher precision arithmetic (e.g. quadruple precision) to prevent the zero division and run the simulation for an extended amount of time to see meaningful results when applying the Lennard-Jones force.

8.3 PDE

We realized that there is a similar problem that was solved few years ago. In his Ph.D. dissertation, J.A. Sethian worked on the evolution of interfaces and developed the Level Set Methods and Fast Marching Method, see [24]. The problem treated in the present work and in the mentioned book have a high similarity. Both consider a boundary that is evolving, but there is an underlying hypothesis that differs. In [24], they consider that the velocity field that causes the boundary to deform is constant. For our work, the particles induce a vector field that is changing in time depending on the amount and position of particles. Prior to the methodology [24] one of the standard techniques to do interface propagation was to sample points from the boundary and use ODE solver for those points in the boundary. These methods were known as marker/string methods. The problems that they had with such formulation are similar to the problems we encountered here, curves self intersect and they might loose differentiability. The core idea in [24] was that the problem can be formulated as a PDE, the Eikonal equation when the velocity field only induces expansion of the boundary, or a Hamilton-Jacobi equation when the boundary propagates inwards and outwards. There are amazing advantages of the PDE formulation. First of all, the problem of self intersection and formation of peaks, are solved. Secondly, due to the nature of the PDE, solvers from hyperbolic PDEs can be used to solve the Hamilton-Jacobi equations. Thirdly, this allowed to use the ideas of propagation of fronds in an incredibly variety of applications.

We believe that it is possible to construct a PDE formulation for differential growth, maybe adapting the arguments in [24]. Such work is definitely out of the scope of a course work project and we believe it could be a publishable work. There are certain parts of the analysis that are hard to formulate in strict mathematical terms, in particular the addition of points into high curvature areas.

A physical interpretation of the difference between the differential growth model and the model in [24] could be that the boundary that is deformed by the vector field, has physical properties, such as elasticity or surface tension that change how that can make an interface to change, without the necessity of an external force acting on it. It is interesting that in [24] the author did not consider intrinsic physical properties of the system.

8.4 Interesting Extensions

8.4.1 3D

We worked with particles in 2D space, but it would be an interesting extension to work with 3D space. This extension would involve, but is not limited to, the following: updating coordinates and the velocities of the nodes, rewriting the forces to work for the 3D coordinate system, and modifying the k-d tree algorithm.

8.4.2 Different Initial Configurations

Our results here use the initial configuration of nodes perturbed on a circle. Running simulations with different and more interesting initial configurations, and then observing how the clusters using the Hausdorff measure analysis look different for different configurations, would be an interesting extension of this project.

References

[1] A. Hoff. Available: https://inconvergent.net/2016/shepherding-random-growth/#container-uniform.

[2] K. Reed. Available: https://www.instagram.com/orangebark/?hl=en.

[3] "Floraform – an exploration of differential growth," Jun 2015. Available: https://n-e-r-v-o-u-s.com/blog/?p=6721d.

[4] "Georg nees." Available: https://en.wikipedia.org/wiki/Georg_Nees.

[5] M. A. Boden and E. A. Edmonds, "What is generative art?," *Digital Creativity*, vol. 20, Jun 2009. doi: 10.1080/14626260902867915.

[6] A. Lomas, "Cellular forms: an artistic exploration of morphogenesis," Jul 2014. doi: 10.1145/2619195.2656282.

[7] C. Huang, Z. Wang, D. Quinn, S. Suresh, and K. J. Hsia, "Differential growth and shape formation in plant organs — pnas." PNAS.org. Available: https://www.pnas.org/content/115/49/12359.

[8] M. J. Razavi, T. Zhang, T. Liu, and X. Wang, "Cortical folding pattern and its consistency induced by biological growth," Sep 2015. Available: https://www.nature.com/articles/srep14477.pdf.

[9] H. S. Hosseini, K. E. Garcia, and L. A. Taber, "A new hypothesis for foregut and heart tube formation based on differential growth and actomyosin contraction," Available: https://journals.biologists.com/dev/article/144/13/2381/48141/A-new-hypothesis-for-foregut-and-heart-tube.

[10] "Information fingerprint examination -terminology, definitions and acronyms." Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/267523/FingerprintTerminology.pdf.

[11] "The differential growth (2d)." Available: https://www.architect-ly.com/the-differential-growth-2d.

[12] "Differential line growth - the quick way." Available: www.youtube.com.https://www.youtube.com/watch?v=SOk5qVXrFQA.

[13] W. Contributors, "Lloyd's algorithm." Available: https://en.wikipedia.org/wiki/Lloyd%27s_algorithm.

[14] O. Deussen, "Aesthetic placement of points using generalized lloyd relaxation," 2009. doi: 10.2312/COMPAESTH/COMPAESTH09/123-128.

[15] "Point relax." Available: https://www.sidefx.com/docs/houdini/nodes/sop/relax.html.

[16] "Jitesh sapra - differential growth." Available: https://jiteshsapra.com/differential-growth.

[17] N. Chee, F. Rembart, and M. Winkel, "A recursive distribution equation for the stable tree," Dec 2018. Available: https://arxiv.org/abs/1812.08636.

[18] W. Contributors, "L-system." Wikipedia. Available: https://en.wikipedia.org/wiki/L-system.

[19] "Binary tree derivatives – uvn lab." Available: http://uvnlab.com/binary-tree-derivatives-en/.

[20] "Tree rings." Available: https://ecophys.utah.edu/tree-rings.html.

[21] J. Dervaux, J. C. Magniez, and A. Libchaber, "On growth and form of bacillus subtilis biofilms." Interface Focus, 2014. Available: https://royalsocietypublishing.org/doi/10.1098/rsfs.2013.0051.

[22] W. Contributors, "Hausdorff distance." Wikipedia. Available: https://en.wikipedia.org/wiki/Hausdorff_distance.

[23] "Hierarchichal clustering." Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html.

[24] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.* 1999.

Appendix A: Division of Labor

Sasha M. Bakker

Sasha has worked on programming the initialization of the toy problem using NetworkX. Furthermore, Sasha worked on the initial forces algorithm, the splitting algorithm, how and where new nodes will be added in between two existing nodes, and has also worked on the checkpoint II write-up. She executed the experiment testing the number of nodes in the system per iteration. Sasha has also worked both the written report and the video.

Sebastian Gutierrez H.

Sebastian has worked on shaping the algorithms, structure of the code, and the initial flowchart, took a lead on creating the functions for the various forces, the use of K-d tree algorithm, implementing the use of the Hausdorff distance, and creating the clustering analysis. Additionally, Sebastian wrote the animation exporting function. Sebastian has also worked both the written report and the video.

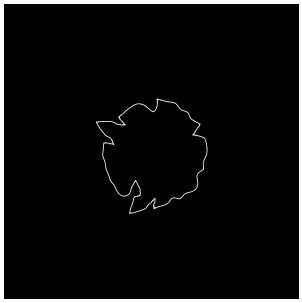
Jieun Seong

Jieun has worked on writing the literature review and organizing our initial work into checkpoint I. Furthermore, Jieun has worked on preparing our code to run through the cluster in the Mathematics Department. Jieun debugged the code for node ordering and curvature computing, especially for the boundary cases. Jieun suggested and implemented a way to prevent the fractal structure. Jieun has also worked both the written report and the video.

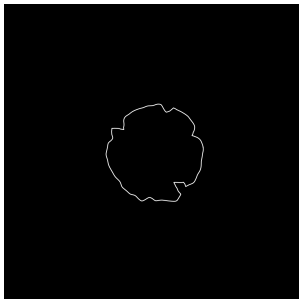
Nathan B. Williams

Nathan has worked on shaping the algorithms, structure of the code, and the initial flowchart. Furthermore, Nathan worked on the initial forces algorithm, the splitting algorithm, how and where new nodes will be added in between two existing nodes, and has also worked on the checkpoint II write-up. Nathan has also worked both the written report and the video.

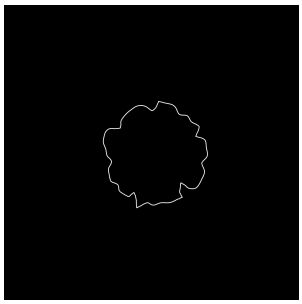
Appendix B: More Figures



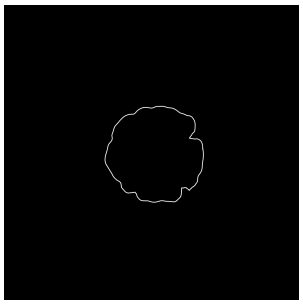
(a) Cluster 1



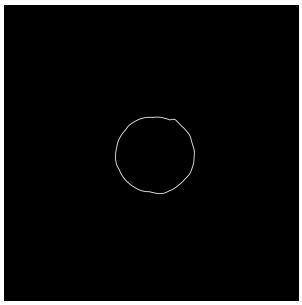
(b) Cluster 2



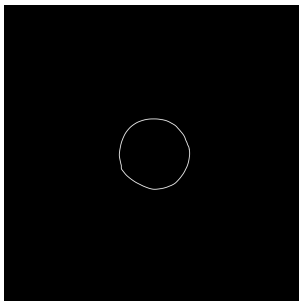
(c) Cluster 3



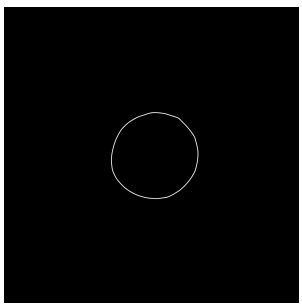
(d) Cluster 4



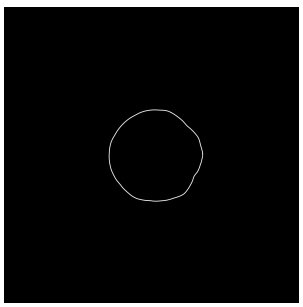
(e) Cluster 5



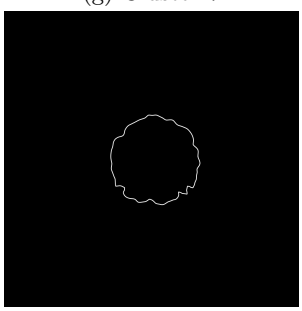
(f) Cluster 6



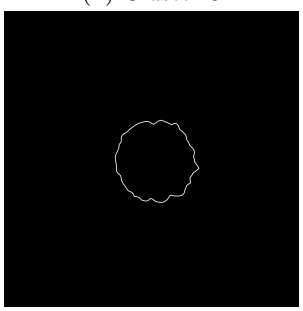
(g) Cluster 7



(h) Cluster 8

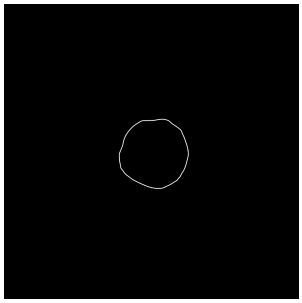


(i) Cluster 9

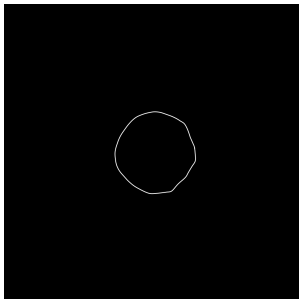


(j) Cluster 10

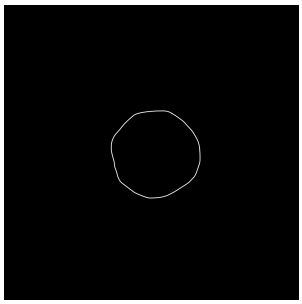
Figure 22: Evolution of Means for Figure 12a



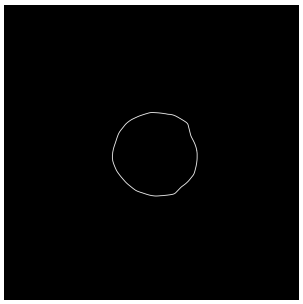
(a) Cluster 1



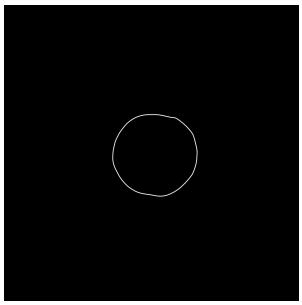
(b) Cluster 2



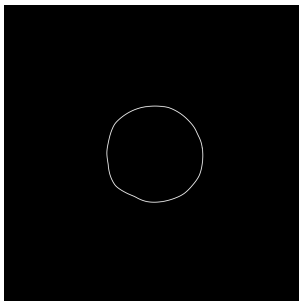
(c) Cluster 3



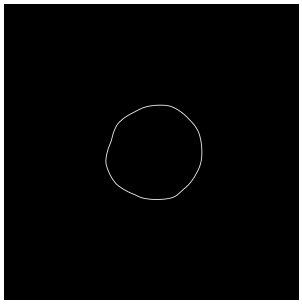
(d) Cluster 4



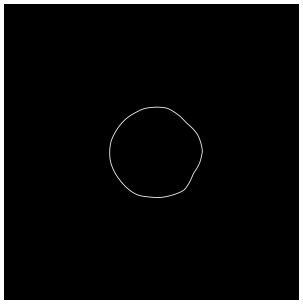
(e) Cluster 5



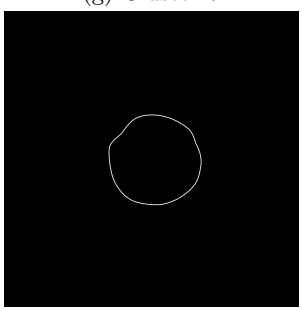
(f) Cluster 6



(g) Cluster 7

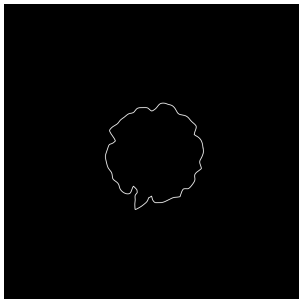


(h) Cluster 8

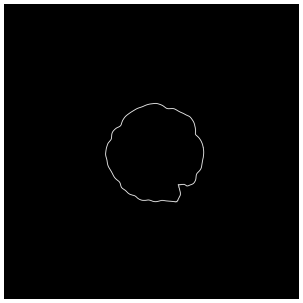


(i) Cluster 9

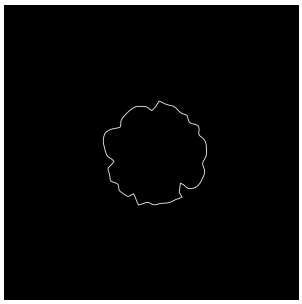
Figure 23: Evolution of Means for Figure 12a



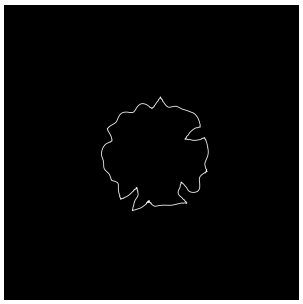
(a) Cluster 1



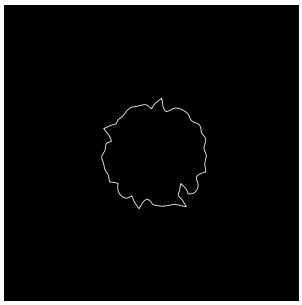
(b) Cluster 2



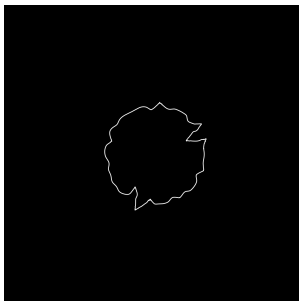
(c) Cluster 3



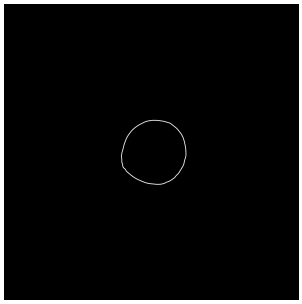
(d) Cluster 4



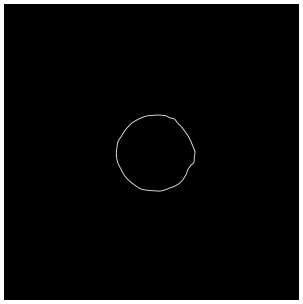
(e) Cluster 5



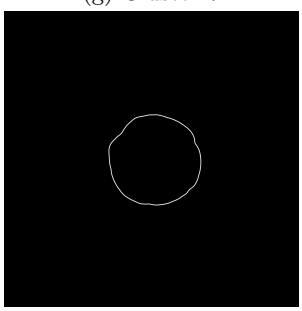
(f) Cluster 6



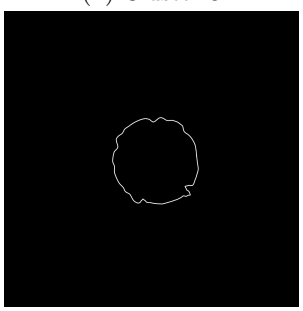
(g) Cluster 7



(h) Cluster 8

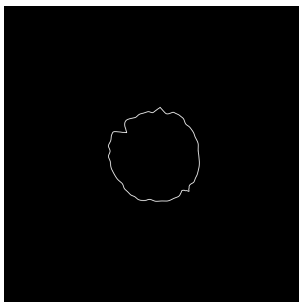


(i) Cluster 9

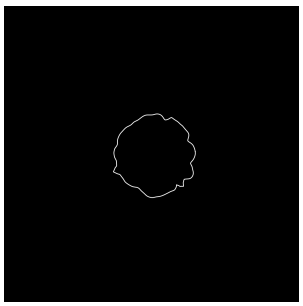


(j) Cluster 10

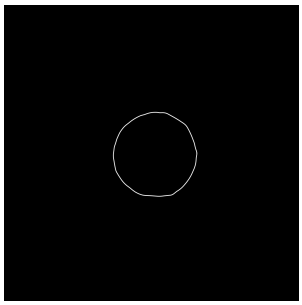
Figure 24: Evolution of Means for Figure 14a



(a) Cluster 11

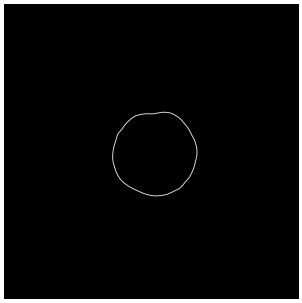


(b) Cluster 12

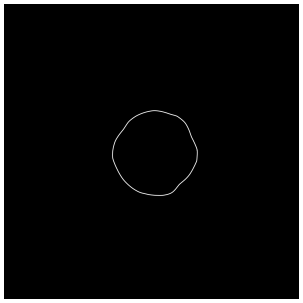


(c) Cluster 13

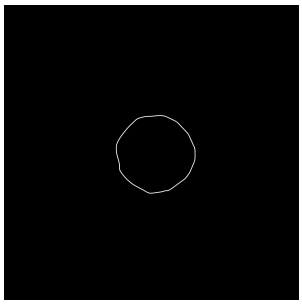
Figure 25: Evolution of Means for Figure 14a Continued



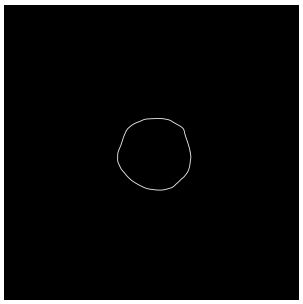
(a) Cluster 1



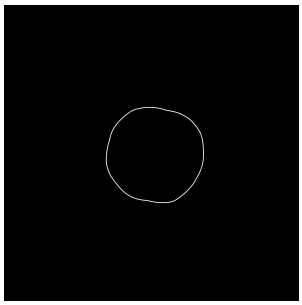
(b) Cluster 2



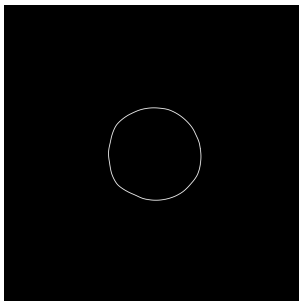
(c) Cluster 3



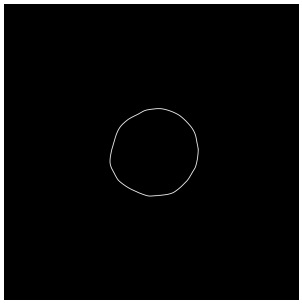
(d) Cluster 4



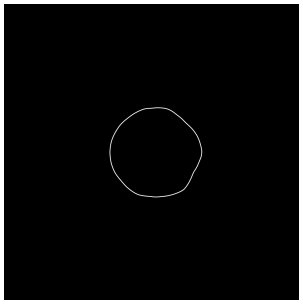
(e) Cluster 5



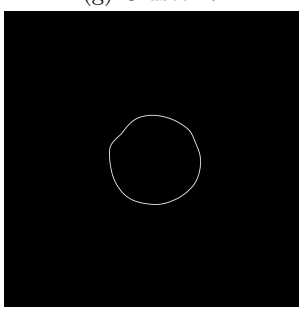
(f) Cluster 6



(g) Cluster 7

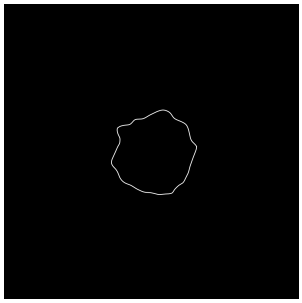


(h) Cluster 8

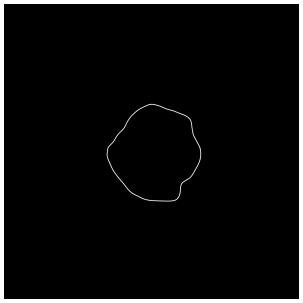


(i) Cluster 9

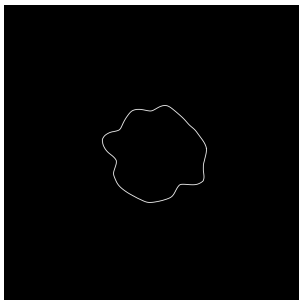
Figure 26: Evolution of Means for Figure 19a



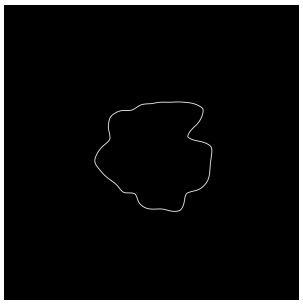
(a) Cluster 1



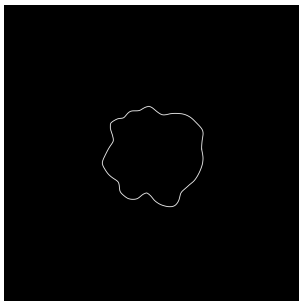
(b) Cluster 2



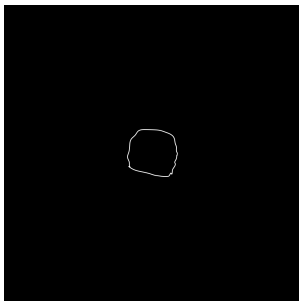
(c) Cluster 3



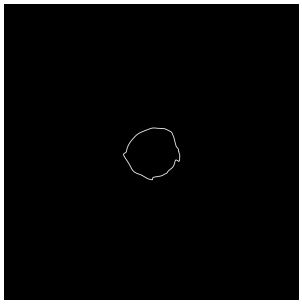
(d) Cluster 4



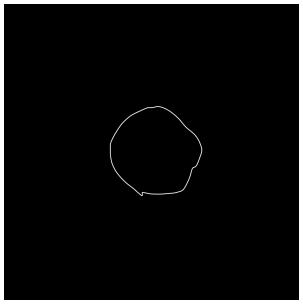
(e) Cluster 5



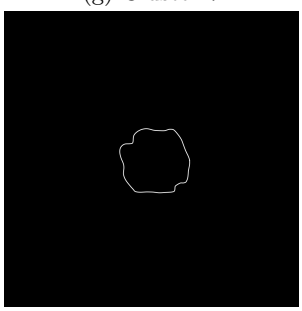
(f) Cluster 6



(g) Cluster 7

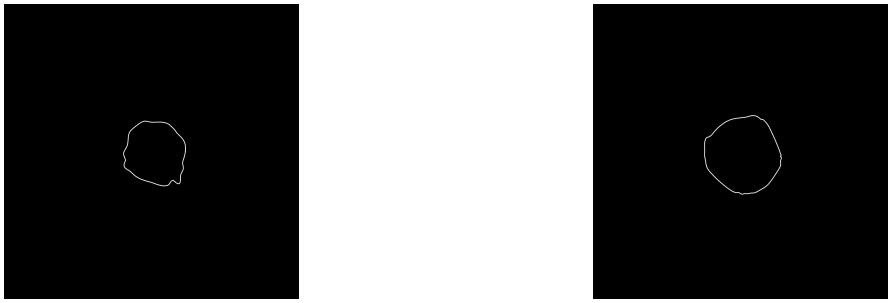


(h) Cluster 8



(i) Cluster 9

Figure 27: Evolution of Means for Figure 19b



(a) Cluster 10

(b) Cluster 11

Figure 28: Evolution of Means for Figure 19b Continued