Universidad de Sevilla

Facultad de Física

Modelando árboles y el sistema vascular del insecto efímera

Autor:
Carlos Rubio Miranda

Director:
Dr.Francisco Jiménez Morales

Curso:
2019-2020

Agradecimientos

En primer lugar quiero agradecer enormemente al **Dr. Francisco Jiménez Morales** por brindarme la oportunidad de desarrollar este proyecto, apoyarme en cada paso que he dado y abrirme las puertas al maravilloso mundo de la programación científica. Desde el principio me transmitió su pasión por el proyecto, pasión que compartí desde los primeros compases del desarrollo.

En segundo lugar, me gustaría agradecer a aquellas personas del equipo docente que se preocuparon de mostrarme los misterios de la física y educarme como científico. Aunque el camino de la Física es arduo, su dedicación me motivó a enfrentarme a todas las dificultades que esta rama de la ciencia entraña.

En tercer lugar, quiero agradecer a todos aquellos compañeros con los que compartí horas y horas de trabajo en equipo. Sin ellos nunca podría haber afrontado los desafíos de la carrera. Como todos sabemos, el grado en Física se supera en equipo y con compañerismo.

En cuarto lugar, quiero agradecer a **Fernando Casares** del Centro Andaluz de Biología del Desarrollo (CSIC- Pablo de Olavide) por ceder los datos experimentales que han formado los cimientos de este proyecto.

En último lugar reservo un espacio protagonista para mi familia. Agradezco a mis padres haber confiado en mí y haberme apoyado día a día, aconsejándome y animándome cada vez que dudaba de mis capacidades. A mi hermano que me ha ayudado inmensamente a adquirir los conocimientos necesarios para el desarrollo de este proyecto tan mezclado con la programación y que me ha servido de inspiración y referencia para perseguir mis metas.

Índice de contenidos

Índice de figuras

1. Introducción

En la naturaleza existe una gran cantidad de sistemas que responden a patrones geométricos caracterizados por ser autosimilares. Esta característica se basa en que la estructura geométrica completa es similar o exacta a las partes de la misma. Estos sistemas se conocen como fractales y se caracterizan por tener una dimensión fractal de valor fraccional [AAA14].

En muchos casos, estos sistemas presentan un alto grado de autosimilitud, es decir, son patrones geométricos muy definidos y con pocas imperfecciones entre sus partes. Debido a esta estabilidad geométrica, estos sistemas pueden ser simulados mediante el uso de procesos iterativos basados en fenómenos concretos. Por un lado, sometiendo a procesos iterativos definidos a un conjunto de valores iniciales aleatorios se obtienen fractales regulares bien definidos. Este es el caso del Fractal de Sierpinski y otros fractales de origen artificial como puede verse en la Figura 1, [WPBF], [JC].

(a) (b)

Figura 1: Fractales originados mediante algoritmos iterativos. (a) Hoja fractal de Barnsley. Se obtiene mediante la iteración de una función afín. (b) Fractal de Sierpinsky. Se genera mediante un proceso aleatorio denominado “el juego del caos"

Por otro lado, métodos como el de difusión limitada por agregación - DLA, [DLA] - o el uso de Sistemas-L son capaces de simular fractales naturales con pocas imperfecciones - como es el caso de la formación molecular de los copos de nieve.

El método de difusión limitada por agregación se basa en el movimiento browniano. A partir de una o un conjunto de partículas raíz, se propone una nueva partícula situada en un punto aleatorio de una circunferencia centrada en la partícula raíz. Sobre esta se plantea un movimiento browniano que no cesa hasta que se sale de la circunferencia o se sitúa en las inmediaciones de una de las partículas centrales, fijándose en esa posición y pasando a formar parte del conjunto central. En ambos casos, el proceso se repite un número elevado de veces hasta que una estructura queda formada. En la figura 2 se observa que la estructura final presenta un gran número de ramificaciones y podría compararse con la formación de los copos de nieve. Este método de simulación, debido a su origen teórico, es muy útil para la representación de estructuras en cuya formación el movimiento browniano es esencial. Como puede ser en estructuras de polvo o el propio copo de nieve.

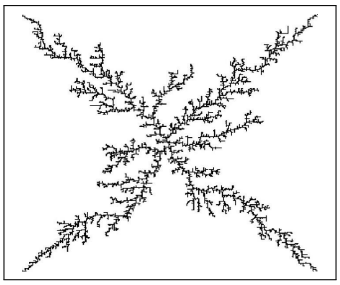


Figura 2: Ejemplo de una simulación basada en DLA

El método de uso de los Sistemas-L, [WPSL], se basa en las características de estos. Un sistema-L es un conjunto de reglas y símbolos que se puede aplicar a un caso concreto. Con esto se especifican unas variables como podría ser una localización espacial, unas reglas de comportamiento y un inicio. Se toma entonces un punto espacial concreto como inicio. Este se constituye en una variable y se le aplican las reglas del modelado. Dando lugar a un nuevo conjunto de variables sobre las que se vuelve a aplicar recursivamente las reglas. Este carácter recursivo condiciona que las estructuras resultantes tengan un fuerte grado de autosimilitud, siendo este un buen sistema de simulación para estructuras fractales puras pero no muy acertado para estructuras fractales naturales.

Sin embargo, existe un gran número de fractales en la naturaleza con un grado mucho menor de autosimilitud. Entre estos figuran los fractales de origen orgánico cuya formación está altamente condicionada por el entorno. Aunque siguen respondiendo a procesos iterativos de formación, estos son más complejos y su alto número de imperfecciones hace que su análisis y simulación se complique. En este proyecto se van a tratar dos tipos de sistemas orgánicos que responden a estas características: Las ramificaciones de los árboles y el sistema vascular del insecto Efímera.

Para salvar las dificultades matemáticas que presentan estos fractales orgánicos este estudio se va a valer de la comparativa visual entre distintos sistemas. De esta manera se podrá contrastar la validez de las simulaciones con las imágenes experimentales así como la influencia de diferentes parámetros en las características finales del sistema.

La simulación de estos sistemas orgánicos se va a basar en el uso del Algoritmo de Colonización Espacial - SCA [ABP07], [FC20]. Este algoritmo responde a un conjunto de procesos iterativos que dependen del entorno y de la estructura ya formada. Los cambios en el entorno se simularan como cambios aleatorios sujetos a limitaciones establecidas. Para reforzar el análisis de las simulaciones mediante la comparativa visual se utilizarán diferentes medidas numéricas sobre los resultados y sobre las imágenes experimentales. Dado que se trata de un sistema fractal, la medida de la dimensión fractal permitirá corroborar el ajuste de la simulación a la realidad aunque debido al gran número de irregularidades que presentan estos fractales orgánicos sus valores variaran con cada simulación concreta que se haga.

2. Teoría: Colonización espacial

En 1971 Hisao Honda [HH71] presenta un primer modelo visual de la estructura de un árbol considerando esta como una ramificación recursiva de la estructura caracterizada por un pequeño numero de atributos geométricos como los ángulos de las ramificaciones y las distancias entre los puntos de ramificación.

Posteriormente, Adam Runions, Brendan Lane, y Przemyslaw Prusinkiewicz [ABP07] plantean en 2007 un algoritmo de simulación destinado a recrear las estructuras ramificadas de algunos árboles. Este algoritmo debido a sus características se bautiza como Algoritmo de Colonización Espacial - SCA por sus siglas en inglés: Spatial Colonization Algorithm.

Es en los resultados de este planteamiento en los que se basa este proyecto puesto que el cuerpo de este estudio es una estructura ramificada natural similar a las utilizados en el artículo. Por ello, en los siguientes apartados se desarrollará en profundidad este algoritmo, su aplicación a los cuerpos vegetales y su posible aplicación en la estructura vascular de los insectos efímera - La cual es el objetivo principal del proyecto.

2.1. Algoritmo de Colonización Espacial (SCA)

El Algoritmo de Colonización Espacial se basa en la existencia de dos unidades básicas: Los nodos y los puntos de atracción. Por un lado, los nodos representan las unidades de la estructura final y albergan la información de la posición en la que se encuentran y la iteración en la que han sido creados. Al conjunto creado por estos nodos lo denotaremos como $N(r, t)$ siendo r la posición y t la iteración. Por otro lado, los puntos de atracción se definen exclusivamente por su posición. Este conjunto de puntos se denotarán como $S(\nu)$, siendo ν la posición que ocupan. Las posiciones tanto de los nodos como de los puntos de atracción se pueden tomar en un espacio tridimensional o bidimensional en función del resultado requerido.

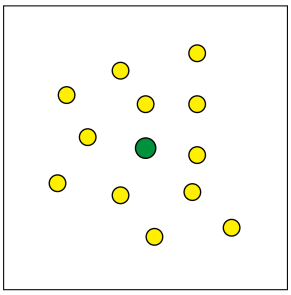
Los puntos de atracción representan la primera diferencia con el algoritmo de difusión por agregación puesto que en el SCA - pese a compartir la necesidad de un nodo inicial - también es necesario definir este conjunto de puntos de atracción que definirán posteriormente la macroestructura del modelo.

En primer lugar, es necesario por tanto definir un conjunto de puntos de atracción y un conjunto de nodos iniciales - Por regla general este conjunto se basará en un solo nodo - como se puede ver en la figura 3, a).

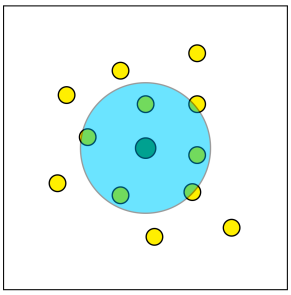
Como se ha mencionado, este algoritmo tiene un carácter iterativo. Es decir, se aplica reiteradamente sobre los dos conjuntos definidos $N(r)$ y $S(\nu)$ modificando ambos conjuntos antes de iniciar la siguiente iteración. La base del algoritmo sigue los siguientes pasos de manera sistemática que se aplican uno a uno a cada nodo del conjunto $N(r)$.

En primer lugar se observa para el nodo en cuestión que puntos de atracción del conjunto $S(\nu)$ se encuentran a una distancia definida o menor a esta. A esta distancia se la denomina radio de atracción, r_a. Los puntos del conjunto $S(\nu)$ ejercen influencia sobre el nodo por tanto si se encuentran a una distancia menor que r_a, es decir, si están dentro del área formada por la circunferencia con radio r_a y con centro situado en la posición

del nodo. En la figura 3, b), se observa este área en azul que contiene cinco puntos de atracción.



(a) Condiciones iniciales de una simulación. En verde el nodo inicial.



(b) Ejemplo del área de influencia para un r_a definido

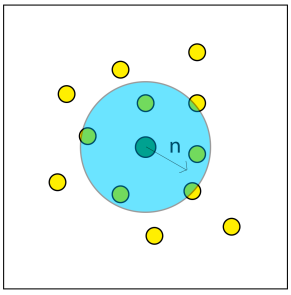
Figura 3: Primer paso del SCA: Busqueda de puntos de atracción

Los puntos de atracción que ejercen influencia sobre el nodo se incluyen en un nuevo conjunto $S'(\nu)$.

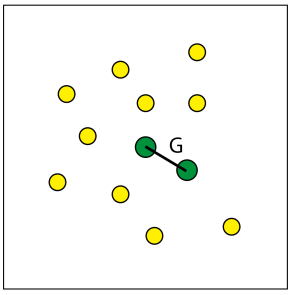
En siguiente lugar, si el conjunto $S'(\nu)$ no es nulo, se calcula la dirección de crecimiento de un nuevo nodo a partir del nodo en cuestión. Esta dirección se define como la suma de los vectores normalizados de la distancia entre cada punto de atracción y el nodo. El vector resultante - Ver figura 4, a)- de esta suma se normaliza finalmente para obtener la dirección unidad de crecimiento del nodo con posición r:

$$\vec{n} = \sum_{S'(\nu)} \frac{\vec{r} - \vec{\nu}}{||\vec{r} - \vec{\nu}||} \rightarrow \vec{n} = \frac{\vec{n}}{||\vec{n}||} \tag{1}$$

Con esta dirección, un nuevo nodo se creará con posición $\vec{r'} = \vec{r} + \vec{n} * G$ dónde G es un parámetro fijo en la simulación y define la distancia fija entre los nodos - Ilustrado en la figura 4, b). Este nuevo nodo se añadirá al conjunto $N(r)$.



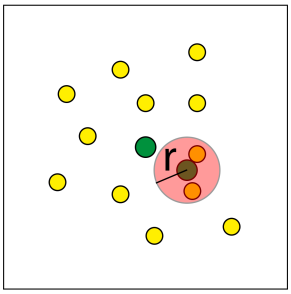
(a) Definición del vector de crecimiento, $\vec{n}$, en función de los puntos de atracción influyentes.



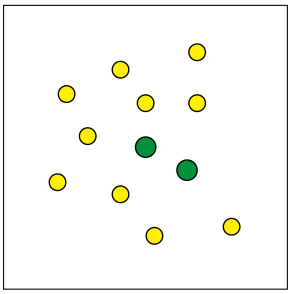
(b) Creación de un nuevo nodo a una distancia G del original

Figura 4: Análisis y creación de un nuevo nodo.

Finalmente, sobre este nuevo nodo, se analizan que puntos de atracción se encuentran a una distancia menor de r_e, parámetro estático denominado radio de eliminación. Observamos en la figura 5, izquierda, como alrededor del nuevo nodo hay dos puntos de atracción que se encuentran en el área de eliminación y en la derecha como quedan eliminados. Estos puntos son retirados del conjunto $S(\nu)$ y, por tanto, no se tienen en consideración en los siguientes cálculos de la simulación.



(a) Cálculo de los puntos de atracción dentro del área de eliminación del nuevo nodo



(b) Puntos de atracción calculados sustraídos del conjunto $S(\nu)$

Figura 5: Proceso de eliminación de puntos de atracción al crecer un nuevo nodo

Una vez aplicado este algoritmo sobre el nodo inicial, la simulación continúa en una nueva iteración en la que este algoritmo se aplica en orden a todo el conjunto de nodos $N(r)$. En el caso de ejemplo se volvería a aplicar sobre el nodo inicial y sobre el nodo recién creado. Es de esperar que de esta iteración, el conjunto de nodos pasara a contener cuatro elementos. El algoritmo se aplica entonces iterativamente al conjunto creciente de nodos hasta una iteración máxima fijada o hasta que el conjunto de puntos de atracción $S(\nu)$ quede vacío.

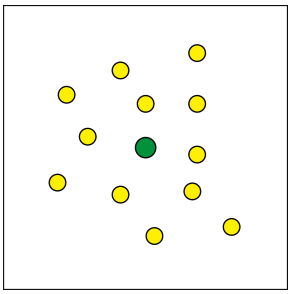
A lo largo de la simulación se puede observar el uso de múltiples parámetros. Es la definición de estos parámetros es lo que establece el resultado final, la forma que presenta y las características de las ramificaciones. Aunque trataremos en más detalle el efecto de estos parámetros en secciones más avanzadas, es preciso definir esquemáticamente estos parámetros y las condiciones iniciales del sistema:

- Condiciones iniciales:
  - $N(r)_{inicial}$: Conjunto de nodos iniciales. La posición de estos nodos dará lugar a diferentes macroestructuras finales.
  - $S(\nu)_{inicial}$: Conjunto de puntos de atracción iniciales. Se toman habitualmente como un conjunto de puntos aleatorios comprendidos en un área definida. Este área definirá una parte de la macroestructura resultante.
- Parámetros estáticos: Las relaciones entre los valores de estos tres parámetros definirán las características de la ramificación de la estructura, es decir, la microestructura de la simulación.

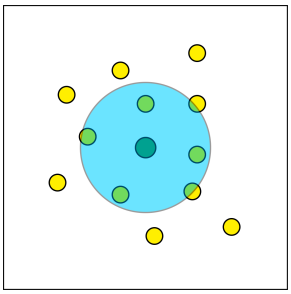
- r_a: Radio de atracción. Distancia a la cual un punto de atracción ejerce influencia sobre un nodo concreto.
- r_e: Radio de eliminación. Distancia a la cual un punto de atracción es eliminado al crecer un nodo concreto.
- G: Parámetro de Crecimiento. Distancia obligada entre un nodo crecido y el nodo respecto al cual crece.

Con todo esto, y a modo de resumen, el Algoritmo de Colonización Espacial quedaría esquematizado de la siguiente manera, apoyándonos en las secciones de la figura 6:

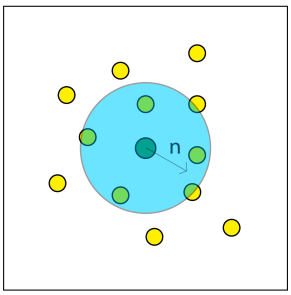
1. Definición de los parámetros iniciales: $N(r)_{inicial}$ y $S(\nu)_{inicial}$. Opcionalmente definir el número máximo de iteraciones - subfigura a:
   - $N(r)_{inicial}$ suele estar compuesto de un solo nodo.
   - $S(\nu)_{inicial}$ suele representar un conjunto de puntos aleatorios con una densidad uniforme que cubre un área definida que se pretende como parte de la macroestructura final
2. Aplicación del algoritmo base a cada nodo del conjunto $N(r)$ en función del conjunto $S(\nu)$, partiendo del nodo primero al último:
   - Cálculo del subconjunto $S'(\nu)$ en función de r_a - subfigura b.
   - Cálculo de la dirección de crecimiento en función del subconjunto $S'(\nu)$ - subfigura c.
   - Creación de un nuevo nodo, r', en función de la dirección de crecimiento y del parámetro G - subfigura d
   - Sustracción del conjunto $S(\nu)$ de los elementos a menor distancia que r_e del nodo creado - subfiguras e y f.
3. Repetición del segundo paso de manera iterativa sobre los conjuntos $N(r)$ y $S(\nu)$ resultantes de las aplicaciones anteriores.
4. Finalización del proceso si se alcanza el número máximo de iteraciones o el conjunto $S(\nu)$ resulta vacío en una de estas.



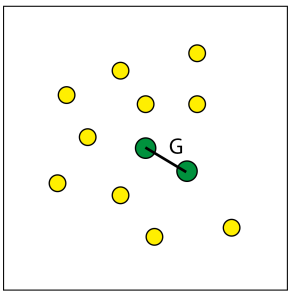
(a) Condiciones iniciales



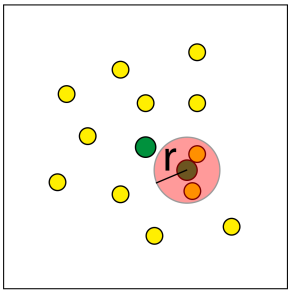
(b) Área de influencia con radio r_a



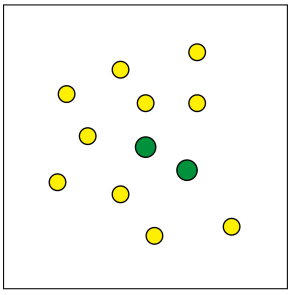
(c) Vector dirección de crecimiento, $\vec{n}$



(d) Nuevo nodo a una distancia G



(e) Área de eliminación con radio r_e



(f) Sistema tras la primera iteración

Figura 6: Esquema de crecimiento de un nodo basado en el algoritmo de colonización espacial

2.2. Aplicación sobre la estructura de árboles

Aunque el SCA se base en el esquema algorítmico aplicado a un nodo concreto, las disposiciones de las condiciones iniciales y de los parámetros fijos tendrán como resultado diferentes simulaciones lo cual permite la aplicación de este algoritmo a diferentes estructuras. En el artículo de Reunions el SCA es aplicado sobre la estructura de árboles y los resultados comparados con estas estructuras encontradas en la naturaleza. Para llegar a estas macroestructuras de "árbol"la definición de las condiciones iniciales es crucial.

Por un lado, se define un conjunto inicial de puntos de atracción con la forma de la copa del árbol que se pretende simular. Como se puede observar en la figura 7 este conjunto se compone de puntos de atracción repartidos arbitrariamente en un volumen de referencia - Aunque la figura al ser una imagen es bidimensional, se entiende que se trata de un cuerpo tridimensional- tratando de mantener uniforme la densidad de los puntos. En la misma figura podemos observar un punto negro bajo el volumen de atracción. Este representa el nodo inicial del que parte la simulación y sobre el que se comienza a aplicar el algoritmo. Cabe denotar su separación del conjunto inicial de puntos de atracción. Esta separación resultará en la formación de un tronco inicial. Es decir, los primeros nodos crecerán en dirección al volumen, formando una sucesión de varios nodos en forma de recta.

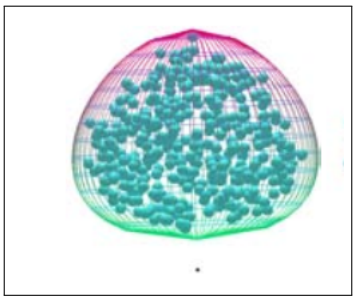
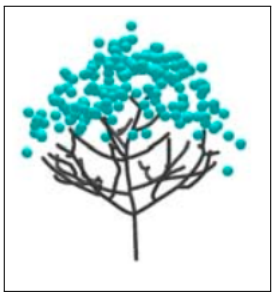


Figura 7: Conjunto inicial de puntos de atracción con forma de copa de árbol

Una vez los nodos alcancen el volumen de atracción los siguientes crecimientos formarán una estructura ramificada de manera que del mismo nodo crezcan en diferentes iteraciones nuevos nodos en direcciones variadas. En la figura 8, sección a, podemos ver como el primer nodo que alcanza el volumen de atracción crece en tres direcciones diferentes. Estos crecimientos se han dado en diferentes iteraciones y consiguen cubrir la parte inferior del volumen de atracción. Al seguir aplicando el algoritmo nacerán nuevas ramificaciones en aquellos puntos interiores al volumen cuyas terminaciones caen sobre los límites de este conjunto. Estas ramificaciones se pueden observar en la sección b de la figura 8.

(a) Primeros nodos de la simulación



(b) Estado intermedio de la simulación

Figura 8: Proceso de crecimiento para las condiciones iniciales de la figura 7

Aunque se defina la macroestructura mediante las condiciones iniciales y el conjunto inicial de nodos, este último también tiene un efecto reseñable en la microestructura. Tal y como se muestra en la figura 9, la densidad de los puntos de atracción - o el número de estos para un volumen fijo - manteniendo fijos el resto de parámetros influye directamente en la densidad de nodos y ramificaciones. Esto se debe a la necesidad de cubrir zonas más concretas del espacio necesitando un mayor número de nodos para lograrlo.

(a) Alta densidad

(b) Baja densidad

Figura 9: Influencia de la densidad del conjunto inicial de puntos de atracción

Por otro lado, los parámetros radio de influencia y radio de eliminación se disponen en el artículo de Reunions como múltiplos del parámetro de crecimiento, G. De esta manera se logra una consistencia en el crecimiento de las estructuras. Estas relaciones entre los parámetros son las que definen finalmente la forma de las ramificaciones, es decir, las características geométricas de la simulación.

Los resultados que se presentan con el uso de este método de simulación están relacionados fuertemente de manera visual con ejemplos de estructuras vegetales presentes en la

naturaleza. En la figura 10 se ejemplifica esta similitud con el caso de una encina y una simulación basada en la relación $r_a = 8G$, dejando el parámetro r_e modificable para un mayor ajuste a individuos concretos.

(a) Simulación con $r_a = 8G$ (b) Fotografía de una encina

Figura 10: Comparativa entre una simulación basada en SCA y una fotografía de una encina

Sin duda, el SCA se presenta como un método de simulación robusto para las estructuras de los árboles, alcanzando resultados visuales sorprendentes. A su vez, su carga computacional no es muy fuerte lo que permite la posibilidad de hacer simulaciones múltiples en tiempo real de un conjunto de árboles que compitan por un mismo volumen de atracción. Sin embargo, estas simulaciones dinámicas van más allá de los propósitos de este proyecto.

Figura 11: Simulación de un árbol con hojas. [ABP07]

2.3. Efemerópteros y SCA

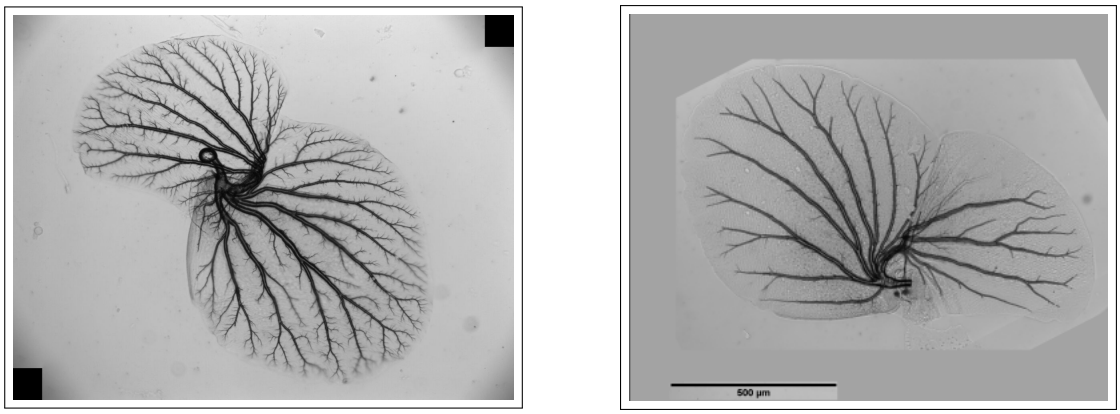
El éxito en la simulación de estructuras arbóreas del SCA permite suponer que este algoritmo será capaz de simular otras estructuras ramificadas similares. En este proyecto vamos a tratar el caso de la estructura vascular de las alas de una especie concreta de insecto efemeróptero. Por ello, es necesario profundizar antes en la morfología de esta estructura y sus causas naturales.

Figura 12: Imagen de un efemeróptero en su segundo estado volador

Los efemerópteros son un orden de insectos englobado en el grupo de los paleóptoros [AS00]. Las familias de insectos que componen este orden ascienden a unos dos mil tipos. Estos insectos en su estado larval, o de ninfa, son predominantemente acuáticos. Posteriormente, pasan por dos estados voladores, algo particular de este tipo de insecto. Es en el primer estado volador en el que se forma, a partir del abdomen, un entramado vascular. En este estado, la efímera - Nombre común de los efemerópteros - permanece sobre la superficie del agua. El entramado vascular mencionado crece a ras de la superficie acuosa preparando el organismo del insecto a su evolución al segundo estado volador, ya fuera del agua. Este entramado vascular será el que se desarrolle en forma de alas, formando una película entre las ramificaciones que termine por conformar una superficie de tejido.

Aunque hay diferentes teorías sobre el origen de las alas en los insectos la más aceptada actualmente se basa en que este entramado vascular es una extensión del sistema branquial de los insectos primitivos cuya funcionalidad residía en la termorregulación de los mismos. La estructura se extendía entonces sobre una superficie acuosa, encargándose de procesar el volumen de oxígeno del sistema, su temperatura y su movimiento - tal y como se discute en el artículo de Andrew Ross [AR17]. Esta teoría gana fuerza al ser coherente con la presencia de alas en insectos no voladores - discusión en más detalle en el artículo de Laurence Ruffieux [LJM98].

Con todo esto, se puede entonces asumir que el crecimiento de las alas de las efímeras comienza en el abdomen y se desarrolla en un entorno bidimensional - superficie acuosa - en pos de abarcar una superficie de la que obtener oxígeno. Las similitudes entre este proceso y el de crecimiento arbóreo refuerza la idea de que el SCA es una buena herramienta para la simulación y estudio de estos entramados vasculares.



(a) Fotografía de referencia

(b) Fotografía con escala métrica

Figura 13: Fotografías tomada con un microscopio de las alas de una efímera

Como se puede observar en la figura 13 los entramados tienen un centro claro y sus ramificaciones mantienen sus terminaciones en el contorno de la superficie que forman, dos características fundamentales de las simulaciones obtenidas con el SCA. Nuestro propósito en este proyecto es obtener simulaciones coherentes con estas imágenes aplicando el algoritmo de colonización espacial. Para ello, definiremos un área de influencia basada en las imágenes del microscopio y un nodo inicial situado en el centro - aproximado- que expone la imagen. En los siguientes apartados desarrollaremos en profundidad este proceso, considerando el efecto de las variaciones en los parámetros del sistema y tomando la medida de la dimensión fractal como parámetro de coincidencia entre las imágenes del microscopio y los resultados de la simulación.

3. Desarrollo práctico

Como hemos visto teóricamente, este proyecto tiene dos módulos de desarrollo: La simulación mediante el Algoritmo de Colonización Espacial del aparato vascular del insecto Efímera y el estudio de la influencia de los parámetros principales en la morfología del mismo.

Ambas ramas del proyecto, debido al alto número de datos y variables, se desarrollarán por métodos numéricos en un entorno de programación. La elección de este entorno debe tener en cuenta el manejo de un gran número de datos y microprocesos puesto que, aunque las operaciones matemáticas involucradas son de carácter sencillo, estas son muy numerosas. Por otro lado, los resultados visuales requieren de herramientas de dibujo más complejas que las requeridas para gráficas simples así como el tratamiento de las imágenes del microscopio. Con todo esto, el lenguaje de programación C# [CS20] y su entorno gráfico más utilizado, Unity [UNITY20], conformarán el entorno de programación y gráfico usado en este proyecto.

El lenguaje de programación C# no dista mucho de otros lenguajes como C, C+ y Python. Todos estos cuentan con un gran número de librerías adicionales y sus funciones básicas están altamente optimizadas para garantizar una buena velocidad de procesamiento. El funcionamiento básico del lenguaje se basa en la definición de clases y funciones, los dos componentes en los que se basarán todos los desarrollos hechos en este proyecto.

Las clases son conjuntos de información y métodos que se asocian a "Unidades" de un sistema. Entre las clases más básicas encontramos las constantes y los vectores. Sin embargo, estos lenguajes dan la posibilidad de definir clases propias con las características requeridas para esas unidades.

Por otro lado, las funciones son métodos o conjuntos de estos que reescriben datos almacenados en variables internas o ceden un resultado a partir de un input previo. Estas funciones pueden trabajar de manera global al sistema de scripts del proyecto o internamente a un script concreto donde residen.

3.1. Implementación del Algoritmo de Colonización Espacial

La implementación del Algoritmo de Colonización Espacial se va a separar en tres bloques. Por un lado, se definirán las clases y las variables que intervienen en el mismo definiendo así el sistema y las unidades involucradas. En segundo lugar, se definirán las funciones implicadas que estructuren así las reglas del comportamiento del sistema y las interacciones entre las unidades. Finalmente, el flujo de estas funciones acabará por especificar la dinámica de la simulación partiendo de unas condiciones iniciales y dando lugar a un sistema final del cual se obtendrán los resultados visuales y numéricos.

3.1.1. Clases y variables

Como se ha explicado en el apartado teórico, el SCA se vale de tres parámetros para definir las características de de la ramificación simulada. Las tres son números racionales constantes en la simulación y en esta se definen como floats. Estas son el radio de atracción, el radio de eliminación y el parámetro de crecimiento. Estas variables son globales, es decir, son accesibles por cualquier función del script pero son solo de lectura por lo que no son modificables por el script pero sí externamente. Junto a estás se va a requerir de dos variables adicionales con propósitos independientes:

- **Número de iteraciones máximo**: Debido a que el algoritmo plantea dinámicas emergentes no es posible desde un inicio asegurar que su funcionamiento será perfecto, por tanto, cabe la posibilidad de que en una simulación haya puntos de atracción lo suficientemente alejados del cluster de nodos como para ser inalcanzables y el algoritmo seguiría iterando hasta el infinito provocando un bucle interminable. Para esto, se define un número máximo de iteraciones que recorrerá el algoritmo antes de dar por finalizada la simulación.
- **Número inicial de puntos de atracción**: En la simulación, el conjunto inicial de puntos de atracción se define geométricamente, es decir, en función de una superficie por lo que no se estipula el número inicial de estos puntos. La implicación de esta variable será la distancia media que tendrán los puntos de atracción entre ellos, algo crucial en las magnitudes que se usarán en las variables principales.

Las clases propias que se van a utilizar en el desarrollo de este algoritmo son dos: los nodos y los puntos de atracción. Estas clases serán las unidades básicas del sistema y se guardarán en memoria a lo largo de la simulación en dos conjuntos listados independientes a los que accederán y recorrerán continuamente las funciones. Las clases tendrán un conjunto de variables en su interior que conformarán la información que defina la misma clase.

- **Nodo**: Unidad referida a los puntos que conforman la estructura de la ramificación.
  - X,Y: Números racionales (floats) que definen la posición del nodo en un espacio bidimensional.
  - Generación: Número entero (int) que define en que iteración de la simulación ha sido creado el nodo. Se usará posteriormente para definir el radio de las ramas en los resultados visuales.
  - Padre: Número entero (int) que define el índice en el conjunto listado de los nodos del nodo predecesor. Para el nodo inicial con índice en la lista 0, este parámetro de establece como -1. Esta variable permitirá en la obtención de resultados visales definir las ramas entre los nodos padre e hijo.
  - Obsoleto: Variable booleana que se establece como verdadera cuando en una iteración el nodo no puede crecer, dando por supuesto que este ya no podrá volver a crecer. La importancia de esta variable reside en la reducción de costes de procesamiento al no considerar en los cálculos globales de crecimiento estos nodos.
  - Lista de puntos atractivos: Lista de individuos de clase "Punto de atracción" que se vaciará y se llenará en las comprobaciones de crecimiento. Esta variable reduce costes de procesamiento a cambio de costes de memoria puesto que las diferentes funciones del proceso de crecimiento sobre el nodo solo recorrerán esta lista definida al comienzo de la iteración en vez de la lista global de puntos de atracción.
- **Punto de atracción**: Unidad referida a los puntos en el espacio que ejercerán influencia en el crecimiento de la ramificación, es decir, en la creación de nuevos nodos.
  - X,Y: Números racionales (floats) que definen la posición del punto de atracción en un espacio bidimensional.

Con estas dos clases definidas, se declaran las dos variables dinámicas del sistema. Estas serán globales además de accesibles y modificables por cualquier función dentro del script. Se trata de la Lista de Nodos y la Lista de Puntos de atracción. A la primera solo se añadirán elementos y se leerán las características de estos elementos (Nodos). A la segunda se añadirán en un inicio todos los elementos y se irán modificando o eliminando a lo largo de la simulación.

3.1.2. Funciones

En este proyecto se establecen dos tipos de funciones: Aquellas que modifican las variables globales y aquellas que requieren de un conjunto de variables de entrada (inputs) para ceder unas variables de salida (outputs). Debido al papel que tienen las diferentes funciones podemos categorizarlas en cuatro bloques: Funciones de SCA, Funciones de creación, Funciones de cálculo y Funciones extra. A continuación se listarán y definirán todas las funciones involucradas:

- **Funciones de cálculo**: Conjunto de funciones destinada a realizar cálculos matemáticos que involucran a las variables globales o no existen dentro de las librerías utilizadas:
  - CalcularRandomFloat(): Calcula un número racional dentro de los límites geométricos establecidos de la simulación. Se llamará por otras funciones para obtener posiciones aleatorias.
  - CalcularDistancia(): Esta función tiene dos vectores bidimensionales como parámetro de entrada. Calcula el módulo de la distancia entre ambos y la cede como parámetro de salida.
- **Funciones de creación**: Conjunto de funciones encargadas de crear los individuos del sistema y añadirlos a las listas globales de control.
  - CrearNodo(): Función cuyos parámetros de entrada son: Posición(x,y), Generación, Padre y Obsoleto (Variables de definición de la clase Nodo). Crea una estructura de clase nodo y define las variables consecuentes. Finalmente lo añade a la lista global de Nodos y calcula la distancia de este nodo a cada punto de atracción en la lista global. Si la distancia entre el nodo y uno de estos puntos de atracción es menor al radio de eliminación, el punto de atracción es extraído de la lista global.
  - CrearPuntoDeAtracción(): Función que tiene una posición (x,y) como parámetro de entrada. Define una estructura de clase Punto de Atracción y su posición consecuente al parámetro de entrada. Finalmente añade este punto a la lista global de Puntos de Atracción.
  - CrearCampoDePuntosDeAtracción(): Esta función toma la geometría de la superficie de puntos de atracción iniciales para llamar a *CrearPuntoDeAtracción()* con posiciones aleatorias dentro de la superficie definida un número de veces estipulado inicialmente en la variable *Número inicial de puntos de atracción.*
- **Funciones de SCA**: Conjunto de funciones encargadas del crecimiento de la ramificación que leen las listas globales y llaman a las funciones de creación para modificarlas.

- AsociarPuntosDeAtracciónANodos(): En primer lugar, se recorre la lista global de puntos de atracción. Para cada punto de atracción se calcula la distancia a cada nodo de la lista global de nodos de manera que a aquel a menor distancia (si esta es menor que el radio de atracción) se le añade a su lista de puntos de atracción (variable interna de la clase nodo) el punto de atracción comentado. Cuando la función termina, todos los nodos tienen su lista interna de puntos de atracción rellena, es decir, se ha asociado cada punto de atracción a su nodo más cercano dentro del radio de atracción.
- CrecerNodo(): Esta función tiene como parámetros el índice en la lista global de nodos de un nodo concreto y la iteración de crecimiento en la que se encuentra. Calcula a partir de la lista interna de puntos de atracción asociada a este nodo y de las variables de crecimiento la posición de un nuevo nodo. Después, llama a la función "CrearNodo()" pasándole como padre el índice del nodo aceptado como parámetro de entrada, como posición la posición calculada, como generación el siguiente entero de la generación del creador y como obsoleto un bool declarado a falso. Finalmente, declara la lista interna del nodo creador de puntos de atracción como una nueva lista vacía.
- CrecerRamificaciones(): En esta función se llama inicialmente a las funciones extra que se apliquen a la simulación. Después se llama a "AsociarPuntosDeAtracciónANodos()". Finalmente se recorre en orden creciente la lista global de nodos llamando en cada iteración a "CrecerNodo()" con el índice de la iteración en que se encuentra.

- **Funciones extra**: Este conjunto de funciones se encarga de modificar en cada iteración de crecimiento la disposición de los puntos de atracción para simular condiciones dinámicas reales en estos.

  - CrearRuidoEnElCampoDePuntosDeAtracción(): Esta función toma para cada elemento de la lista global de puntos de atracción la posición asociada y la varía aleatoriamente en un pequeño rango para crear una simulación de movimiento browniano en los puntos.

3.1.3. Flujo

El flujo del algoritmo de simulación de colonización espacial será el siguiente:

1. CrearCampoDePuntosDeAtracción() :Cálculo de la superficie inicial de los puntos de atracción y creación del campo de puntos de atracción.
2. CrearNodo(): Creación del primer nodo en el centro estipulado externamente.
3. CrecerRamificaciones().
4. Si la lista global de puntos de atracción no está vacía ni se ha alcanzado el número máximo de iteraciones Se vuelve al paso 3. Si no, se pasa al paso 5.
5. Fin de la simulación.

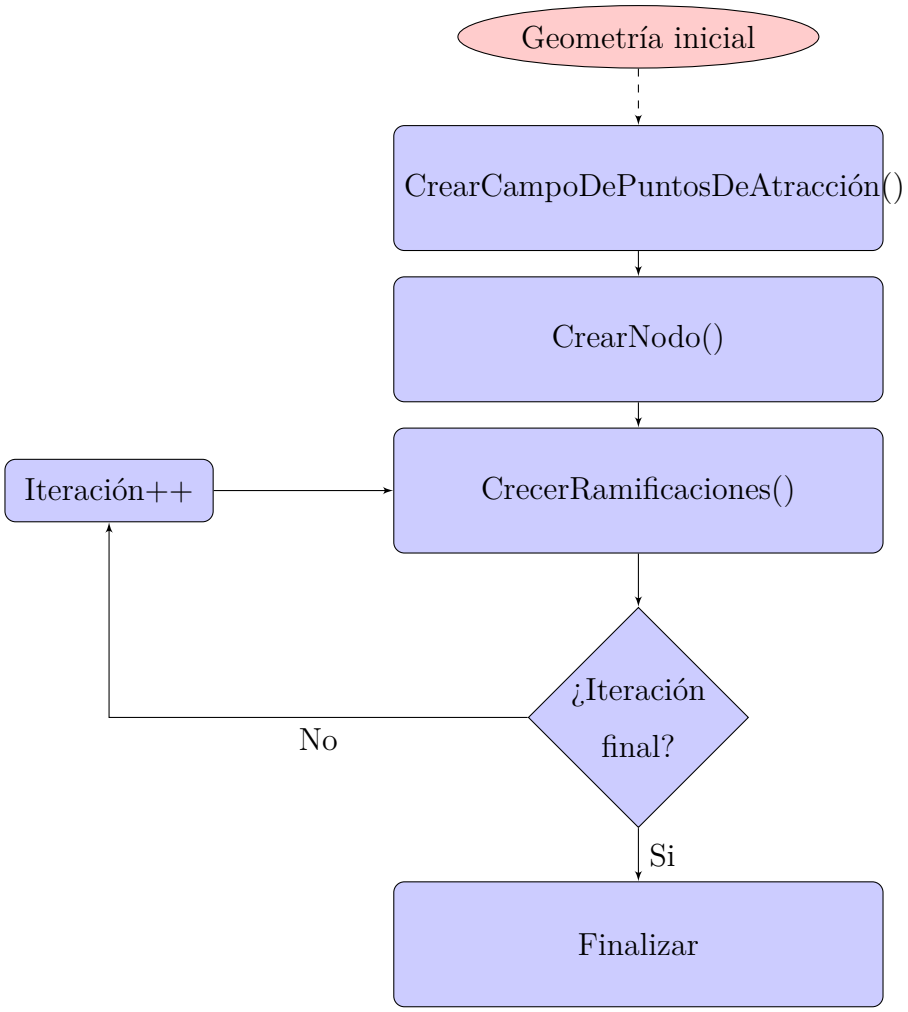
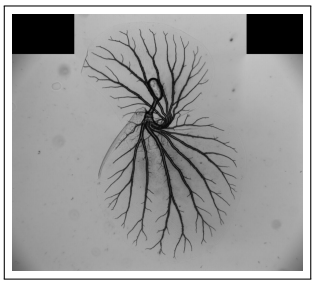


Figura 14: Flujo del Algoritmo de Simulación de Colonización Espacial

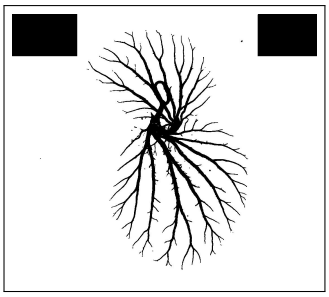
3.2. Tratamiento de las imágenes del insecto efímera

Como se ha mencionado anteriormente, este proyecto pretende aplicar el SCA sobre el aparato vascular de los insectos Efimeraes y comparar una variable numérica, la dimensión fractal, entre la simulación y la realidad. Para ello se usarán como referencia imágenes de las alas de estos insectos al microscopio. Por un lado, se obtendrá una superficie de puntos de atracción con la forma de estas alas como referencia para aplicar el SCA y por otro se analizará la dimensión fractal de la imagen 15 a) para, en último lugar, compararla con la obtenida mediante la simulación.

En primer lugar, se tratará esta imagen hasta obtener un mapa de píxeles blancos y negros, eliminando los gradientes grises y ensanchando las ramificaciones. Para ello se recorrerá la imagen píxel por píxel truncando los valores de estos píxeles a blanco y negro en función de una constante divisoria. El resultado 15 b) es una imagen clara de la ramificación con valores binarios. Es decir, se puede tratar como un mapa booleano.



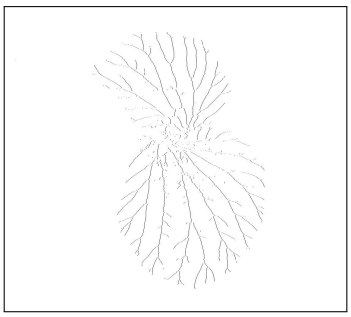
(a) Imagen original de referencia



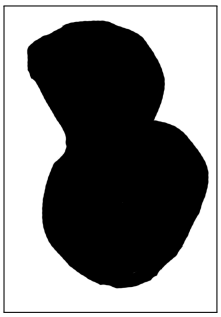
(b) Mapa de píxeles.

Figura 15: Tratamiento de imagen del microscopio a mapa lógico de píxeles

En segundo lugar, se limpiarán manualmente las imperfecciones de la imagen - manchas de la imagen original -. Con esto, se volverá a recorrer la imagen de izquierda a derecha, píxel por píxel, siguiendo la siguiente regla: Al encontrar un píxel negro se guarda en memoria y se sigue recorriendo hasta encontrar un píxel blanco. Cuando este se encuentra, todos los píxeles guardados se pintan de blanco menos el píxel que esté en el centro que se pinta de negro. Como resultado, figura 16 a), de este tratamiento, se tiene un nuevo mapa en el que cada píxel negro corresponde en este caso a un nodo de la ramificación. Así, se ha eliminado el grosor de las ramificaciones ya que esta variable no es necesaria para el análisis de la dimensión fractal de la ramificación.



(a) Mapa de nodos de la ramificación de la imagen de referencia



(b) Campo de puntos de atracción para inicializar la simulación.

Figura 16: Resultados finales del tratamiento de imágenes: Mapa de nodos dispuesto para el análisis de la dimensión fractal y campo de puntos de atracción inicial de la simulación

Esta última imagen permitirá convertir los datos experimentales en una lista de valores de posición en un entorno bidimensional. Para ello se guardan las posiciones de los píxeles negros en una lista de Vectores bidimensionales.

Por otro lado, a partir de la imagen 16 a), se dibuja una superficie negra sobre el área que ocupa la ramificación: figura 16 b). Para crear la lista de puntos de atracción que de inicio a la simulación, se crean puntos aleatorios dentro de la superficie descrita por la imagen. De esta manera obtenemos un campo de puntos de atracción inicial referenciados directamente a la imagen del microscopio original.

Una vez se tiene preparada una superficie de puntos de atracción, es necesario buscar el centro del cual partirá la simulación. Este centro se toma en referencia a la imagen

original y de manera visual. Con un punto aproximado como centro, los dos conjuntos de datos iniciales necesarios para lanzar la simulación están preparados. El resultado de esta simulación son un conjunto de nodos, es decir, entidades con información de posición en el espacio bidimensional, referenciadas al nodo desde el cual han crecido y con información temporal sobre el estadio de la simulación en la que se crearon.

Para recrear una rama entre dos nodos se usa un trapecio cuyas bases tienen de longitudes l_n y l_{n+1} siendo n la generación del nodo más viejo. De esta manera, las ramas de generación más temprana tendrán un ancho mayor que las más tardías. La decadencia del ancho con la generación influye en la representación final de la figura aunque no es relevante en la forma de la ramificación. En las siguientes imágenes se ha escogido un ancho inicial y dividido este por la generación del nodo para el cálculo de la longitud de las bases.

Usando de referencia la imagen del microscopio presentada en el apartado anterior, el resultado de la simulación recreado en el sistema de ramas se observa en la figura 17

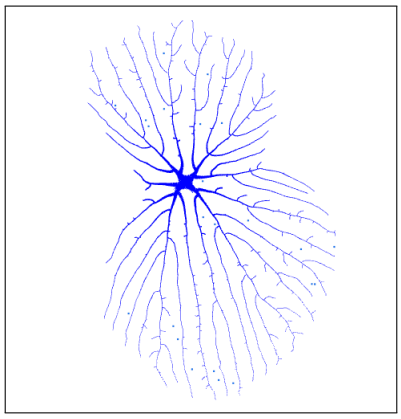


Figura 17: Simulación mediante el algoritmo de colonización espacial sobre una superficie de puntos de atracción obtenida a partir de una imagen al microscopio del insecto efímera. Parámetros: Radio de eliminación: 1'1; Radio de atracción: 4; Parámetro de crecimiento: 0'2

Este resultado permitirá la comparación visual entre las imágenes del microscopio y las simulaciones referenciadas a esta. Así, se podrá ver los efectos de los parámetros básicos del SCA sobre las ramificaciones y extrapolar los parámetros que rigen a las imágenes originales.

Por otro lado, a partir de los conjuntos de nodos - tanto de la imagen real como de la simulación- se hará un cálculo de su dimensión fractal siguiendo el método de Hausdorff [FS11]. Para ello, se centrará el conjunto de nodos al cero, es decir, se cambiará su posición para que el centro del conjunto coincida con el punto cero del espacio bidimensional de trabajo. Con esto, se tomará un círculo con un radio inicial y se observará el conjunto de puntos que caen dentro de este círculo. Estas medidas se repetirán con radios crecientes hasta el radio que englobe el total del conjunto de nodos. Como resultado se tendrá un conjunto de datos de la siguiente forma: $N(r)$ y r.

Se define entonces la dimensión fractal de Hausdorff, D, como:

$$N = r^D \rightarrow D = \frac{ln(N(r))}{ln(r)} \tag{2}$$

Para obtenerla a partir del conjunto de puntos $(N(r), r)$, se calculan el conjunto $(ln(N(r)), ln(r))$ y se toma una regresión lineal de estos. Como resultado de esta regresión se tiene la dimensión fractal de Hausdorff como la pendiente de la misma. En la figura 18 podemos observar la regresión lineal aplicada al conjunto de datos relativo a la figura 17.

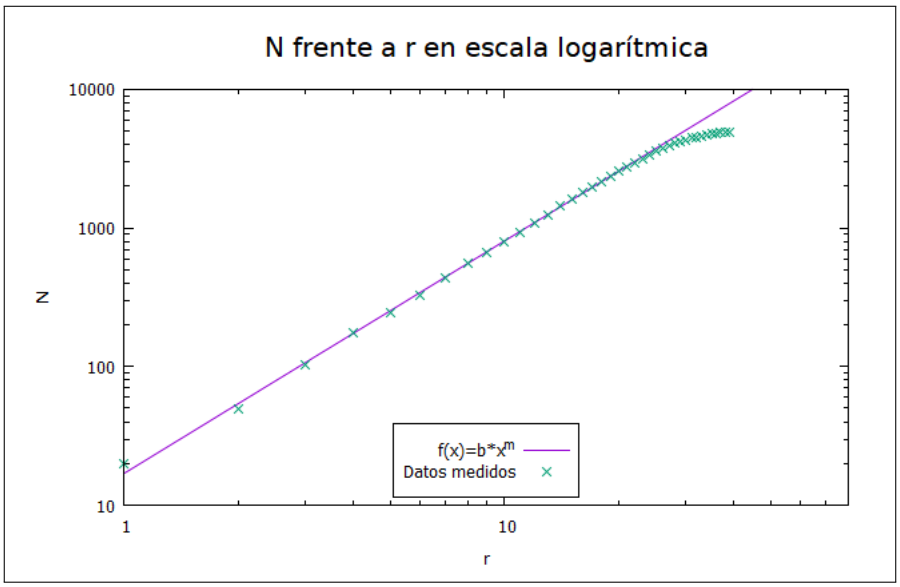


Figura 18: Número de nodos frente al radio en escala logarítmica correspondiente a una tendencia lineal con pendiente de 1'677 siendo esta la dimensión fractal analizada dentro de los límites esperados

Entonces, se puede determinar que la dimensión fractal asociada a la figura 17 es de $1'677$, lo cual entra en los límites esperados - Entre 1 y 2 debido al carácter bidimensional de la simulación - y su valor, relativamente alto, se puede asociar a la fuerte ramificación de la figura de ejemplo.

No se ha planteado un estudio de la dimensión fractal a partir del método del contado de cajas ya que estas figuras no se forman lógicamente por segmentos si no por puntos. Esto hace que el método del contado de cajas no pueda formular resultados coherentes, cayendo en valores menores que uno.

4. Resultados

Usando el método detallado en los apartados anteriores, se puede obtener una simulación a partir de una imagen al microscopio de las alas del insecto efímera. Sin embargo, el parecido en las ramificaciones ha de encontrarse a partir del afinamiento de los parámetros fundamentales de la simulación, estos son: el radio de eliminación, el radio de atracción y el parámetro de crecimiento. En un primer acercamiento, se han utilizado un conjunto estático de parámetros fundamentales: 1'1, 4 y 0'2 para el radio de eliminación, el radio de atracción y el parámetro de crecimiento respectivamente. Con estos parámetros se han simulado cuatro casos a partir de cuatro imágenes diferentes, la comparativa se puede observar en la figura 19.

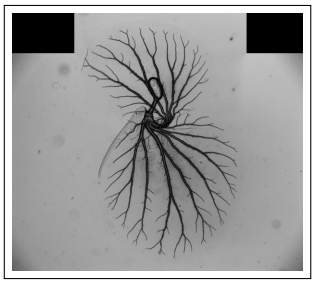
Analizando estos cuatro casos, cabe destacar la fuerte similitud en el par M-4. Las ramas principales son coincidentes y las pequeñas ramificaciones son muy similares en ambos casos. Aunque esta comparativa es puramente visual, se ha analizado la dimensión fractal de Hausdorff sobre la imagen y sobre la simulación. Los resultados son, respectivamente, $1'408$ y $1'466$. Es decir, la similitud visual es respaldada en este caso por la coincidencia en la dimensión fractal con una separación entre los valores de $0'058$. Se podría afirmar en este caso que la simulación es funcional y se asemeja fuertemente a lo ocurrido en la realidad.

Por otro lado, en el caso del par M-2 obtenemos las siguientes dimensiones fractales: $1'368$ y $1'520$ para el caso real y para la simulación respectivamente. Esto nos indica que la ramificación en la simulación es más compacta, es decir, hay un mayor número de ramificaciones en la simulación que en la imagen. Esto se corrobora con la comparativa visual: Se observa un menor número de ramificaciones en el caso real que en la simulación. De nuevo, la dimensión fractal corrobora los resultados visuales que se han encontrado con la simulación.

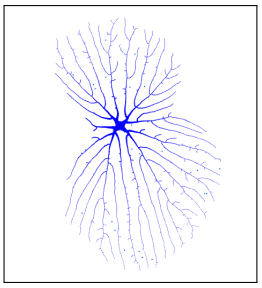
La dimensión fractal de Hausdorff hace un análisis radial de la figura fractal. Es decir, contempla una simetría radial. En nuestros casos, las figuras no tienen esta simetría, ocupando diferentes cuadrantes de manera exclusiva. Esto concluye en que esta medida no es del todo rigurosa pero si permite contrastar los resultados con una métrica numérica aproximada.

En los siguientes apartados se va a analizar el efecto de los parámetros fundamentales de la simulación en los resultados obtenidos. Para ello usaremos la figura M-4 como referencia constante, de manera que los datos resultantes tengan una correlación para después poder compararlos. Se ha elegido esta imagen dado que es la que tiene un carácter más radial, con su centro aproximadamente en el centro de densidad de la figura. Para el análisis del radio de eliminación y el radio de atracción, se mantendrá el parámetro de crecimiento en $0'2$ y se tomarán valores de los parámetros como múltiplos de este. Además, todos los conjuntos de parámetros deberán mantener una relación estricta: El radio de eliminación ha de ser siempre menor que el radio de atracción. Si esto no fuera así, la simulación terminaría en la primera iteración al eliminar todos los puntos de atracción dentro del radio de atracción del nodo.

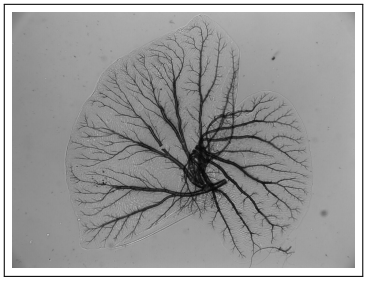
Cabe mencionar que los parámetros fundamentales son magnitudes de longitud relativas al tamaño de la imagen, por lo que añadirles una unidad sería irrelevante.



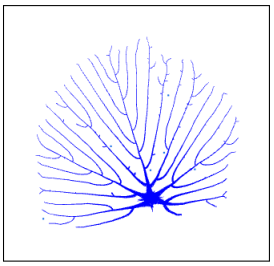
(a) M-1 Imagen real



(b) M-1 Simulación

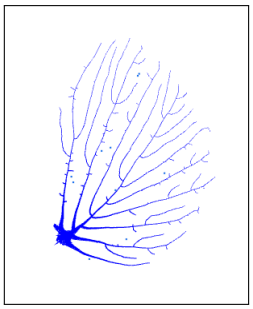


(c) M-2 Imagen real

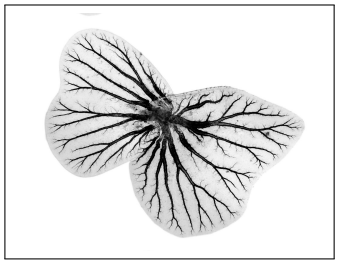


(d) M-2 Simulación

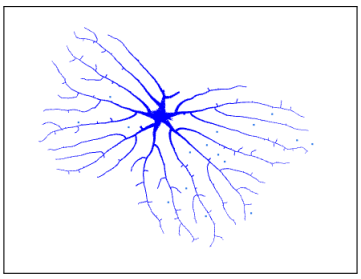
(e) M-3 Imagen real



(f) M-3 Simulación



(g) M-4 Imagen real



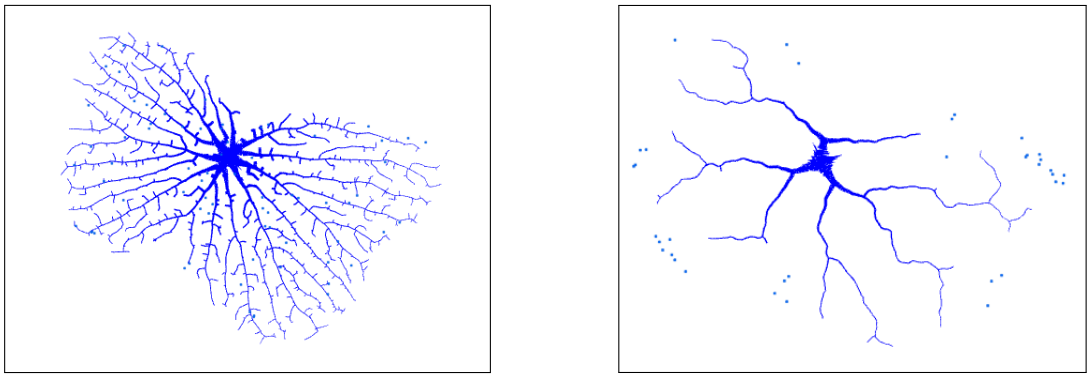
(h) M-4 Simulación

Figura 19: Comparativa entre imágenes del microscopio y simulaciones mediante el SCA. Parámetros: Radio de eliminación: 1'1; Radio de atracción: 4; Parámetro de crecimiento: 0'2. Pone de manifiesto la fidelidad de las simulaciones del SCA con las imágenes experimentales

4.1. Influencia del radio de eliminación (r_e)

Al crecer un nuevo nodo, se observan que puntos de atracción están a una distancia menor o igual al radio de eliminación y estos son eliminados. Por tanto, el radio de eliminación es un parámetro que define cuanto tienen que crecer las ramificaciones para eliminar los puntos de atracción. Para valores pequeños de este parámetro, las ramas principales tendrán que ramificarse mucho para alcanzar todos los puntos de atracción cercanos. Para los valores más grandes, habrá un menor número de ramificaciones puesto que las ramas alcanzarán un mayor número de puntos de atracción a eliminar.

Los resultados que se han obtenido para este apartado se referencian a la imagen M-4 del microscopio. Se mantiene un parámetro de crecimiento $G = 0'2$ y un radio de atracción $r_a = 4'0 = 20G$. Respetando las condiciones entre los parámetros, el radio de eliminación quedará dentro de los límites $r_e \in [G, 19G]$



(a) Radio de eliminación de G

(b) Radio de eliminación de 17G

Figura 20: Comparativa de la simulación con valores extremos del radio de eliminación. Mayor número de ramificaciones para valores pequeños y menor para valores grandes.

Como se puede observar en la comparativa de la figura 20, para el valor extremo mínimo se observa una ramificación abundante, necesaria para alcanzar todos los puntos de atracción. En el caso del valor más grande no se ha tomado el extremo que coincidiría con $19G$ puesto que en este caso - al estar muy próximo el radio de eliminación y el de atracción - el algoritmo no encuentra puntos de influencia cercanos. Este fenómeno se debe a que los puntos de atracción tienen una disposición aleatoria respetando la densidad. Si se eliminan todos los puntos cercanos al nuevo nodo, es posible que al buscar después nuevos puntos de atracción, estos no existan en la cercanía del radio de atracción como ya se observa en la figura 20 b. En este caso de valor máximo, la ramificación es muy poco abundante y con unas pocas ramas principales se cubre el campo de puntos de atracción. Estos dos resultados corresponden a unas dimensiones fractales de $1'599$ y $1'351$ para el valor menor y el mayor respectivamente. Es decir, se refleja un entramado más compacto y autocontenido en el caso de $r_e = G$ y uno menos compacto y menos autocontenido en el caso $r_e = 17G$.

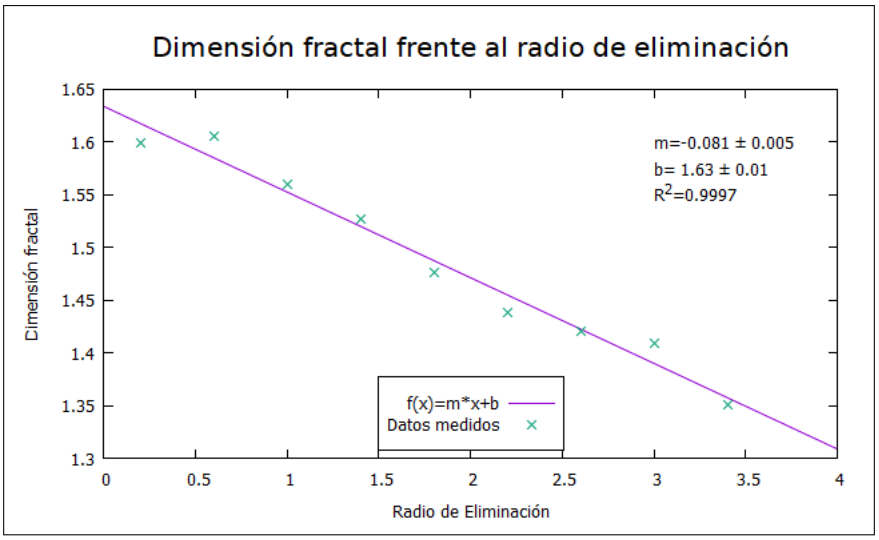


Figura 21: Relación entre el radio de eliminación y la dimensión fractal en la simulación mediante SCA de la imagen M-4. Ajuste a una tendencia lineal decreciente con pendiente $-0'081$

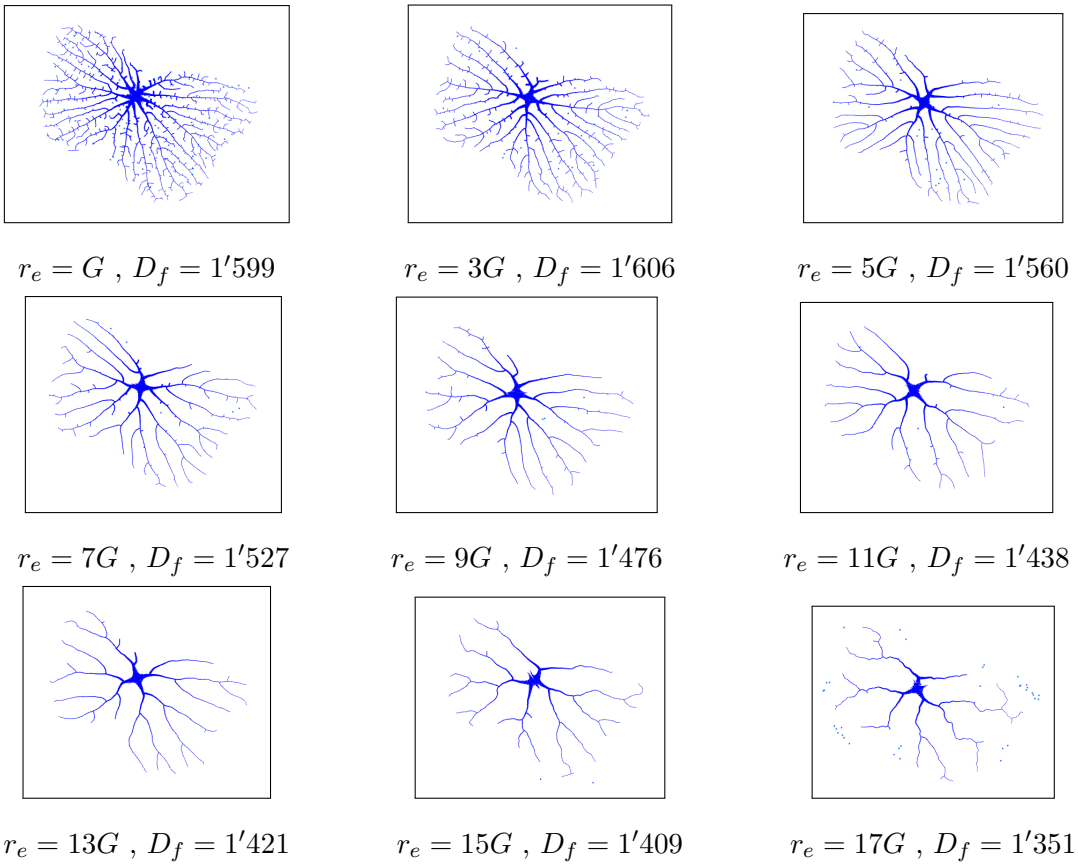


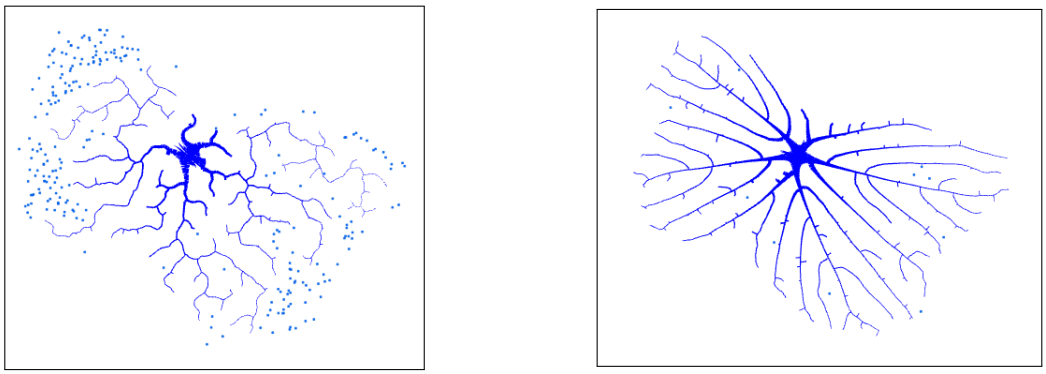
Figura 22: Influencia del parámetro r_e en la simulación de SCA para la imagen M-4. Disminución del número de ramificaciones y la dimensión fractal con el aumento del valor del radio de eliminación

Podemos entonces afirmar que existe una relación entre la dimensión fractal resultante de la simulación y el parámetro r_e. En la figura 21 se establece que esta relación entre la dimensión fractal y el parámetro r_e es lineal y muestra un decrecimiento de la dimensión fractal con el aumento del radio de eliminación. Cabe destacar además que el hecho de necesitar un menor número de ramificaciones induce a la necesidad de un menor número de nodos en las estructura, siendo este otro parámetro que depende del radio de eliminación aunque en menor medida.

4.2. Influencia del radio de atracción (r_a)

El radio de atracción, r_a, es el parámetro que establece a que distancia los puntos de atracción se consideran influyentes al nodo. Es decir, si este parámetro tiene valores muy grandes, se verá influenciado por un gran número de puntos de atracción. Como estos se disponen aleatoriamente respetando una densidad, si el parámetro es muy grande, concluirá que la dirección de crecimiento se tomará hacia la zona con mayor número de puntos de atracción, ya estén lejos o cerca. Esto implica que las ramas se extenderán rápidamente hacia las zonas externas no pobladas de la superficie. Como resultado, cabe esperar una forma en las ramas más recta. Por otro lado, para valores pequeños del parámetro, lo puntos de atracción influyentes serán los más cercanos al nodo creciente. Esto implica que habrá un mayor número de posibilidades de que la rama se desvíe de una trayectoria recta influenciada por los puntos de atracción residuales cercanos.

Los resultados obtenidos para este apartado se referencian a la imagen M-4 del microscopio. Se mantiene un parámetro de crecimiento $G = 0'2$ y un radio de eliminación $r_e = 1'0 = 5G$. Teniendo en cuenta que el radio de atracción debe de ser mayor que el radio de eliminación, este primero debe encontrarse en los límites $r_a \in [5G, \infty)$



(a) Radio de atracción de 9G

(b) Radio de atracción de 100G

Figura 23: Comparativa de la simulación con valores extremos del radio de atracción. Mayor presencia de desviaciones en las ramas para los valores pequeños mientras que para valores grandes las ramas tienden a seguir una trayectoria rectilínea

De nuevo, para el par de valores r_e y r_a muy próximos entre sí, aparece el fenómeno en el cual la simulación no alcanza nuevos crecimientos puesto que los puntos de atracción en las inmediaciones de los nuevos nodos han sido eliminados y estos no se ven influidos

por otros. Esto se puede observar en la figura 23 a.

Observando la comparativa 23 vemos corroborado lo esperado sobre la forma de las ramas. En la parte (a), con un valor muy pequeño del radio de atracción, las ramas se doblan sobre si mismas, influenciadas por los puntos de atracción cercanos. En la parte (b), con un valor en el extremo superior, las ramas son rectas y apuntan directamente a los límites de la superficie.

En la figura 24 podemos observar la influencia del parámetro r_a sobre la forma de la estructura ramificada de la simulación. Sin embargo, en todos los casos, la dimensión fractal asociada oscila entre los valores $1'5$ y $1'6$ sin seguir ninguna tendencia aparente. Esto indica que el parámetro r_a no tiene una influencia directa sobre la dimensión fractal ni sobre el número de nodos resultante, aunque si una fuerte influencia sobre la forma que toman las ramas.

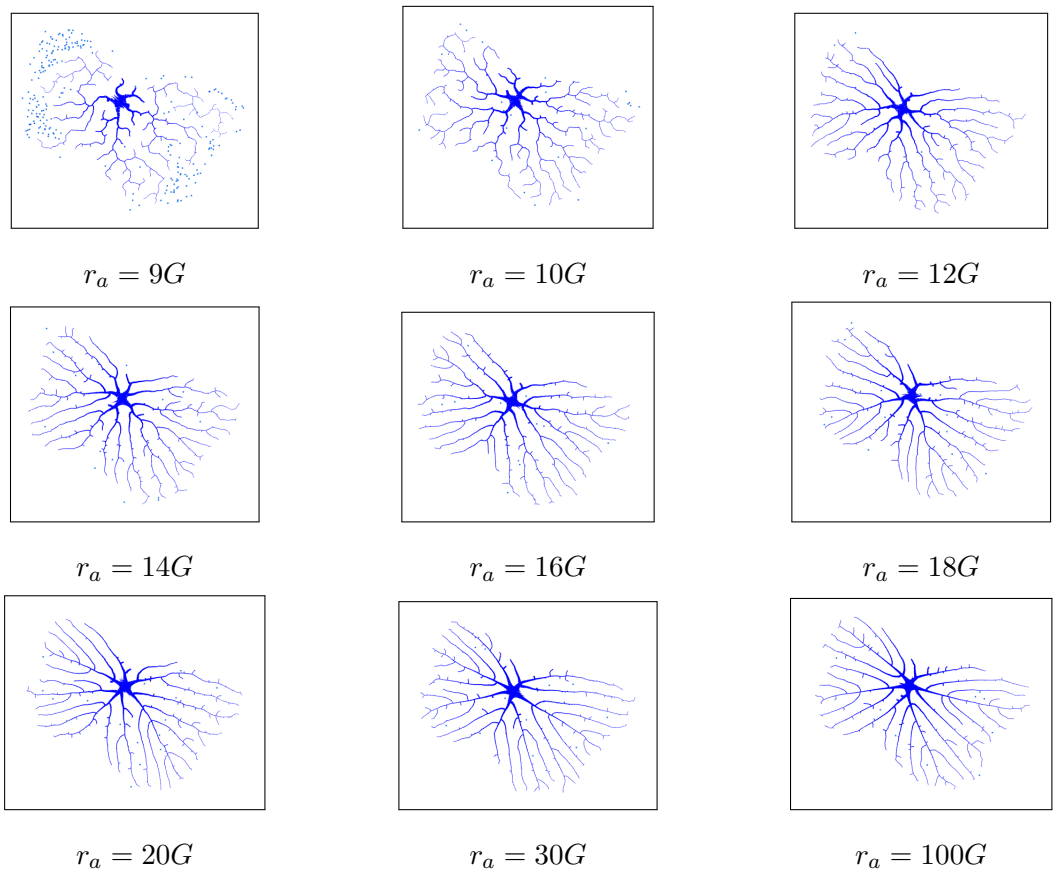


Figura 24: Influencia del parámetro r_a en la simulación de SCA para la imagen M-4. Tendencia a trayectorias rectilíneas de las ramas con el aumento del radio de atracción

4.3. Influencia del parámetro de crecimiento (G)

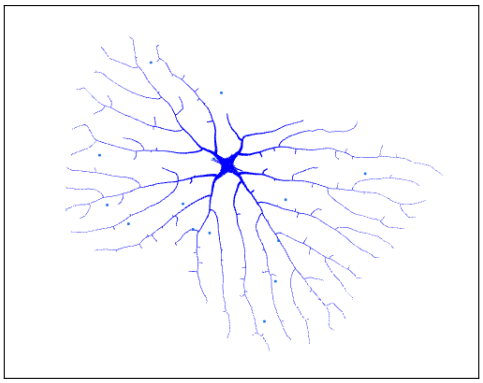
Como se ha visto, para iniciar la simulación, se parte de una imagen al microscopio. Esta tiene una resolución en píxeles fija. Tras su tratamiento, concluimos en un conjunto de puntos en un espacio bidimensional cuya unidad espacial define la distancia que se le adjudica a estos píxeles. En conclusión, la nueva superficie tendrá un tamaño referenciado a la imagen original mediante esta unidad espacial.

Por su parte, el parámetro de crecimiento define la distancia a la cual se posicionará un nuevo nodo creado respecto al nodo creador. Esta distancia se toma respecto a la unidad espacial que define la superficie. Por tanto, su valor tendrá efectos distintos en función del tratamiento que se haga sobre la imagen. Como todas las imágenes han sufrido la misma conversión de píxeles a posiciones en el espacio bidimensional, podemos asumir que el efecto de un mismo valor del parámetro de crecimiento afectará por igual a todos los casos. Estas consideraciones no eran necesarias al discutir sobre los otros dos parámetros pues se toman en referencia al parámetro de crecimiento.

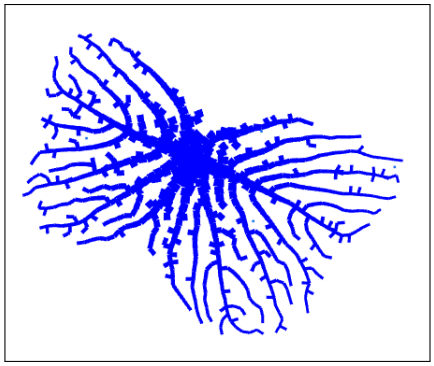
Al aumentar el parámetro de crecimiento, los nodos se crearán más separados entre sí, cubriendo una mayor superficie del campo de puntos de atracción. Es de esperar entonces que el aumento de este parámetro concluya en un número menor de nodos resultantes de la simulación. Sin embargo, puesto que el campo de puntos de atracción tiene una densidad relativamente constante, no cabe esperar un cambio en la dimensión fractal de los distintos resultados con el parámetro de crecimiento.

Los resultados de este apartado se han obtenido manteniendo fijos los parámetros $r_a = 4'0$ y $r_e = 1'0$. En este caso, el parámetro de crecimiento deberá de ser menor siempre que el radio de eliminación y mayor que cero, quedando dentro de los límites $G \in (0, r_e = 1'0]$. Estos parámetros se mantienen fijos puesto que si se mantuvieran proporcionales al parámetro de crecimiento, los resultados tendrían las mismas relaciones entre parámetros variando exclusivamente el "tamaño" de la imagen, es decir, resultando en una estructura más grande, el mismo efecto que utilizar una superficie más pequeña.

En la figura 25 podemos observar una comparativa entre los valores extremales del parámetro de crecimiento. El bajo grosor de las ramas en la parte (a) indica que hay un mayor número de generaciones - a mayor generación, menor grosor en las ramas - indicando a su vez que hay un mayor número de nodos puesto que se han creado más. En la parte (b) se observa un grosor mucho mayor en las ramas, indicando que se han creado menos generaciones de nodos. Esto afecta a la forma de las ramas, aunque sus direcciones de atracción al crecer son las mismas, alcanzan zonas más lejanas con cada crecimiento, resultando en ramificaciones más toscas y directas para el caso del parámetro de crecimiento mayor. Sin embargo, es observable que ambos casos presentan el mismo carácter compacto.



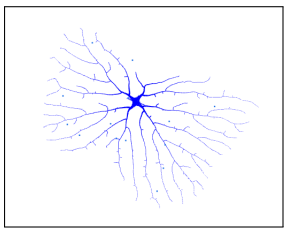
(a) Parámetro de crecimiento de 0'1



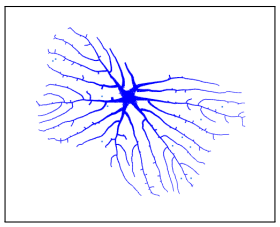
(b) Parámetro de crecimiento de 1'0

Figura 25: Comparativa de la simulación con valores extremos del parámetro de crecimiento. Menor número de nodos para los valores altos del parámetro de crecimiento y mayor para los valores bajos

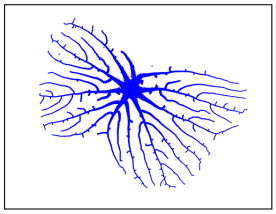
Existe una tendencia de decrecimiento del número de nodos de la simulación con el parámetro de crecimiento. Esta tendencia corresponde a una relación potencial entre el parámetro de crecimiento y el número de nodos de la estructura. Sin embargo, los valores de la dimensión fractal no presentan ninguna tendencia y oscilan entre los valores $1'5$ y $1'6$ al igual que ocurría con la variación del radio de atracción. Estas oscilaciones hacen referencia a que cada simulación, debido a los comportamientos aleatorios de disposición de los puntos de atracción, es siempre diferente cada vez que se realiza. Además, la dimensión fractal no da un rigor métrico puesto que tratamos con fractales naturales, que, aunque se asemejan a los fractales, no terminan de serlo matemáticamente.



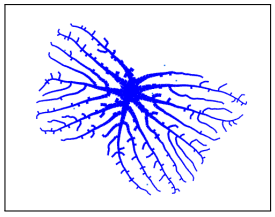
$G = 0'1$



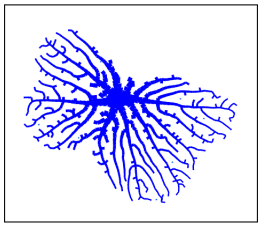
$G = 0'3$



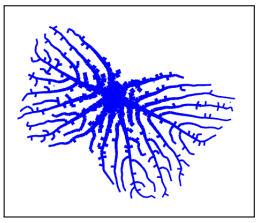
$G = 0'5$



$G = 0'7$



$G = 0'9$



$G = 1'0$

Figura 26: Influencia del parámetro G en la simulación de SCA para la imagen M-4.Disminución del número de nodos con el aumento del parámetro de crecimiento.

5. Conclusiones

En este proyecto se ha puesto de manifiesto la utilidad del algoritmo de colonización espacial para el estudio de estructuras fractales orgánicas. Estructuras con características no lineales y que presentan un reto hoy día. Se ha tomado como centro del proyecto el sistema vascular del insecto efímera aunque se ha demostrado la aplicabilidad previa que tiene sobre las estructuras vegetales. Aunque las medidas numéricas de la dimensión fractal están sujetas a las fuertes imperfecciones de las estructuras y de las propias imágenes de referencia, los resultados se han presentado coherentes y conexos entre las simulaciones y los análisis. Las conclusiones de este proyecto y su realización son las siguientes:

1. Los códigos los hemos hecho sobre los lenguajes Python y C#. Cada uno se ha utilizado para sacar el máximo rendimiento que ofrece. Por un lado, el tratamiento de imágenes en Python se ha valido de la fuerte optimización de su entorno para los cálculos algebraicos básicos y las iteraciones masivas. Por otro, C# y su desarrollado sistema de clases, funciones estáticas y corrutinas han permitido que el algoritmo (Tratando órdenes de 10^7 operaciones en cada iteración) funcione de manera fluida. En último lugar, el entorno gráfico de Unity basado en el lenguaje C# se ha utilizado para las representaciones finales de las simulaciones, haciendo que estas sean visualmente definibles y entendibles.

2. Hemos tratado las imágenes del sistema vascular del insecto efímera para obtener mediante la definición de Hausdorff su dimensión fractal. Este tratamiento se ha hecho desde cero usando el mapa de píxeles de las propias imágenes.

3. Hemos realizado más de 10 simulaciones para cada una de las cinco imágenes de referencia escogidas. Estas han presentado un alto grado similitud con las imágenes originales y sus medidas de la dimensión fractal son coherentes entre sí.

4. En la discusión de los parámetros fundamentales del SCA hemos observado la influencia de los parámetros sobre la morfología resultante de la simulación:

   a) El radio de atracción se presenta como un parámetro que define la curvatura de las ramas, sin presentar efectos sobre el número de nodos ni sobre la dimensión fractal resultante.

   b) El radio de eliminación presenta una relación lineal con la dimensión fractal resultante, sin efecto sobre la forma de las ramas pero afectando en cierta medida sobre el número final de nodos.

   c) El parámetro de crecimiento se ha presentado como un modificador del número de nodos, teniendo una relación de decrecimiento potencial y no afectando sobre la dimensión fractal o la forma de las ramas finales.

5. Aunque en este proyecto se ha tratado con estructuras orgánicas estáticas, el SCA tiene un lógica dinámica. Esto se debe a que sus iteraciones se presentan como crecimientos en el tiempo. Estos crecimientos se podrían interpretar como movimientos de partículas y los puntos de atracción como puntos libres. Con este cambio de perspectiva, el algoritmo simularía el movimiento de un conjunto de partículas que buscan espacios libres. Es decir, podría aplicarse a la simulación del movimiento de multitudes en espacios pequeños tanto para personas como para animales como rebaños o bancos de peces. Esta aplicación podría ser muy interesante de cara al simulacro de medidas de contención y flujo de individuos en estos casos.

6. Anexo: Código

6.1. Algoritmo de Colonización Espacial

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

public class Simulation : MonoBehaviour
{
    //--CLASES--//
    public class Node
    {
        public float x;
        public float y;
        public int gen;
        public bool deprecated;
        public int parent;
        public List<AP> atractedPoints = new List<AP>();
    }
    public class AP
    {
        public float x;
        public float y;
    }
    //--PRIVATES--//
    private List<Node> nList = new List<Node>();
    private List<AP> apList = new List<AP>();

    //--PUBLICS--//
    [Header("Par metros de control")]
    public float kDist;
    public float aDist;
    public float growRate;
```

```
    [Header("Condiciones iniciales")]
    public int initialAPS;
    public Vector2 firstNodePos= Vector2.zero;
    public float iterations = 1000;
    [Header("Drawer and readers")]
    public Drawer drawer;
    public EfimeraReader reader;
    [Header("Par metros extra")]
    public float noiseScaler = 10;


    public void createField()
    {
        createReadedAPField();
      //  drawer.Draw(nList, apList);
    }

    //--FUNCIONES DE SIMULACI N--//
    public (float, float, float, List<Vector2>) startSimulation()
    {
        createReadedAPField();
        createFirstNode();


        for (int i = 0; i < iterations; i++)
        {
            float actualCount = nList.Count;
            growTree();
            if (actualCount >= nList.Count)
            {
                Debug.Log("Paramos en la iteraci n n  " + i);
                //Ya no hay mas crecimiento, paramos.
                break;
            }

        }
        drawer.Draw(nList, apList);
```

```
        //Creamos la lista de las posiciones de los nodos para mandar a
    analizar.
        List<Vector2> points = new List<Vector2>();
        for (int i = 0; i < nList.Count; i++)
        {
            //Escribimos los puntos centrados en 0 al rededor del primer
    nodo.
            Vector2 point = new Vector2(nList[i].x-firstNodePos.x, nList[i
    ].y-firstNodePos.y);
            points.Add(point);
        }
        return (kDist,aDist,growRate,points);
    }

    //--FUNCIONES DE CREACION--//
    public void createFirstNode()
    {
        createNP(firstNodePos.x, firstNodePos.y, 0, false, -1);
    }
    private void CreateAP(float x, float y)
    {
        AP newAp = new AP();
        newAp.x = x;
        newAp.y = y;
        apList.Add(newAp);
    }
    public void createNP(float x, float y, int gen, bool deprecated, int
    parent)
    {
        Node newNode = new Node();
        newNode.x = x;
        newNode.y = y;
        newNode.gen = gen;
        newNode.deprecated = deprecated;
        newNode.parent = parent;
        nList.Add(newNode);
        for (int i = apList.Count - 1; i >= 0; i--)
        {
            float dist = calculateDist(x, y, apList[i].x, apList[i].y);
```

```
            if (dist <= kDist)
            {
                apList.RemoveAt(i);
            }
        }
    }

    //--CREADORES DE LOS PUNTOS DE ATRACCIN INICIALES--//
    public void createRandomAP()
    {
        float x = randomFloat();
        float y = randomFloat();
        AP newAP = new AP();
        newAP.x = x;
        newAP.y = y;
        apList.Add(newAP);
    }
    public void createRandomFieldOfAP()
    {
        int n = (int)Mathf.Round(scaler * density);
        for (int i = 0; i < n; i++)
        {
            createRandomAP();
        }
    }
    private void createReadedAPField()
    {
        List<Vector2> aps= reader.readText();
        boundaries = reader.TextBoundaries();
        foreach (Vector2 apPos in aps)
        {
            float randomx = Random.Range(-1f, 1f);
            float randomy = Random.Range(-1f, 1f);
            CreateAP((apPos.x +randomx)* scaler / boundaries.x, (apPos.y+
    randomy) * scaler / boundaries.x);
        }
    }
    private void createOvalAPField()
    {
```

```
        //Numero de atractores
        for (int i = 0; i < initialAPS; i++)
        {
            float x = Random.Range(0f, 2*a);
            float calc = Mathf.Pow(x - a, 2) / Mathf.Pow(a, 2);
            float limit = b * Mathf.Sqrt(1 - calc);
            float y = Random.Range(-limit, limit);
            CreateAP(x, y);
        }


    }



    //--FUNCIONES DE CALCULO--//

    public float randomFloat()
    {
        float rf = 0;


        rf = Random.Range(-1f,1f);
        rf = scaler * rf;
        return rf;
    }

    public float calculateDist(float x1, float y1, float x2, float y2)
    {
        float xd = x2 - x1;
        float yd = y2 - y1;
        float diff = (float)Mathf.Sqrt(Mathf.Pow(xd, 2) + Mathf.Pow(yd, 2))
;
        return diff;
    }

    //--FUNCIONES DE SCA--//
```

```
//Hasta ahora, se han calculado los puntos de atracci n para cada nodo
//Ahora vamos a probar a calcular los puntos de atracci n antes de
crecer asociandolos a cada nodo.
public void AssociateAtractedPointsToNodes()
{
    for (int i = 0; i < apList.Count; i++)
    {
        int minIndex=-1;
        float minDist=10000;
        for (int j = 0; j < nList.Count; j++)
        {
           float dist= calculateDist(nList[j].x, nList[j].y, apList[i].
x, apList[i].y);
            if(dist<= minDist)
            {
                minDist = dist;
                minIndex = j;
            }
        }
        //Si hemos encontrado puntos cercanos
        if (minIndex != -1)
        {
            //Si est  3en el radio de influencia
            if (minDist <= aDist)
            {
                nList[minIndex].atractedPoints.Add(apList[i]);
            }

        }
    }
}
public void growNodePoint(int index)
{
    //Limpiamos la lista de puntos de atracci n del nodo:
    List<AP> listOfAtractedPoints = nList[index].atractedPoints;
    float x;
    float y;
    int gen;
```

```
        bool deprecated = false;

        int length = listOfAtractedPoints.Count;

        gen = nList[index].gen + 1;
        if (length == 0)
        {
            nList[index].deprecated = true;
            return;
        }
        float sumX = 0;
        float sumY = 0;
        for (int i = 0; i < length; i++)
        {
            float xVar = listOfAtractedPoints[i].x - nList[index].x;
            float yVar= listOfAtractedPoints[i].y - nList[index].y;
            float module1 = Mathf.Sqrt(Mathf.Pow(xVar, 2) + Mathf.Pow(yVar,
    2));
            sumX += xVar / module1;
            sumY += yVar / module1;
        }
        //Normalizamos el vector n obtenido con su m dulo
        float module = Mathf.Sqrt(Mathf.Pow(sumX, 2) + Mathf.Pow(sumY, 2));
        sumX = sumX / module;
        sumY = sumY / module;
        //Aplicamos el par metro de crecimiento al vector n ya normalizado
    y se lo sumamos al vector posicion actual:
        x = sumX * growRate + nList[index].x;
        y= sumY*growRate +nList[index].y;

        nList[index].atractedPoints = new List<AP>();
        createNP(x, y, gen, deprecated, index);
    }



    public void growTree()
    {
        moveAPs();
```

```
        AssociateAtractedPointsToNodes();
        int length = nList.Count;
        //Si contamos los nodos desde el primero usar este bucle
        for (int i = 0; i < length; i++)
        {

            if (!nList[i].deprecated)
            {
                //    Console.WriteLine("Nodo creciendo: " + i);
                growNodePoint(i);
            }
        }


    }

    //--FUNCIONES EXTRA --//

    public void moveAPs()
    {
        for (int i = 0; i < apList.Count; i++)
        {
            float randomx = Random.Range(-1f, 1f);
            float randomy = Random.Range(-1f, 1f);
            apList[i].x += randomx/noiseScaler;
            apList[i].y += randomy/noiseScaler;
        }
    }


}
```

6.2. Análisis de la dimensión fractal

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
public static class Analyzer
{

    public static float GetFractalDimension(List<Vector2> points, float
    distBetweenRads)
    {
        List<Vector2> resultsList = new List<Vector2>();
        float biggestRadius = 0;
        foreach (Vector2 point in points)
        {
            float newR = Mathf.Abs(point.magnitude);
            if (newR >= biggestRadius)
            {
                biggestRadius = newR;
            }
        }
        bool ended = false;
        float ri = 0;
        while (!ended)
        {

            float ni = 0;
            foreach (Vector2 point in points)
            {
                float newR = Mathf.Abs(point.magnitude);
                if (newR <= ri)
                {
                    ni += 1;
                }
            }
            resultsList.Add(new Vector2(ri, ni));
            ri += distBetweenRads;
            if (ri >= biggestRadius)
```

```
        {
            ended = true;
            Debug.Log("Analisis finalizado");
        }
    }
    //Puntos (N,r) guardados en resultsList
    //Pasamos a hacer la estimaci n lineal.

    float xm = 0;
    float ym = 0;
    float s2x = 0;
    float s2y = 0;
    float sxy = 0;
    float fd = 0;
    resultsList.RemoveAt(0);
    float length = resultsList.Count;
    foreach (Vector2 point in resultsList)
    {
        float logx = Mathf.Log(point.x);
        float logy = Mathf.Log(point.y);
        xm += logx;
        ym += logy;
    }
    xm = xm / length;
    ym = ym / length;
    foreach (Vector2 point in resultsList)
    {
        float logx = Mathf.Log(point.x);
        float logy = Mathf.Log(point.y);
        s2x += Mathf.Pow(logx - xm, 2);
        s2y += logy - ym;
        sxy += (logx - xm) * (logy - ym);
    }
    s2x = s2x / length;
    s2y = s2y / length;
    sxy = sxy / length;
    fd = sxy / s2x;

    return fd;
```

```
    }
}
```

6.3. Tratamiento de imágenes

```
using System;
using System.Drawing;
using System.IO;

namespace Fractal_Analizer
{

    class Program
    {
        static void Main()
        {
            Console.Write("Empezando!");
            Color Threshold = Color.FromArgb(90, 90, 90);
            Color Black = Color.FromArgb(0, 0, 0);
            Color White = Color.FromArgb(255, 255, 255);
            int smoother = 2;

            Bitmap imageNormal = new Bitmap(@"C:\Users\charl\Desktop\TFG\
    Fractal Analizer IMGS\prueba6.png");
            Bitmap imageNormal2 = new Bitmap(@"C:\Users\charl\Desktop\TFG\
    Fractal Analizer IMGS\prueba6.png");
            Bitmap imageNormal3 = new Bitmap(@"C:\Users\charl\Desktop\TFG\
    Fractal Analizer IMGS\prueba6.png");
            // Do some processing
            for (int x = 0; x < imageNormal.Width; x++)
            {
                for (int y = 0; y < imageNormal.Height; y++)
                {
                    imageNormal3.SetPixel(x, y, White); //Seteamos toda la
    imagen a blanco.
                    Color pixelColor = imageNormal.GetPixel(x, y);

                    if (pixelColor.R >= Threshold.R && pixelColor.B >=
    Threshold.B && pixelColor.G >= Threshold.G)
                    {
                        imageNormal.SetPixel(x, y, White);
                        imageNormal2.SetPixel(x, y, White);
```

```
                }
                else
                {
                    imageNormal.SetPixel(x, y, Black);
                    if(x<imageNormal.Width -smoother && y<imageNormal.
Height - smoother && x >  smoother && y > smoother)
                    {

                        for (int i = 0; i < smoother; i++)
                        {
                            for (int j = 0; j < smoother; j++)
                            {
                                imageNormal2.SetPixel(x+i, y+j, Color.
Black);
                                imageNormal2.SetPixel(x - i, y - j,
Color.Black);
                            }
                        }
                    }
                }

            }
        }
        imageNormal.Save(@"C:\Users\charl\Desktop\TFG\Fractal Analizer
IMGS\resultado1.jpg");
        imageNormal2.Save(@"C:\Users\charl\Desktop\TFG\Fractal Analizer
 IMGS\resultado2.jpg");



        Color lastColor = Color.Red;
        int countdown = 0;


        for (int i=0; i< imageNormal.Height; i++) //barrido vertical
        {
            countdown = 0;
            for(int j=10; j < imageNormal.Width; j++) //barrido
horizontal
```

```
                {
                    Color pixelColor = imageNormal2.GetPixel(j, i);
                    //Si cambiamos a blanco seteamos en funcion de cuanto
    llevasemos en negro contando con el countdown. Se reinicia el countdown
                    if (pixelColor != lastColor && pixelColor==White)
                    {
1070                    imageNormal3.SetPixel(j - (countdown / 2), i, Black
    );
                        countdown = 0;
                    }


                    if (pixelColor==lastColor && pixelColor==Black) //Si
    seguimos en negro contamos.
                    {
                        countdown++;
                    }

1080                lastColor = pixelColor;

                }
            }
            imageNormal3.Save(@"C:\Users\charl\Desktop\TFG\Fractal Analizer
     IMGS\resultado3.jpg");
        }
    }
}
```

Bibliografía

[AAA14] Ankit Garg,Akshat Agrawal and Ashish Negi. A Review on Natural Phenomenon of Fractal Geometry. International Journal of Computer Applications (0975-8887). Volume 86 - No4, January 2014.

[HH71] Honda H.: Description of the form of trees by the parameters of the tree-likebody:Effects of the branching angle and the branch length on the shape of the tree-likebody. Journal of Theoretical Biology31(1971), 331– 338.

[ABP07] Adam Runions, Bendan Lane, and Przemyslaw Prusinkiewic: Modeling Trees with a Space Colonization Algorithm. Eurographics Workshop on Natural Phenomena. 2007.

[AS00] Arghavan Salles: Ephemeroptera. Part of a Biology 1B project for Section 107, under Brian R. Speer (GSI). 2000.

[AR17] Andrew Ross: Insect Evolution: The Origin of Wings. Department of Natural Sciences, National Museum of Scotland, Chambers Street, Edinburgh EH1 1JF, UK. 2016

[LJM98] Laurence Ruffieux, Jean-Marc Elouard and Michel Sartori:Flightlessness in mayflies and its relevance to hypotheses on the origin of insect flight. Proc. R. Soc. Lond. B. 1998.

[CS20] Documentación en línea del lenguaje de programación C# por parte de © Microsoft 2020. `https://docs.microsoft.com/es-es/dotnet/csharp/` . 2020.

[UNITY20] Documentación en línea del entorno gráfico Unity por parte de © 2020 Unity Technologies. https://docs.unity3d.com/Manual/index.html. 2020.

[FS11] M. Fernández-Martínez,1, M.A. Sánchez-Granero. Fractal dimension for fractal structures: A Hausdorff approach. Area of Geometry and Topology, Faculty of Science, Universidad de Almería, 04120 Almería, Spain. 2011.

[FC20] A. Ruiz-Sobrino, C.A. Martín-Blanco, T. Navarro, I. Almudí, G. Masiero,M. Jimenez-Caballero, D.B. Buchwalter, D.H. Funk, J.L. Gattolliat, M.C. Lemos,F. Jimenez, F. Casares. Space colonization by branching trachea explains the morphospace of a simple respiratory organ. Developmental Biology 462, 50-59 (2020).

[WPBF] `https://en.wikipedia.org/wiki/Barnsley_fern`

[WPSL] `https://es.wikipedia.org/wiki/Sistema-L`

[JC] `https://www.microsiervos.com/archivo/ciencia/juegos-caoticos-fractales.html`

[DLA] `http://paulbourke.net/fractals/dla/‘`