

Calculo de la dimensión fractal de objetos 3D

Grado en Ingeniería Informática



Trabajo Fin de Grado

Autor:

Gacel Ivorra Rodríguez

Tutor/es:

Miguel Ángel Cazorla

Sergio Ramón

Junio 2017

1.- Resumen

Este TFG consiste en una investigación y experimentación del cálculo de la dimensión fractal por medio del algoritmo de conteo de cajas, en inglés “Box Counting”, aplicado sobre nubes de puntos 3D.

Se ha detectado que no existe documentación alguna donde podamos ver que se haya intentado realizar este tipo de cálculos sobre objetos representados mediante nubes de puntos, por lo que la mayor parte del trabajo se centra en, una vez desarrollado el algoritmo, realizar una extensa experimentación tratando de obtener la mejor solución posible variando los parámetros que modifican el comportamiento de éste, y realizando una comparativa de resultados sobre estas distintas variantes.

Para la experimentación se utilizan algunas formas cuya dimensión fractal es conocida teóricamente y se pretende en este trabajo ajustar el algoritmo de tal manera que sea capaz de dar un resultado igual o muy aproximado a la dimensión teórica del objeto en cuestión.

2.- Motivación

Este proyecto viene motivado por el impulso de desarrollar y poner en práctica técnicas que hagan uso de lo que se conoce como geometría fractal.

La geometría fractal es un campo relativamente joven y reciente de las matemáticas que aporta un enfoque muy interesante ya que es capaz de “descodificar” o reconocer patrones existentes en objetos aparentemente caóticos los cuales no se podían llegar a describir mediante matemáticas tradicionales.

Como más adelante se explicará en este documento y ya es conocido por muchas personas que han trabajado o estudiado acerca de este enfoque matemático, existe un enorme y profundo potencial en el aspecto científico sobre esta materia ya que a raíz de este se han obtenido unas de las técnicas matemáticas que más se aproximan a lo que sería una descripción natural y simple de nuestra realidad, el universo y la naturaleza en su conjunto.

Desde el punto de vista científico y objetivo, no hay ningún objeto o forma física existente que pueda representarse de manera estricta mediante líneas rectas, planos, esferas perfectas, etc. ¿Por qué es cierta esta afirmación? Se ha demostrado desde hace muchos años que, haciendo zoom sobre cualquier objeto aparentemente plano, a medida en que se van obteniendo mayores ampliaciones, se comienzan a observar más y más rugosidades sobre éste, y si pudiéramos llegar hasta el fondo veríamos que ese objeto al final es un conjunto de átomos agrupados formando el objeto original pero que jamás podríamos considerar de manera objetiva como una superficie lisa.

Es una manera simple de explicar algo extremadamente complejo, pero podemos concluir que se hace ver una carencia en las matemáticas tradicionales para describir formas y fenómenos existentes en la naturaleza como pueden ser un árbol, una nube, una montaña, etc. de una manera relativamente sencilla.

Teniendo en cuenta todo lo anterior, siendo conscientes de la sabiduría innata de la naturaleza, y sabiendo que gran parte de la tecnología y los grandes inventos desarrollados por los seres humanos se basan en imitar estos patrones naturales, se hace una obligación el explorar y seguir desarrollando ideas basadas en esta “matemática de lo natural”.

A continuación, se muestran una serie de imágenes de fractales naturales donde se observa que la naturaleza utiliza los mismos patrones repetitivos desde la escala macro a la micro, en todos los niveles (esto aplica también para todo tipo de patrones existentes, no solo de formas):

Col Romanesco

Copo de nieve

Nubes

Árbol

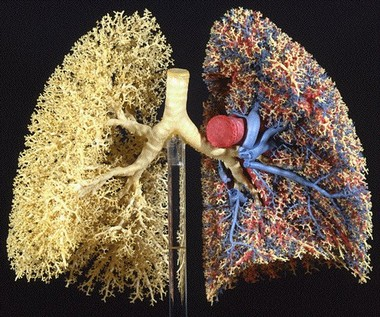


Rayo

Desembocadura del rio Lena, Siberia



Molde de árbol bronquial humano



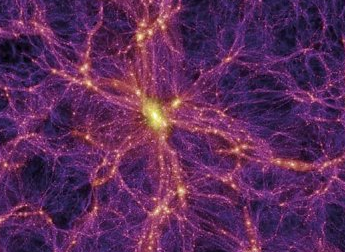
Red de neuronas (cerebro humano)



Vía láctea



Cúmulo de galaxias y filamentos



3.- Objetivos

Los objetivos principales de este trabajo son:

* Implementar el algoritmo Box Counting aplicado sobre figuras representadas por nubes de puntos 3D.
* Investigar sobre la factibilidad de aplicar técnicas de cálculo de dimensión fractal sobre objetos representados mediante nubes de puntos.
* Realizar una experimentación sobre qué solución del algoritmo proporciona los mejores resultados.
* Establecer las premisas a seguir para aplicar Box Counting sobre nubes de puntos 3D mediante las conclusiones obtenidas de la investigación y experimentaciones realizadas.
* Obtener una herramienta software que permita calcular de manera fiable la dimensión fractal de objetos 3D.
* Profundizar y divulgar en la medida de lo posible sobre técnicas de geometría fractal.

Aunque existe una motivación de fondo por utilizar la dimensión fractal como una característica de peso para realizar clasificación de objetos 3D, queda fuera del ámbito de este proyecto la aplicación de dicha característica sobre este ni sobre ningún otro fin en concreto, ya que si se realiza el trabajo con éxito se conseguirá una utilidad que sirva para cualquier campo de aplicación en el que se desee utilizar.

Las diferentes aplicaciones de la medida conocida como dimensión fractal serán detalladas más adelante a lo largo de este documento.

2.- Introducción

Para empezar con el desarrollo de este trabajo, es necesario explicar una serie de conceptos clave para su entendimiento.

2.1.- ¿QUÉ ES UN FRACTAL?

Un **fractal** es un ente geométrico cuya estructura básica se repite a diferentes escalas. El término fue propuesto por el matemático Benoît Mandelbrot en 1975 y deriva del latín fractus, que significa quebrado o fracturado.

Un **fractal ideal** es una figura geométrica que los matemáticos crean por medio de un algoritmo iterativo o regla repetitiva que tiene una forma, bien sea sumamente irregular, bien sumamente interrumpida o fragmentada, y sigue siendo así a cualquier escala que se produzca el examen. Los fractales matemáticos cumplen con la propiedad de autosimilitud.

Un **fractal** **natural** es un elemento de la naturaleza que puede ser descrito mediante la geometría fractal. Las nubes, las montañas, el sistema circulatorio, las líneas costeras o los copos de nieve son fractales naturales. Esta representación es aproximada, pues las propiedades atribuidas a los objetos fractales ideales, como el detalle infinito, tienen límites en el mundo físico.

2.2.- CARACTERISTICAS DE UN FRACTAL

Un objeto es fractal cuando es demasiado irregular para ser descrito en términos geométricos tradicionales.

De forma general, podemos caracterizar los fractales mediante las siguientes propiedades:

* Tienen una estructura compleja a cualquier resolución.
* Tienen una dimensión no entera.
* Tienen un perímetro de longitud que tiende a infinito, pero un área limitada.
* Son auto-similares e independientes de la escala

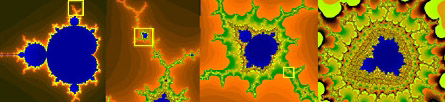
La característica más destacada es la autosimilitud. Según Benoit Mandelbrot, un objeto es autosimilar o autosemejante si sus partes tienen la misma forma o estructura que el todo, aunque pueden presentarse a diferente escala y pueden estar ligeramente deformadas.

Los fractales pueden presentar tres tipos de autosimilitud:

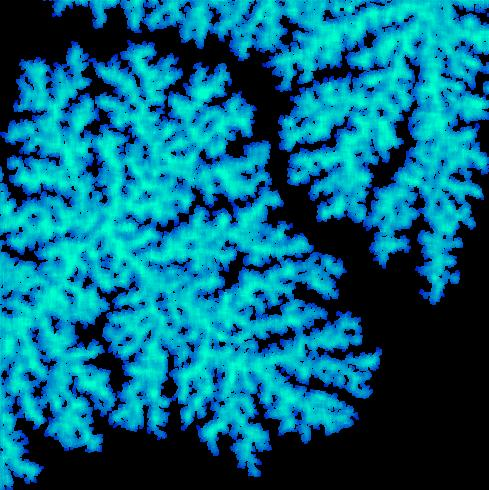
* **Autosimilitud exacta**. Este es el tipo más restrictivo de autosimilitud: exige que el fractal parezca idéntico a diferentes escalas. A menudo la encontramos en fractales definidos por sistemas de funciones iteradas (IFS), un ejemplo es el triángulo de Sierpinsky:



* **Cuasiautosimilitud**: Exige que el fractal parezca aproximadamente idéntico a diferentes escalas. Los fractales de este tipo contienen copias menores y distorsionadas de sí mismos. Los fractales definidos por relaciones de recurrencia son normalmente de este tipo. En este grupo encontramos, por ejemplo, el famoso fractal de Mandelbrot:



* **Autosimilitud estadística**. Es el tipo más débil de autosimilitud: se exige que el fractal tenga medidas numéricas o estadísticas que se preserven con el cambio de escala. Los fractales aleatorios son ejemplos de fractales de este tipo. A continuación, un fractal con autosimilitud estadística generado por el proceso de agregación limitada por difusión:



2.3.- ¿QUÉ ES LA GEOMETRÍA FRACTAL?

La geometría fractal ofrece un modelo alternativo que busca una regularidad en las relaciones entre un objeto y sus partes a diferentes escalas. Esta forma de regularidad no precisa el encorsetamiento del objeto en otras formas geométricas que, aunque elementales, no dejan de ser externas al mismo, sino que busca la lógica interna del propio objeto mediante relaciones intrínsecas entre sus elementos constitutivos cuando estos se examinan a diferentes escalas. De esta forma no se pierden ni la perspectiva del objeto global, ni del aspecto del mismo en cada escala de observación. La geometría fractal busca y estudia los aspectos geométricos que son invariantes con el cambio de escala.

2.4.- DIMENSIÓN FRACTAL

Para dar una definición de qué es la dimensión fractal y qué significado tiene veámoslo con un ejemplo:

Una hoja de papel es un objeto tridimensional ya que por fina que parezca, tiene un cierto grosor. Supongamos que esto no fuera así y que fuera un plano perfecto de 2 dimensiones. En ese caso podríamos coger la hoja y arrugarla hasta formar una bola. El objeto tendría volumen y sería sólido, pero no sería tridimensional porque la bola está llena de huecos y discontinuidades. Para convertirla en una esfera tendríamos que hacer un largo número de interpolaciones lineales. Todo esto explica porque es tan difícil modelar la naturaleza con la geometría euclideana. La mayoría de objetos en el mundo real no son sólidos en el sentido de Euclides pues tienen hoyos y deformaciones. A pesar de residir en el espacio tridimensional, su dimensión es fraccionaria entre uno y dos, o entre dos y tres.

La **dimensión fractal** intenta medir en qué grado un objeto 2D llena el espacio 3D, o un objeto de dimensión 1 se asemeja a una superficie 2D.

Esta última definición es perfecta cuando estamos hablando de objetos geométricos, pero esta misma idea también se da con una infinidad de patrones de toda clase que se pueden presentarse como una forma geométrica, por ejemplo una gráfica de datos. De esta manera podríamos obtener una estimación de la dimensión fractal de la curva del valor de las acciones en bolsa durante diferentes etapas y/o tendencias y podríamos hacer una comparativa en el tiempo para luego realizar predicciones. Se puede asignar una dimensión fractal a una trayectoria (1D) en el plano (2D), o incluso en el espacio (3D), por ejemplo, para analizar y comparar los movimientos de un ratón en respuesta a diferentes medicamentos y poder tener un índice de la excitación o relajación producida en respuesta a cada medicamento. Puede extrapolarse esta medida a infinidad de patrones a los que se nos ocurra aplicarlo, muchos conocidos y muchos otros aún por descubrir e investigar, y se ha demostrado que funciona muy bien en diferentes campos de la ciencia.

En resumen, la dimensión fractal es una medida capaz de captar la esencia de un patrón repetitivo, parecido a la manera en que la naturaleza se manifiesta en la realidad, tanto a nivel de formas (ríos, montañas, arboles, plantas, vasos sanguíneos, nubes, etc.), como de patrones de todo tipo que seamos capaz de extrapolar a esta matemática.

3.- Estado del arte

4.- Metodología

Veamos los pasos seguidos para el desarrollo de este trabajo:

En una primera fase, se ha realizado un estudio sobre las técnicas disponibles para el cálculo de la dimensión fractal. De forma paralela, se ha buscado información sobre trabajos previos realizados sobre objetos representados como nube de puntos para saber cuál era el punto de partida y las ideas que se habían desarrollado previamente y de las que pudiéramos hacer uso.

Al no encontrar nada similar que pudiera servir de referencia, se ha optado por desarrollar el algoritmo Box Counting que, aunque no sea el mejor de todos a nivel de resultados, es el más extendido en todas las áreas debido a que es más sencillo de implementar. Proporciona buenos resultados y es rápido, lo que nos va a permitir desarrollar una experimentación más fluida y poder abarcar más pruebas.

Una vez implementado el software necesario, se ha realizado una experimentación bastante extensa, aplicando el algoritmo sobre objetos con dimensión fractal conocida, analizando los resultados mediante gráficas, y qué solución proporciona los mejores resultados.

También se realizarán pruebas sobre familias de objetos cuya dimensión fractal exacta no es conocida pero sabiendo que siguen una escala de rugosidad que debería verse reflejada por el resultado obtenido para cada uno de ellos.

Esto último se debe a que aunque la solución final obtenida no proporcionara la dimensión fractal de forma exacta, seguiría siendo un algoritmo válido si es capaz de conservar la escala ya que permitiría hacer comparaciones entre objetos.

5.- Plataforma, recursos y herramientas utilizadas

A continuación, se presentarán los recursos utilizados para el desarrollo de este trabajo:

5.1.- Características del equipo utilizado

* Procesador: Intel Core i5 M450 @ 2.40GHz 64 bits
* Memoria RAM: 4GB DDR2 800MHz
* Sistema operativo: Linux Mint 17.2 Rafaela

5.2.- Librerías software

**Lenguaje de programación**

Se ha decidido realizar el desarrollo mediante el lenguaje de programación C++ debido a que es un lenguaje muy potente, estable y que produce software muy rápido. También porque es el lenguaje nativo de la PCL.

**Point Cloud Library (PCL)**

La herramienta conocida como PCL (Point Cloud Library) consiste en una librería desarrollada para el tratamiento completo de nubes de puntos 3D, liberada bajo licencia BSD, que cuenta con varios métodos de alta eficiencia computacional.

PCL surgió ante la necesidad de que los robots tengan la capacidad de percibir el mundo tal como lo hacemos los seres humanos, es decir, que puedan determinar las diferentes características y detalles que el ojo humano puede observar. Esta librería se ha potencializado debido al gran avance y al bajo costo de los sensores de adquisición de imágenes 3D, en el que cabe destacar en sensor Kinect® de la compañía Microsoft el cual se basa en la tecnológica PrimeSense.

Esta librería dispone de varios módulos que proporcionan métodos computacionales para manipulación de las nubes de puntos, entre ellos los que se han utilizado para este trabajo:

* **Filtros** (libpclfilters)**:** PCL posee métodos de filtrado digital como por ejemplo eliminadores de datos atípicos, filtro tipo vóxel, eliminador con condición, suavizado, índices de extracción y proyecciones. Debido al volumen de datos que genera una nube de puntos también existe la posibilidad de utilizar filtros que además de eliminar valores atípicos, hacen una gran reducción de datos, permitiendo así una mayor rapidez en la computación.
* **KdTree** (libpclKdtree):Esta biblioteca consiste en una estructura tipo árbol que almacena un conjunto de puntos k-dimensional con el fin de realizar búsquedas eficientes del vecino más cercano.
* **Reconstrucción de Superficies** (libpclsurface):Esta biblioteca se utiliza para mejorar la visualización de un modelo tridimensional procesando los puntos de una nube para obtener una representación en malla o una superficie alisada.
* **Visualización** (libpclvisualization): Esta librería permite la rápida visualización de algoritmos que operan sobre nubes de puntos 3D. Cuenta con métodos de procesamiento configuración de propiedades visuales como colores, tamaños de punto y opacidad.

<http://pointcloudlibrary.blogspot.com.es/2013/12/herramientas-de-la-libreria-pcl.html>

**MeshLab**

MeshLab es un software de código abierto para procesar y editar mallas 3D. Proporciona un conjunto de herramientas para editar, limpiar, inspeccionar, renderizar, texturizar y convertir mallas. Ofrece características para procesar datos en bruto producidos por otras herramientas o dispositivos de digitalización 3D y para preparar modelos para la impresión 3D.

En este trabajo se ha utilizado para poder visualizar y renderizar figuras durante la generación de fractales y como herramienta de apoyo para algunas otras operaciones durante la experimentación.

**Librería QT4**

Qt es una amplia plataforma de desarrollo gratuita y de código abierto que incluye clases, librerías y herramientas para la producción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas.

Una de esas herramientas es Qt Creator, un IDE que facilita la creación de formularios, botones y ventanas de dialogo con el uso del ratón.

Por todo ello, es más que adecuada para poder crear una interfaz sencilla donde visualizar las nubes de puntos y realizar algunas operaciones sobre ellas.

[https://es.wikibooks.org/wiki/Programaci%C3%B3n\_con\_Qt4](https://es.wikibooks.org/wiki/Programación_con_Qt4)

**Gnuplot y Gnuplot-iostream**

Gnuplot es una utilidad gráfica de línea de comandos para Linux, OS / 2, MS Windows, OSX, VMS y muchas otras plataformas. Es gratuita y fue creada para permitir a los científicos y estudiantes visualizar funciones matemáticas y datos de forma gráfica.

Gnuplot-iostream es una interfaz que permite que gnuplot sea controlado desde un programa en C++.

Esta herramienta ha sido la utilizada para representar los resultados del algoritmo implementado en una gráfica XY.

7.- Generador de fractales

Para poder realizar el estudio aplicando el algoritmo box counting en 3D, se ha implementado un programa que genere dos fractales simples de los que conocemos a priori la dimensión fractal, los cuales servirán de modelos con los que ir ajustando el algoritmo cada vez más.

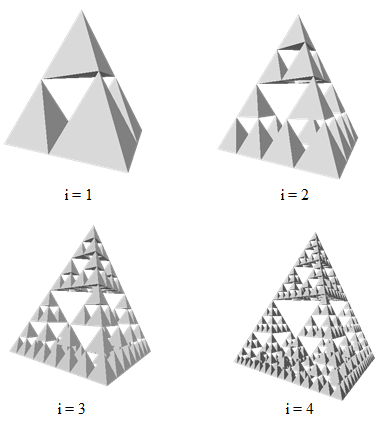
Las figuras generadas estarán representadas como nubes de puntos, los cuales representan cada uno de los vértices de la figura, en formato PLY.

**Tetraedro de Sierpinsky**

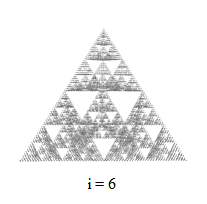
El tetraedro de Sierpinsky es la versión 3D del triángulo de Sierpinsky, una de las figuras fractales más conocidas. Se ha generado de la forma más sencilla y rápida computacionalmente mediante el siguiente algoritmo iterativo:

1. Partimos de un tetraedro equilátero de lado l = 100 representado por sus 4 vértices.
2. Se realizan tres copias del tetraedro inicial realizando un desplazamiento a los puntos de cada uno para que entre los 4 tetraedros formen un tetraedro que duplica las medidas del tetraedro inicial (no se duplican los puntos que tengan las mismas coordenadas).
3. Vamos a generar **dos versiones del tetraedro** que se diferencian en este punto según cual de las dos decisiones posibles se tomen. Nos viene bien tener estas dos versiones para poder demostrar que el tamaño del objeto no es relevante para el calculo de la dimensión fractal:
   1. Se dividen entre 2 las coordenadas de todos los puntos para mantener la escala inicial (en todas las iteraciones el tetraedro tendrá l = 100).
   2. No se realiza ninguna operación en este punto, por lo que el tetraedro resultante de cada iteración doblará las medidas del de la anterior.
4. Se comienza en el punto 1 hasta realizar todas las iteraciones que se deseen realizar.

En las imágenes se muestran los tetraedros representados por polígonos y no por los vértices para que sea más claro a la vista, aunque posteriormente trabajaremos solo con los vértices como una nube de puntos.



A medida que aumentan las iteraciones utilizadas, más fácilmente puede diferenciarse la forma solamente visualizando los vértices.

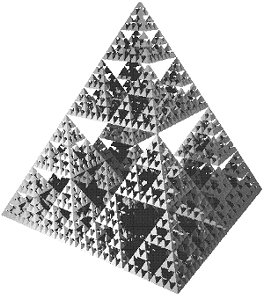


Sabemos que la dimensión fractal de este objeto se puede calcular como

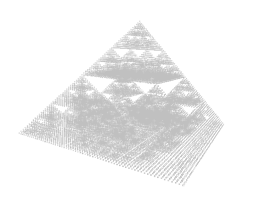
D = log(4)/log(2) = 2

**Pirámide de Sierpinsky**

La pirámide de Sierpinsky es una variante del tetraedro, pero en lugar de partir de un tetraedro equilátero, se parte de una pirámide. La generación se realiza siguiendo los mismos pasos descritos para el tetraedro.

Iteración 5

Iteración 5 solo puntos

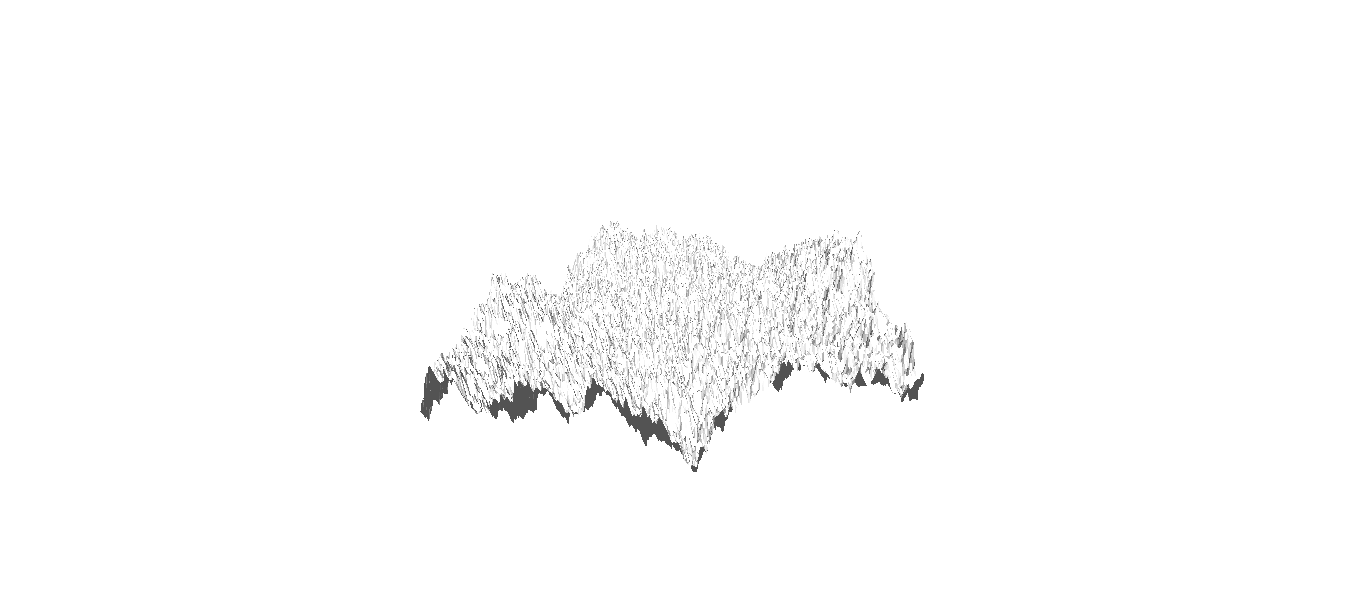
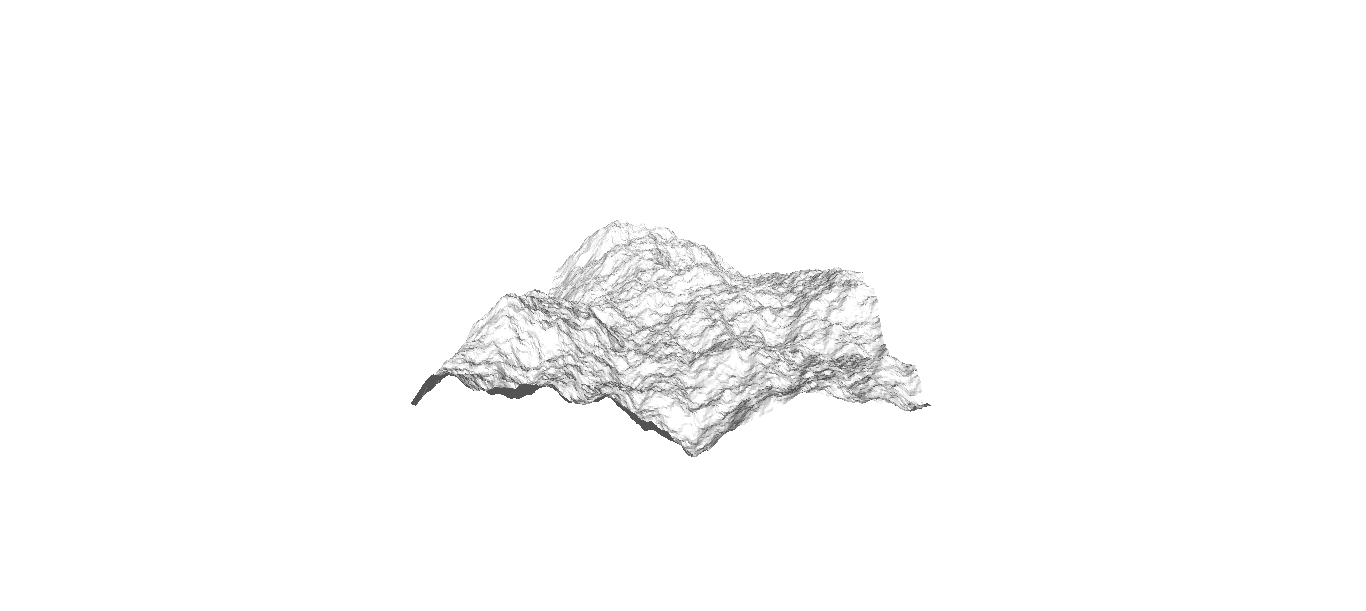
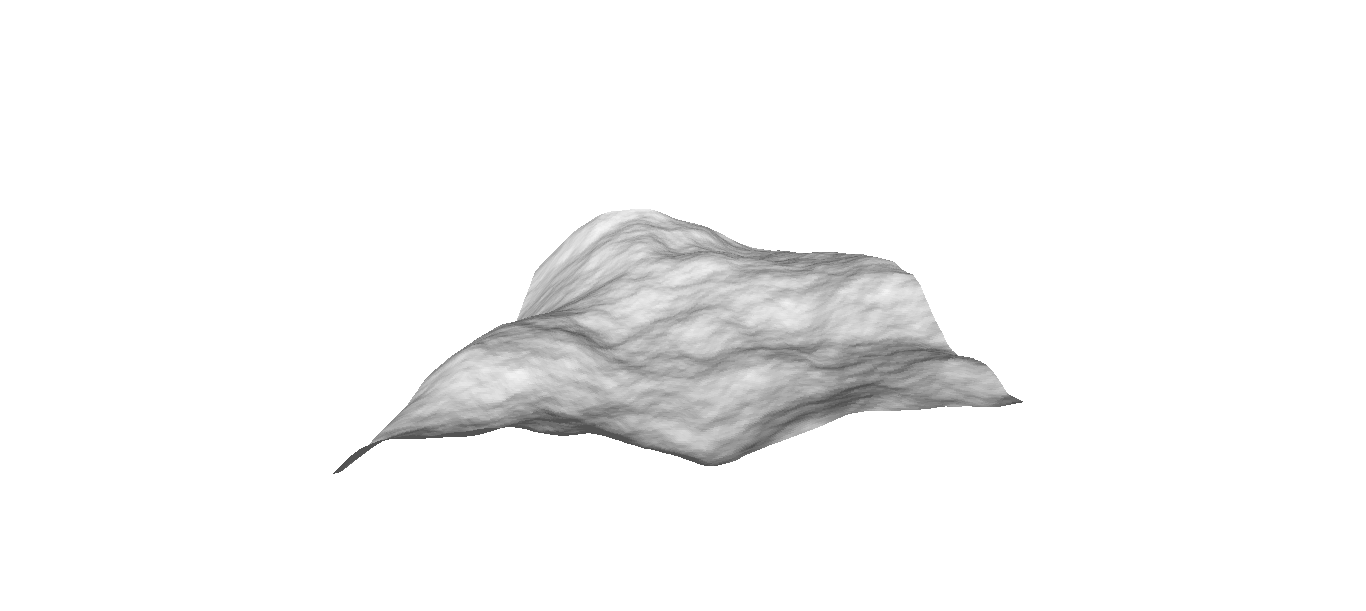


Sabemos que la dimensión fractal de este objeto se puede calcular como

D = log(5)/log(2) = 2.3219

**Terrenos rugosos**

Mediante el algoritmo fBM (Fractal Brownian Motion) software MeshLab se han generado 13 terrenos de los cuales no sabemos su dimension fractal exacta pero si que siguen una escala de rugosidad. A continuación se muestran varios de esos modelos generados, no todos ya que no es relevante, simplemente aquí se expone que tenemos una serie de modelos de los que sabemos que varia su rugosidad y los resultados del algoritmo deberán mantener esa escala reflejada:



6.- Algoritmo Box Counting

El algoritmo conocido como box counting es el algoritmo más extendido en todas las ramas del conocimiento para el cálculo de la dimensión fractal debido a su relativamente sencilla implementación y la velocidad a la que realiza los cálculos.

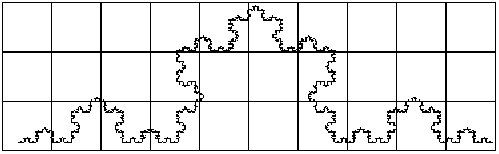
Se va a realizar una explicación del algoritmo en 2 dimensiones para que se pueda entender más fácilmente, y a continuación se describirá la implementación realizada en este trabajo para 3 dimensiones.

Como su propio nombre indica, lo que realiza el algoritmo es un conteo de cajas. Es un algoritmo iterativo el cual realiza los siguientes pasos:

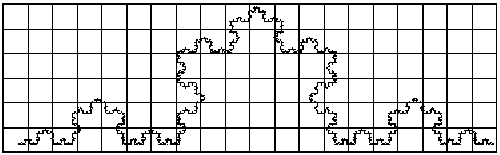
1. Calcula el rectángulo que recubre toda la curva o figura
2. Realiza una división del rectángulo en cajas o celdas del mismo tamaño *t.*
3. Se realiza un conteo de las celdas que contienen parte de la curva
4. Se almacena un punto (x,y) en el que la x se corresponde al tamaño de las cajas en esta iteración y la y al número de cajas que contienen parte de la curva
5. Se realiza un decremento sobre el tamaño de las cajas y se vuelve al punto 1 tantas veces como iteraciones se quieran realizar.
6. Una vez completadas todas las iteraciones y teniendo el resultado de cada iteración, se calculan los puntos Log(x) Log(y) para cada punto obtenido en cada iteración.
7. Se realiza la recta de regresión sobre los puntos (Log(x),Log(1/y))
8. La pendiente de la recta obtenida es una estimación de la dimensión fractal del objeto

A continuación, se muestra un ejemplo de 3 iteraciones realizado sobre la curva de Koch, otro de los fractales matemáticos más conocidos y utilizados para demostraciones.

Iteración i = 1, tamaño t = 10

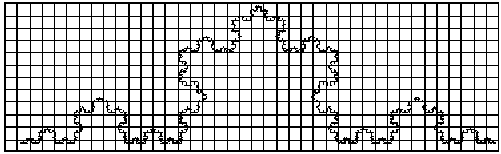
18 de 30 cajas contienen parte de la curva, obtenemos el punto p1 = (10, 18)

Iteración i = 2, tamaño t = 5



41 de 120 cajas contienen parte de la curva, obtenemos el punto p2 = (5, 41)

Iteración i = 3, tamaño t = 2.5

105 de 480 cajas contienen parte de la curva, obtenemos el punto p3 = (2.5, 105)

En este punto, si realizamos el calculo Log(x), Log(1/y) para cada punto obtenemos los siguientes puntos:

**p1'** = (Log(10), Log(1/18)) = **(1, -1.255272505)**

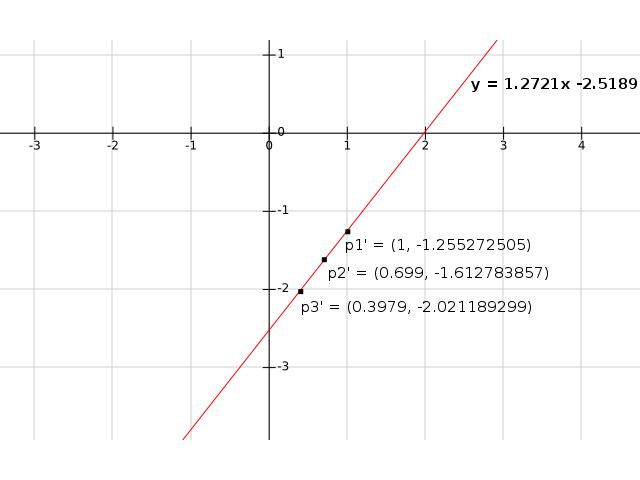
**p2'** = (Log(5), Log(1/41)) = **(0.699, -1.612783857)**

**p3'** = (Log(2.5), Log(1/105) = **(0.3979, -2.021189299)**

Sabemos que la curva de Koch tiene una fractal dimensión teórica es:

**D = Log(4)/Log(3) = 1.2619**

Pues bien, si ahora calculamos la recta de regresión sobre p1', p2' y p3' y lo representamos mediante una gráfica obtenemos:

La recta se alinea perfectamente con los 3 puntos con una pendiente, o lo que es lo mismo, la dimensión fractal calculada D\_bc = 1.2721 ~ (D = 1.2619).

Sorprendentemente, con este algoritmo tan simple y en tan solo 3 iteraciones obtenemos una aproximación muy ajustada a la dimensión fractal teórica del objeto analizado.

Lo que se va a realizar en este trabajo es una implementación de este algoritmo, llevado a 3 dimensiones, con objetos más complejos y discontinuos, representados como nubes de puntos en un espacio 3D.

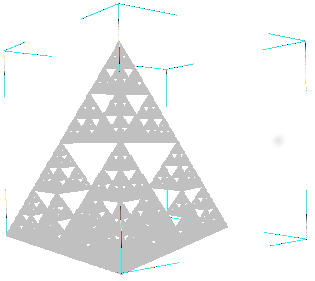
6.1.- Implementación del algoritmo

Box Counting no tiene ninguna diferencia sustancial en los cálculos al pasar de 2D a 3D, simplemente debemos adaptar las divisiones del espacio por cada iteración, de cuadrados a cubos. Una peculiaridad que a priori parece que si que puede ser más problemática es la representación de los objetos de forma discontinua.

Se va a describir el algoritmo implementado en una primera fase, y los cambios que se han ido realizando en las siguientes fases buscando mejorar los resultados obtenidos.

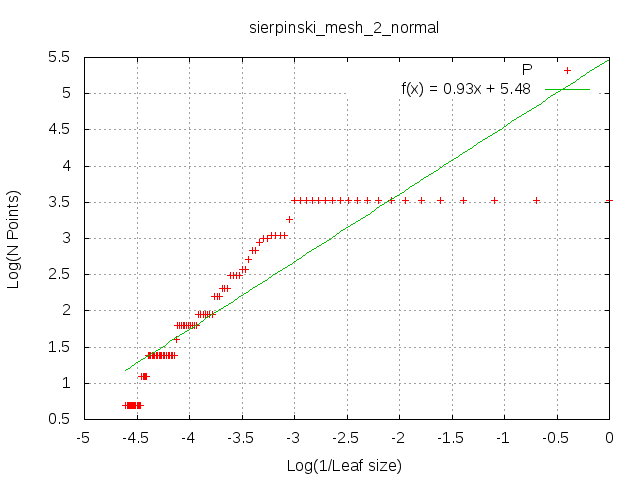
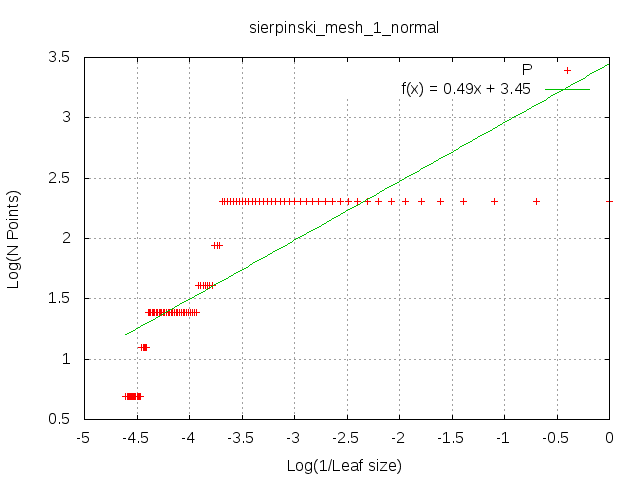
**FASE 1**

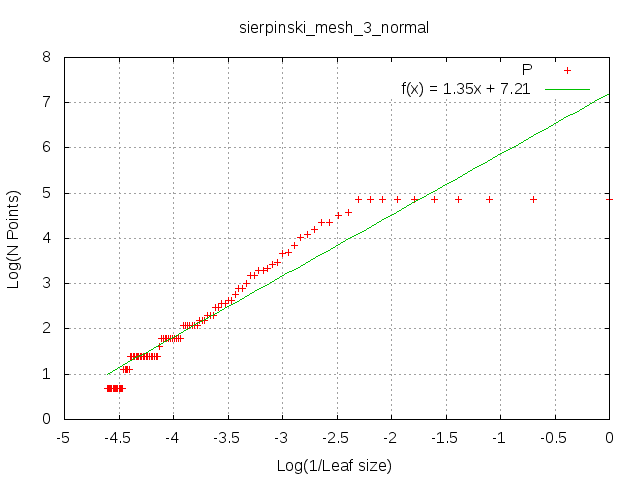
En esta fase los pasos realizados para el calculo de la dimensión fractal son los siguientes:

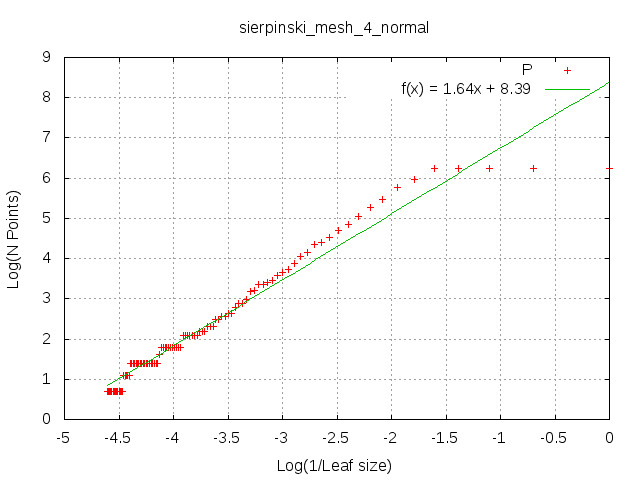
1. Calculo del bounding box. Bounding box se traduce como caja delimitadora y esto se traduce en la obtención de dos puntos 3D, min(min\_x, min\_y, min\_z) y max (max\_x, max\_y, max\_z):
2. Obtenemos la longitud de la figura en cada eje y guardamos el que tenga mayor longitud max(max\_x – min\_x, max\_y – min\_y, max\_z – min\_z) y se almacena en la variable “*tamaño\_max*”.
3. Dividimos el valor máximo entre un número de iteraciones especificadas por parámetros. El valor obtenido será el tamaño inicial de los cubos o vóxeles en los que se fragmentará la figura para realizar los cálculos, y también será el incremento que se realizará en cada iteración sobre el tamaño del vóxel. Es importante señalar que el elemento estructurante, es decir, las cajas, son de forma cuadrada y no rectangular.
4. Se asigna el valor a las variables “*tamaño\_voxel”* e “*incremento*”.
5. Se realiza la división del bounding box en cubos de tamaño “*tamaño\_voxel”,* se realiza el conteo de la cantidad de ellos que contiene algún punto de la nube y se almacena en la variable N.
6. Se almacena en el vector de pares “*resultados”* el par (o punto 2D) (*tamaño\_voxel*, *N*).
7. Se incrementa *tamaño\_voxel* con el valor de la variable *incremento*.
8. Se vuelve al punto 5 hasta que se hayan completado el número de iteraciones especificado.
9. Calculamos el vector “*resultados\_log\_log”* aplicando la fórmula ( Log(*tamaño\_voxel*), Log(1/*N*) ).
10. Se obtiene la recta de regresión sobre los puntos del vector *resultados\_log\_log.*
11. La dimensión fractal calculada es igual a la pendiente de la recta obtenida.

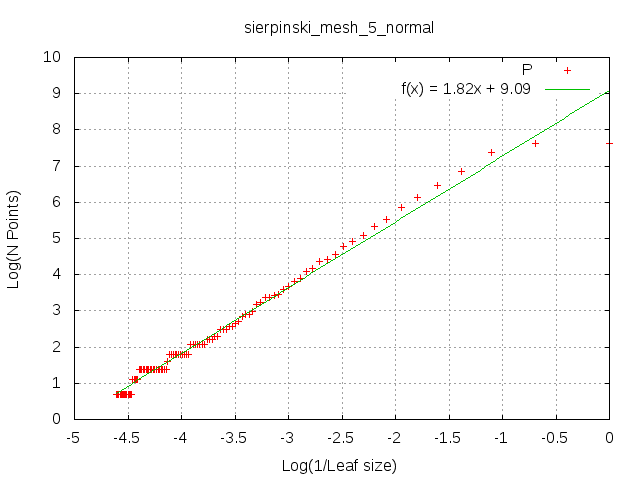
**Experimentación**

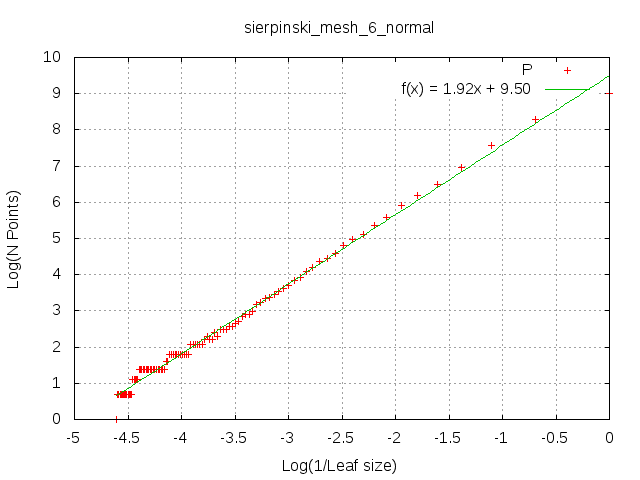
A continuación se muestran las gráficas de los resultados obtenidos con esta versión inicial del algoritmo para el tetraedro de Sierpinski, que recordemos, tiene una dimensión fractal teórica igual a 2. Los resultados mostrados corresponden a la misma figura, pero generada con diferentes iteraciones, de 1 a 9 (el número de iteraciones de la figura correspondiente se indica en el nombre de cada gráfica), con un tamaño de la figura constante para cada iteración.

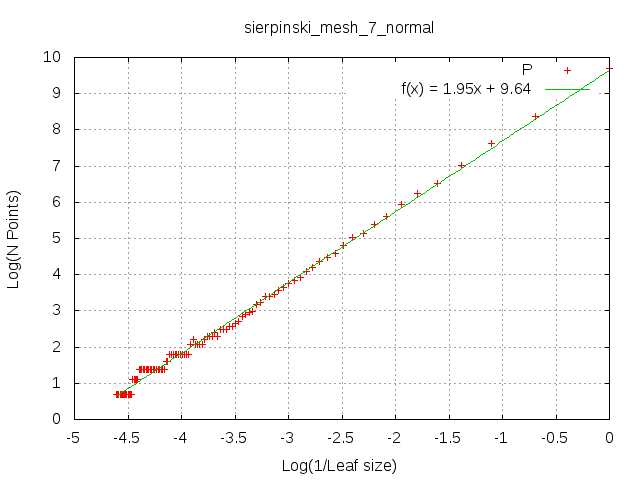


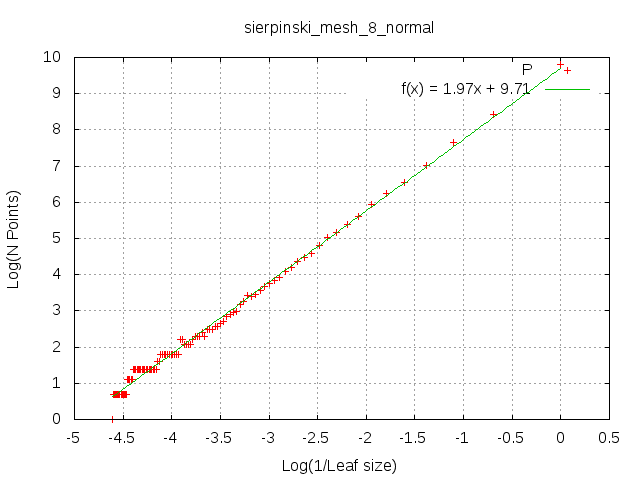


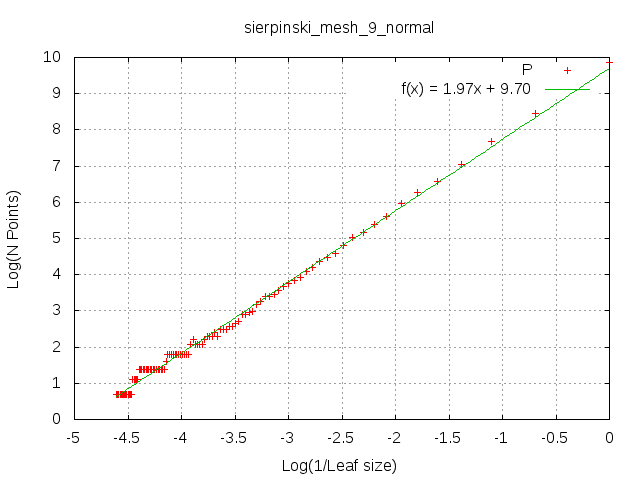


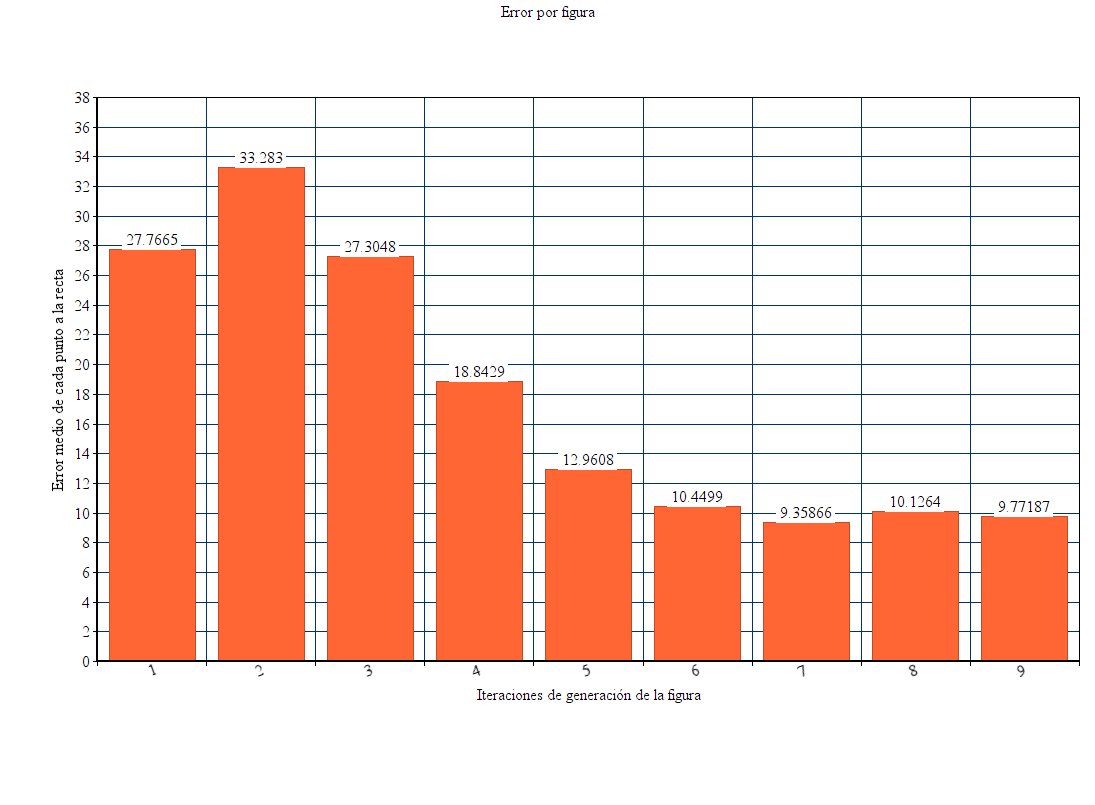








****

**Conclusiones obtenidas**

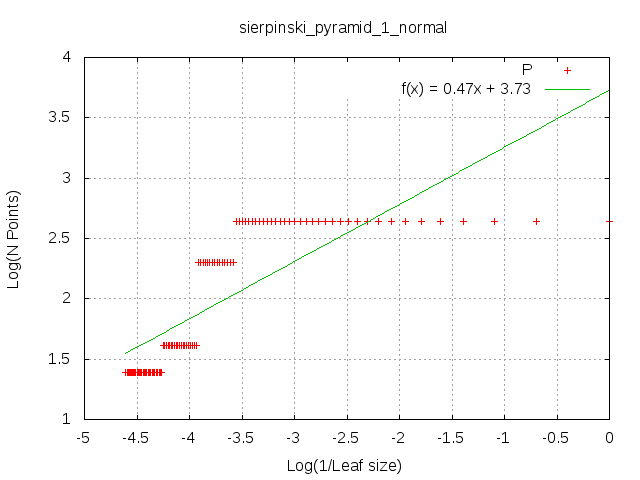
Para una fácil lectura, a continuación se enumeran las conclusiones obtenidas de esta primera experimentación:

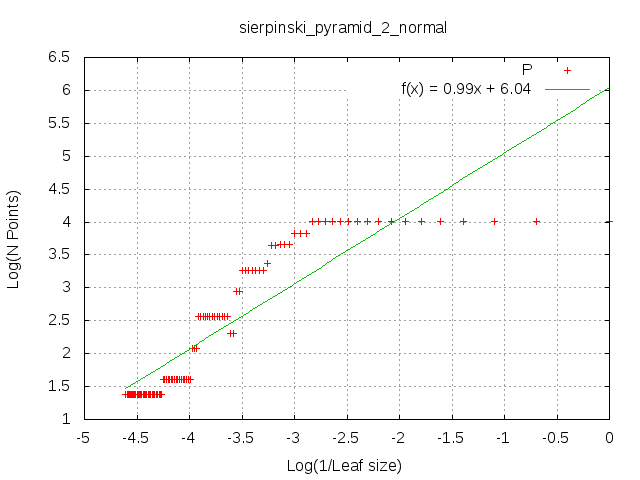
* Las figuras generadas con menos iteraciones (1, 2 y 3) contienen menos puntos por lo que es mas difícil obtener datos significativos realizando el mismo número de iteraciones que para las figuras con una mayor cantidad de puntos. Además, contienen menos repeticiones del patrón de la estructura por lo que es más difícil de calcular. Teóricamente, con estas figuras, realizando una generación a partir de la 5-6 iteración se alcanza el punto crítico en el que ya no se puede apreciar diferencias a medida que se realizan más iteraciones. Por estos motivos, en siguientes experimentaciones no se va a dar ninguna atención a resultados para figuras generadas con menos de 4 iteraciones.

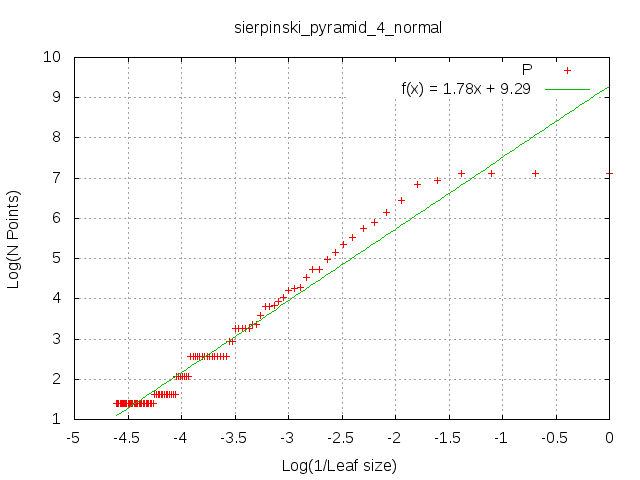
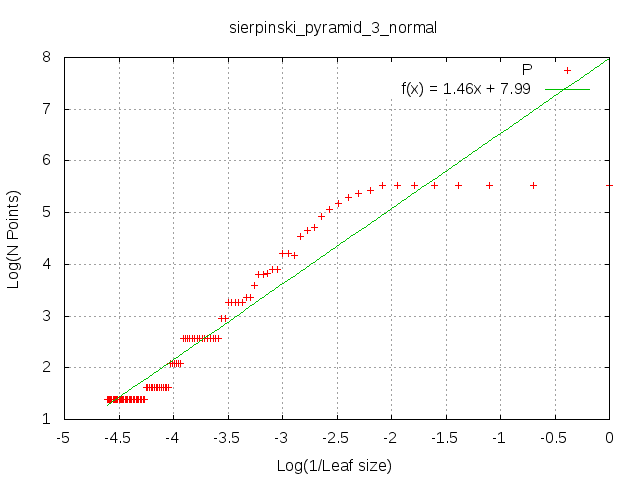
Esto también abre la cuestión de si tiene sentido especificar el número de iteraciones por parte del “usuario” en lugar de que el algoritmo sea capaz de decidir las iteraciones que va a realizar en función de la figura, pero de momento este punto se va a dejar para más adelante.

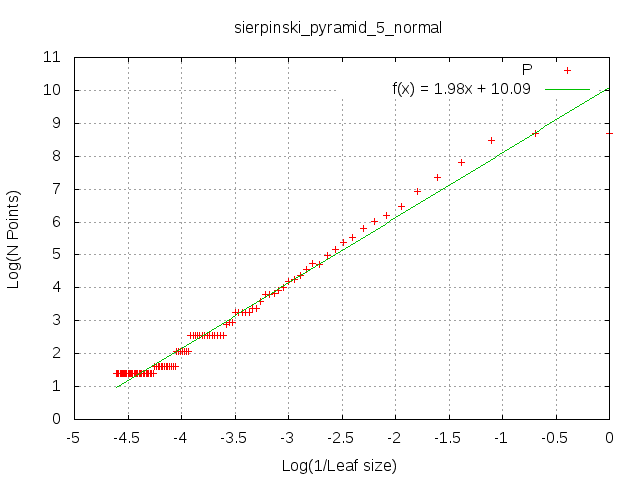
* Conforme se van obteniendo los resultados para las figuras con mayores iteraciones, se observa que los valores de la dimensión fractal empiezan a converger entre si, aproximándose cada vez más entre el resultado de una figura a otra. También se va estabiliza el error cometido en la regresión lineal.
* Con esta versión del algoritmo, se obtienen resultados más aproximados a el resultado teórico (D = 2) a medida que más veces existe el patrón estructural en la figura, además de que el error sobre la recta de regresión va disminuyendo.
* Los puntos x,y no se alinean de una manera muy precisa con la recta de regresión calculada. Se aprecia que muchos de los puntos si que siguen una tendencia pero hay muestras que crean ruido en el calculo de la regresión lineal, y por tanto no se obtiene la pendiente que marcaría la dimensión fractal correcta.
* El algoritmo parece válido ya que los resultados obtenidos con las figuras mas iteradas para la dimensión fractal se aproximan bastante bien a la dimensión teórica, pero se debe ajustar mejor ya que las gráficas muestran que existe demasiado error en las muestras a la hora de calcular la regresión lineal.
* La variación del tamaño del vóxel en cada iteración del algoritmo parece afectar, sobre todo, en algunos intervalos de la gráfica. Esto tiene una relación con la distribución de los puntos de la nube.
* Para unas figuras, los tamaños de vóxel iniciales (más a la derecha en la gráfica) provocan ruido y para otras ocurre para los tamaños finales de voxel (más a la izquierda en la gráfica). De aquí sale la siguiente conclusión.
* El tamaño de vóxel inicial y el final afectan de una manera u otra en función de la figura que se está analizando.

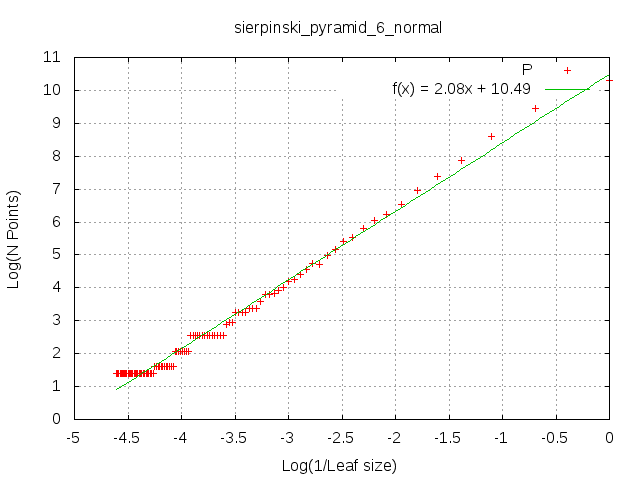
Se muestran a continuación los resultados para la pirámide de Sierpinski para ver si las conclusiones extraídas del análisis del tetraedro también son válidas el caso de la pirámide, cuya dimensión fractal, recordemos, es igual a 2.3219:

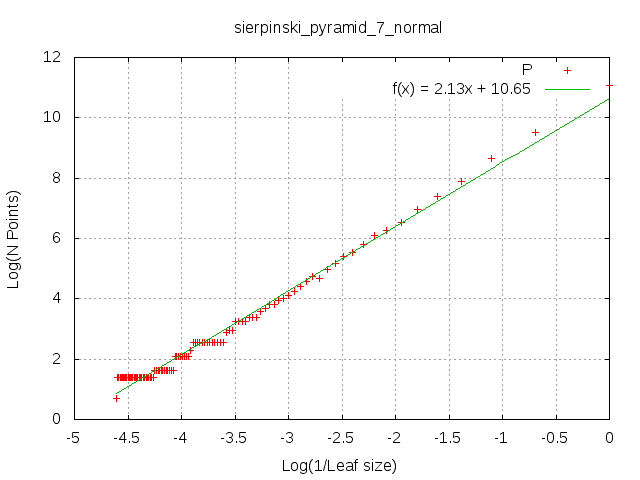


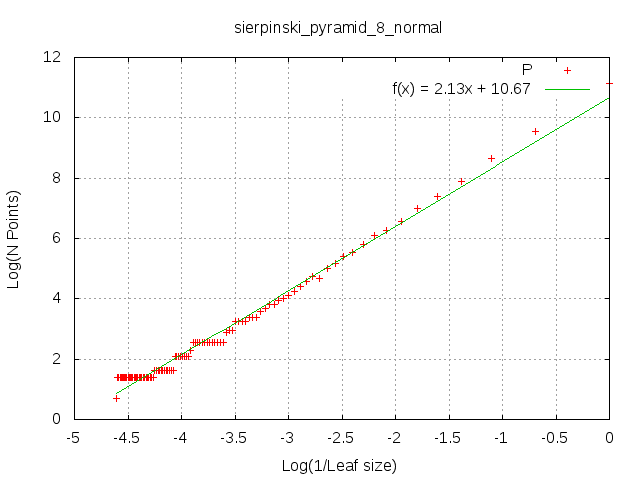


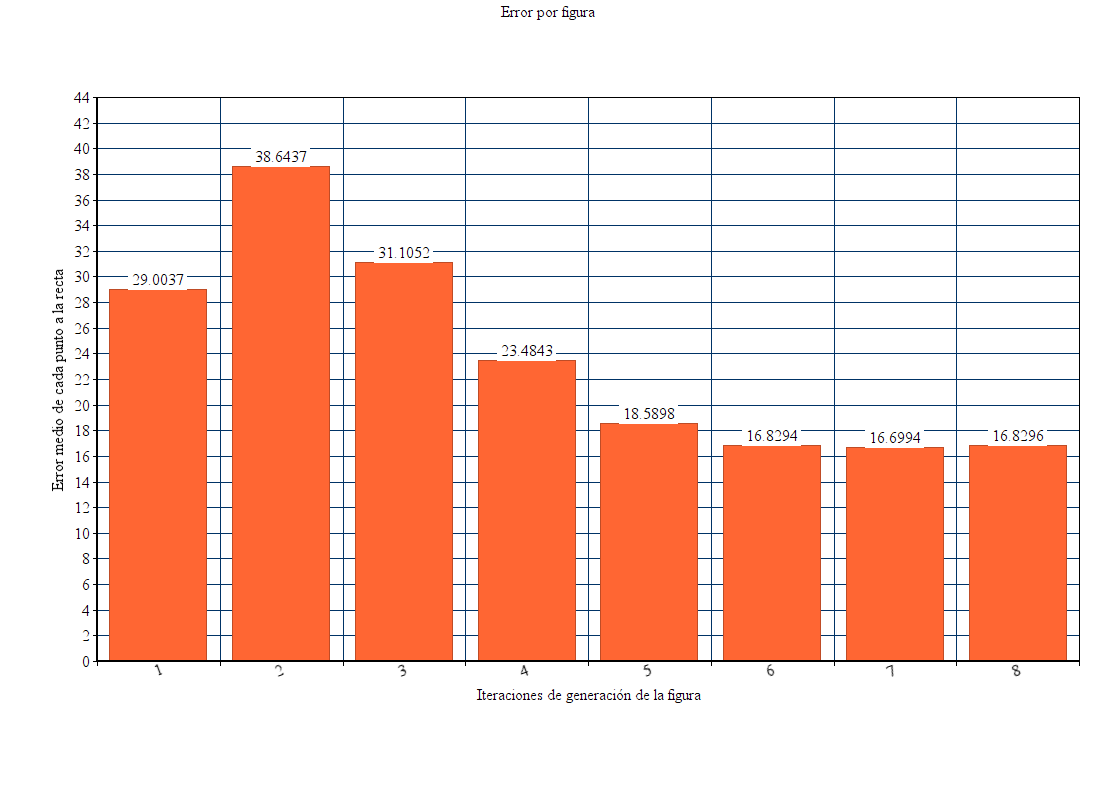




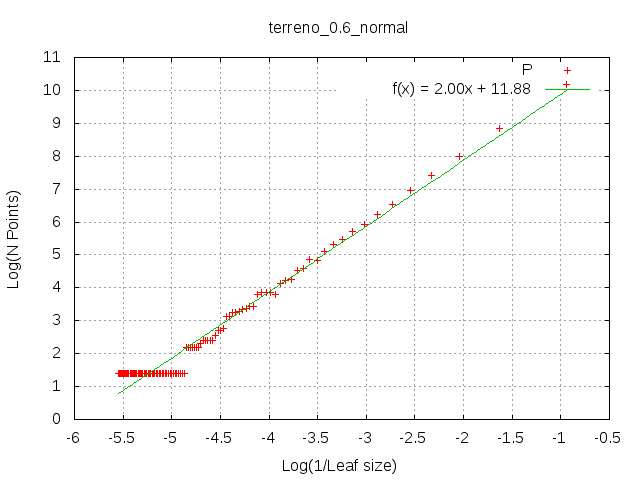


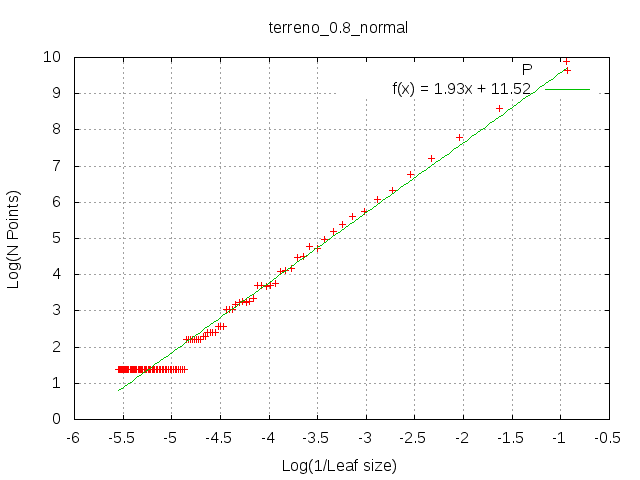


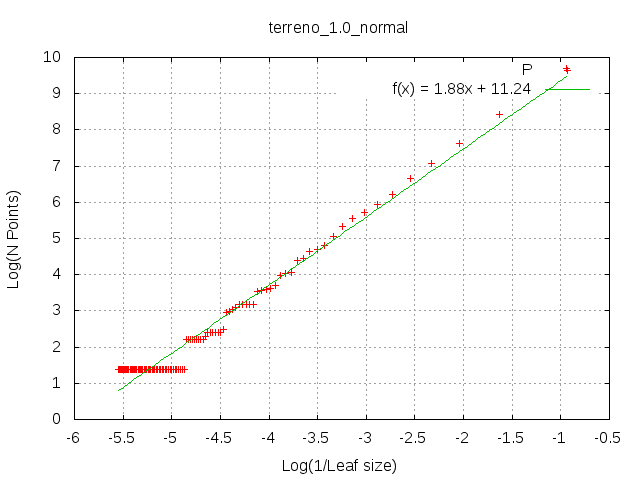


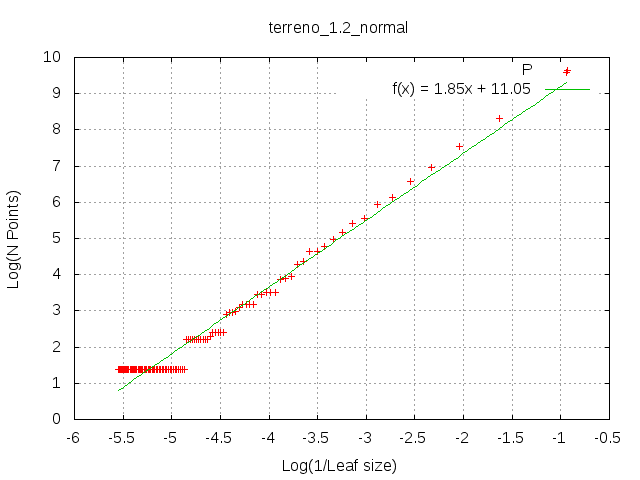
Efectivamente, tanto las gráficas de resultados como la del error son muy similares para el tetraedro y la pirámide de Sierpinski, con la diferencia de que en las gráficas de la pirámide podemos ver un aumento del ruido en las muestras de los tamaños de vóxel finales (más cercanos al tamaño de la figura original), lo que refuerza alguna de las hipótesis concluidas anteriormente como que el tamaño del vóxel, su variación en cada iteración y el número de iteraciones a realizar deben estar en función de la figura que se está analizando en cada momento.

Veamos ahora que resultados se producen al procesar algunos de los terrenos fractales (ordenados de mas rugoso a más suave):



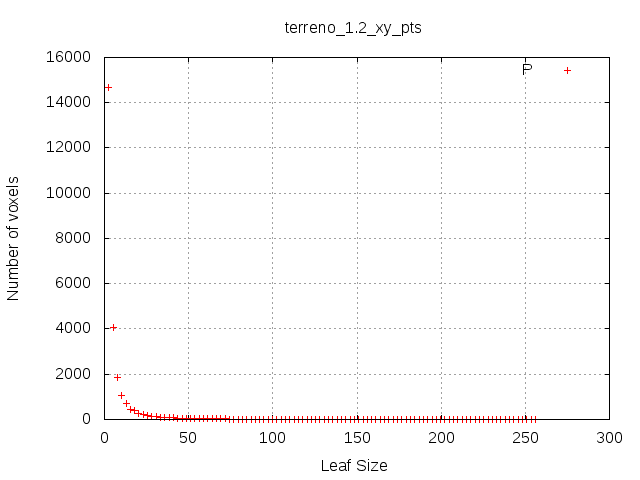






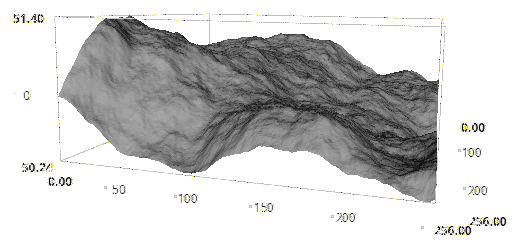
En este caso vemos más claramente como una gran parte de las iteraciones realizadas por el algoritmo (las de mayor tamaño de vóxel) son contraproducentes ya que no aportan información y afectan negativamente al calculo de la recta de regresión.

Es fácil darse cuenta que cuando el tamaño de los vóxeles es mayor a la variable calculada *tamaño\_max/2*, no se puede captar prácticamente información ya que habrá muy poca variación en los resultados de las siguientes iteraciones. Una gráfica que puede ayudarnos a entender esto mejor es la siguiente:



Aquí vemos en el eje y el número de vóxeles que contienen algún punto de la nube y en el eje x el tamaño del vóxel. Se puede ver que en las primeras iteraciones se está perdiendo la mayor parte de los puntos que forman el objeto, en este caso el terreno, por lo tanto en el resto de las iteraciones no obtenemos casi información ya que lo que buscamos es captar la variabilidad de los resultados en las diferentes iteraciones, y cuando no hay variabilidad, se está introduciendo error.

También existe un problema con objetos de este tipo ya que los puntos que forman el terreno se distribuyen mayormente a lo largo de dos de los tres ejes de coordenadas, como vemos en la siguiente imagen, el tamaño en los ejes x y z (256) es cinco veces mayor que el tamaño en el eje y, 51.4:



Esto implica que cuando se están usando vóxeles de tamaño mayor a 51.4 solo se capta información en los ejes x y z.

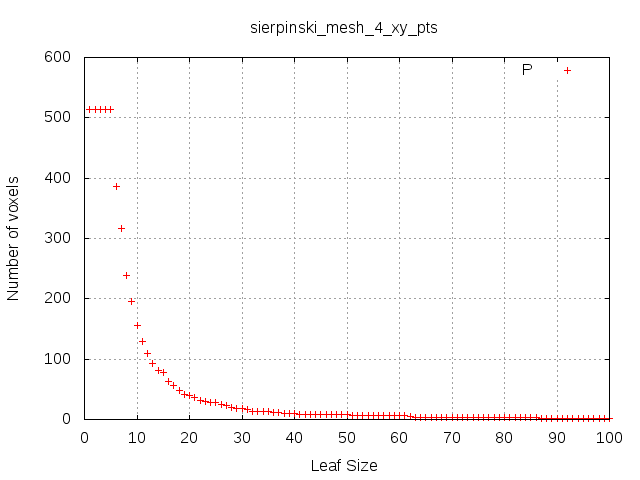
**FASE 2**

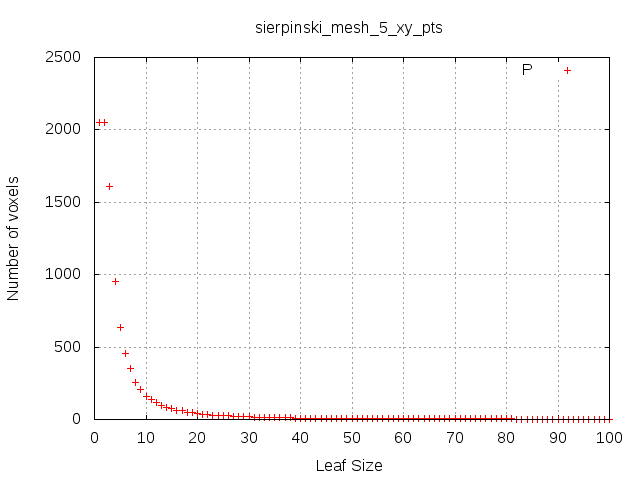
**Problemas a resolver**

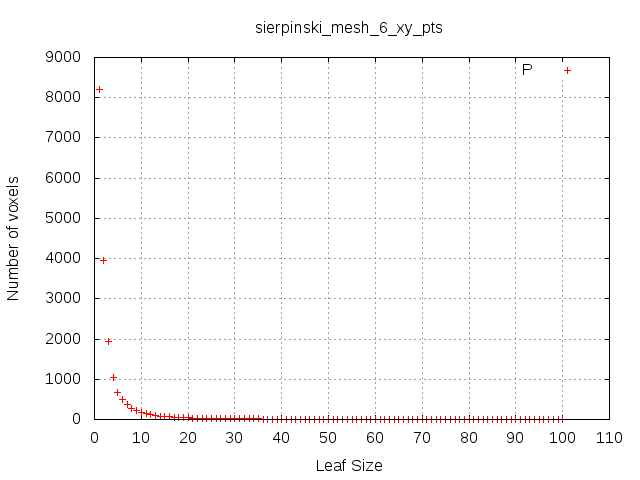
En esta fase se va a intentar dar una solución para obtener mayor variabilidad de los resultados en las iteraciones con tamaño de vóxeles mayores.

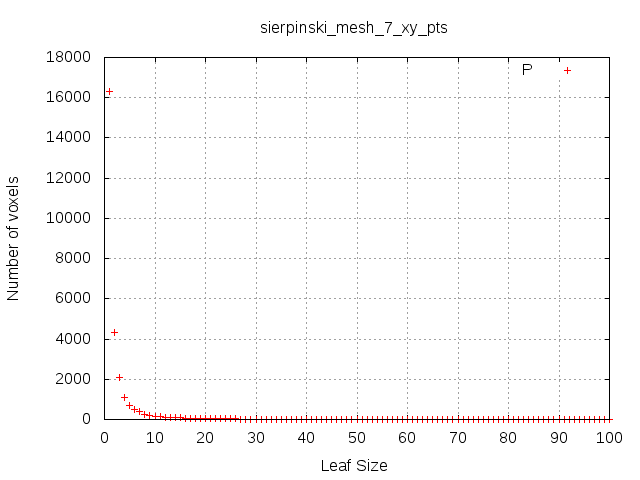
Vamos a comprobar también con otras figuras si tenemos el mismo problema de la perdida de información al realizar las primeras iteraciones.

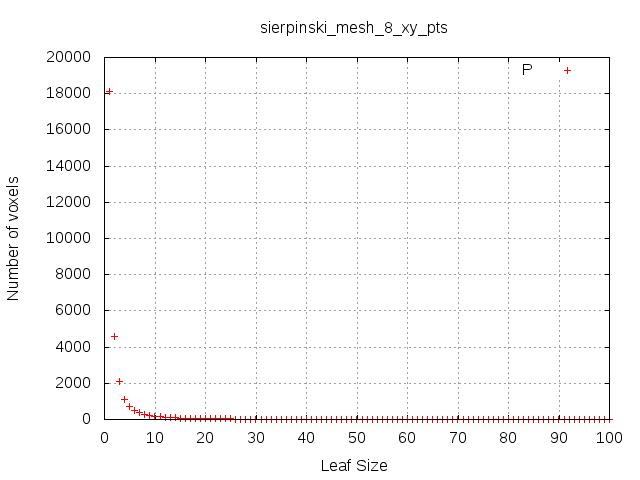
Tetraedro de Sierpinski

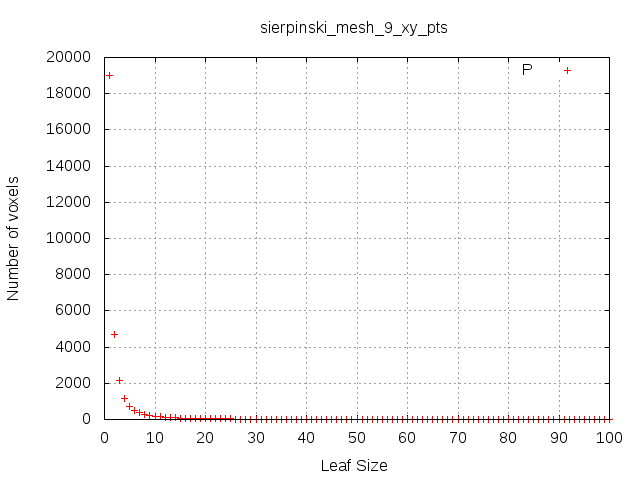










Debido a que las diferentes nubes de puntos que representan el tetraedro de Sierpinski que estamos analizando tienen todas el mismo tamaño, pero cada una tiene más detalle que el anterior, lo que nos está diciendo este grupo de gráficas es que si utilizamos solo el tamaño del objeto para decidir el tamaño de los vóxeles para cada iteración, estaremos perdiendo mucha información en las figuras que tienen más detalle, y es precisamente lo contrario que queremos, queremos captar el patrón con el máximo detalle posible.

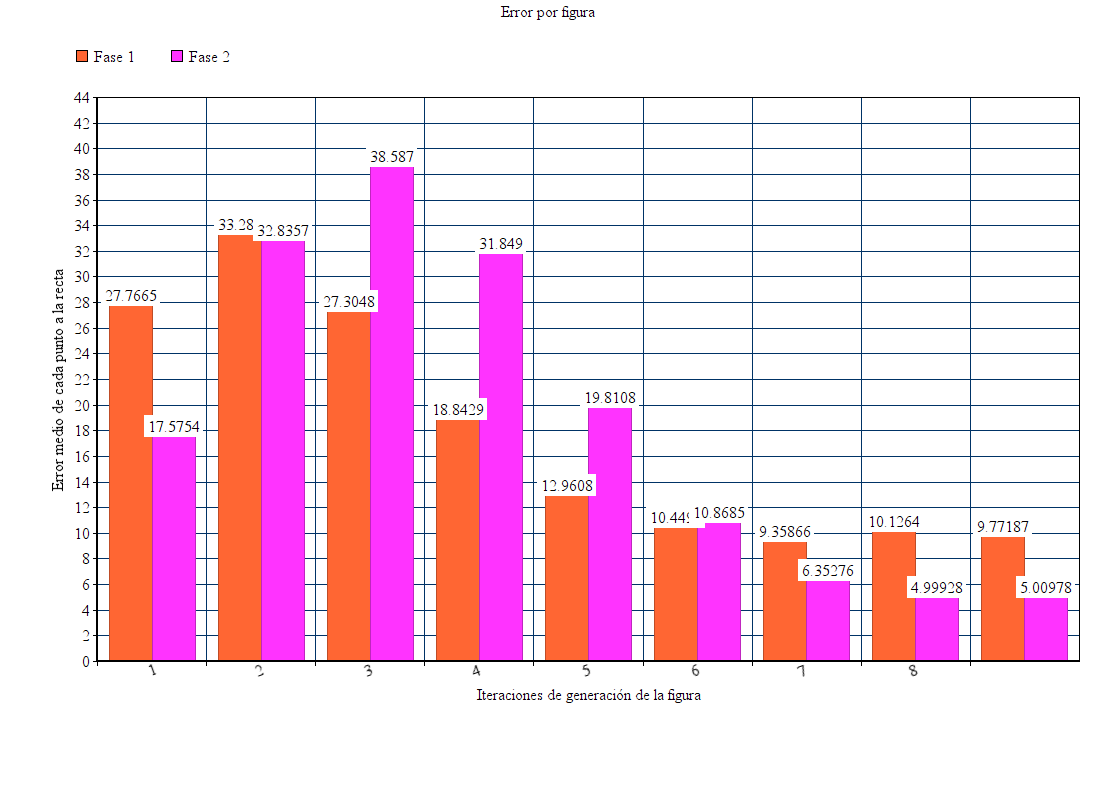
También evidencian que iterar hasta el tamaño máximo de la figura no aporta más información si no que es contraproducente.

**Modificaciones de la fase 2**

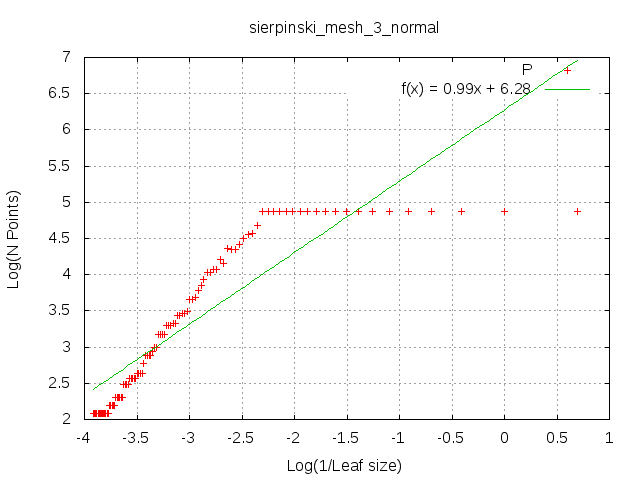
Esta segunda fase se va a restringir el tamaño máximo de los vóxeles, es decir, el tamaño en la última iteración, a la mitad del tamaño máximo de la figura y ver si obtenemos mayor variabilidad en las diferentes iteraciones.

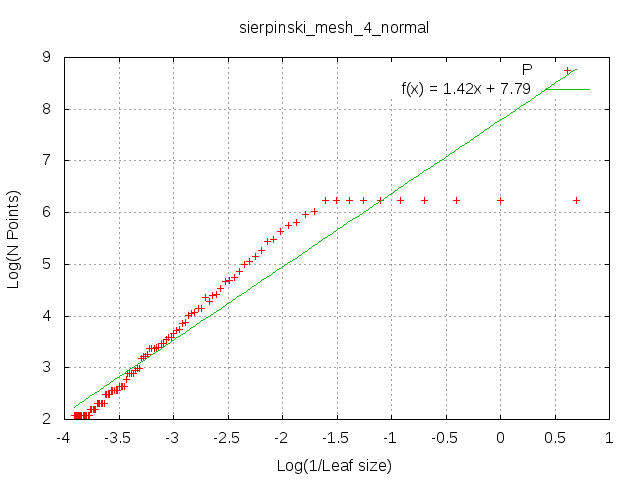
**Experimentación**

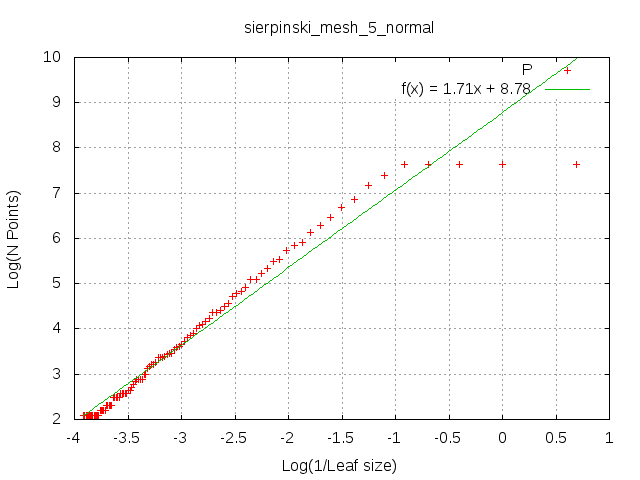
Empezamos con el tetraedro de Sierpinski, haciendo una comparativa del error obtenido en la regresión lineal entre la fase 1 y la fase 2:

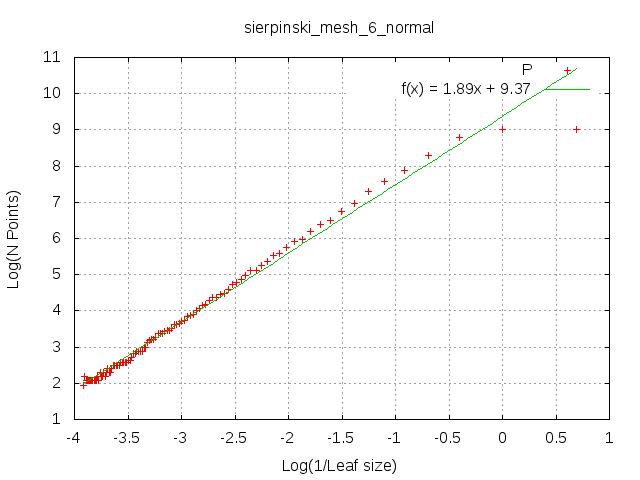


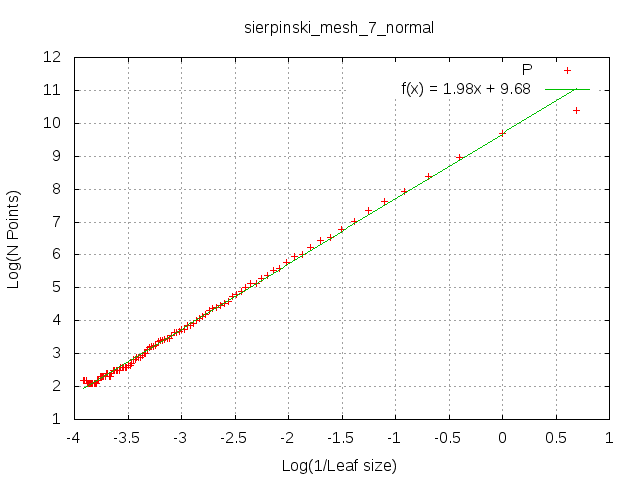
Se puede observar como el error ha aumentado de forma considerable con respecto a la fase 1 en las figuras 3, 4 y 5. En la figura 6 se obtienen errores similares, y después para las figuras 7, 8 y 9 con mayor nivel de detalle empieza a verse una mejora. Veamos las gráficas de regresión lineal de las figuras mencionadas:

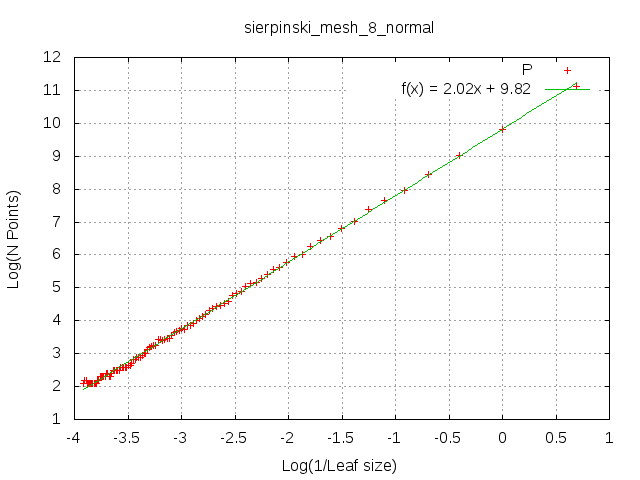


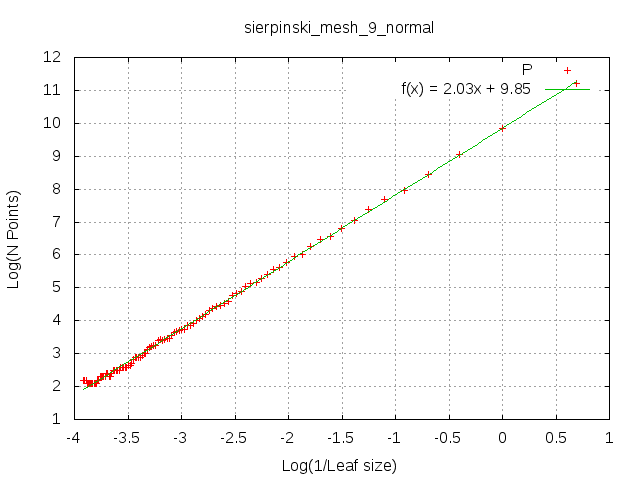


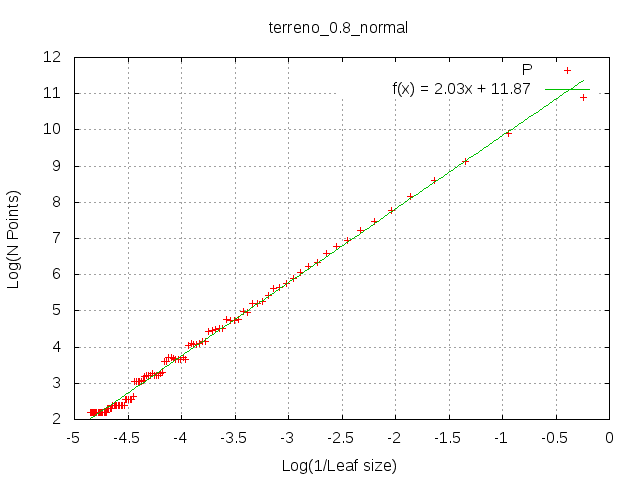


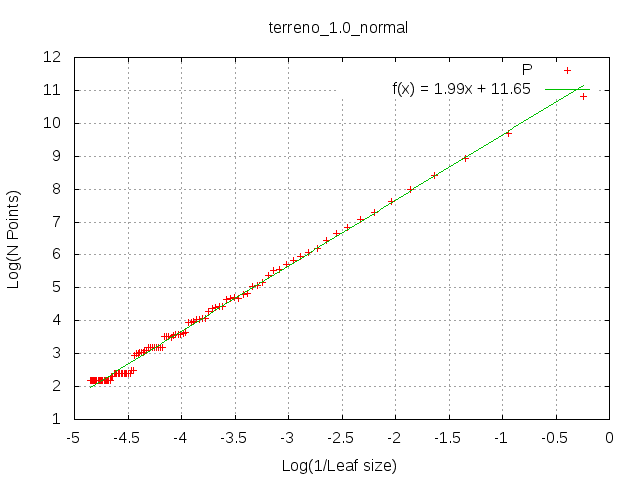


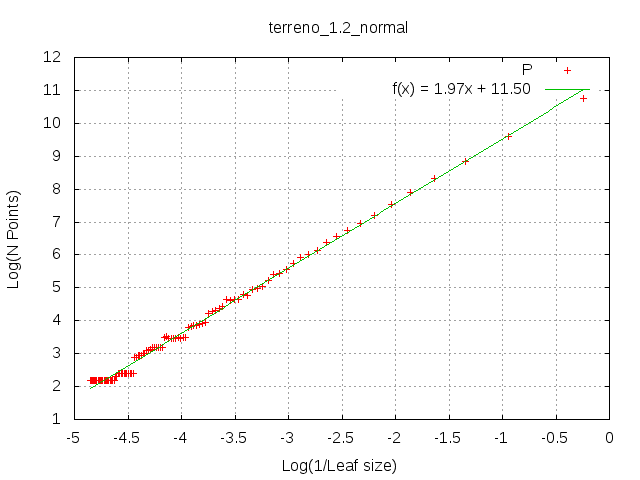




Se muestran también varias gráficas correspondientes a los terrenos:







**Conclusiones obtenidas**

A continuación se enumeran las conclusiones obtenidas de esta experimentación:

* Al limitar el tamaño máximo de voxel a *tamaño\_max*/2 podemos ver que efectivamente los resultados de las últimas iteraciones (mas a la izquierda en la gráfica) muestran mayor variabilidad que en la fase 1.

En el caso de los terrenos, se ha eliminado por completo un grupo de puntos que se alineaban de forma horizontal en los tamaños más grandes de vóxel en el caso de la fase 1, y se aprecia una mejor alineación de los puntos sobre la recta de regresión calculada.

* Al mismo tiempo que estamos obteniendo mejores resultados en las iteraciones con tamaños más grandes, estamos introduciendo error en las que tienen tamaños más pequeños en el caso de figuras con menos densidad o detalle del tetraedro. Esto se debe a que al calcular el tamaño inicial de vóxel y el incremento en cada iteración en función al tamaño máximo, en esta fase dos tendríamos tamaños iniciales e incrementos más pequeños para una misma figura, por lo que parece que se están haciendo las divisiones en vóxeles a un tamaño más pequeño que la resolución del objeto, por lo que no se obtiene variación en los resultados de las primeras iteraciones.
* Aunque los resultados obtenidos sigan sin ser ideales, parece una buena decisión el limitar el tamaño máximo de vóxeles, aunque será necesario tomar otras decisiones para seguir ajustando el algoritmo.
* Parece también que sería buena idea ajustar los tamaños más pequeños de vóxel en función de la densidad de la nube de puntos que se está procesando.

**FASE 3**

**Problemas a resolver**

El objetivo de esta fase es similar a la fase dos, pero en este caso se pretende ajustar el tamaño inicial de los vóxeles a un tamaño mínimo, que estará en función de la densidad del objeto, y así solventar el error que se está introduciendo en los resultados al realizar iteraciones del algoritmo con tamaños demasiado pequeños.

Se va a mantener la modificación de la fase 2 ya que, aunque hay que seguir ajustando, ofrece mejoras frente a la fase 1 y parece de sentido común.

**Modificaciones de la fase 3**

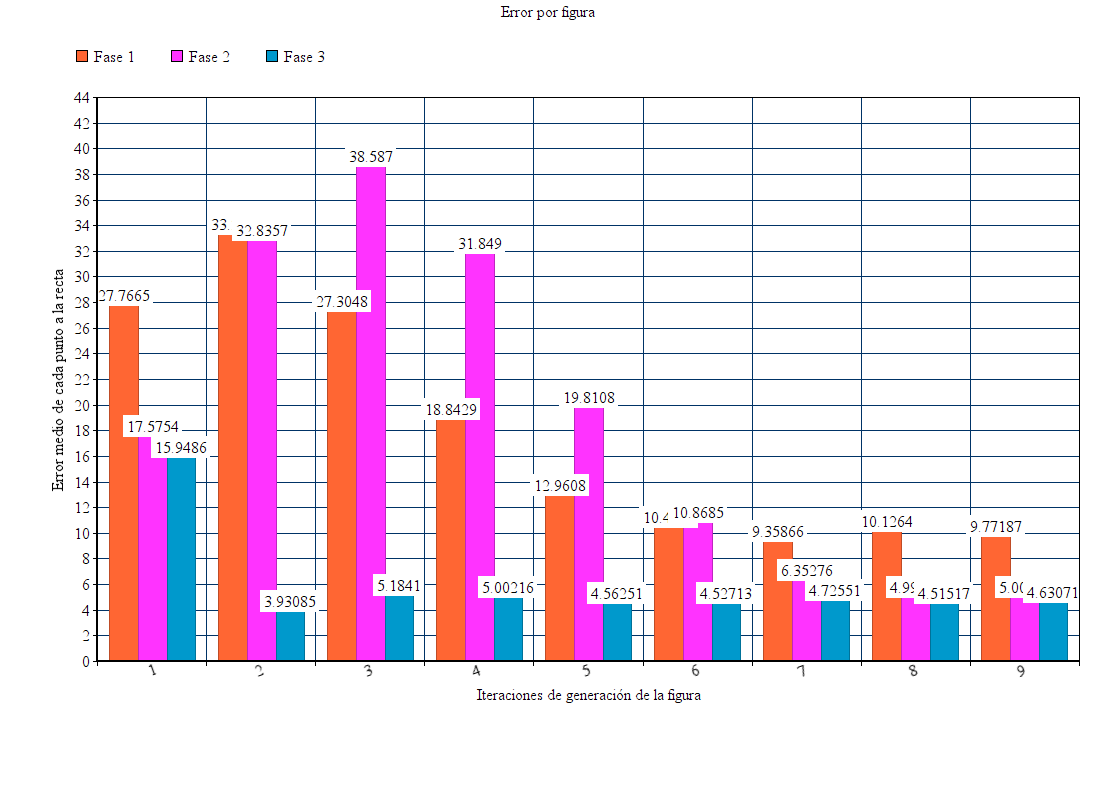
En el caso de las nubes de puntos 3D, y en lo que afecta al algoritmo que se está implementando, podríamos utilizar como medida de resolución máxima del objeto la distancia entre los puntos que componen su superficie.

Una forma de no utilizar tamaños demasiado pequeños es restringir el tamaño inicial del algoritmo en función de la media de distancias entre vecinos más cercano, esto es, el sumatorio de la distancia de cada punto a su vecino mas cercano y dividido entre el número de puntos que contiene la nube. Como tampoco tiene sentido llegar a un tamaño tan pequeño, se va a usar el doble de este valor como tamaño inicial en esta fase, y el incremento por cada iteración será calculado como:

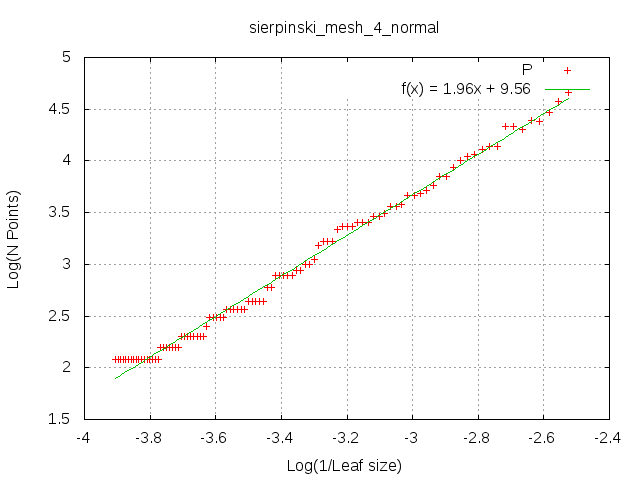
*(tamaño\_final – tamaño\_inicial) / iteraciones*

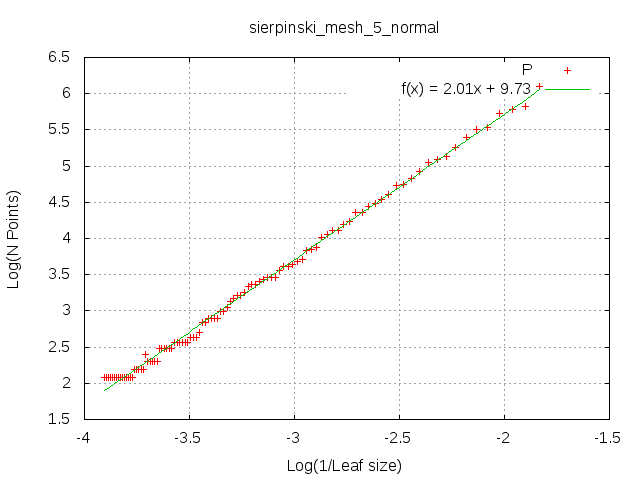
**Experimentación**

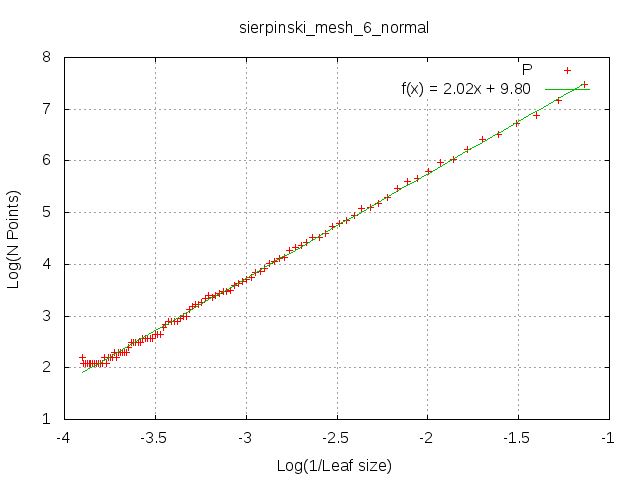
Empezamos igual que en la fase 2 haciendo una comparativa del error en la regresión lineal en las distintas fases de experimentación:

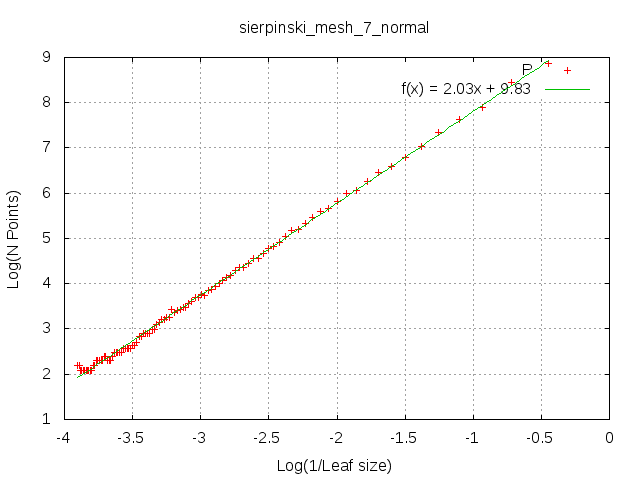


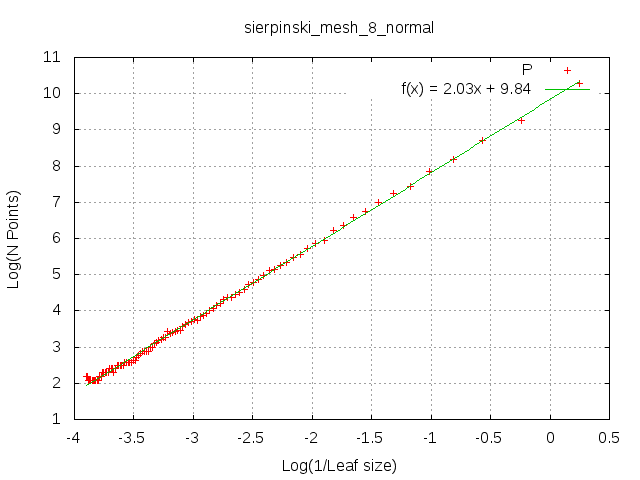
En esta gráfica podemos ver que el cambio realizado mejora muy considerablemente los resultados de la regresión lineal en cuanto al error obtenido. Veamos como son las gráficas generadas con los resultados de la fase 3 para el tetraedro de Sierpinski:

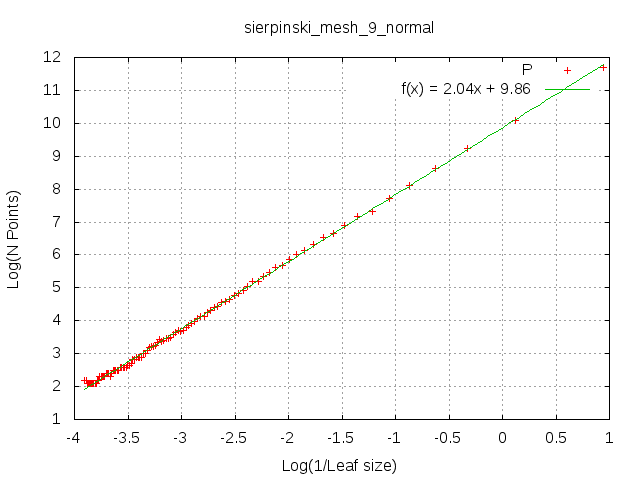




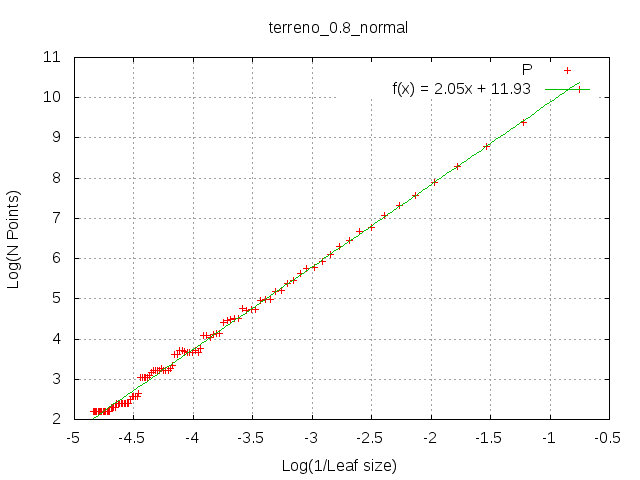


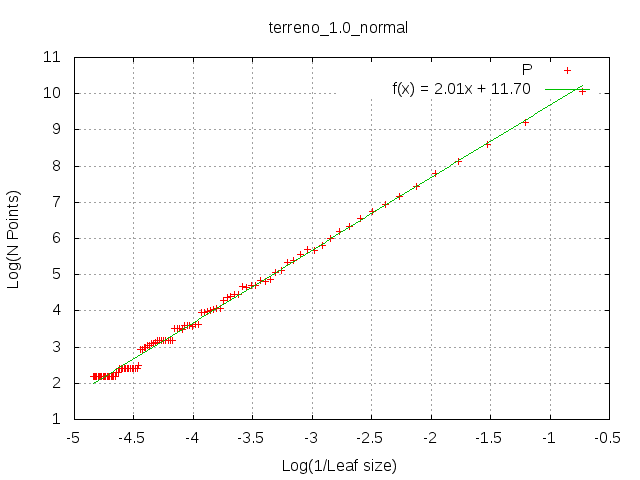


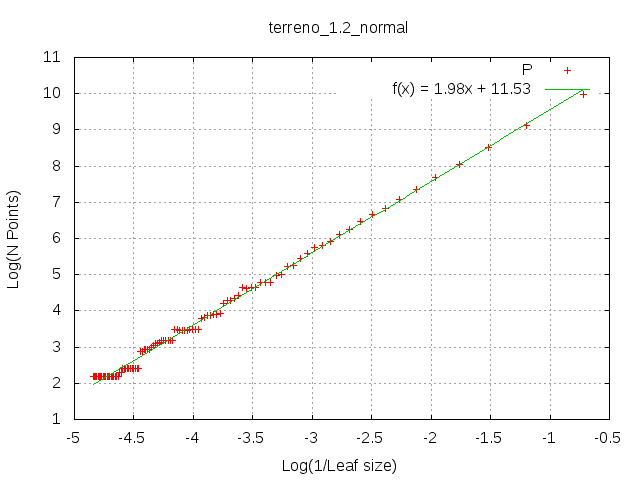




Se muestran también resultados obtenidos sobre los terrenos fractales:







**Conclusiones obtenidas**

A continuación se enumeran las conclusiones obtenidas de esta fase:

* En cuanto a el error producido en la regresión lineal con los resultados del algoritmo, tenemos una mejora en todos los casos del tetraedro de Sierpinski, en especial para el caso de los que contienen menos repeticiones del patrón geométrico que define la figura, exactamente como se pretendía hacer.

Si hacemos una comparativa de las gráficas de regresión lineal en la fase 2 y la fase 3 para una misma figura, se puede perfectamente apreciar por que se está reduciendo tanto el error, y es que en la fase dos los puntos correspondientes a las primeras iteraciones están provocando una inclinación muy fuerte sobre la recta de regresión. Cuando no tenemos esos puntos, la recta puede ajustarse mucho mejor a el conjunto de puntos de forma global y de esta forma reducir tanto el error.

* Podemos ver que esa mejora se ve también reflejada en una mejor aproximación de la dimensión fractal calculada respecto a la dimensión fractal teórica de nuestro objeto, y por lo tanto, gracias a este cambio el algoritmo da mejores resultados y esta bastante más ajustado que en fases anteriores.
* El partir de un tamaño de vóxel inicial mayor hace que el incremento en cada iteración sea diferente a las anteriores fases, lo que también esta afectando a los resultados
* En el caso de los terrenos, no se ven apenas diferencias ya que estos no presentaban el problema que se quería solventar en esta fase.
* El algoritmo puede mejorar o empeorar mucho en función de como se elijan los tamaños de vóxel inicial y final. Hemos visto que estos parámetros tienen mucho peso sobre los resultados que se producirán, por lo que no es descabellado pensar que el incremento realizado en cada iteración también pueda ser determinante.
* Aunque con mejoras en los resultados, seguimos viendo que hay parte de los puntos que no se acaban de alinear de forma suave sino que tienen cierta forma escalonada. Esto son iteraciones que nos están introduciendo error.
* En este punto se pueden barajar distintas opciones. Pensar una manera de incrementar el tamaño de vóxel en cada iteración distinta a como se hace actualmente, o intentar utilizar un método de eliminación de ruido o métodos de suavizado que nos proporcionen un mejor conjunto de puntos sobre el que calcular la recta de regresión.

**FASE 4**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

**FASE X**

**Problemas a resolver**

**Modificaciones de la fase 3**

**Experimentación**

**Conclusiones**

8.- Experimentación

**Primeros pasos**

<http://www.dma.fi.upm.es/recursos/aplicaciones/geometria_fractal/proyectos/movimiento_browniano/geometriafractal.htm>

<https://es.wikipedia.org/wiki/Fractal>

<http://casanchi.com/mat/03_gfractal01.pdf>

tesis: <http://eprints.uanl.mx/377/1/1020114994.PDF>

La Dimensión Fractal:

<http://catarina.udlap.mx/u_dl_a/tales/documentos/lme/ojeda_s_r/capitulo4.pdf>

Longitud y Área de Curvas Fractales. Dimensión Fractal: <http://www.dma.ulpgc.es/profesores/personal/aph/ficheros/resolver/ficheros/fractales.pdf>

Fractales: la frontera entre el arte y las matemáticas – Universidad Jaen, muy bueno

http://matema.ujaen.es/jnavas/web\_recursos/archivos/fractales%20datos/Modulo%205-fractales\_%20jnavas.pdf

CAOS, FRACTALES Y COSAS RARAS:

http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen3/ciencia3/150/htm/sec\_7.htm

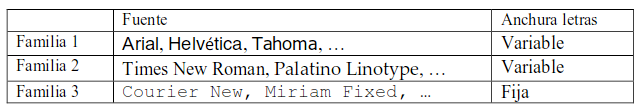
Imagenes:

<http://es.gizmodo.com/diez-bellisimos-ejemplos-de-fractales-en-la-naturaleza-1677114869>

<http://paulbourke.net/fractals/gasket/>

Box counting:

http://www.wahl.org/fe/HTML\_version/link/FE4W/c4.htm



NORMA UNE 50135:1996

* Parte inicial
  + portada
  + resumen
  + indice
  + glosario de signos, simbolos, unidades, abreviaturas, acronimos, …
  + prefacio, si es necesario
* Cuerpo del informe
  + Introduccion
    - Enfoque fractal
    - Tecnicas y algoritmos
  + Marco teórico o Estado del arte
  + Objetivos
  + Metodología - Herramientas utilizadas
  + Nucleo del informe con ilustraciones esenciales y tablas
    - Aplicaciones
* Conclusiones y recomendaciones
* Agradecimientos..
* Bibliografia y Referencias
* Anexos

Contenido:

* Justificación y objetivos
* Introduccion
* Estado del arte
* Objetivos
* D
* Box Counting
  + Primeras experimentaciones