CODIFICACIÓN FRACTAL DE IMÁGENES

Juan Antonio Pérez Ortiz

septiembre 1997 – julio 1998

Prólogo

La geometría fractal, cuyos primeros desarrollos datan de finales del siglo pasado, ha recibido durante los últimos veinte años, desde la publicación de los trabajos de Mandelbrot, una atención y un auge crecientes. Lejos de ser simplemente una herramienta de generación de impresionantes paisajes virtuales, la geometría fractal viene avalada por la teoría geométrica de la medida y por innumerables aplicaciones en ciencias tan dispares como la Física, la Química, la Economía o, incluso, la Informática.

Dentro de esta corriente, la teoría matemática denominada *sistemas de funciones iteradas*, desarrollada en 1981 por Hutchinson, se convirtió a finales de la década de los 80 con los trabajos de Barnsley en una de las técnicas más innovadoras y prometedoras en el campo de la compresión de imágenes. Aunque las expectativas iniciales fueron de alguna manera exageradas (situación ésta que también sobreestimó en sus inicios a técnicas como la lógica difusa o las redes neuronales), el paso de los años ha ido configurando una teoría factible que comienza ya a dar sus primeros productos comerciales.

Aunque el principal cometido de este trabajo es conocer el estado actual de la codificación fractal y los principios matemáticos que la configuran, no por ello deja de ser también una revisión detallada de los fundamentos de la geometría fractal.

Al mundo de los fractales muchos llegan (el autor incluido) atraidos por el estallido de color de alguna representación del conjunto de Mandelbrot o de un conjunto de Julia. Sin embargo, una vez que se profundiza en la magia de los fractales, uno no sabe que admirar más, si las cascadas multicolor o la belleza de las matemáticas que las engendran. En esta obra los turistas que se asomen por primera vez a este mundo encontrarán una introducción desde cero a los principios elementales de la revolución fractal.

Hay también quienes disfrutan desde hace ya tiempo generando intrincados bosques fractales o fotografías a profundidades abismales tras iterar fórmulas sencillas mediante programas confeccionados, incluso, por ellos mismos. Algunos serán los geómetras del siglo XXI armados esta vez de ordena-

dores cada vez más rápidos que, aun así, cuando se trata de fractales, siempre parecen quedarse pequeños. Con todo, y quizá debido al nivel superficial con el que muchas publicaciones divulgativas afrontan el tema, muchas veces se desconoce la teoría matemática que aguarda tras cada fractal. Conceptos como los de dimensión de Hausdorff o conjunto autosemejante son de vital importancia para abordar con ciertas garantías de éxito la exploración de nuevos continentes fractales y se mostrarán aquí detalladamente.

Por último, hay quienes pueden acercarse a este proyecto para ampliar sus conocimientos sobre la compresión de imágenes con pérdidas. En realidad, la primera parte de la obra trata de crear un clima adecuado para poder abordar la codificación fractal de imágenes, una de las tecnologías de compresión más en boga en los últimos años. Aunque el mayor peso recae sobre el esquema de compresión fractal, no quedan sin analizar con cierto detalle otros enfoques alternativos, principalmente los basados en wavelets y en la transformada discreta del coseno.

Contenido de la obra

Los requisitos para acceder a esta obra son mínimos ya que basta con que el lector conozca suficientemente algunas de las herramientas proporcionadas por un curso inicial de cálculo. Aun así, se presentan cuando es necesario la mayor parte de los conceptos utilizados y cuando esto no es posible, por requerir una gran cantidad de información, se remite al lector a las referencias oportunas. De cualquier forma lo anterior ocurre con aspectos nunca básicos del trabajo cuya no completa asimilación no debe repercutir en la comprensión del resto de la obra. Se ha tratado de realizar un trabajo lo más autocontenido posible.

El capítulo 1 presenta una introducción a las ideas básicas de la geometría fractal. Se realiza allí una revisión histórica del estudio de los conjuntos fractales a la vez que se presenta el comportamiento de los sistemas caóticos, primos hermanos de los fractales. Como elementos ineludibles se presentan la constante de Feigenbaum, los conjuntos de Julia y el conjunto de Mandelbrot.

En el segundo capítulo se realiza un primer acercamiento serio al concepto de autosemejanza, crucial en la geometría fractal. Este acercamiento se realiza de la mano de uno de los enfoques estructurales más elegantes con los que describir los fractales, los sistemas L.

El tercer capítulo muestra la teoría de conjuntos autosemejantes de Hutchinson, quizá la más consolidada hoy día. Antes de afrontarla se describen

todos los conceptos fundamentales de la teoría de espacios métricos en los que se sustenta, especialmente el teorema del punto fijo.

Los sistemas de funciones iteradas del capítulo 4 son la base de las técnicas actuales de compresión fractal. Estos sistemas generalizan la concepción de autosemejanza del capítulo anterior, constituyendo la herramienta básica para la aproximación mediante fractales de figuras reales. El teorema del collage, como culminación del capítulo, asegura que bajo ciertas condiciones esto es posible.

Antes de utilizar los fractales para la compresión de imágenes, el capítulo 5 estudia los conceptos generales de tal compresión a la vez que presenta esquemas alternativos como la cuantización vectorial o los basados en transformadas como la del coseno o la transformada con wavelets.

Por fin es posible abordar con seguridad la compresión fractal de imágenes, denominada también transformada fractal. Esto se lleva a cabo en el capítulo 6 donde se presenta un esquema básico, de fácil comprensión, pero de dudosa eficiencia.

El capítulo 7, el último, estudia alguna de las posibles mejoras que pueden llevarse a cabo sobre la técnica básica del capítulo 6 para hacer factible su implementación. Aquí la oferta es amplísima y a falta de determinar qué alternativas son las más adecuadas, se presentan las más interesantes propuestas durante los últimos años.

Algunos aspectos secundarios del trabajo, pero no por ello menos interesantes, se han relegado a los apéndices. El apéndice A aborda la medida de Lebesgue y la dimensión de Hausdorff, ésta última como herramienta imprescindible para medir y comparar fractales. El apéndice B presenta un resumen de la teoría de los wavelets, fundamento de uno de los más duros adversarios de la compresión fractal como se explica en el capítulo 5.

Finalmente, se presenta una bibliografía comentada que evita la mera descripción catalográfica de muchos trabajos y que es de gran importancia para orientarse entre las numerosas fuentes existentes. Cierran la obra el índice de materias y un vocabulario bilingüe al que se puede recurrir al estudiar alguna de las referencias (prácticamente todas) escritas en inglés.

Créditos

Este trabajo se realizó como memoria del proyecto desarrollado por el autor para la obtención del título de Ingeniero en Informática. Es el resultado de cientos de horas de trabajo desde septiembre de 1997 a julio de

1998 bajo la tutela del profesor José Oncina Carratalá del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. El fuente de esta obra fue realizado en LaTeX. El proyecto cumple, además, las esperanzas del autor de adentrarse en la dimensión siempre fascinante de los fractales.

Juan Antonio Pérez Ortiz
Alicante, *8 de julio de 1998*

Índice general

Capítulo 1

Monstruos matemáticos

A pesar de que la historia de los fractales comienza en los últimos días del siglo XIX, gran parte del XX permanece ajena a ellos. En las últimas décadas del siglo, y casi paralelamente a la evolución de la investigación de los sistemas caóticos, los fractales van cobrando un auge creciente, hasta convertirse en un concepto cada vez más extendido en todas las ciencias.

En este capítulo nos introduciremos en la temática mostrando algunos de los fractales más famosos. También abordaremos brevemente la aparición del caos en sistemas dinámicos y el importante descubrimiento que supuso la constante de Feigenbaum. Por último, descubriremos uno de los más bellos y complejos objetos matemáticos, el conjunto de Mandelbrot, que puede considerarse una enciclopedia en la que cada una de sus entradas es un conjunto de Julia.

Este capítulo se basa en información obtenida de muy diversas fuentes y medios. Por citar algunas, puede considerarse [BAR 93b, GUZ 93]. Referencias adicionales aparecen donde se considera oportuno a lo largo del texto. Los objetivos de este capítulo son puramente descriptivos por lo que se omitirá toda demostración de los resultados obtenidos.

1.1. Fractales

A finales del siglo pasado, el matemático Charles Hermite tildaba de "plaga lamentable" la fascinación que algunos otros matemáticos sentían por determinadas curvas que desafiaban los cimientos de la geometría de la época. Muchos como él consideraban patológicas aquel tipo de curvas, desentendiéndose de sus insólitas propiedades. Uno de aquellos primeros

monstruos geométricos era el denominado conjunto de Cantor. Su definición es muy sencilla: se toma un segmento de determinada longitud (por ejemplo el intervalo $[0, 1]$ de la recta real) y se divide en tres subsegmentos de igual longitud, se suprime el segmento central y el proceso se repite con los dos nuevos segmentos resultantes. El resultado de iterar este proceso infinitas veces (paso al límite) es el conjunto de Cantor.

Ahora bien, ¿tiene elementos el conjunto de Cantor? Un espectador infinitesimal que contemplase la iteración anterior durante una eternidad, ¿no terminaría por ver desaparecer la totalidad de los puntos? El consolidado sistema de medidas de la época (medida Lebesgue) daba para dicho conjunto longitud nula. Tarde o temprano se tuvo que aceptar que aquel sistema de medidas era insuficiente.

En 1890, Peano ideó otro de tales monstruos: una curva que rellenaba el plano ¿Cómo podía una región cuadrada del plano ser una curva? Años más tarde, Hilbert ideó una curva con idéntica propiedad pero de más sencilla elaboración.

Otro ejemplo lo constituye la curva ideada por el matemático sueco Helge von Koch en 1904. Un segmento se divide en tres partes iguales, sustituyendo la central por los dos segmentos que junto a dicha parte formarían un triángulo equilátero. El proceso se repite *ad infinitum* con los cuatro segmentos resultantes. La curva de Koch oculta otra característica sorprendente: un perímetro infinito aloja un área finita.

Todas estas formas que se retuercen sobre sí mismas terminaron por revolucionar muchos de los conceptos dados por válidos hasta el siglo pasado, desembocando en la denominada teoría geométrica de la medida desarrollada en las primeras décadas de nuestro siglo. Uno de los aspectos más relevantes surgidos de esta teoría es la redefinición del concepto de dimensión a cargo de Hausdorff, que permite que estas curvas tengan dimensión fraccionaria. Así la curva de Koch tiene una dimensión de Hausdorff de 1,2618 lo cual indica que está más cerca de ser una recta (dimensión 1) que un área (dimensión 2). La curva de Hilbert, por tanto, tiene una dimensión de Hausdorff de 2. Los trabajos de Haussdorf fueron continuados durante la década de los años 20 por Besicovitch derivando en la teoría geométrica de la medida.

Hoy en día todas las curvas anteriores se incluyen dentro de una clase más amplia de objetos matemáticos denominados *fractales*. El término fue acuñado por Benoit Mandelbrot (descubridor de uno de los más bellos y complejos conjuntos matemáticos, que lleva su nombre) hace apenas veinte años como un neologismo derivado de la palabra latina *fractus*[1], estando aún

[1]Aunque Madelbrot definió el sustantivo *fractal* con genero femenino, son raras las referencias en castellano que se refieren a *las fractales* y gran mayoría las que lo hacen

por establecer una definición exacta y definitiva del término. Sin embargo, de algo no hay duda: las curvas descritas anteriormente son genuinamente fractales.

Básicamente los fractales se caracterizan por dos propiedades: *autosemejanza* (o autosimilitud) y *autorreferencia*. La autorreferencia determina que el propio objeto aparece en la definición de sí mismo, con lo que la forma de generar el fractal necesita algún tipo de algoritmo recurrente. La autosemejanza implica *invarianza de escala*, es decir, el objeto fractal presenta la misma apariencia independientemente del grado de ampliación con que lo miremos. Por más que se amplíe cualquier zona de un fractal, siempre hay estructura, hasta el infinito, apareciendo muchas veces el objeto fractal inicial, contenido en sí mismo.

De todas formas, no fue hasta los años 70 que comenzaron a vislumbrarse las aplicaciones de los fractales. En su tan citada obra *The Fractal Geometry of Nature*[2], Mandelbrot razonó que la naturaleza entiende mucho más de geometría fractal que de geometría diferenciable[3]. La geometría fractal aborda el estudio de formas geométricas no diferenciables o quebradas a cualquier escala que se miren. La geometría diferenciable, por otra parte, asume que a pequeña escala formas que no lo son se suavizan, con lo que se pierde la perspectiva global del objeto. La geometría fractal ofrece un modelo alternativo que busca una regularidad en las relaciones entre un objeto y sus partes a diferentes escalas. El objeto se expresa como el límite de un proceso geométrico iterativo, el cual puede provocar en cada iteración una ruptura (fractura o quebramiento) de la suavidad que lleva a la ausencia de diferenciabilidad en el objeto límite.

También fue crucial la publicación por Hutchinson en 1981 de un trabajo en el que se desarrolla el concepto de conjunto autosemejante, de gran trascendencia en el desarrollo posterior de la geometría fractal.

A partir de ahí, muchos científicos se han encontrado fractales en sus campos de estudio (el título de uno de los libros sobre el tema es bastante sugerente, *Fractals Everywhere*). La distribución de las galaxias, los procesos físicos de ramificación, agregación y turbulencia, la aparición de ruido en señales eléctricas (precisamente una especie de conjunto de Cantor en su distribución) e incluso los fenómenos económicos o sociológicos son algunos de los lugares en los que se esconde el serpenteo incansable de los fractales.

---

a *los fractales*. Por ello, y siguiendo esta tendencia, utilizaremos en esta obra el genero masculino.

[2]Editada en castellano en 1997, veinte años después de su publicación original, por la editorial Tusquets.

[3]Es más correcto contraponer la geometría fractal a la geometría diferenciable que a la euclidiana, aunque muchas fuentes la opongan a esta última.

Un congreso multidisciplinar sobre fractales (Fractal 98, Valletta, Malta) incluye entre los temas a tratar los siguientes:

- Aplicaciones en Biología, Medicina, Ingeniería, Economía y Sociología
- Autómatas celulares
- Estructuras coherentes
- Difusión
- Sistemas desordenados
- Superficies y volúmenes fractales
- Fenómenos de crecimiento
- Sistemas de funciones iteradas
- Análisis y síntesis de imágenes
- Sistemas L
- Multifractales
- Sistemas dinámicos no lineales
- Formación de estructuras
- Transiciones de fase
- Autoorganización y fenómenos de cooperación
- Turbulencia
- Visualización
- Ondas e interacciones

Resulta curioso que los matemáticos que sentaron las bases de la teoría geométrica de la medida a comienzos de este siglo, lo hicieron desde un punto de vista completamente teórico, sin intuir las tremendas consecuencias que sus trabajos tendrían varias décadas después en multitud de disciplinas científicas. Aunque no es correcto atribuir a Mandelbrot la paternidad de la geometría fractal, no puede negarse su vital aportación al renacimiento de ésta y su visión de la potencia de los fractales para modelizar la realidad.

La naturaleza es fractal

Es muy común encontrar afirmaciones como la que titula este apartado en la literatura sobre el tema. Sin embargo, es necesario advertir aquí que, realmente, *la naturaleza no es fractal*. Cuando decimos que un objeto real como una costa o la red capilar del sistema venoso es un fractal estamos

queriendo decir que existe un modelo fractal que aproxima con bastante precisión dicho objeto.

En el mundo real no existen fractales, como tampoco existen rectas ni esferas. Hablar de la dimensión fractal de una costa no es más absurdo que hablar del radio de la Tierra. La ciencia avanza gracias a todas estas aproximaciones, aunque probablemente las cosas no comprendan en su esencia nuestros modelos matemáticos.

Los fractales, además, abren la puerta a numerosas conjeturas sobre la complejidad del mundo. Las pautas de generación de fractales son extremadamente sencillas si se comparan con los resultados obtenidos. Es posible que numerosos comportamientos de la naturaleza que hoy día se nos antojan extremadamente complicados respondan de igual forma a mecanismos de enome sencillez. La geometría fractal es una rama muy joven cuyos progresos deben repercutir muy directamente en una creciente utilidad de la geometría fractal para el estudio de la realidad.

El conjunto de Cantor

El conjunto de Cantor es un ejemplo clásico de conjunto no numerable con el mismo cardinal que el continuo, pero, a pesar de ello, con medida de Lebesgue unidimensional (longitud) nula. Una breve descripción de esta medida puede encontrarse en el apéndice A.

Para construir el conjunto de Cantor partiremos del intervalo unidad $E_0 = [0, 1] \subset \mathbf{R}$. Dividimos dicho intervalo en tres partes iguales y consideramos los intervalos cerrados de los extremos

$$E_{11} = \left[0, \frac{1}{3}\right] \qquad E_{12} = \left[\frac{2}{3}, 1\right]$$

cada uno de ellos de longitud $1/3$.

El proceso anterior se repite sobre los nuevos conjuntos obtenidos. Cada uno de estos intervalos se divide en tres intervalos de igual longitud para prescindir del intervalo central y considerar los cuatro intervalos cerrados

$$E_{21} = \left[0, \frac{1}{9}\right] \quad E_{22} = \left[\frac{2}{9}, \frac{1}{3}\right] \quad E_{23} = \left[\frac{2}{3}, \frac{7}{9}\right] \quad E_{24} = \left[\frac{8}{9}, 1\right]$$

cada uno de ellos de longitud $1/9$.

Si continuamos indefinidamente de esta forma, en la etapa k-ésima tendremos 2^k intervalos cerrados E_{kj} con $j = 1, 2, \ldots, 2^k$ cada uno de ellos de longitud 3^{-k}.

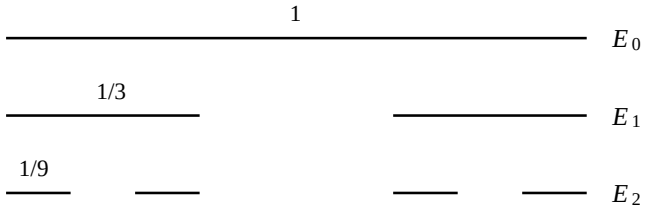


Figura 1.1: El conjunto ternario de Cantor se obtiene de manera inductiva comenzando por el segmento unidad y quitando en cada etapa a cada intervalo el segmento medio resultante de dividirlo en tres partes iguales.

Consideremos ahora para cada $k = 1, 2, \ldots$ el conjunto

$$E_k = \bigcup_{j=1}^{2^k} E_{kj}$$

Observamos que los conjuntos E_k, $k = 1, 2, \ldots$, forman una sucesión monótonamente decreciente, esto es

$$E_{k+1} \subset E_k \quad \forall k$$

El conjunto límite de este proceso

$$E = \bigcap_{k=1}^{\infty} E_k$$

se denomina *conjunto ternario de Cantor* . En la figura 1.1 se muestran las primeras etapas de la generación del conjunto de Cantor.

Las propiedades asombrosas de este conjunto son abundantes. Veamos unas cuantas. En primer lugar observamos que E no es vacío ya que en cada E_k están, como mínimo, los extremos de los 2^k intervalos cuya unión nos da E_k y, por lo tanto, también están en E. Además, el conjunto de Cantor es cerrado por ser intersección de cerrados.

Con todo, éstos no son los únicos puntos de E; si así fuera, se trataría de un conjunto numerable. Pero E es no numerable. Veámoslo.

Cada punto de E es representable de forma única mediante

$$a = \frac{a_1}{3} + \frac{a_2}{3^2} + \cdots + \frac{a_n}{3^n} + \cdots$$

donde cada a_i es 0 ó 2. Podemos entonces escribirlo en base tres como

$$a = 0.a_1\, a_2 \ldots a_n \ldots$$

Recíprocamente, cada expresión de este tipo corresponde a un punto de E. Si E fuera numerable[4] podríamos ordenar sus elementos. Supongamos que es cierto lo anterior y que E es numerable

$$a^1 = 0.a_1^1\, a_2^1 \ldots$$
$$a^2 = 0.a_1^2\, a_2^2 \ldots$$
$$a^3 = 0.a_1^3\, a_2^3 \ldots$$
$$\ldots\ldots\ldots\ldots$$

y formemos un punto $0.b_1 b_2 \ldots$ a partir de la sucesión anterior con la regla siguiente

$$\begin{array}{ll} \text{si } a_n^n = 0, & b_n = 2 \\ \text{si } a_n^n = 2, & b_n = 0 \end{array}$$

El número así formado no está en la sucesión anterior y, sin embargo, pertenece claramente a E y, por tanto, E no puede ser numerable.

Este procedimiento es muy similar a la famosa técnica utilizada por Cantor para demostrar la no numerabilidad de $\mathbf{R}$.

Aun así, el conjunto de Cantor tiene medida Lebesgue unidimensional nula. Esta medida se discute en el apéndice A. Para cualquier etapa k, la familia de intervalos $\{E_{kj}\}$, $j = 1, \ldots, 2^k$, es un recubrimiento de E formado por intervalos disjuntos. Así se tiene, por las propiedades de la medida de Lebesgue, que

$$L^1(E) \leq L^1\left(\bigcup_{j=1}^{2^k} E_{kj}\right) = \sum_{j=1}^{2^k} L^1(E_{kj}) = 2^k 3^{-k} = \left(\frac{2}{3}\right)^k$$

Puesto que la desigualdad es cierta para todo k y $(2/3)^k$ tiende a cero cuando k tiende a infinito, se obtiene $L^1(E) = 0$.

Aunque aquí no lo demostraremos, puede comprobarse, además, que el conjunto E no contiene intervalos, es decir, es infinitamente poroso.

Curvas de Peano y Hilbert

En 1890 Peano construyó una curva continua que pasa por todos los puntos del cuadrado unidad $[0,1]^2$. Era el primer ejemplo de una curva que

[4] Un conjunto es *infinito* si tiene el mismo cardinal que una parte estricta suya, esto es, si puede establecerse una aplicación biyectiva entre el conjunto y un subconjunto propio suyo. Un conjunto es *numerable* si tiene el mismo cardinal que $\mathbf{N}$. Cantor demostró que $\mathbf{Q}$ es numerable y que $\mathbf{R}$ no es numerable.

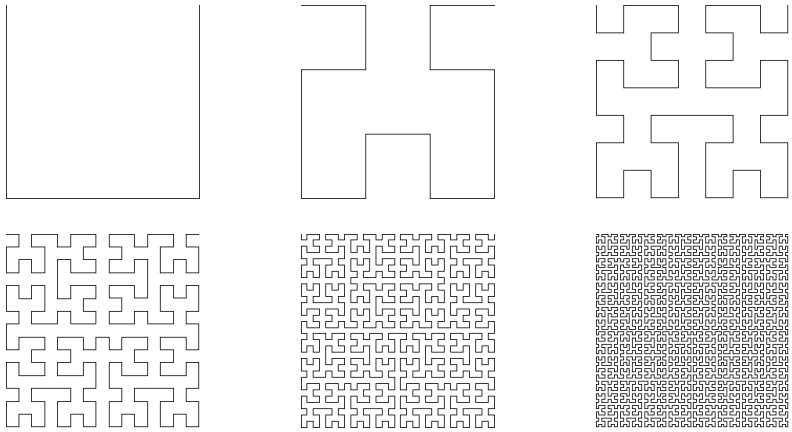


Figura 1.2: Primeras etapas de la generación de la curva de Hilbert. La curva de Hilbert es un ejemplo de curva que llena el plano, por lo que su dimensión fractal es 2.

llena un espacio. Años más tarde, Hilbert construye otra del mismo tipo con una construcción geométrica más simple de describir.

La curva de Hilbert se construye iterando el procedimiento que puede observarse en la figura 1.2. En cada etapa cada segmento se sustituye por otros cuatro con la mitad de longitud. La curva límite de tales poligonales llena el cuadrado unidad.

Curva de Kock

Esta curva fue construida en 1904 por el matemático Helge von Kock. Se parte del segmento unidad $[0, 1]$ y se divide en tres partes iguales, sustituyendo la parte central por los dos segmentos que junto con dicha parte formarían un triángulo equilatero. Con cada uno de los cuatro segmentos que así queden determinados se repite la operación anteriormente descrita.

Se procede indefinidamente de esta forma obteniendo en cada etapa k una poligonal de longitud $(4/3)^k$. La curva de Kock se define como la curva límite a que converge la sucesión cuando k tiende a infinito. Se trata, por tanto, de una curva de longitud infinita pues $(4/3)^k$ tiende a infinito con k. Más aún, la longitud de la parte de curva comprendida entre dos puntos cualesquiera de la misma también es infinita. El área bajo la curva, por otra parte, viene dada por la serie

$$1 + \left(\frac{4}{9}\right) + \left(\frac{4}{9}\right)^2 + \left(\frac{4}{9}\right)^3 + \ldots$$

Figura 1.3: Primeros pasos del proceso de construcción de la curva de Koch. En el límite, dados dos puntos cualesquiera de la curva es imposible llegar a uno de ellos desde el otro por encima de la curva. La longitud de cualquier tramo de curva es infinita.

que converge a 9/3 asumiendo que el área bajo el triángulo de la primera iteración es 1. En la figura 1.3 pueden verse las primeras etapas de la generación de la curva de Koch.

Funciones de Weierstrass

Weierstrass dio otro ejemplo de una curva con comportamiento análogo a las anteriores, pero definida de forma analítica

$$f(x) = \sum_{i=1}^{\infty} \lambda^{(s-2)i} \operatorname{sen} \lambda^i x$$

con $1 < s < 2$ y $\lambda < 1$. Esta función es continua, pero no es diferenciable en ningún punto.

Otros fractales

Los apartados anteriores han mostrado algunos conjuntos fractales de reconocida fama y prestigio. Sin embargo, no son, ni mucho menos, los únicos.

La figura 1.4 muestra el triángulo de Sierpinski. Este fractal se genera a partir de un triángulo equilatero relleno de lado l del que se extrae el subtriángulo formado por los tres puntos medios de los lados del triángulo. El proceso se repite con los tres nuevos triángulos de lado $l/2$ así obtenidos. Si continuamos de esta manera, en la etapa k tendremos 3^k triángulos equilateros con lados de longitud $l2^{-k}$. La figura 1.4 muestra el conjunto obtenido.

El proceso seguido para la construcción del conjunto de Cantor puede generalizarse a dimensiones superiores. La generalización a tres dimensiones

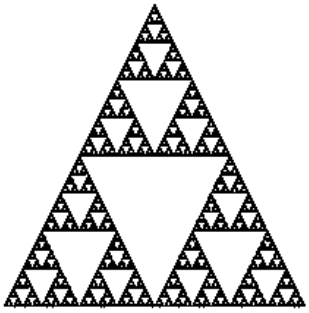


Figura 1.4: Imagen final aproximada del triángulo de Sierpinski. Sabiendo que el conjunto inicial es un triángulo equilatero relleno, no es difícil deducir el proceso iterativo que permite obtenerlo.

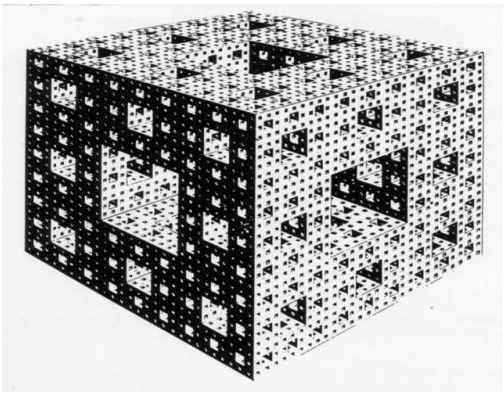


Figura 1.5: Esponja de Menger. Puede considerarse una generalización del conjunto de Cantor. Se comienza por un cubo y se divide en 27 cubos más pequeños, extrayendo el cubo central y los situados en el centro de cada cara del cubo original. El proceso se repite con cada uno de los cubos restantes y así sucesivamente. La dimensión de la esponja de Menger es $2,727$ lo que indica que está más cerca de ser un cuerpo solido que una curva suave.

produce la denominada esponja de Menger que puede verse en la figura 1.5.

Unos años antes de los primeros desarrollos de Mandelbrot, algunos científicos comenzaron a ponerse de acuerdo en la explicación de ciertos fenómenos irregulares que surgían en multitud de sistemas dinámicos. Eran los primeros intentos de descubrir algunos viejos trucos de magia de la naturaleza.

1.2. El caos y el orden

El descubrimiento y formalización del caos se ha dado en considerar como una nueva revolución en la Física del siglo XX, comparable a la que en su día provocaron la relatividad y la teoría cuántica.

Un sistema dinámico (siempre no lineal) se considera caótico[5] si presenta un comportamiento aperiódico (esto es, resultado de oscilaciones regulares que no se repiten nunca, de periodo infinito) resultado de un modelo totalmente determinista y que presenta gran sensibilidad a las condiciones iniciales.

La sensibilidad a las condiciones iniciales implica que existe una divergencia exponencial de trayectorias inicialmente muy próximas en el espacio de fases, fenómeno que se conoce como *estirado*. Otra propiedad existente sobre el espacio de fases y opuesta al estirado es el *plegamiento* que conlleva que dos trayectorias muy lejanas pueden eventualmente acercarse.

Si representamos el retrato fase de un sistema dinámico, veremos que las dos fuerzas anteriores entran en acción de forma que se genera una estructura confinada en una región del espacio de fases que se conoce como *atractor extraño*. Antes del descubrimiento del caos, los ciclos límite eran los atractores más complejos que se conocían. Hoy día se puede decir que cada sistema caótico lleva asociado un atractor de características peculiares.

Las trayectorias del espacio de fases nunca intersectan entre sí, pues esto supondría un comportamiento periódico. Como la región en la que está ubicado el atractor es finita, se tiene, al seguir una trayectoria cualquiera, una curva de longitud infinita encerrada en un área finita o, dicho de otra forma, un atractor extraño posee estructura *fractal*.

El ordenador facilita el proceso iterativo de los sistemas dinámicos y es un arma imprescindible para aproximarse a la geometría de los atractores extraños.

Duplicación de periodo y constante de Feigenbaum

La ecuación logística se ha convertido en la manera usual de introducir las características del caos. Se trata de una ecuación en diferencias que fue formulada por Verhulst en el siglo pasado para explicar el crecimiento de

[5] Para este apartado puede consultarse *Dinámica clásica* de Antonio Rañada, editado por Alianza. También *Biofísica: procesos de autoorganización en Biología* de Francisco Montero y Federico Morán, editado por Eudema, donde se discute ampliamente la teoría de bifurcaciones y la constante de Feigenbaum.

una población perteneciente a la misma especie y que se reproduce en un entorno cerrado sin ningún tipo de influencia externa. Pese a su aparente sencillez, constituye un buen ejemplo para mostrar el comportamiento de los sistemas caóticos. La ecuación se puede escribir como

$$x_{n+1} = rx_n(1 - x_n)$$

donde el parámetro r es una constante denominada *parámetro de crecimiento* (generalmente entre 0 y 4) y la variable x_n puede verse como la fracción máxima de población que el ambiente puede soportar en el instante t_n. Considerando que la población límite

$$x_\infty = \lim_{n\to\infty} x_n$$

existe, queremos investigar la forma en la cual x_n depende del parámetro de crecimiento r. Si estudiamos experimentalmente el sistema, observaremos que para valores de $r < 3$ el sistema converge a un punto fijo estable, que es cero cuando $r < 1$. Cuando $r > 3$, el punto fijo se hace inestable y el valor de x_∞ oscila entre dos valores; se ha obtenido lo que se conoce como una *duplicación de periodo*.

Si se aumenta r ligeramente, por ejemplo $r = 3{,}44\ldots$, el número de puntos sobre los que oscila x_∞ es de 4. Si se sigue aumentando el valor de r, aparece una nueva duplicación de periodo para $r = 3{,}544\ldots$, obteniendo un periodo de 8. Y así sucesivamente hasta llegar a obtener una sucesión de infinitos valores para x_∞ correspondiente al caos. Nótese cómo los valores de r para los que se producen las sucesivas duplicaciones de periodo están cada vez más cerca unos de otros.

El comportamiento de la ecuación logística en función de r puede observarse visualmente a través de un *diagrama de bifurcación*. En el eje horizontal se representa un cierto intervalo de valores de r y entonces se dibujan los valores de x generados por la iteración en el eje vertical. La figura 1.6 muestra el diagrama de bifurcación de la ecuación logística en el rango $2 \leq r \leq 4$.

La duplicación de periodo es un signo ineludible del comportamiento caótico de un sistema. Son muchos, y cada día más, los sistemas dinámicos en los que se observa este fenómeno y que desembocan, variando alguno de sus parámetros, en caos. Es más, un famoso artículo publicado en los inicios de la teoría del caos demostró que cualquier sistema en el que, para algún valor de sus parámetros, se registrara una periodicidad de periodo 3, desembocará para otros valores de sus parámetros en comportamiento caótico.

Lo anterior hace pensar en una universalidad del caos todavía no muy bien conocida que hace que sistemas muy diferentes muestren pautas similares de comportamiento. Un hecho que vino a corroborar esto y a mostrarnos

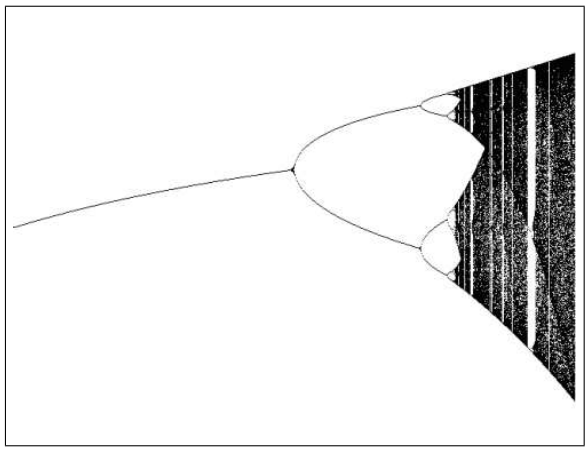


Figura 1.6: Diagrama de bifurcación de la ecuación logística $x_{n+1} = rx_n(1 - x_n)$ en el rango $2 \leq r \leq 4$. Puede observarse la ruta del caos: sucesivos desdoblamientos de periodo que desembocan en un periodo infinito.

que existe un cierto orden en el caos fue el descubrimiento por parte de Feigenbaum a mediados de los setenta de la constante que lleva su nombre. Una vez obtenido el diagrama de bifurcación de la ecuación logística, se puede calcular el incremento del parámetro entre dos bifurcaciones contiguas

$$\Delta i = r_i - r_{i-1}$$

y dividiendo por el incremento en el siguiente intervalo

$$\frac{\Delta i}{\Delta i + 1} = \frac{r_i - r_{i-1}}{r_{i+1} - r_i}$$

Feigenbaum encontró que la fracción anterior convergía hacia un valor determinado al ir haciendo i mayor, de modo que en el límite se obtenía

$$\delta = \lim_{i \to \infty} \frac{\Delta i}{\Delta i + 1} = 4{,}669201609102990671853203820466 2\ldots$$

Feigenbaum calculó el límite anterior para otras ecuaciones en diferencias y obtuvo el mismo valor para δ. Posteriormente se ha encontrado el mismo valor de δ en algunos sistemas continuos e incluso en sistemas experimentales, todos de muy diversa procedencia. Hoy sabemos que δ, conocida como *constante de Feigenbaum*, es una constante universal tan fundamental como π o e y que ha provocado una nueva forma de ver el mundo.

El comportamiento caótico descrito anteriormente no sólo surge bajo sistemas discretos. Multitud de sistemas dinámicos de ecuaciones diferenciales presentan fenómenos caóticos que generan atractores extraños. Por mostrar uno de ellos, veremos como el caos puede anidar incluso en sistemas clásicos aparentemente sencillos.

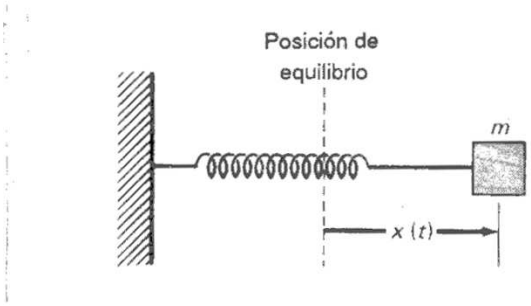


Figura 1.7: Cuando se considera que la fuerza ejercida por un muelle sobre una masa m no sigue la ley de Hooke, sino que esta fuerza es función no lineal de x y, además, hacemos que una fuerza externa actúe sobre la masa, el sistema puede comportarse de forma caótica.

Ecuación forzada de Duffing

La ecuación diferencial de segundo orden

$$m\ddot{x} + c\dot{x} + kx + \beta x^3 = 0$$

puede utilizarse para modelar las vibraciones libres amortiguadas por la velocidad de una masa m sobre un muelle no lineal como se muestra en la figura 1.7. El término kx representa la fuerza ejercida sobre la masa por un muelle lineal, mientras que el término βx^3 representa la no linealidad de un muelle real.

Vamos a analizar las vibraciones forzadas que resultan cuando una fuerza externa $F(t) = F_0 \cos \omega t$ actúa sobre la masa. Con esta fuerza sumada al sistema obtenemos la ecuación forzada de Duffing

$$m\ddot{x} + c\dot{x} + kx + \beta x^3 = F_0 \cos \omega t$$

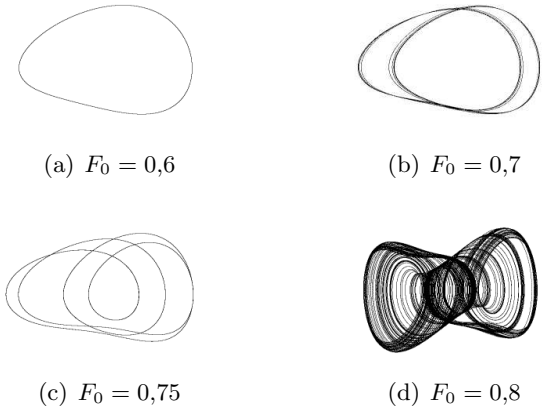
para el desplazamiento $x(t)$ de la masa de su posición de equilibrio. Para simplificar el modelo supondremos que $k = -1$ y que $m = c = \beta = \omega = 1$, con lo que la ecuación diferencial es

$$\ddot{x} + \dot{x} - x + x^3 = F_0 \cos t.$$

Pasando el sistema anterior a variables de estado obtenemos

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ (1 - x^2) & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ F_0 \cos t \end{pmatrix}$$

que puede integrarse mediante el método de Euler para obtener el retrato fase asociado al sistema.



(a) $F_0 = 0{,}6$ (b) $F_0 = 0{,}7$

(c) $F_0 = 0{,}75$ (d) $F_0 = 0{,}8$

Figura 1.8: Ruta hacia el caos de la ecuación forzada de Duffing. Las figuras muestran las duplicaciones de periodo directamente sobre el atractor extraño (también podría haberse hecho con un diagrama de bifurcación). En algún punto entre $F_0 = 0{,}75$ y $F_0 = 0{,}8$ el caos irrumpe en el sistema obligándolo a un comportamiento aperiódico. Las duplicaciones de periodo respetan la constante de Feigenbaum.

Variando el valor de F_0 cuidadosamente desde $F_0 = 0{,}6$ a $F_0 = 0{,}8$, como en la figura 1.8, pueden observarse las sucesivas duplicaciones de periodo que llevan al caos. Es curioso que aunque esta ecuación se estudió durante décadas, sin ordenadores nadie pudo vislumbrar en ella los signos del caos.

1.3. Conjuntos de Julia

En las secciones anteriores hemos estudiado la evolución de sistemas dinámicos sobre el plano real. Sin embargo, algunos de los resultados más espectaculares obtenidos con la iteración de un sistema dinámico se dan cuando consideramos funciones de variable compleja. Esta espectacularidad se muestra en dos frentes distintos: el estético y el matemático.

La teoría de sistemas dinámicos complejos data de comienzos de este siglo, con los trabajos de los matemáticos franceses Gaston Julia y Pierre Fatou. Aquí nos centraremos en el estudio de sistemas dinámicos complejos cuadráticos. La discusión de otros sistemas se sale del objetivo de esta introducción.

Podemos definir el conjunto de Julia de un polinomio de variable compleja como la frontera del conjunto de puntos que escapan al infinito al iterar dicho polinomio. Esto significa que la órbita de un elemento del conjunto de Julia no escapa al infinito, pero existen puntos arbitrariamente cerca de

él que sí lo hacen. La órbita de un punto x bajo un sistema dinámico de función f es la sucesión de puntos $\{f^n(x)_{n=0}^{\infty}\}$.

Para simplificar las cosas nuestro estudio girará en torno a funciones polinómicas cuadráticas de la forma:

$$f(z) = z^2 + c$$

donde c y z son números complejos. A pesar de su sencillez, la iteración de la función anterior genera estructuras muy complicadas, hecho éste que ya fue vislumbrado por Julia y Fatou a comienzos de siglo.

Los valores que no tienden a infinito los representaremos dibujando en negro su pixel asociado. Con este procedimiento se han obtenido los conjuntos de la figura 1.9. Como se dijo antes, es la frontera de las regiones dibujadas en negro lo que constituye realmente el conjunto de Julia. A la región completa se le suele denominar *conjunto de Julia relleno*.

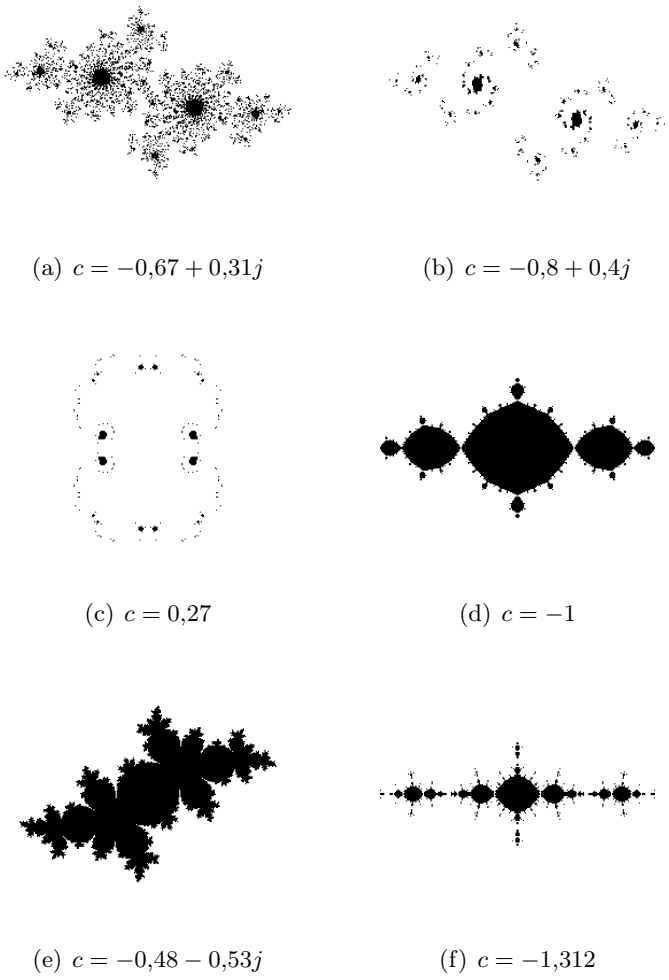
Aunque algunos matemáticos intuían la diversidad de conjuntos de Julia que se derivaba de la utilización de distintos valores para c, la llegada de los primeros ordenadores con capacidades gráficas y a precios asequibles a finales de la década de los setenta hizo que se sobrepasara cualquier expectativa.

Si observamos los distintos conjuntos de Julia rellenos representados en esta sección veremos que pueden clasificarse en dos grandes grupos según su estructura. Algunos de ellos parecen estar formados por una única pieza, de manera que parece factible poder caminar por su frontera interminablemente. Otros, sin embargo, parecen fragmentados o disconexos. Esta clasificación de hecho no es arbitraria y su estudio dio lugar a uno de los objetos matemáticos más complejos que existen: el conjunto de Mandelbrot.

1.4. El conjunto de Mandelbrot

Julia probó que la órbita de $z = 0$ juega un papel esencial para saber si un conjunto de Julia es conexo. Si esta órbita escapa al infinito, el conjunto aparece fragmentado como polvo fractal. En caso contrario, el conjunto de Julia es conexo.

El resultado anterior nos proporciona un método preciso y cómodo para determinar la conectividad de un conjunto de Julia. Ahora bien, ¿cuándo podemos considerar que la órbita de $z = 0$ diverge a infinito? Esta pregunta queda resuelta por la teoría de iteraciones que nos asegura que la órbita divergirá a infinito si en algún momento uno de sus puntos tiene módulo mayor o igual a 2.



(a) $c = -0{,}67 + 0{,}31j$ (b) $c = -0{,}8 + 0{,}4j$

(c) $c = 0{,}27$ (d) $c = -1$

(e) $c = -0{,}48 - 0{,}53j$ (f) $c = -1{,}312$

Figura 1.9: Conjuntos de Julia para distintos valores del parámetro c. Estos conjuntos se pueden dividir en dos grupos, los que están formados por una sola pieza y los que parecen estar fragmentados en muchas piezas. Los tres primeros pertenecen a la última clase.

Aunque no lo demostraremos, la clasificación anterior es todavía de carácter más fuerte, ya que, según el valor del parámetro c, el conjunto de Julia es o bien conexo o bien completamente disconexo, es decir, formado por una nube de puntos dispersos con la misma estructura que los conjuntos de Cantor. Estos puntos aparecen dispuestos en grupos densos de forma que cualquier disco finito que envuelva a un punto contiene, como mínimo, otro punto del conjunto.

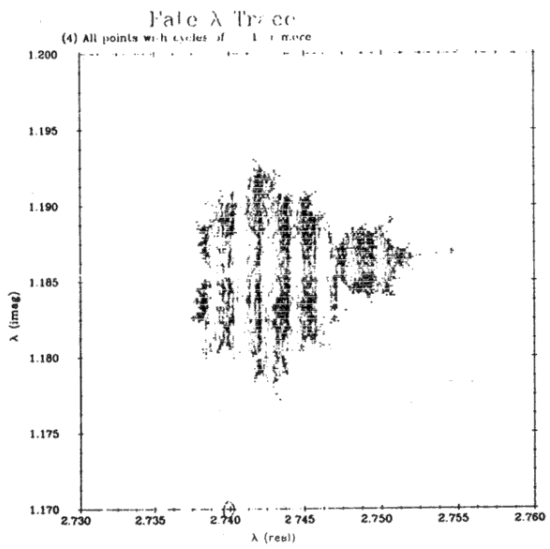


Figura 1.10: Una de las primeras fotografías del nuevo continente descubierto por Mandelbrot. Se trata de una de las primeras imágenes de una cardioide distinta a la principal y fue tomada en 1980. En tan sólo unos años se ha hecho posible obtener imágenes a enormes resoluciones y con millones de colores con tan solo un ordenador personal. Aun así esta representación ya significaba mucho: en los años sesenta los primeros atractores extraños se representaron por impresora mediante caracteres alfanuméricos.

Dada esta división de los conjuntos de Julia, parece natural preguntarse qué valores de c de la ecuación $f(z) = z^2 + c$ generan conjuntos de uno u otro tipo. Esta cuestión tan simple no fue resuelta hasta 1978 cuando Mandelbrot representó en un plano todos los valores de c que producían conjuntos de Julia conexos, consiguiendo la primera representación del conjunto que hoy lleva su nombre. Por aquellas fechas y con los medios disponibles las primeras y toscas impresiones de ordenador que obtuvo Mandelbrot eran del tipo de la de la figura 1.10.

Una representación más visible es la mostrada en la figura 1.11. Lo primero que salta a la vista es la gran región a la derecha que conforma una *cardioide*.[6] A la izquierda de la gran cardioide podemos observar un disco tangente a ella. Éste, no obstante, no es el único disco tangente a la cardioide, ya que pueden apreciarse multitud de pequeños discos tangentes a

[6]Una cardioide es la curva engendrada por el movimiento de un punto de una circunferencia que rueda exteriormente sobre otra fija de igual radio. La ecuación de la cardioide en coordenadas cartesianas es $(x^2 + y^2)^2 - 4ax(x^2 + y^2) = 4a^2y^2$ donde a es el radio de la circunferencia fija.

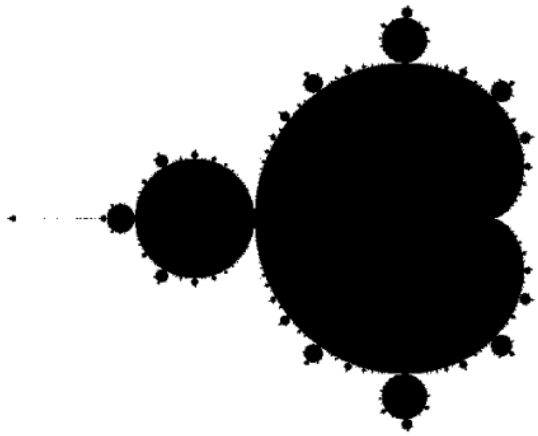


Figura 1.11: El conjunto de Mandelbrot. Puede apreciarse la cardioide y la serie de círculos cada vez más pequeños pegados a ella.

ella rodeándola. Si ampliáramos[7] algunas zonas de la imagen, veríamos que unidos a estos discos existen otros aún más pequeños, a los que se unen otros y así sucesivamente. Si se estudia detenidamente la sucesión de circulos cada vez más pequeños que se extiende horizontalmente en el sentido negativo del eje de las x y obtenemos los diámetros de los sucesivos círculos $d_1, d_2, \ldots$, podemos obtener el límite

$$\delta = \lim_{i \to \infty} \frac{d_i}{d_{i+1}} = 4{,}669201609102990671853203820 \ldots$$

cuyo valor es la constante de Feigenbaum. La universalidad de la constante de Feigenbaum es un hecho fascinante que hoy por hoy desafía a la ciencia.

Existen muchas otras cardioides repartidas por el plano, realmente infinitas. Todas estas cardioides están unidas a la cardioide principal por medio de filamentos cargados de nuevas cardioides. Estos filamentos se ramifican siguiendo pautas muy complejas haciendo que el conjunto de Mandelbrot sea conexo. La demostración de la conectividad del conjunto de Mandelbrot es una labor compleja que todavía presenta gran cantidad de cuestiones abiertas.

Existen series dudas sobre la autosemejanza del conjunto de Mandelbrot. Aunque es posible encontrar pequeñas copias en miniatura del conjunto por

[7] Cuando hablamos de ampliar una zona de la imagen de un conjunto fractal, nos referimos a representar el conjunto entre unos intervalos más pequeños que los de la imagen inicial y no a nada parecido a un *zoom* fotográfico que no aportaría ninguna información adicional.

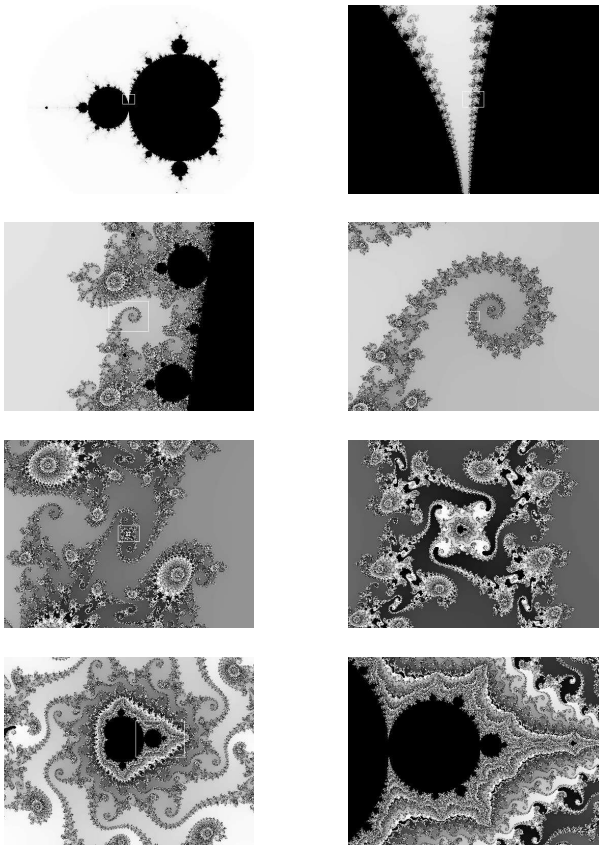


Figura 1.12: De izquierda a derecha y de arriba a abajo, sucesivas imágenes de una inmersión en el conjunto de Mandelbrot que demuestran el carácter casi autosemejante del conjunto. El centro de las imágenes corresponde al punto $-0{,}74650920 + 0{,}09848028j$ y la última muestra una escala de casi tres millones de aumentos sobre la primera. Para generarlas se utilizó el algoritmo de tiempo de escape.

todo el plano, éstas siempre están envueltas en filamentos cuyo aspecto varía notablemente según donde observemos. A diferencia de los conjuntos de Julia, que sí son autosemejantes, dadas dos copias del conjunto de Mandelbrot, podríamos identificar, en teoría, bajo qué escala del plano se han obtenido. Podemos, por tanto, considerar por ahora al conjunto de Mandelbrot como *casi autosemejante*.

Las imágenes de la figura 1.12 pueden dar una idea de la variedad de estructuras que es posible encontrar en el conjunto de Mandelbrot. Estas imágenes muestran de izquierda a derecha y de arriba a abajo un *zoom* sobre la imagen precedente a cada una.

```
1. Sea Δp = (x2 − x1)/(x̃ − 1)
2. Sea Δq = (y2 − y1)/(ỹ − 1)
3. Hacer desde p = 0 hasta x̃
    3.1. Hacer desde q = 0 hasta ỹ
        3.1.1. Hacer p0 = x1 + p · Δp
        3.1.2. Hacer q0 = y1 + q · Δq
        3.1.3. Hacer z = 0 + 0j
        3.1.4. Desde k = 1 hasta iteraciones
            3.1.4.1. Hacer z = z^2 + (p0 + q0 j)
            3.1.4.2. Si |z| > 2, pintar el punto (p, q) y
                        salir del bucle de la variable k
4. Fin
```

Figura 1.13: Algoritmo para representar el conjunto de Mandelbrot. Las dimensiones del modo de vídeo utilizado son $\tilde{x} \times \tilde{y}$. La parte a representar del conjunto es la comprendida entre las coordenadas (x_1, y_1) y (x_2, y_2) que conformarán, respectivamente, la esquina superior izquierda y la esquina inferior derecha de la imagen obtenida.

Explosiones cromáticas

La representación del conjunto de Mandelbrot en una pantalla de ordenador es extremadamente sencilla. El único problema que puede surgir es la discretización que impone la pantalla. El programa en pseudocódigo de la figura 1.13 asume que las dimensiones del modo de vídeo utilizado son $\tilde{x} \times \tilde{y}$ y que se desea representar la parte del conjunto de Mandelbrot comprendida entre las coordenadas (x_1, y_1) y (x_2, y_2) que conformarán, respectivamente, la esquina superior izquierda y la esquina inferior derecha de la imagen.

Aunque el valor de *iteraciones* del programa de la figura 1.13 puede mantenerse en torno a 200, deberá incrementarse conforme se vaya reduciendo el intervalo del conjunto de Mandelbrot a representar.

Una de las formas de representar el conjunto de Mandelbrot que proporciona las imágenes de mayor belleza es mediante el denominado *algoritmo de tiempo de escape*. Para representar el conjunto de Mandelbrot mediante este procedimiento seguimos pintando, como hasta ahora, los puntos pertenecientes al conjunto de color negro. Los puntos que divergen a infinito, sin embargo, se pintan con distintos colores según la *velocidad* de escape hacia infinito de su órbita asociada. Lo anterior se lleva a cabo en la práctica considerando una paleta de colores en la que cada uno lleva asociado un número distinto. Un punto no perteneciente al conjunto se pinta de un color n si son necesarias n iteraciones para que el módulo de su órbita sea mayor que 2 (condición ésta, como ya vimos, suficiente para que la orbita se fugue hacia infinito). El algoritmo de tiempo de escape también puede aplicarse,

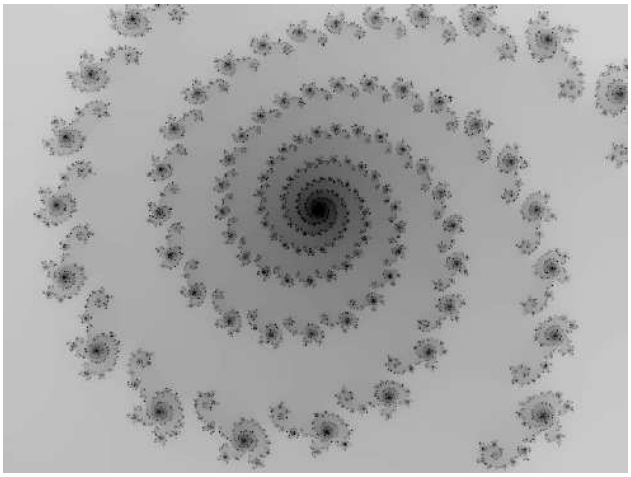


Figura 1.14: El algoritmo de tiempo de escape convierte en un arte la representación de conjuntos de Julia. Hoy día es casi tan importante la selección de la zona de visualización como la de una adecuada paleta de colores. El conjunto aquí mostrado corresponde a $c = -0{,}204812 - 0{,}663316j$.

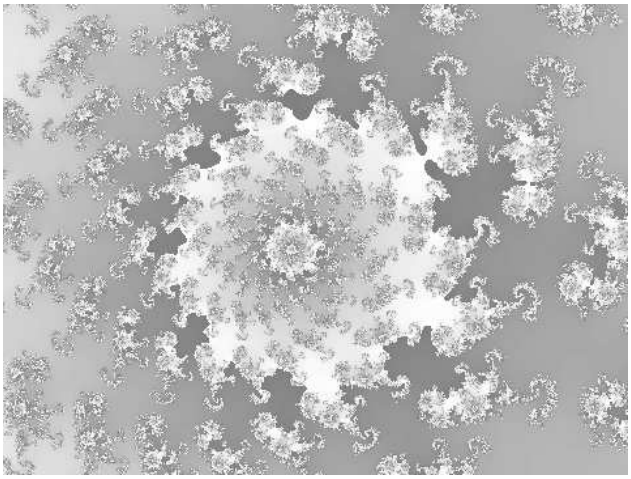


Figura 1.15: Otro conjunto de Julia merecedor de una visita. El parámetro c de este conjunto es $-0{,}833062 + 0{,}194798j$.

por supuesto, a la representación de conjuntos de Julia. Un par de conjuntos de Julia obtenidos con esta técnica se muestran en las figuras 1.14 y 1.15. El algoritmo de tiempo de escape, en la sencilla versión aquí comentada y en variaciones más sofisticadas, ha convertido en un arte la representación de conjuntos fractales: se venden pósters, camisetas o postales con estos motivos y se organizan exposiciones por todo el planeta.

Capítulo 2

Lenguajes fractales

En el capítulo anterior se describió la forma de construir varios fractales de muy diversa índole. Sin embargo, salvo en el caso de los sistemas cuadráticos complejos, no se mostró un método sencillo para generarlos.

Los sistemas D0L proporcionan las pautas para la obtención de multitud de fractales, basándose en la interpretación de ciertos códigos que almacenan la información que permite la construcción de una sucesión de conjuntos convergentes al fractal. No es el único enfoque posible (veremos más adelante los sistemas de funciones iteradas y existen otros métodos como los basados en automatas celulares o en teragones), pero sí es uno de los más sencillos y potentes. Los contenidos de este capítulo pueden encontrarse también en [BAR 93b] y especialmente en [BLA 94].

2.1. Teoría de lenguajes

Antes de presentar los sistemas D0L es necesaria una pequeña introducción a los conceptos básicos de la teoría de lenguajes. Unas cuantas definiciones nos darán las herramientas básicas para comprender el resto del capítulo.

Para empezar consideremos un conjunto finito Σ al que denominaremos *alfabeto*. A los elementos de Σ también los denominaremos *símbolos*. Las letras y los dígitos son ejemplos de símbolos utilizados frecuentemente. Una *cadena* (o *palabra*) es una secuencia posiblemente infinita de símbolos yuxtapuestos. Por ejemplo, partiendo del alfabeto $\Sigma = \{a, b, c\}$ podemos construir cadenas como $abbc$ o $aaaaaa\ldots$

Consideremos ahora el conjunto de todas las cadenas de longitud finita

sobre Σ, incluyendo la cadena vacía ϵ; dicho conjunto recibe a menudo el nombre de *lenguaje universal* y se denota por $\Sigma^\star$. Con Σ^+ nos referiremos al conjunto de todas las cadenas no vacías (esto es, $\Sigma^\star = \Sigma^+ \cup \{\epsilon\}$).

Para $x \in \Sigma^\star$, $|x|$ es la longitud de x, esto es, el número de elementos que forman la cadena x. Por lo tanto, $|\epsilon| = 0$.

El conjunto de cadenas infinitas sobre Σ se escribe Σ^ω y el conjunto de cadenas posiblemente infinitas $\Sigma^\infty = \Sigma^\star \cup \Sigma^\omega$. El concepto más importante y el que da utilidad y sentido a la teoría de lenguajes es precisamente el de *lenguaje*. Un lenguaje es cualquier subconjunto de $\Sigma^\star$. Así, $\mathcal{L}_1 = \{aa, \epsilon, bca\}$ y $\mathcal{L}_2 = \{c, cc, ccc, cccc, \ldots\}$ son lenguajes sobre el alfabeto anterior.

2.2. Fractales sintácticos

No daremos aquí intencionadamente una definición detallada de fractal, idea ésta que se irá concretando a lo largo de sucesivos capítulos. Preferimos que el lector maneje la idea intuitiva de conjunto fractal que adquirió en el capítulo anterior y que en éste se concretará todavía más sin llegar a una definición rigurosa. Puede considerarse, por tanto, un fractal como un subconjunto de $\mathbf{R}^n$ con propiedades peculiares, especialmente la de autosemejanza.

Las técnicas sintácticas para generar fractales que se discuten a continuación son una forma agradable y casi natural de familiarizarse con los conjuntos fractales bajo $\mathbf{R}^2$, aunque su generalización a espacios mayores es casi inmediata. Una de las razones de su popularidad es que los objetos que se procesan realmente son símbolos relacionados con primitivas geométricas y no con desarrollos numéricos que pueden ser menos sencillos de entender. La idea es generar mediante ciertas reglas predeterminadas una secuencia de cadenas convergente a un cierto fractal. El estudio de los fractales se traslada de esta forma, independientemente de la dimensión del espacio inicial, al dominio de las palabras infinitas.

Aquí estudiaremos los *sistemas D0L*, que son un tipo particular de *sistemas L*. Los sistemas L fueron introducidos en 1968 por el matemático y biólogo danés Aristid Lindenmeyer con el propósito de simular el crecimiento de organismos vivos. El modelado de organismos a través de los sistemas L permite comprobar ciertas hipótesis relativas a los mecanismos existentes tras determinados fenómenos biológicos.

Tanto los sistemas L como los D0L son estructuras claramente discretas, por lo que cabe preguntarse por su utilidad para aproximarse a conjuntos

fractales. En general, no hay una dualidad directa entre un fractal en $\mathbf{R}^n$ y un modelo discreto; es más, uno espera intuitivamente que muchos conjuntos (los conjuntos de Julia, por ejemplo) no puedan describirse mediante tal modelo discreto. Con todo, los estudios realizados sobre sistemas L aseguran que se pueda capturar mediante modelos sintácticos la estructura fractaliforme de multitud de conjuntos autosemejantes.

2.3. Sistemas D0L

Dado un alfabeto finito Σ, los sistemas D0L generan cadenas autosemejantes al iterar un morfismo respecto a la operación de concatenación $\phi : \Sigma^\star \longrightarrow \Sigma^\star$ (endomorfismo de $\Sigma^\star$) sobre una cadena inicial s para construir la secuencia $\phi(s), \phi^2(s) = \phi(\phi(s)), \phi^3(s) \ldots$ Por ser un morfismo, ϕ viene completamente definido por el conjunto de sus valores $\phi(x)$ para $x \in \Sigma$. A pesar de su sencillez, este modelo computacional permite el cálculo de cadenas con propiedades topológicas complejas como veremos más adelante. Ahora formalicemos la idea anterior.

Definición 2.1 (Sistema DT0L) *Un sistema DT0L es un triplete* $D = (\Sigma, \Phi, s)$ *donde* Σ *es un conjunto finito,* Φ *es un conjunto de* p *morfismos* $\Sigma^\star \longrightarrow \Sigma^\star$ *y* s *es una cadena inicial o axioma.*

Consideremos el conjunto de todas las cadenas que es posible generar mediante la aplicación de cualquier posible secuencia de los morfismos de Φ sobre la cadena inicial s. A este lenguaje lo designaremos por $\mathcal{L}(D)$.

Ejemplo Sea el sistema DT0L $\Gamma = (\{a, b\}, \{\phi_1, \phi_2\}, a)$ con los morfismos definidos por

$$\begin{array}{ll} \phi_1(a) = aba, & \phi_1(b) = aa \\ \phi_2(a) = bab, & \phi_2(b) = b \end{array}$$

Tendremos entonces

$$\begin{aligned} \mathcal{L}(\Gamma) &= \{\phi_1(a), \phi_2(a), \phi_1(\phi_1(a)), \phi_2(\phi_1(a)), \\ &\qquad \phi_1(\phi_2(a)), \phi_2(\phi_2(a)), \phi_1(\phi_1(\phi_1(a))), \ldots\} \\ &= \{aba, bab, abaaaaba, babbbab, aaabaaa, \\ &\qquad bbabb, abaaaabaabaabaabaaaaba, \ldots\} \end{aligned}$$

que es el lenguaje asociado al sistema.

Un sistema D0L es un sistema DT0L con $p = 1$, esto es, con un único morfismo.[1] El 0 en el acrónimo D0L significa que la reescritura es independiente del contexto (la reescritura de un símbolo es independiente de su ubicación en la cadena), la D significa determinista y la L es por el apellido del creador de los sistemas L, Lindenmeyer. En lo siguiente sólo consideraremos sistemas D0L y designaremos por ϕ al morfismo (único) del sistema.

Para dibujar las cadenas de $\mathcal{L}(D)$ utilizaremos una aplicación $K : \Sigma^\star \longrightarrow \mathbf{R}^2$ seguida normalmente de una función de reescalado $L : \mathbf{R}^2 \longrightarrow \mathbf{R}^2$. Por simplificar nos centraremos en representaciones sobre el plano.

Una posibilidad para K es que traduzca cada símbolo de la cadena a vectores unitarios con, posiblemente, diferentes sentidos y cada uno con origen o punto de aplicación en el extremo del vector inmediatamente anterior. La forma de K influirá decisivamente en el tipo de conjuntos que se puedan generar.

El cometido de L es meramente estético. La función L provoca una reducción de escala en cada iteración sucesiva de manera que el conjunto generado queda confinado en una determinada zona del plano. De lo contrario, la expansión de la cadena inicial aumentaría, posiblemente exponencialmente, el tamaño del conjunto generado.

Si la secuencia de curvas $K(\phi(s)), K(\phi^2(s)), K(\phi^3(s)), \ldots$ converge a una curva particular $\mathcal{C}$ (según una métrica apropiada), entonces es razonable considerar la cadena infinita $w = \lim_{n\to\infty} \phi^n(s)$ en lugar de $\mathcal{C}$ e intentar encontrar propiedades combinatorias y topológicas de w en vez de características geométricas en $\mathcal{C}$. Se ha demostrado que bajo ciertas condiciones esta convergencia puede ser garantizada. Nosotros no llegaremos tan lejos. En la siguiente sección mostraremos un enfoque para caracterizar a la aplicación K, ligeramente distinto al sugerido anteriormente con vectores unitarios, que nos permitirá alcanzar el objetivo inicial: generar fractales mediante sistemas D0L.

2.4. Curvas fractales y sistemas D0L

La aplicación K, que transforma las cadenas del lenguaje asociado a un sistema D0L en un conjunto geométrico sobre el plano, nos da la clave para convertir cadenas autosemejantes en fractales. Una de las aproximaciones más sencillas a la modelización de K consiste en interpretar algunos de los

[1] Si en un sistema DT0L la aplicación de los distintos morfismos se lleva a cabo de forma aleatoria puede obtenerse algo similar a los denominados fractales aleatorios, pero aquí nos contentaremos con sistemas D0L y fractales autosemejantes en un sentido estricto.

símbolos de las cadenas del lenguaje generado por un sistema D0L como pautas de comportamiento para una tortuga geométrica al estilo de la del lenguaje de programación Logo.

Ampliemos la definición de sistema D0L para incluir la determinación de un ángulo cuyo significado se verá más adelante. Definamos, por tanto, un *sistema D0L modificado* como $D = (\Sigma, \phi, s, \alpha)$ donde todo es como antes excepto por la aparición de α, que indica un ángulo de giro en radianes. Además, Σ incluirá como mínimo el símbolo F y opcionalmente alguno de los símbolos del conjunto $\{G, +, -, [,]\}$, que tienen para nuestra aplicación K el significado especial mostrado en la tabla 2.1, aunque pueden utilizarse en el morfismo ϕ como cualquier otro símbolo.

Nótese que es posible que ϕ mantenga invariable algún símbolo de Σ haciendo $\phi(x) = x$ para algún $x \in \Sigma$. De hecho, este es el comportamiento más habitual con los símbolos del conjunto $\{+, -, [,]\}$. De forma contraria, es frecuente que $\phi(F) \neq F$ y que $\phi(G) \neq G$.

Ejemplo El sistema D0L $\Gamma_1 = (\{F, G\}, \phi_1, F, \pi)$ con $\phi_1(F) = FGF$ y $\phi_1(G) = GGG$ genera cadenas que cuando son interpretadas según la aplicación K descrita anteriormente convergen al conjunto ternario de Cantor. El lector puede comprobarlo generando manualmente las primeras cadenas del lenguaje. Nótese que en este caso, por tratarse de un fractal plano, el valor del ángulo es indiferente. Se ha mantenido en Γ_1 por consistencia con la definición.

Símbolo	Función
F	avanza un paso la tortuga dibujando
G	avanza un paso la tortuga sin dibujar
+	gira la tortuga a la izquierda α radianes
−	gira la tortuga a la derecha α radianes
[	almacena en una pila la posición y ángulo actual de la tortuga
]	saca de la pila nuevos valores para la posición y el ángulo de la tortuga

Cuadro 2.1: Algunos símbolos del alfabeto del sistema D0L modificado tienen un significado especial cuando son interpretados por la aplicación K. El número de símbolos especiales puede aumentarse para dotar de mayor poder de representación al sistema.

2.5. Instrumentación

Las curvas fractales pueden generarse en la pantalla de un ordenador de muy distintas formas. Dada la autorreferencia que las caracteriza, una forma evidente (y utilizada con bastante frecuencia) es mediante algún algoritmo recursivo. Esta es una solución bastante potente en muchas situaciones, pero implica la elaboración de un programa para cada curva distinta y el aburrido enfrentamiento con errores inherentes a la propia programación y no a la curva en sí. Los sistemas D0L brindan un mecanismo elegante para representar ciertas formas fractales, permitiendo obtener con un único programa multitud de fractales según el sistema D0L suministrado como entrada.

El mecanismo de actuación del programa sería el siguiente: introducido el sistema D0L como entrada al programa, se genera la cadena derivable tras el número de pasos de derivación indicados por el usuario. Dicha cadena se interpreta símbolo a símbolo según la tabla 2.1, generando la curva en pantalla. Aquellos símbolos que aparezcan en la cadena y no sean alguno de los símbolos especiales, simplemente no se interpretan, procediendo automáticamente con el símbolo siguiente de la cadena.

Ejemplo El sistema D0L $\Gamma_2 = (\{F, +, -, [,]\}, \phi_2, ++++F, \pi/8)$ con $\phi_2(F) = FF-[-F+F+F]+[+F-F-F]$ y la identidad como imagen de ϕ_2 para los símbolos distintos de F genera cadenas que convergen a una especie de arbusto fractal.

Vamos a analizar los tres primeros niveles de derivación del sistema D0L Γ_2. En el nivel cero (todavía no se ha realizado ninguna sustitución) la cadena es $++++F$ (la cadena inicial), lo que provoca que la tortuga gire un poco y pinte una recta. Tras la primera derivación la cadena a interpretar es $++++FF-[-F+F+F]+[+F-F-F]$. La segunda derivación hace la cadena un poco más larga, resultando

$$++++FF-[-F+F+F]+[+F-F-F]FF-[-F+F\ldots$$

El lector puede generar la cadena completa e intentar dibujar su interpretación en papel. La cadena generada en el nivel 4 permite obtener la sorprendente imagen que se muestra en la figura 2.1.

Ejemplo El sistema D0L $\Gamma_3 = (\{F, X, +, -\}, \phi_3, FXF--FF--FF, \pi/3)$ con $\phi_3(F) = FF$, $\phi_3(X) = --FXF++FXF++FXF--$ y la identidad como imagen de ϕ_3 para los símbolos restantes genera cadenas que cuando son interpretadas según la aplicación K descrita en la tabla 2.1 convergen al triángulo de Sierpinski.

Otros fractales famosos se generan también de manera sencilla mediante sistemas D0L. La curva de Koch o la de Hilbert son ejemplos de ello.

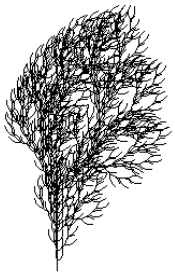


Figura 2.1: Los sistemas D0L son ideales para la modelización de plantas como ésta, obtenida del sistema Γ_2 tras 4 iteraciones. Una de las primeras aplicaciones de estos sistemas fue la representación gráfica de estructuras presentes en la naturaleza.

Ejemplo El sistema D0L $\Gamma_4 = (\{F, +, -\}, \phi_4, F, \pi/3)$ con $\phi_4(F) = F + F - -F + F$, y la identidad como imagen de ϕ_4 para los símbolos restantes genera cadenas que cuando son interpretadas según la aplicación K descrita en la tabla 2.1 convergen a la curva de Koch.

Ejemplo El sistema D0L $\Gamma_5 = (\{F, X, Y, +, -\}, \phi_5, X, \pi/2)$ con $\phi_5(X) = -YF + XFX + FY-$, $\phi_5(Y) = +XF - YFY - FX+$ y la identidad como imagen de ϕ_5 para los símbolos restantes genera cadenas que cuando son interpretadas según la aplicación K de la tabla 2.1 convergen a la curva de Hilbert.

2.6. Un poco de Botánica

El arbusto fractal de la figura 2.1 no es un ejemplo aislado de la aproximación a la naturaleza de los fractales. Aunque operan perfectamente con muchas de las clásicas curvas fractales, los sistemas D0L también producen modelizaciones de plantas, árboles y arbustos de aspecto casi real. Precisamente, éste fue el primer uso que se hizo de los sistemas L (recordemos, un superconjunto de los sistemas D0L) asociado a gráficos por ordenador. Fueron A. R. Smith en 1984 y P. Prusinkiewicz en 1986 los creadores de este método.

En la figura 2.2 se muestran dos plantas más generadas con sistemas D0L. El lector puede intentar encontrar los morfismos que las generan. Aunque no obtenga un sistema exacto, seguro que es capaz de crear un modelo muy similar.

Dentro de los procesos de crecimiento fractal existe uno que emula con gran realismo el crecimiento de muchas especies: la ramificación. La ramificación puede observarse en un gran número de árboles, plantas, algas, musgos, líquenes y corales. Los sistemas D0L permiten generar muchas de estas pautas de ramificación tales como la ramificación dicotómica, la monopódica o la simpódica mediante sistemas extremadamente sencillos.

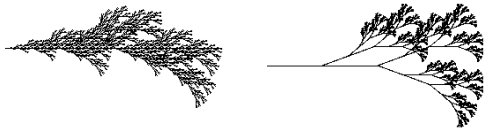


Figura 2.2: Las estructuras fractaliformes modelan con bastante realismo muchos tipos de vegetación. Estas dos plantas, generadas con sistemas D0L como los discutidos en este capítulo, son un ejemplo de ello.

Dentro del cuerpo humano abundan también las estructuras fractaliformes. Las ramificaciones fractales amplían notablemente la superficie de las áreas de absorción como en el intestino, de distribución o recolección como ocurre en los vasos sanguíneos o en los tubos bronquiales, y de proceso de información como en las terminaciones nerviosas. Además, debido a su estructura fractal, la redundancia de operadores similares dota a estas partes de una gran resistencia ante las lesiones. Evidentemente, aun en estos casos, la estructura no es totalmente fractal (la ramificación no se extiende hasta el infinito pues existe un límite determinado, por ejemplo, por el nivel atómico) pero el modelo fractal supone una excelente aproximación.

En el sentido anterior, también es imposible representar curvas fractales por medio de un ordenador (o por cualquier medio) ya que la resolución de pantalla o la memoria disponible imponen un límite al nivel de profundización. En el caso de los sistemas D0L, una cadena w generada será un fractal si y sólo si su longitud es infinita o, lo que es lo mismo, si y sólo si se deriva de la cadena inicial en un número infinito de derivaciones. Esto tiene la consecuencia de que la función que genera un fractal es no computable. De nuevo, las aproximaciones gráficas que podemos obtener por medio de un ordenador son más que suficientes para hacernos una idea del aspecto *final* del fractal.

2.7. Más allá de los sistemas D0L

Los sistemas utilizados pueden complicarse todo lo que uno quiera. Pueden hacerse dependientes del contexto para permitir, por ejemplo, que en la generación de un árbol, una rama demasiado profunda se convierta en una explosión de hojas. Pueden utilizarse distintas aplicaciones K con nuevos símbolos para manejo de color o saltar a los sistemas DT0L. Una de las modificaciones más espectaculares permitiría la generación de curvas tridimensionales: la generalización de muchas de las curvas fractales presentadas a tres dimensiones es casi inmediata. Usamos la expresión tres dimensiones en un sentido amplio, ya que la mayor parte de dichas curvas tendrían una dimensión fractal comprendida entre 2 y 3.

Los sistemas DT0L pueden considerarse como una especie de gramáticas independientes del contexto[2] en las que no hay distinción entre símbolos terminales y no terminales. Una gramática independiente del contexto $\mathcal{G}_D$ en forma normal de Chomsky puede derivarse fácilmente a partir de un sistema DT0L D sustituyendo para cada $a \in \Sigma$ el valor $\phi(a) = w$ por una regla de derivación $a \longrightarrow \phi(a)$. $\mathcal{G}_D$ genera codificaciones más realistas que el sistema DT0L inicial; es más, el modelo puede mejorarse añadiendo probabilidades a las reglas (gramáticas estocásticas).

El problema inverso todavía permanece poco explorado. El problema inverso consiste en calcular el sistema D0L que genera un conjunto fractal dado. Algunos desarrollos se han realizado utilizando gramáticas independientes del contexto (véase, por ejemplo, [BLA 94]). Sin embargo, esta técnica se encuentra muy lejos en cuanto a su aplicación a la compresión de imágenes del modelo matemático en el que se centra este trabajo, los sistemas de funciones iteradas, una evolución de la teoría de conjuntos autosemejantes.

[2]No se describirán aquí las gramáticas. Puede encontrarse más información en el libro de Hopcroft y Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

Capítulo 3

Conjuntos autosemejantes

Nadie ha escapado nunca a la divertida sensación que producen algunos libros en cuya portada un personaje muestra un libro cuya portada es igual a la del primer libro y en la que, por tanto, aparece el mismo personaje sosteniendo un libro con una portada igual a la del libro... Aunque, evidentemente, se trata de un montaje fotográfico y el nivel de profundización no es infinito, no nos resulta complicado imaginar una sucesión interminable del personaje sosteniendo un libro en el que aparece él mismo mostrando la misma portada.

La situación anterior posee en cierta forma estructura fractal, ya que la invarianza a escala y la autosemejanza se manifiestan de manera notoria. Las matemáticas de los conjuntos autosemejantes modelizan el comportamiento anterior y son la pista de despegue hacia nuestro destino: la compresión de imágenes mediante sistemas de funciones iteradas.

En este capítulo se presenta también la teoría de los espacios métricos y las nociones de topología imprescindibles para enfrentarse a cualquier exposición seria sobre fractales. La referencia principal para todo este capítulo es [GUZ 93], aunque también se presentarán ideas de [FIS 95] y [BAR 93a]. Muchas otras de las referencias que aparecen en la bibliografía discuten con mayor o menor profundidad algunos de los conceptos aquí estudiados.

3.1. Modelo matemático de autosemejanza

Desde un punto de vista intuitivo, un *conjunto autosemejante* es aquél que puede ser descompuesto en partes, cada una de las cuales es semejante al conjunto total.

La autosemejanza es una propiedad universalmente extendida en la naturaleza. Se han reconocido rasgos de autosemejanza en fenómenos como las variaciones climatológicas, los flujos en régimen de turbulencia, los precios de un mercado o la formación de masas coralinas. Los fractales que presentan propiedades de autosemejanza en la naturaleza lo suelen hacer en un sentido aleatorio. Aquí, sin embargo, sólo trataremos el caso de autosemejanza en sentido estricto o autodeterminista.

A pesar de que los conjuntos autosemejantes se encuentran entre los primeros tipos de conjuntos fractales conocidos, su estudio sistemático no se produjo hasta la década de los 80. Existen diferentes aproximaciones matemáticas a la noción de autosemejanza. En este capítulo estudiaremos el enfoque más extendido, que parte del crucial trabajo *Fractals and self-similarity* desarrollado por J. Hutchinson en 1981.

Una *semejanza* es la correspondencia que transforma una figura en otra semejante. Una transformación Ψ de $\mathbf{R}^n$ es una semejanza si y sólo si para cierto $r \in \mathbf{R}$ y para cualesquiera $x, y \in \mathbf{R}^n$ se tiene

$$d(\Psi(x), \Psi(y)) = rd(x, y)$$

donde la función d expresa la distancia entre puntos de $\mathbf{R}^n$. Al factor r se le denomina *razón de semejanza* y expresa la reducción o dilatación operada sobre el tamaño de las figuras sobre las que actúa la semejanza. En el capítulo siguiente se dará una descripción detallada de las ecuaciones analíticas de las semejanzas del plano.

Definición 3.1 *Un conjunto $E \subseteq \mathbf{R}^n$ es autosemejante si existe una colección $\Psi_1, \Psi_2, \ldots, \Psi_m$ de semejanzas de $\mathbf{R}^n$, todas ellas con razones menores a la unidad (es decir, contractivas), tales que*

a) $E = \bigcup_{i=1}^{m} \Psi_i(E)$

b) Para cierto s (no necesariamente entero) se tiene que $H^s(E) > 0$ y que $H^s(\Psi_i(E) \cap \Psi_j(E)) = 0$, si $i \neq j$

La condición *a)* indica que el conjunto se obtiene como unión de partes semejantes al total (cada $\Psi_i(E)$ es una de tales partes). La condición *b)* precisa la forma en que los trozos $\Psi_i(E)$ pueden solapar entre sí (solapamiento de las piezas que componen E), exigiendo que este solapamiento sea despreciable en relación a la medida total de E cuando medimos en cierta dimensión s. En la sección A.5 de los apéndices se define la medida de Hausdorff $H^s(E)$; aunque no es un concepto trascendental para el resto del capítulo, sería interesante su estudio detallado, ya que es una idea que subyace en toda la teoría fractal.

La condición *b)* de la definición 3.1 se verifica automáticamente cuando no existe solapamiento (por ejemplo, en el conjunto de Cantor en $\mathbf{R}^2$), pero muchas curvas autosemejantes presentan un cierto contacto entre sus piezas (por ejemplo, en la curva de Koch los solapamientos consisten en un único punto) por lo que no podemos relajar la condición *b)* por la de no solapamiento.

Más adelante veremos que la condición *b)* puede ser substituida por la *condición de abierto*, de utilización más simple que *b)*, pero menos drástica que la exigencia de solapamiento vacío.

3.2. Conjuntos autosemejantes famosos

Como muestra de conjuntos autosemejantes veremos tres conjuntos célebres. En el primer caso, el conjunto de Cantor, los resultados obtenidos en el capítulo 1 nos permitirán demostrar la autosemejanza del conjunto.

Conjunto de Cantor

Consideremos un sistema $\{\Psi_1, \Psi_2\}$ de dos contracciones de $\mathbf{R}$ de ecuaciones $\Psi_1(x) = x/3$ y $\Psi_2(x) = x/3 + 2/3$. La primera transforma $I = [0, 1]$ en el intervalo $[0, 1/3]$, mientras que $\Psi_2(I) = [2/3, 1]$.

Sabemos que el conjunto de Cantor E construido en el capítulo 1 no es otro que el conjunto de los números reales incluidos en I tales que en sus expresiones decimales en base tres sólo figuran ceros y doses. Observemos que si x es uno de tales números, $x/3$ también los es (la división por 3 en base 3 se efectúa corriendo la coma decimal un lugar a la izquierda). También $x/3 + 2/3$ estará en el conjunto de Cantor, ya que ahora tras correr la coma un lugar a la izquierda sumaremos 0,2 (en base 3).

Los conjuntos $\Psi_1(E)$ y $\Psi_2(E)$ resultan ser aquí, respectivamente, aquellos puntos del conjunto de Cantor cuyas expresiones decimales comienzan por 0,0 y aquéllos que comienzan por 0,2. Entre ambos reúnen todos los puntos de E, siendo vacía su intersección. Hemos probado que el conjunto de Cantor es autosemejante con arreglo a la definición dada anteriormente.

Conjunto de Cantor en $\mathbf{R}^2$

Para obtener el conjunto de Cantor en $\mathbf{R}^2$ mostrado en la figura 3.1 partiendo de un cuadrado utilizamos un sistema de cuatro semejanzas: las

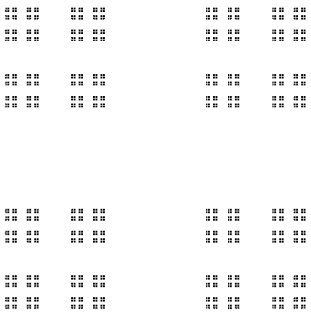


Figura 3.1: El conjunto de Cantor en $\mathbf{R}^2$ es un conjunto autosemejante bajo el sistema de cuatro semejanzas que transforman el cuadrado inicial en cada uno de los cuatro cuadrados de las esquinas.

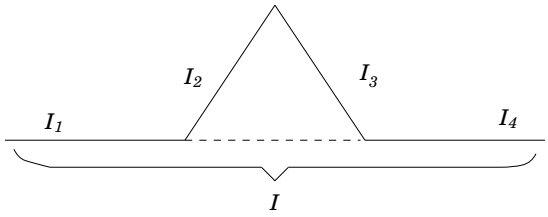


Figura 3.2: La curva de Koch se puede construir sustituyendo el segmento I por los segmentos I_1, I_2, I_3, I_4 y repitiendo en cada uno de ellos este proceso indefinadamente.

que transforman el cuadrado inicial en cada uno de los cuadrados pequeños que ocupan sus cuatro esquinas. Más concretamente, si Ψ_1 y Ψ_2 son las semejanzas del ejemplo anterior, nuestras cuatro semejanzas son ahora

$$\Phi_{ij} = (\Psi_i(x), \Psi_j(y)), \qquad 1 \leq i, j \leq 2$$

El conjunto E se descompone en la unión de cuatro copias semejantes, que son precisamente las $\Phi_{ij}(E)$. En este ejemplo tampoco existe solapamiento entre las copias.

Curva de Koch

Consideremos las cuatro semejanzas del plano que transforman el segmento unitario I en los cuatro segmentos de la figura 3.2.

Puede demostrarse que la curva de Koch es autosemejante respecto a es-

tas cuatro semejanzas, cada una de las cuales tiene razón 1/3. En el capítulo siguiente se dan las ecuaciones exactas de tales semejanzas.

3.3. Espacios métricos

Antes de profundizar en las características de los conjuntos autosemejantes, es necesario mostrar algunos conceptos sobre topología y espacios métricos cuya comprensión es vital no sólo para éste, sino para el resto de capítulos del libro. Aunque en un principio puedan parecer conceptos excesivamente abstractos, se verá más adelante su gran utilidad a la hora de conformar una base teórica sólida de la geometría fractal. El lector con conocimientos suficientes sobre espacios métricos, completitud, compacidad y el teorema del punto fijo puede saltar directamente a la sección 3.4.

Se dice que d es una *métrica* o *distancia* definida en un conjunto X si a cada par de puntos $x, y \in X$ se les puede asignar un número real $d(x, y)$ tal que

1. Para todo $x, y \in X$, $d(x, y) \geq 0$ y $d(x, y) = 0 \Leftrightarrow x = y$
2. Para todo $x, y \in X$, $d(x, y) = d(y, x)$
3. Para todo $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$ *(desigualdad triangular)*

Al par (X, d) se le denomina *espacio métrico*. Un ejemplo característico de espacio métrico es el espacio $\mathbf{R}^n$ con la distancia euclídea habitual

$$d(x, y) = |x - y| = \sqrt{\sum_{k=1}^{n} (x_k^2 - y_k^2)}$$

con $x, y \in \mathbf{R}^n$.

Si (X, d) es un espacio métrico, todo $A \subset X$ admite de forma natural una métrica d_A, dada para $x, y \in A$ por

$$d_A(x, y) = d(x, y)$$

lo que convierte (A, d_A) en un espacio métrico del que se dice es *subespacio métrico* de X.

En un espacio métrico (X, d), dado un punto $x \in X$ y un número real $r > 0$ se define *bola abierta de centro x y radio r* como el conjunto

$$B(x, r) = \{y \in X : d(x, y) < r\}$$

Si en esta definición se cambia $<$ por $\leq$ se obtiene la definición de *bola cerrada con centro x y radio r*.

Un subconjunto A de un espacio métrico se dice *acotado* si está incluido en alguna bola del espacio métrico.

Definición 3.2 *Dado un conjunto acotado $A \subset \mathbf{R}$ no vacío, el supremo de A, que representaremos por* $\sup A$*, cumple las dos condiciones siguientes:*

a) para cualquier $x \in A$ se verifica $x \leq \sup A$

b) dado $\epsilon > 0$, por pequeño que sea, existe $x \in A$ tal que $x > (\sup A) - \epsilon$, es decir, un punto x tan próximo al supremo de A como queramos.

Si el conjunto A es cerrado (incluye a su frontera), no sólo ocurre lo anterior, sino que de hecho existe $x \in A$ tal que $x = \sup A$, y entonces $\sup A = \max A$, esto es, el supremo se convierte en máximo. A partir de esta definición de supremo es sencillo obtener la de ínfimo y mínimo de un conjunto.

Dado un subconjunto acotado A en un espacio métrico (X, d) se define *diámetro* de A como

$$|A| = \sup_{x,y \in A} \{d(x, y)\}$$

Si A y B son conjuntos acotados de X (en particular cuando alguno de ellos se reduce a un punto), se define distancia[1] entre A y B como

$$d(A, B) = \inf_{x \in A, y \in B} \{d(x, y)\}$$

En un espacio métrico (X, d) un conjunto A se llama *abierto* si para cada $x \in A$ hay una bola $B(x, r) \subseteq A$. Un conjunto B se llama *cerrado* si su complementario $X - B$ es abierto.

En un espacio métrico (X, d) dado $A \subseteq X$ se llama *adherencia* de A al conjunto

$$\mathrm{adh}(A) = \{x \in X : \text{para toda bola } B(x, r),\ \ B(x, r) \cap A \neq \emptyset\}$$

La adherencia de un conjunto es el mínimo conjunto cerrado que lo contiene y también

$$\mathrm{adh}(A) = \{x : d(A, x) = 0\}$$

[1] Esta distancia no coincide con la métrica de Hausdorff d_H que veremos más adelante; de hecho, ni siquiera es una métrica según la definición anterior ya que no cumple el apartado 1.

Un conjunto es cerrado si y sólo si coincide con su adherencia.

Dado $A \subseteq X$ se llama *interior* de A al conjunto

$$\mathrm{Int}(A) = \{x \in A : \text{existe } B(x,r) \subseteq A\}$$

El interior de un conjunto es el mayor conjunto abierto contenido en él. Un conjunto es abierto si y sólo si coincide con su interior.

Una sucesión $\{x_n\}$ de puntos de un espacio métrico (X,d) es *convergente* si existe un número x que verifique que para cualquier $\epsilon > 0$ existe un natural N tal que si $n > N$, $d(x,x_n) < \epsilon$. Entonces se escribe $x = \lim_{n\to\infty} x_n$.

Una aplicación $f : X \longrightarrow Y$ entre dos espacios métricos es *continua* en $x \in X$ si para todo $\epsilon > 0$ existe δ tal que

$$d(x,y) < \delta \Longrightarrow d(f(x),f(y)) < \epsilon$$

Si f es continua en todo punto de X, se dice que es continua en X. Una condición necesaria y suficiente para que f sea continua en x es que, para toda $\{x_n\}$ convergente a x, sea

$$\lim f(x_n) = f(\lim x_n) = f(x)$$

Una condición necesaria y suficiente para que f sea continua en X es que para todo $A \subseteq X$ sea

$$f(\mathrm{adh}(A)) \subseteq \mathrm{adh}(f(A))$$

Espacios métricos completos y compactos

En un espacio métrico (X,d) una sucesión $\{x_n\}$ se llama *de Cauchy* si para todo $\epsilon > 0$ existe un N tal que si $p,q > N$, $d(x_p,x_q) < \epsilon$. Toda sucesión convergente es de Cauchy, pero puede haber sucesiones de Cauchy que no sean convergentes. Cuando toda sucesión de Cauchy es convergente a un punto de X, al espacio métrico se le denomina *completo* .

Un espacio métrico es *compacto* si toda sucesión $\{x_n\}$ de puntos de X admite una subsucesión convergente a un punto de X. Son ejemplos característicos de conjuntos compactos los conjuntos cerrados y acotados de $\mathbf{R}^n$. La imagen de un conjunto compacto por una aplicación continua entre espacios métricos es un conjunto compacto.

Aplicaciones contractivas en espacios métricos

Una aplicación $f : X \longrightarrow X$, donde (X,d) es un espacio métrico, es *contractiva* si para $x,y \in X$, $d(f(x),f(y)) \leq k \cdot d(x,y)$ para cierto $k < 1$

llamado *constante de la contracción*, *módulo de la contracción* o *razón de contractividad*. En estas condiciones se verifica la siguiente proposición.

Proposición 3.1 *Si la aplicación $f : X \longrightarrow X$ sobre el espacio métrico X es contractiva, entonces se cumple:*

a) f es continua

b) Si $g : X \longrightarrow X$ es contractiva de módulo k', entonces $g \circ f$ es contractiva de módulo $k \cdot k'$

c) f^n es contractiva de módulo k^n

La demostración es sencilla y puede encontrarse en [GUZ 93, p. 150].

Teorema del punto fijo

Estamos en condiciones ahora de dar un teorema vital para nuestro trabajo, ya que sin él la compresión fractal (al menos tal como hoy la conocemos) no sería posible. Se trata, además, de un resultado muy util en muchas áreas de las matemáticas.

Teorema 3.1 (Teorema del punto fijo) *Si X es un espacio métrico completo y $f : X \longrightarrow X$ es una aplicación contractiva de módulo k, entonces existe un único $x \in X$ denominado punto fijo de la contracción tal que $f(x) = x$.*

Demostración No pueden existir x e y tales que $f(x) = x$ y $f(y) = y$, ya que en tal caso

$$d(f(x), f(y)) = d(x, y)$$

y, sin embargo, la contractividad de f impone

$$d(f(x), f(y)) < d(x, y)$$

Esto prueba que si x existe, es único. □

Teorema 3.2 *Si X es un espacio métrico completo y $f : X \longrightarrow X$ es una aplicación contractiva de módulo k, entonces, si x es el punto fijo de la contracción tal que $f(x) = x$, se tiene que para cualquier $y \in X$*

a) $x = \lim_{n\to\infty} f^n(y)$

b) $d(x,y) \leq \frac{1}{1-k}\, d(y, f(y))$

DEMOSTRACIÓN Veamos en primer lugar la demostración de *a)*. Probaremos a continuación que, dado un $y \in X$ arbitrario, la sucesión cuyo término general es $y_n = f^n(y)$ es de Cauchy.

Para $p \geq 1$ arbitrario

$$d(y_p, y_{p+1}) = d(f(y_{p-1}), f(y_p)) \leq k \cdot d(y_{p-1}, y_p)$$

Aplicando esta fórmula repetidas veces se tiene

$$d(y_p, y_{p+1}) \leq k^p d(y_0, y_1)$$

Para $p < q$ arbitrarios, en virtud de la desigualdad triangular

$$\begin{aligned} d(y_p, y_q) &\leq d(y_p, y_{p+1}) + d(y_{p+1}, y_{p+2}) + \cdots + d(y_{q-1}, y_q) \\ &\leq \sum_{i=p}^{\infty} d(y_i, y_{i+1}) \\ &\leq d(y_0, y_1) \sum_{i=p}^{\infty} k^i \\ &= d(y_0, y_1) \frac{k^p}{1-k} \end{aligned}$$

y esta última expresión se hace más pequeña que un ϵ arbitrario tomando p suficientemente grande.

Como $\{y_n\}$ es de Cauchy en el espacio completo (X, d), debe ser convergente. Si x es su límite, en virtud de la continuidad de f se tiene

$$\begin{aligned} f(x) &= f(\lim_{n\to\infty} y_n) \\ &= \lim_{n\to\infty} f(y_n) \\ &= \lim_{n\to\infty} y_{n+1} \\ &= x \end{aligned}$$

y esto prueba que x es el punto fijo de la contracción y concluye la demostración de *a)*.

Para demostrar *b)*, tomando $p = 0$ en

$$d(y_p, y_q) \leq d(y_0, y_1)\, \frac{k^p}{1-k}$$

se obtiene

$$d(y_0, y_q) \leq \frac{1}{1-k}\, d(y_0, y_1)$$

Tomando límites cuando q tiende a infinito

$$\lim_{q\to\infty} d(y_0, y_q) = d\left(y_0, \lim_{q\to\infty} y_q\right) = d(y_0, x) \leq \frac{1}{1-k}\, d(y_0, f(y_0))$$

donde $y_0 \in X$ es arbitrario. □

3.4. Invarianza respecto a un sistema de semejanzas

Expondremos ahora un resultado que proporciona un criterio más eficaz para probar la autosemejanza de conjuntos al permitir su construcción directa a partir de *sistemas de autosemejanzas*. En los ejemplos de la sección 3.2 hemos encontrado el sistema de semejanzas *a posteriori* basándonos en el conocimiento que teníamos del proceso de construcción de los conjuntos. Además, la demostración rigurosa de la autosemejanza del conjunto de Cantor fue posible porque disponíamos de las expresiones decimales de sus puntos.

Conseguiremos ahora, por tanto, un método flexible para la construcción y caracterización de autosemejantes (con el que se puede probar de forma elemental la autosemejanza de todos los fractales mencionados en el apartado 3.2).

Teorema 3.3 *Dado un sistema $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ de semejanzas contractivas de $\mathbf{R}^n$ (esto es, todas ellas de razón menor a la unidad) existe un único compacto y no vacío $E \subset \mathbf{R}^n$ tal que*

$$E = \bigcup_{i=1}^{m} \Psi_i(E)$$

Observemos que el conjunto E cuya existencia conocemos a partir de cualquier sistema de semejanzas contractivas, dado *a priori*, satisface la condición *a)* de la definición de autosemejante, pero nada podemos asegurar respecto de la condición *b)*. La demostración de este teorema se irá construyendo durante las próximas páginas.

Construcción de teragones

Como ejemplo de la utilización del teorema anterior se muestra la construcción de unas curvas denominadas *teragones*. Se empieza con el llamado

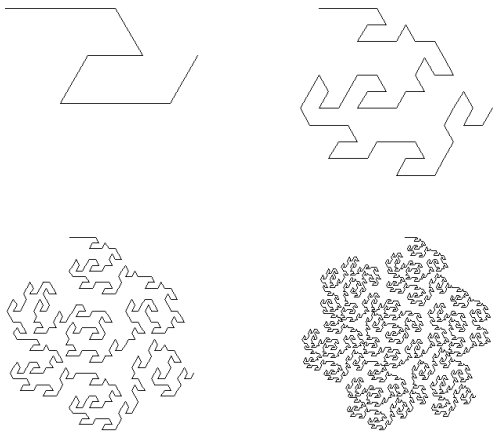


Figura 3.3: Primeras etapas de la construcción de un teragón que llena la isla de Koch. La primera figura es el conjunto generador.

conjunto generador F que es una curva poligonal formada por segmentos rectilíneos colocados de forma consecutiva en el plano.

Sean $x_1, x_2, \ldots, x_{n+1}$ los extremos de estos segmentos. Entonces se selecciona un sistema de semejanzas $\{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ tales que Ψ_k transforma la poligonal F en una poligonal semejante con extremos en x_k y x_{k+1}, siendo Ψ_k contractiva.[2]

Según el teorema 3.3, sabemos que debe existir un conjunto invariante para este sistema de semejanzas. De nuevo es necesario advertir que las curvas así obtenidas no tienen por qué ser conjuntos autosemejantes en el sentido estricto de la definición, ya que, aunque se verifica la condición *a)* nada sabemos de la condición *b)*. Sólo si conseguimos probar que tal condición se verifica (o si se verifica la *condición de abierto* que veremos más adelante), podremos estar seguros de la autosemejanza de la curva fractal. La figura 3.3 muestra las primeras etapas de la generación de un teragón.

3.5. Transformación asociada a un sistema de semejanzas

La demostración del teorema 3.3 requiere ideas y métodos fundamentales para la geometría fractal.

[2]Los teragones pueden también construirse con ayuda de los sistemas D0L del capítulo 2.

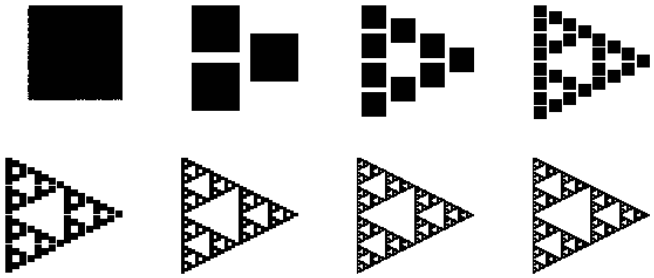


Figura 3.4: La sucesión de imágenes obtenidas mediante una transformación $S\Psi$ se estabiliza hacia el mismo conjunto independientemente del conjunto inicial como puede observarse comparando esta figura con la 3.5.

Comenzaremos definiendo a partir del sistema de semejanzas contractivas $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ una transformación de conjuntos $S\Psi$ tal que a cada $F \subset \mathbf{R}^n$ le hace corresponder el conjunto $S\Psi(F)$ definido por

$$S\Psi(F) = \bigcup_{i=1}^{m} \Psi_i(F)$$

En estos términos el teorema afirma la existencia de un único compacto E tal que $E = S\Psi(E)$. Por tal razón suele llamarse a E *conjunto invariante respecto a la transformación de conjuntos* S o también *respecto al sistema de semejanzas* S.

La idea central que conduce a una demostración constructiva del teorema consiste en explotar las propiedades de la transformación $S\Psi$. La propiedad que nos interesa puede verse en la figuras 3.4 y 3.5. Si partiendo de un conjunto compacto F arbitrario (que puede reducirse a un único punto), obtenemos en un ordenador imágenes de los conjuntos

$$S\Psi(F), S\Psi^2(F) = S\Psi(S\Psi(F)), \ldots, S\Psi^n(F) = S\Psi(S\Psi^{n-1}(F))$$

podemos observar cómo, al aumentar n, la sucesión de imágenes se va estabilizando rápidamente hacia una forma fractal cuyo aspecto es siempre el mismo con independencia del conjunto F de partida.

Supongamos por un momento que, tal como sugieren estos experimentos, tuviera sentido la expresión

$$\lim_{n\to\infty} S\Psi^n(F)$$

y que el límite conmutara con S. En tal caso

$$S\Psi\left(\lim_{n\to\infty} S\Psi^n(F)\right) = \lim_{n\to\infty} S\Psi^{n+1}(F) = \lim_{n\to\infty} S\Psi^n(F)$$

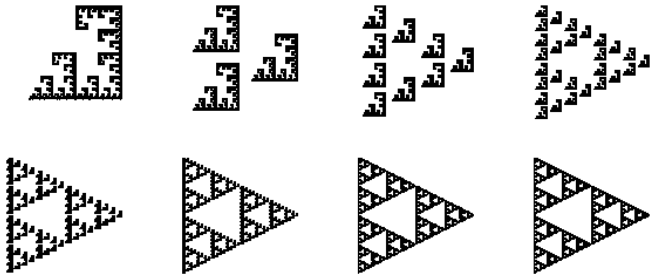


Figura 3.5: La sucesión de imágenes proporcionadas por la iteración de $S\Psi$ converge hacia el mismo fractal con independencia del conjunto inicial sobre el que se aplicó la transformación (¡incluso aunque éste sea fractal!) como puede apreciarse comparando esta figura y la 3.4.

lo que supondría que $\lim_{n\to\infty} S\Psi^n(F)$ sería precisamente el conjunto invariante para el sistema de semejanzas S.

3.6. Espacio $(\mathcal{H}(\mathbf{R}^n), d_H)$

¿Cómo formalizamos la idea anterior? ¿Qué significa límite de una sucesión de conjuntos? La noción de límite está muy vinculada a la de distancia. Para poder definir el límite de una sucesión de conjuntos es necesario hablar previamente de distancia entre conjuntos. No nos vale la distancia usada normalmente en geometría como la distancia entre los puntos más próximos de los conjuntos, ya que si los conjuntos tienen algún punto en común su distancia es cero, aunque sean muy diferentes. La distancia que nosostros precisamos debe ser tal que dos conjuntos próximos respecto a ella sean parecidos entre sí. Tal requisito lo cumple la llamada *distancia de Hausdorff*.

Definición 3.3 *Dados A y B, subconjuntos compactos y no vacíos de $\mathbf{R}^n$, definimos la distancia de Hausdorff entre A y B como*

$$d_H(A,B) = \max\{\max_{x\in B}\{\min_{y\in A} d(x,y)\}, \max_{x\in A}\{\min_{y\in B} d(x,y)\}\}$$

donde $d(x,y)$ expresa la distancia habitual entre puntos de $\mathbf{R}^n$.

Una forma alternativa para definir esta distancia se basa en la noción de cuerpo paralelo-δ a un conjunto. Dado un conjunto $A \subseteq \mathbf{R}^n$ se define su

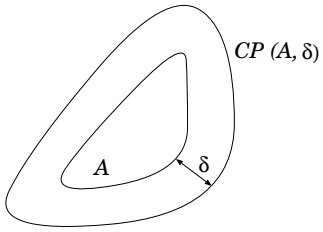
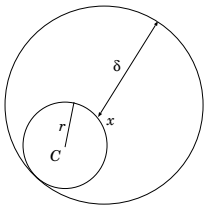


Figura 3.6: El cuerpo paralelo-δ de un conjunto A se define como el conjunto de puntos cuya distancia a A es menor o igual a δ.

cuerpo paralelo-δ $CP(A, \delta)$ como el conjunto de puntos cuya distancia a A es menor o igual a δ como se muestra gráficamente en la figura 3.6. Si el conjunto A es compacto, $CP(A, \delta)$ es la unión de todas las bolas cerradas centradas en A y con radio δ.

Dados dos conjuntos compactos A y $B \subseteq \mathbf{R}^n$, si considaremos el menor δ_1 tal que A está incluido en el cuerpo paralelo-δ_1 a B, y el menor δ_2 tal que B está incluido en el cuerpo paralelo-δ_2 a A, entonces la distancia de Hausdorff entre A y B es el mayor de los dos números δ_1 y δ_2.

Ejemplo La distancia de Hausdorff entre un círculo C de radio r y un punto x de su borde es el diámetro del círculo.



$$\left.\begin{array}{c} x \in CP(C,0) \\ C \subseteq CP(x,\delta) \Rightarrow \delta \geq 2r \end{array}\right\} \Rightarrow d_H(x,C) = 2r$$

La distancia de Hausdorff tiene un significado sencillo: dos conjuntos están próximos cuando tienen parecida forma, tamaño y ubicación.

Ejemplo Sea $C \subset [0,1] \subset \mathbf{R}$ el conjunto clásico de Cantor y sea $C_k = \bigcup_{i=1}^{2^k} I_i^k$, $k \geq 1$, la unión de los 2^k intervalos cerrados de longitud 3^{-k} que se obtienen en el paso k de la construcción inductiva del conjunto de Cantor.

Puesto que $C \subset [0,1]$, la distancia de cualquier punto de C al conjunto $[0,1]$ es cero. Luego, si queremos calcular $d_H(C,[0,1])$ tendremos que hallar la distancia del punto de $[0,1]$ que esté más alejado del conjunto C a dicho conjunto. A la vista de la construcción de C, este punto es

el punto central del intervalo que se elimina en la construcción de C_1, puesto que el resto de puntos de $[0,1]$ que se van quitando en el proceso de construcción de C distan menos que él de dicho conjunto. Luego,

$$d_H(C,[0,1]) = d\left(\frac{1}{2}, C\right) = d\left(\frac{1}{2}, \frac{1}{3}\right) = \left|\frac{1}{2} - \frac{1}{3}\right| = \frac{1}{6}$$

Un razonamiento análogo se puede aplicar para calcular $d(C, C_k)$. Puesto que $C \subset C_k$, tendremos que buscar el punto de C_k que diste una mayor distancia del conjunto C. Este punto será el centro de cualquiera de los intervalos que se eliminen en el proceso de construcción de C_{k+1}. Puesto que todos estos intervalos tienen longitud $3^{-(k+1)}$ y sus extremos pertenecen a C se tiene

$$d_H(C, C_k) = \frac{1}{2} \cdot 3^{-(k+1)} = \frac{1}{6 \cdot 3^k}$$

Puede demostrarse fácilmente que la distancia de Hausdorff cumple las propiedades esenciales de toda distancia.

La distancia de Hausdorff nos permite utilizar el espacio métrico $(\mathcal{H}(\mathbf{R}^n), d_H)$ cuyos puntos serán los subconjuntos compactos y no vacíos de $\mathbf{R}^n$, separados por la distancia d_H.

Podemos ahora hablar de límite de sucesiones, con lo que ya tiene todo su sentido formular nuestra conjetura: el conjunto invariante para el sistema de semejanzas es el límite de la sucesión $\{S\Psi^n(F)\}$, donde F es un compacto no vacío arbitrario.

El espacio $(\mathcal{H}(\mathbf{R}^n), d_H)$ posee, además, una valiosa propiedad, la completitud; no en todos los espacios métricos las sucesiones que se estabilizan convergen a un límite.

Ejemplo La sucesión $1, 3/2, 7/5, \ldots, p/q, (p+2q)/(p+q), \ldots$ en el espacio métrico $(\mathbf{Q}, d(x,y) = |x-y|)$ se estabiliza (sus términos son cada vez más parecidos), pero converge a $\sqrt{2}$ que no es racional.

Los espacios métricos en los que las sucesiones que se estabilizan (sucesiones de Cauchy) convergen a un límite, se llaman *espacios métricos completos*. El llamado *teorema de selección de Blaschke* garantiza que el espacio $(\mathcal{H}(\mathbf{R}^n), d_H)$ es completo.

3.7. Teorema del punto fijo

Una aplicación contractiva $f : \mathbf{R}^n \longrightarrow \mathbf{R}^n$, por ser continua, transforma cada compacto A en un compacto $f(A)$, induciendo una aplicación

$f : \mathcal{H}(\mathbf{R}^n) \longrightarrow \mathcal{H}(\mathbf{R}^n)$. Se verifica la siguiente proposición.

Proposición 3.2 *Si $f : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ es contractiva de módulo k, la aplicación inducida $f : \mathcal{H}(\mathbf{R}^n) \longrightarrow \mathcal{H}(\mathbf{R}^n)$ es contractiva de módulo k en $(\mathcal{H}(\mathbf{R}^n), d_H)$.*

La demostración de esta proposición puede encontrarse en [GUZ 93, p. 150]. Podemos retomar ahora el teorema 3.1 adecuándolo a nuestros intereses.

Teorema 3.4 (Teorema del punto fijo) *Si (X, d) es un espacio métrico completo y Φ es una transformación contractiva en X, es decir, si existe una constante de contracción $k < 1$ tal que para un par arbitrario x, y de puntos de X es*

$$d(\Phi(x), \Phi(y)) \leq kd(x, y)$$

entonces existe un único $x \in X$ tal que $\Phi(x) = x$, y, además, dado cualquier $y \in X$

$$x = \lim \Phi^n(y)$$

Bastaría ahora con probar que $S\Psi$ es una transformación contractiva en $(\mathcal{H}(\mathbf{R}^n), d_H)$ para probar la existencia de un único compacto E tal que $S\Psi(E) = E$. Puesto que las semejanzas contractivas son aplicaciones contractivas (para las cuales la desigualdad $\leq$ se convierte en igualdad), este resultado depende ya exclusivamente de la siguiente proposición.

Proposición 3.3 *Dada una colección $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ de transformaciones contractivas de $\mathbf{R}^n$, la transformación $S\Psi : \mathcal{H}(\mathbf{R}^n) \longrightarrow \mathcal{H}(\mathbf{R}^n)$ definida por*

$$S\Psi(F) = \bigcup_{i=1}^{m} \Psi_i(F)$$

es contractiva en $(\mathcal{H}(\mathbf{R}^n), d_H)$ con módulo de contracción igual al máximo de los módulos de contracción de las Ψ_i, $1 \leq i \leq m$.

La demostración puede encontrarse en [GUZ 93, p. 151]. En el capítulo siguiente se muestran numerosos ejemplos de aplicaciones contractivas. En definitiva, podemos concluir el siguiente resultado.

Teorema 3.5 *Dada una colección $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ de semejanzas contractivas, el conjunto E tal que*

$$S\Psi(E) = \bigcup_{i=1}^{m} \Psi_i(E) = E$$

verifica

$$E = \lim_{n\to\infty} S\Psi^n(F)$$

siendo F cualquier compacto de $\mathbf{R}^n$ no vacío.

Este resultado nos proporciona un medio constructivo para la obtención del conjunto invariante para $S\Psi$ y será fundamental para la construcción de conjuntos autosemejantes.

3.8. Condición de abierto

Retomemos nuestro camino volviendo a considerar la parte *b)* de la definición de conjunto autosemejante. Daremos una forma más sencilla de asegurar el cumplimiento de la definición de autosemejanza basada en la condición de abierto.

Definición 3.4 *Se dice que el sistema $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ de semejanzas de $\mathbf{R}^n$ cumple la condición de abierto si existe un conjunto acotado y abierto $V \in \mathbf{R}^n$ tal que*

$$S\Psi(V) \subset V \quad y \quad \Psi_i(V) \cap \Psi_j(V) = \emptyset \quad si \quad i \neq j$$

Resulta sencillo fabricar ejemplos que verifiquen la condición de abierto. Tómese para ello, en este caso, un triángulo equilatero F como el de la figura 3.7. Formemos imágenes semejantes a él tales que estén incluidas en F, pero que se solapen entre sí como máximo en puntos del borde. En nuestro ejemplo consideraremos los tres triángulos F_1, F_2 y F_3, que solapan solamente en tres vértices.

En tales condiciones el abierto V resultante de quitar a F el borde verifica la condición de abierto respecto de las semejanzas que transforman en los F_i al conjunto F. El conjunto invariante para el sistema de semejanzas así construido es, en nuestro caso, el triángulo de Sierpinski.

3.9. Red de recubrimientos básicos

Veamos cómo puede ser utilizada sistemáticamente la condición de abierto para construir el conjunto invariante E en un proceso de selección por étapas. Utilizamos el compacto $F = \mathrm{adh}(V)$ para obtener el conjunto invariante E como límite de $S\Psi^n(F)$.

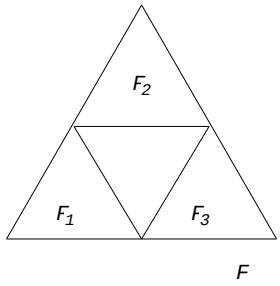


Figura 3.7: El abierto V resultante de quitar el borde a F verifica la condición de abierto respecto a las semejanzas que transforman F en cada una de las tres partes indicadas. El conjunto invariante para este sistema de semejanzas es el triángulo de Sierpinki.

Antes hagamos una observación importante. No es posible obtener E directamente como límite de $S\Psi^n(V)$ porque V no es compacto. Podríamos preguntarnos por qué no exigimos directamente que V sea compacto. La razón es que en muchas ocasiones tal compacto no puede encontrarse (por ejemplo, en la curva de Koch o en el triángulo de Sierpinski). En estos casos sucede que las piezas que forman V no tienen solapamiento, pero sí existe solapamiento en sus fronteras.

Ahora, usando la condición de abierto, puede probarse la siguiente proposición.

Proposición 3.4 *Dada el sistema de semejanzas* $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ *que cumple la condición de abierto respecto al conjunto* V *y dado* $F = \mathrm{adh}(V)$ *se cumple que*

$$F \supset S\Psi(F)$$

Demostración Sabemos por definición que $S\Psi(F) = \bigcup_{i=1}^{m} \Psi_i(F)$. Ahora bien, como cada Ψ_i es una semejanza y por tanto continua (ver sección 3.3), es $\Psi_i(F) = \Psi_i(\mathrm{adh}(V)) \subseteq \mathrm{adh}(\Psi_i(V))$, de donde

$$\begin{aligned} S\Psi(F) &\subseteq \bigcup_{i=1}^{m} \mathrm{adh}(\Psi_i(V)) \\ &= \mathrm{adh}\left(\bigcup_{i=1}^{m} \Psi_i(V)\right) \\ &\subset \mathrm{adh}(V) \\ &= F \end{aligned}$$

Observemos que $S\Psi(F)$ es un conjunto formado por m piezas de la forma

$\Psi_i(F)$ semejantes, por tanto, a F, y todas ellas incluidas en F. □

Tomando en la relación $S\Psi(F) \subset F$ que acabamos de demostrar imágenes por $S\Psi$ en ambos miembros sucesivas veces se obtiene $F \supset S\Psi(F) \supset S\Psi^2(F) \ldots \supset S\Psi^k(F) \supset \ldots$

Consideremos el conjunto

$$E = \bigcap_{i=1}^{\infty} S\Psi^i(F)$$

Tomando imágenes por $S\Psi$ se obtiene

$$\begin{aligned} S\Psi(E) &= \bigcap_{i=1}^{\infty} S\Psi^{i+1}(F) \\ &= F \cap \bigcap_{i=1}^{\infty} S\Psi^i(F) \\ &= F \cap E \\ &= E \end{aligned}$$

de lo que se deduce que E es precisamente el conjunto invariante para el sistema de semejanzas S. La condición de abierto proporciona pues el tipo de proceso geométrico de construcción que buscábamos.

Profundizaremos ahora en este proceso, investigando más detalladamente las propiedades de la colección de recubrimientos $S\Psi^k(F)$ a la que llamaremos *red de recubrimientos básicos* (sabemos que para todo k es $E \subset S\Psi^k(F)$, luego tales conjuntos son, efectivamente, recubrimientos de E que actúan como una colección de filtros con poros cada vez más finos; tras filtrar al conjunto F a través de todos ellos, lo que resta es el conjunto E). Cada $S\Psi^k(F)$ es una *aproximación* a E (recuérdese que E se obtiene como límite de los $S\Psi^k(F)$).

Para $S\Psi^2(F)$ se tiene

$$\begin{aligned} S\Psi^2(F) &= S\Psi\left(\bigcup_{i=1}^{m} \Psi_i(F)\right) \\ &= \bigcup_{j=1}^{m} \Psi_j\left(\bigcup_{i=1}^{m} \Psi_i(F)\right) \\ &= \bigcup_{1\le i,j\le m} \Psi_j \circ \Psi_i(F) \end{aligned}$$

que, si convenimos en definir $F_{i,j} = \Psi_i \circ \Psi_j(F)$ para cada i, j, se escribirá

$$S\Psi^2(F) = \bigcup_{1\le i,j\le m} F_{i,j}$$

Esta relación se generaliza para cualquier k

$$S\Psi^k(F) = \bigcup_{i_1,i_2,\ldots,i_k \in A_k} F_{i_1,i_2,\ldots,i_k}$$

donde A_k representa el conjunto de todas las posibles sucesiones de k números enteros comprendidos entre 1 y m, y, como antes,

$$F_{i_1,i_2,\ldots,i_k} = \Psi_{i_1} \circ \Psi_{i_2} \circ \ldots \Psi_{i_k}(F) \tag{3.1}$$

Este proceso consiste, por tanto, en partir de un conjunto inicial F del que seleccionamos en una primera etapa los trozos $\Psi_i(F)$, $1 \leq i \leq m$, que componen el conjunto $S\Psi(F)$. En la siguiente etapa seleccionamos en cada $\Psi_i(F)$ los m trozos que componen $S\Psi(\Psi_i(F))$, y así sucesivamente.

Los recubrimientos $S\Psi^k(F)$ están formados por piezas de la forma $F_{i_1,i_2,\ldots,i_k}$ que son semejantes a F a través de la cadena de semejanzas indicadas en la ecuación 3.1. Podemos, por lo tanto, calcular con exactitud cuál es la razón de la semejanza resultante, ya que en aplicación de sucesivas semejanzas se multiplican las razones. Esto permite el cálculo exacto del diámetro de cada pieza

$$|F_{i_1,i_2,\ldots,i_k}| = r_{i_1} r_{i_2} \cdots r_{i_k} |F|$$

y, por consiguiente, los diámetros de todas las piezas deben tender a cero, ya que las razones son menores que uno.

3.10. Dimensión de Hausdorff de conjuntos autosemejantes

Obsérvese cómo, en las construcciones en las que se verifica la condición de abierto, el solapamiento de los trozos $\Psi_i(F)$ se produce exclusivamente en las fronteras de los mismos, es decir, los solapamientos son pequeños en medida en comparación con F. Es natural conjeturar que en el límite también será pequeña la medida del solapamiento de los $\Psi_i(E)$ en relación a la de E si medimos el conjunto E en la dimensión adecuada; es decir, que la condición de abierto implica la condición *b)* de la definición de conjunto autosemejante.

El siguiente teorema da toda su potencia y utilidad al método de construcción de fractales que hemos desarrollado en este capítulo.

Teorema 3.6 *Sea $S = \{\Psi_1, \Psi_2, \ldots, \Psi_m\}$ un sistema de semejanzas contractivas de $\mathbf{R}^n$ con razones r_i, $1 \leq i \leq m$, que verifican la condición de*

abierto. Entonces, el compacto E invariante para S es autosemejante y tanto su dimensión de Hausdorff como su dimensión fractal (véase el apéndice A) son iguales y vienen dadas por el único número real no negativo s, tal que

$$1 = \sum_{i=1}^{m} (r_i)^s$$

verificándose, además, que $0 < H^s(E) < \infty$.

La demostración de este teorema es bastante laboriosa y puede encontrarse en [GUZ 93, p. 112].

Comprobemos cómo puede ser utilizado este teorema para hallar la dimensión de Hausdorff de fractales autosemejantes que conocemos. En el caso del triángulo de Sierpinski S tenemos tres semejanzas de razón $1/2$ con lo que la ecuación anterior se convierte en

$$1 = 3\left(\frac{1}{2}\right)^s$$

de donde $1/3 = (1/2)^s \Rightarrow \log 1/3 = s \log 1/2 \Rightarrow -\log 3 = -s \log 2$ lo que nos da una dimensión $\dim(S) = s = \log 3/\log 2$. De igual forma puede obtenerse que la dimensión de la curva de Koch es $\log 4/\log 3$ y que $\log 2/\log 3$ es la del conjunto de Cantor en $\mathbf{R}$.

Capítulo 4

Sistemas de funciones iteradas

Como ya vimos en el capítulo anterior, J. E. Hutchinson fue en 1981 el primer matemático que, estudiando las propiedades comunes (compacidad, autosemejanza etc.) de los fractales ya conocidos, elaboró una teoría unificada para la obtención de una amplia clase de conjuntos fractales: los conjuntos autosemejantes.

En 1985, M. F. Barnsley generalizó el método de Hutchinson. Mientras que éste utiliza semejanzas contractivas, Barnsley utiliza aplicaciones contractivas, lo que permite ampliar notablemente la familia de fractales obtenidos, de la que ahora los conjuntos autosemejantes son un subconjunto.

El método de Barnsley para la generación de conjuntos fractales que vamos a presentar, los sistemas de funciones iteradas, se mostrará sobre $\mathbf{R}^n$, que es el espacio natural en que lo vamos a aplicar. Debe quedar claro, en todo caso, que los desarrollos presentados son aplicables en cualquier espacio métrico completo.

La referencia principal sobre los sistemas de funciones iteradas es [BAR 93a]. También seguiremos en este capítulo a [GUZ 93]. Además, es posible encontrar exposiciones más o menos profundas sobre el tema en muchas otras de las referencias de la bibliografía.

4.1. El espacio de los fractales

Debe tenerse muy en cuenta que durante este capítulo (y sólo durante él) consideraremos fractal en sentido amplio a todo conjunto compacto, es decir, a cualquier conjunto no vacío acotado y que contenga a su frontera. Esta consideración surge del hecho de poder unificar bajo un nombre común a todos los conjuntos que se pueden derivar de un sistema de funciones iteradas, independientemente de que posean o no estructura fractal. De cualquier modo, los resultados obtenidos serán aplicados únicamente a los auténticos conjuntos fractales.

Llamaremos, por tanto, *fractal* a cualquier subconjunto compacto y no vacío de $\mathbf{R}^n$ y *espacio de los fractales*, o espacio donde van a vivir los fractales, de $\mathbf{R}^n$ al conjunto de todos los fractales de dicho espacio, es decir, al conjunto

$$\mathcal{H}(\mathbf{R}^n) = \{K : K \subset \mathbf{R}^n,\ K \neq \emptyset \text{ y } K \text{ es compacto}\}$$

Puesto que aquí vamos a tratar sobre el problema de aproximar objetos naturales (fractales en un cierto espacio $\mathbf{R}^n$) mediante fractales que nosotros podamos generar, es necesario disponer de una métrica que nos dé la distancia entre elementos del espacio $\mathcal{H}(\mathbf{R}^n)$. Nosotros consideraremos la métrica de Hausdorff d_H que ya vimos en 3.6. Como ya se dijo allí, d_H es una métrica sobre el espacio $\mathcal{H}(\mathbf{R}^n)$ y el espacio de los fractales $(\mathcal{H}(\mathbf{R}^n), d_H)$ es un espacio métrico completo.

4.2. Aplicaciones contractivas

Para construir un fractal autosemejante partíamos de un número finito de transformaciones que eran semejanzas contractivas; aquí vamos a estudiar lo que ocurre cuando el conjunto de transformaciones está formado por aplicaciones de una clase mucho más amplia: *aplicaciones contractivas*. Como ya vimos en el capítulo anterior, toda aplicación contractiva es continua e induce una aplicación contractiva en el espacio métrico completo $(\mathcal{H}(\mathbf{R}^n), d_H)$ de igual razón.

Intuitivamente una aplicación contractiva $f : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ es aquella que acerca los puntos y contrae las figuras como se refleja en la figura 4.1.

Entre dos figuras semejantes y distintas del plano euclídeo siempre existe una aplicación contractiva que transforma la mayor en la menor. Esta aplicación contractiva es una composición de *isometrías* (traslaciones, giros

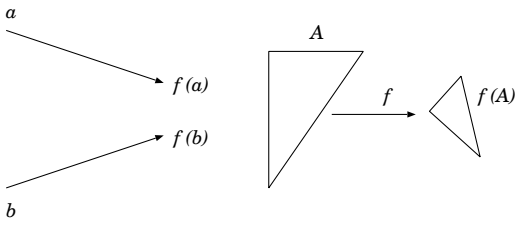
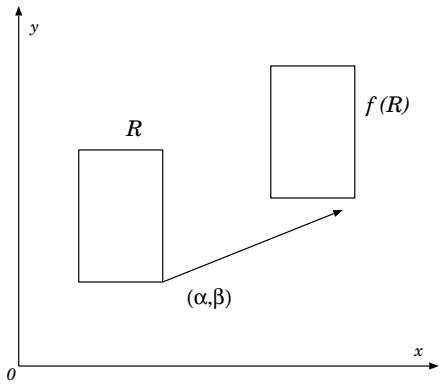


Figura 4.1: Una aplicación contractiva f acerca los puntos y contrae, por tanto, los conjuntos sobre los que se aplica.

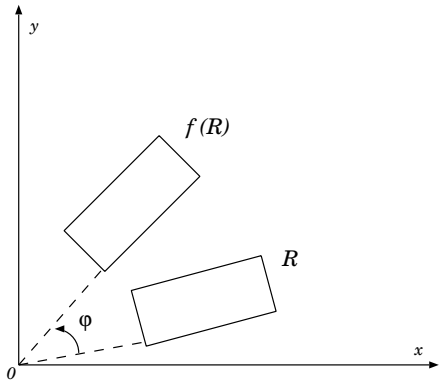
y simetrías) y una *homotecia contractiva*. A continuación se muestran las transformaciones elementales del plano euclídeo. Cualquier otro giro, simetría u homotecia se puede obtener por composición de las transformaciones elementales siguientes.

1. Traslación de vector (α, β):



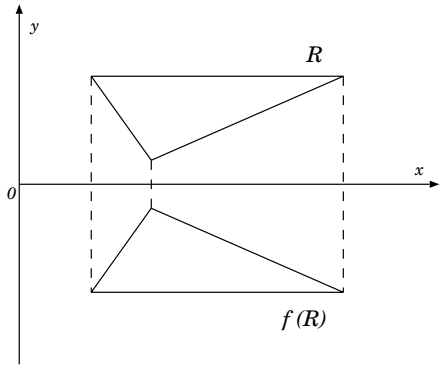
$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

2. Giro del ángulo φ y centro en el origen:



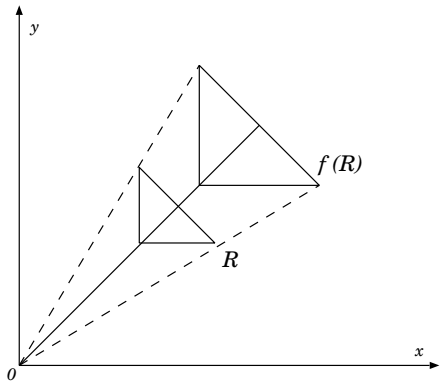
$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\operatorname{sen}\varphi \\ \operatorname{sen}\varphi & \cos\varphi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

3. Simetría respecto del eje de abcisas:



$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

4. Homotecia centrada en el origen de razón k:



$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Si las isometrías y homotecias que definen una aplicación contractiva son fáciles de determinar, entonces dicha aplicación se puede obtener como composición de las mismas. Si, por el contrario, son difíciles de determinar, se puede proceder directamente a calcular la semejanza teniendo en cuenta que toda semejanza es una transformación afín y que, por tanto, sus ecuaciones son

$$f(x, y) = f(ax + by + e, cx + dy + f)$$

o bien

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

Para determinar los coeficientes a, b, c, d, e y f se procede a determinar las imágenes de tres puntos y a resolver el correspondiente sistema de seis

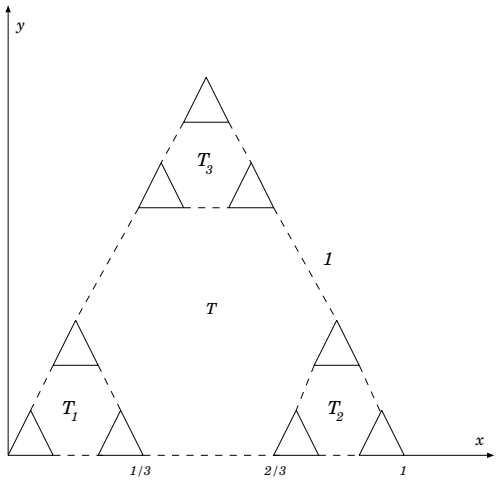


Figura 4.2: Cada parte T_i, $1 \leq i \leq 3$, del triángulo de Sierpinski es semejante al triángulo total T.

ecuaciones con seis incógnitas que nos dará sus valores:

$$\begin{aligned} f(x_1, y_1) &= (ax_1 + by_1 + e, cx_1 + dy_1 + f) = (x_1', y_1') \\ f(x_2, y_2) &= (ax_2 + by_2 + e, cx_2 + dy_2 + f) = (x_2', y_2') \\ f(x_3, y_3) &= (ax_3 + by_3 + e, cx_3 + dy_3 + f) = (x_3', y_3') \end{aligned}$$

Veamos algunos ejemplos.

Ejemplo Consideremos el triángulo de Sierpinski $T \subset \mathbf{R}^2$ y la representación de la figura 4.2.

Podemos poner $T = T_1 \cup T_2 \cup T_3$, siendo T_1, T_2 y T_3 las partes del triángulo de Sierpinski que caen dentro de los tres triángulos de lado $1/3$ que aparecen en la figura. Cada parte T_i, $1 \leq i \leq 3$, del triángulo de Sierpinski es semejante al conjunto total T. Luego existirán semejanzas contractivas f_1, f_2 y f_3 tales que $f_i(T) = T_i$, $1 \leq i \leq 3$, que hacen que

$$T = f_1(T) \cup f_2(T) \cup f_3(T)$$

Vamos a determinar esas transformaciones. La semejanza f_1 es una homotecia de centro el origen y razón $1/3$, luego

$$f_1(x, y) = \left(\frac{x}{3}, \frac{y}{3}\right)$$

o matricialmente

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

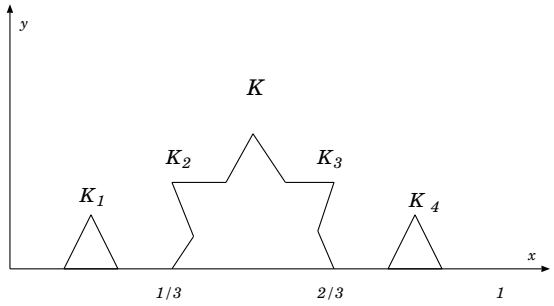


Figura 4.3: Cada una de las partes K_i, $1 \leq i \leq 4$, de la curva de Koch indicadas es semejante a la curva total K.

La semejanza f_2 es una homotecia de centro el origen y razón $1/3$, seguida de una traslación de vector $(2/3, 0)$, luego

$$f_2(x, y) = \left(\frac{x}{3}, \frac{y}{3}\right) + \left(\frac{2}{3}, 0\right)$$

o bien

$$f_2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{2}{3} \\ 0 \end{pmatrix}$$

y la semejanza f_3 es una homotecia de centro el origen y razón $1/3$ seguida de una traslación de vector $((2/3)\cos 60°, (2/3)\operatorname{sen} 60°)$, luego

$$f_3(x, y) = \left(\frac{x}{3}, \frac{y}{3}\right) + \left(\frac{2}{3}\cos 60°, \frac{2}{3}\operatorname{sen} 60°\right)$$

o en forma matricial

$$f_3 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{2}{3} \begin{pmatrix} \cos 60° \\ \operatorname{sen} 60° \end{pmatrix}$$

Es fácil observar que cada una de las aplicaciones contractivas f_i, $1 \leq i \leq 3$, tiene razón $1/3$.

Ejemplo

Consideremos la curva de Koch $K \subset \mathbf{R}^2$. Entonces $K = \bigcup_{i=1}^{4} K_i$ siendo K_i, $1 \leq i \leq 4$, las partes de la curva de Koch que se indican en la figura 4.3 y que son semejantes a la curva total K. Luego existirán semejanzas contractivas f_i, $1 \leq i \leq 4$, tales que $f_i(K) = K_i$ y, por tanto, tales que

$$K = \bigcup_{i=1}^{4} f_i(K)$$

Vamos a determinar estas transformaciones. La semejanza f_1 es una homotecia de centro el origen y razón $1/3$, luego

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

o lo que es lo mismo

$$f_1(x,y) = \left(\frac{x}{3}, \frac{y}{3}\right)$$

La semejanza f_2 es una homotecia de centro el origen y razón $1/3$, seguida de un giro de centro el origen y ángulo $60°$ y seguido de una traslación de vector $(1/3, 0)$, luego

$$f_2\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos 60° & -\operatorname{sen} 60° \\ \operatorname{sen} 60° & \cos 60° \end{pmatrix}\begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{3} \\ 0 \end{pmatrix}$$

y desarrollando

$$f_2(x,y) = \left(\frac{x\cos 60° - y\operatorname{sen} 60° + 1}{3}, \frac{x\operatorname{sen} 60° + y\cos 60°}{3}\right)$$

La semejanza f_3 es una homotecia de centro el origen y razón $1/3$, seguida de un giro de centro el origen y ángulo $-60°$ y seguida de una traslación de vector $(\frac{\sqrt{3}}{3}\cos 30°, \frac{\sqrt{3}}{3}\operatorname{sen} 30°) = (1/2, \sqrt{3}/6)$, luego

$$f_3\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(-60°) & -\operatorname{sen}(-60°) \\ \operatorname{sen}(-60°) & \cos(-60°) \end{pmatrix}\begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{6} \end{pmatrix}$$

y desarrollando

$$f_3(x,y) = \left(\frac{x\cos 60° + y\operatorname{sen} 60°}{3} + \frac{1}{2}, \frac{-x\operatorname{sen} 60° + y\cos 60°}{3} + \frac{\sqrt{3}}{6}\right)$$

Por último, la semejanza f_4 es una homotecia de centro el origen y razón $1/3$, seguida de una traslación de vector $(2/3, 0)$, luego

$$f_4\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{2}{3} \\ 0 \end{pmatrix}$$

o bien

$$f_4(x,y) = \left(\frac{x+2}{3}, \frac{y}{3}\right)$$

Todas las aplicaciones f_i, $1 \leq i \leq 4$, son contractivas de razón $1/3$.

Si $f : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ es una aplicación contractiva, entonces la aplicación $f : \mathcal{H}(\mathbf{R}^n) \longrightarrow \mathcal{H}(\mathbf{R}^n)$ es también contractiva. Aplicando el teorema del punto fijo de la página 40 a la aplicación f en $\mathbf{R}^n$ existirá un único punto $x_f \in \mathbf{R}^n$ tal que $f(x_f) = x_f$ y aplicándolo a f en $(\mathcal{H}(\mathbf{R}^n), d_H)$ existirá un único conjunto $K_f \subset \mathbf{R}^n$ compacto y no vacío $K_f \in \mathcal{H}(\mathbf{R}^n)$ tal que $f(K_f) = K_f$ y

$$\lim_{k\to\infty} f^k(B) = K_f, \text{ para todo } B \in \mathcal{H}(\mathbf{R}^n)$$

en la métrica de Hausdorff.

Una familia finita de aplicaciones contractivas definidas sobre un mismo espacio $\mathbf{R}^n$ es lo que llamaremos un *sistema de funciones iteradas*. Más concretamente:

Definición 4.1 *Llamaremos sistema de funciones iteradas (SFI) en* $\mathbf{R}^n$ *a cualquier familia finita* $\{f_i\}_{i=1}^N$ *de aplicaciones contractivas* $f_i : \mathbf{R}^n \longrightarrow \mathbf{R}^n$, $1 \leq i \leq N$. *Tal sistema de funciones iteradas se representará por*

$$\{f_1, f_2, \ldots, f_N\}$$

y llamaremos razón de contractividad del SFI a

$$r = \text{máx}\{r_1, r_2, \ldots, r_N\}$$

donde r_i, $0 \leq r_i < 1$, *es la razón de contractividad de* f_i *(obviamente,* $0 \leq r < 1$*).*

> **Ejemplo** En $\mathbf{R}^2$, sea f_i la aplicación contractiva que transforma el triángulo de Sierpinski $T = T_1 \cup T_2 \cup T_3$ en T_i, $1 \leq i \leq 3$, según la figura 4.2. Estas aplicaciones son las presentadas en el ejemplo de la pagina 59.
>
> Entonces $\{f_1, f_2, f_3\}$ es un SFI y, puesto que la razón de cada una de las aplicaciones contractivas f_i, $1 \leq i \leq 3$, es $1/3$, la razón de contractividad de este SFI es $1/3$.

> **Ejemplo** En $\mathbf{R}^2$ sea f_i la aplicación contractiva que transforma la curva de Koch $K = K_1 \cup K_2 \cup K_3 \cup K_4$ en K_i, $1 \leq i \leq 4$, de la figura 4.3. Más concretamente, cada f_i responde a la forma dada en el ejemplo de la página 60.
>
> Ahora $\{f_1, f_2, f_3, f_4\}$ es un SFI y, puesto que la razón de cada una de las aplicaciones contractivas f_i, $1 \leq i \leq 4$, es $1/3$, la razón de contractividad de este SFI es también $1/3$.

En los dos ejemplos anteriores existe un conjunto que es igual a la unión de sus imágenes obtenidas al aplicarle cada una de las aplicaciones contractivas. En el primer caso, si $T \subset \mathbf{R}^2$ es el triángulo de Sierpinski, se cumple que $T = \bigcup_{i=1}^3 f_i(T)$; en el segundo, si $K \subset \mathbf{R}^2$ es la curva de Koch, $K = \bigcup_{i=1}^4 f_i(K)$.

Lo anterior nos sugiere plantearnos las siguientes cuestiones. Dado un SFI $\{f_1, f_2, \ldots, f_N\}$ en $\mathbf{R}^n$,

1. ¿Existirá un conjunto $A \subset \mathbf{R}^n$ tal que $A = \bigcup_{i=1}^N f_i(A)$? (invariante respecto del SFI) y, si la respuesta a esta pregunta es afirmativa,

2. ¿será único?, ¿cómo se obtendrá?

Si $\{f_1, f_2, \ldots, f_N\}$ es un SFI de razón r y $K \subset \mathbf{R}^n$ es un conjunto compacto no vacío, entonces $f_i(K)$, $1 \leq i \leq N$, será también, por la continuidad de f_i, compacto y no vacío.

Puede demostrarse que la unión finita de conjuntos compactos es un conjunto compacto. Con ello se tendrá que la aplicación

$$F : \mathcal{H}(\mathbf{R}^n) \longrightarrow \mathcal{H}(\mathbf{R}^n)$$

definida por

$$F(K) = \bigcup_{i=1}^{N} f_i(K), \quad \forall K \in \mathcal{H}(\mathbf{R}^n)$$

está bien definida.

Los cuestiones planteadas anteriormente se traducen ahora, en el contexto de la función F, en el estudio de la existencia y unicidad de algún punto fijo para esta aplicación, es decir, de algún conjunto $A \in \mathcal{H}(\mathbf{R}^n)$ tal que

$$F(A) = \bigcup_{i=1}^{N} f_i(A) = A$$

Como ya vimos en el capítulo anterior, la aplicación F es contractiva de razón r en el espacio métrico completo $(\mathcal{H}(\mathbf{R}^n), d_H)$. Aplicando el teorema del punto fijo existirá un único $A \in \mathcal{H}(\mathbf{R}^n)$ tal que

$$F(A) = A$$

y, además, para todo $B \in \mathcal{H}(\mathbf{R}^n)$ se cumple que

$$\lim_{k \to \infty} F^k(B) = A$$

en el espacio métrico $(\mathcal{H}(\mathbf{R}^n), d_H)$.

Con todo esto hemos probado el siguiente resultado.

Teorema 4.1 *Sea $\{f_1, f_2, \ldots, f_N\}$ un sistema de funciones iteradas en $\mathbf{R}^n$ de razón de contractividad r $(0 \leq r < 1)$. Entonces existe un único fractal $A \in \mathcal{H}(\mathbf{R}^n)$ tal que*

$$F(A) = \bigcup_{i=1}^{N} f_i(A) = A$$

Además, para cualquier fractal $B \in \mathcal{H}(\mathbf{R}^n)$ se cumple

$$\lim_{k \to \infty} F^k(B) = A$$

en el espacio métrico completo $(\mathcal{H}(\mathbf{R}^n), d_H)$.

Definición 4.2 *Sea $\{f_1, f_2, \ldots, f_N\}$ un SFI sobre $\mathbf{R}^n$. Se llama atractor del SFI al único fractal $A \in \mathcal{H}(\mathbf{R}^n)$ que verifica*

$$F(A) = \bigcup_{i=1}^{N} f_i(A) = A$$

cuya existencia y unicidad queda asegurada por el teorema anterior.

Si A es el atractor asociado a un SFI $\{f_1, f_2, \ldots, f_N\}$ de razón r, el teorema anterior nos sugiere un método para la obtención del conjunto A. Este método consiste en partir de un conjunto compacto y no vacío $B \subset \mathbf{R}^n$ e iterar la aplicación F sobre B, hallando los primeros términos de la sucesión $\{F^k(B)\}_{k=0}^{\infty}$. El teorema 3.2 nos proporciona también un método para calcular en cada paso, una cota de la distancia de Hausdorff entre el atractor A y su aproximación $F^k(B)$. Esta fórmula es

$$d_H(F^k(B), A) \leq \frac{1}{1-r} \cdot d_H(F^k(B), F^{k+1}(B))$$

Veamos a continuación unos ejemplos de SFI en los que determinaremos el atractor a través de estas aproximaciones así como la distancia de Hausdorff entre el atractor y sus aproximaciones.

Ejemplo Sean $f_i : \mathbf{R} \longrightarrow \mathbf{R}$, $i = 1, 2$, las aplicaciones contractivas definidas por

$$f_1(x) = \frac{x}{3} \qquad \text{y} \qquad f_2(x) = \frac{x+2}{3}$$

ambas de razón $1/3$. Entonces $\{f_1, f_2\}$ es un SFI de razón $r = 1/3$ cuyo atractor es el conjunto clásico de Cantor $C \subset \mathbf{R}$, ya que este conjunto, como hemos visto, verifica que $C = f_1(C) \cup f_2(C)$.

Vamos a aplicar el proceso iterativo de obtención del atractor, sugerido por el teorema anterior, partiendo del conjunto $B = [0, 1] \subset \mathbf{R}$. Entonces

$$\begin{aligned} B &= [0,1] \\ F(B) &= f_1(B) \cup f_2(B) \\ &= \left[0, \frac{1}{3}\right] \cup \left[\frac{2}{3}, 1\right] \\ F^2(B) &= f_1(F(B)) \cup f_2(F(B)) \\ &= \left[0, \frac{1}{9}\right] \cup \left[\frac{2}{9}, \frac{3}{9}\right] \cup \left[\frac{6}{9}, \frac{7}{9}\right] \cup \left[\frac{8}{9}, 1\right] \end{aligned}$$

que, como se puede observar en la figura 4.4, nos va dando los intervalos que generan por inducción el conjunto clásico de Cantor.

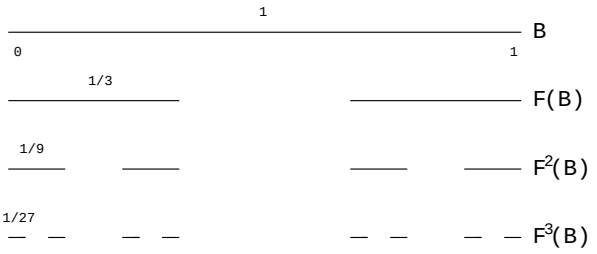


Figura 4.4: Intervalos convergentes al conjunto de Cantor obtenidos mediante el SFI $f_1(x) = x/3$ y $f_2(x) = (x+2)/3$ a partir del intervalo unidad.

Teniendo en cuenta que la distancia de Hausdorff entre dos conjuntos es la máxima distancia entre un punto de un conjunto y el otro conjunto, se puede observar que

$$\begin{aligned} d_H(B, F(B)) &= \frac{1}{6} \\ d_H(F(B), F^2(B)) &= \frac{1}{18} \\ d_H(F^2(B), F^3(B)) &= \frac{1}{54} \end{aligned}$$

y, en general,

$$d_H(F^k(B), F^{k+1}(B)) = \frac{1}{2 \cdot 3^{k+1}}, \quad \forall k \geq 0$$

Luego, una cota de la distancia de Hausdorff entre el conjunto de Cantor C (atractor) y sus aproximaciones es

$$\begin{aligned} d_H(C, F^k(B)) &\leq \frac{1}{1-r} d_H(F^k(B), F^{k+1}(B)) \\ &= \frac{1}{1-\frac{1}{3}} \frac{1}{2 \cdot 3^{k+1}} \\ &= \frac{3}{2} \frac{1}{2 \cdot 3^{k+1}} \\ &= \frac{1}{4 \cdot 3^k}, \quad k \geq 0 \end{aligned}$$

lo que nos permite hallar el conjunto de Cantor con la aproximación deseada. Es obvio que la obtención del atractor la podíamos haber abordado desde cualquier conjunto $B \subset \mathbf{R}$ compacto y no vacío.

Ejemplo Sean $f_i : \mathbf{R}^2 \longrightarrow \mathbf{R}^2$, $1 \leq i \leq 3$, las aplicaciones contractivas definidas por

$$\begin{aligned} f_1(x, y) &= r(x, y) \\ f_2(x, y) &= r(x, y) + (1 - r, 0) \\ f_3(x, y) &= r\left(x - \frac{1}{2}, y - \frac{\sqrt{3}}{2}\right) + \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) \end{aligned}$$

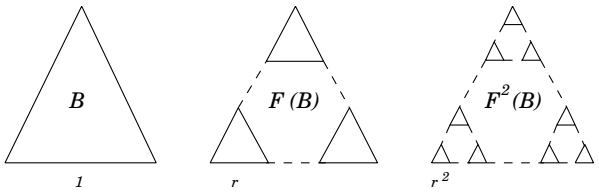


Figura 4.5: Primeras iteraciones del SFI asociado al triángulo de Sierpinski a partir de un triángulo de lado unidad.

con $0 < r \leq 1/2$, siendo r la razón de contractividad de cada una de ellas. Entonces $\{f_1, f_2, f_3\}$ es un SFI de razón r, cuyo atractor será un cierto conjunto $T_r \in \mathcal{H}(\mathbf{R}^n)$ con $n = 2$ tal que

$$T_r = \bigcup_{i=1}^{3} f_i(T_r)$$

Conviene observar que si $r = 1/3$, entonces el atractor $T_{1/3}$ es el triángulo de Sierpinski. Para un r genérico, obtendremos el *triángulo generalizado de Sierpinski*. Vamos a aplicar el proceso iterativo de obtención del atractor al triángulo con vértices en los puntos (0,0), (1,0) y $(1/2, \sqrt{3}/2)$. Sea B este triángulo. Las primeras iteraciones de este conjunto B se pueden ver en la figura 4.5.

A partir de la figura 4.5 se puede ver mediante cálculos geométricos elementales que

$$d_H(B, F(B)) = \sqrt{\left(\frac{1-2r}{2}\right)^2 + \left(\frac{\sqrt{3}}{6}\right)^2 - \left(\frac{r}{2}\right)^2} = \frac{\sqrt{3}}{2}\left(\frac{2}{3} - r\right)$$

En general, y por semejanza, se tiene que

$$d_H(F^k(B), F^{k+1}(B)) = \frac{\sqrt{3}}{2} r^k \left(\frac{2}{3} - r\right), \quad \forall k \geq 0$$

Luego una cota de la distancia de Hausdorff entre el atractor T_r y sus primeras aproximaciones es

$$\begin{aligned} d_H(T_r, F^k(B)) &\leq \frac{1}{1-r} d_H(F^k(B), F^{k+1}(B)) \\ &= \frac{1}{1-r} \frac{\sqrt{3}}{2} r^k \frac{2-3r}{3} \\ &= \frac{r^k (2-3r)}{2\sqrt{3}(1-r)} \end{aligned}$$

4.3. Obtención del fractal asociado a un SFI

Sólo consideraremos aquí el caso de los SFI definidos sobre $\mathbf{R}^2$ por ser de más sencilla elaboración. Describiremos dos algoritmos distintos, uno determinista y otro aleatorio. Ambos, sin embargo, proporcionan el mismo resultado.

Algoritmo determinista

Las pautas anteriores para la obtención del atractor de un SFI pueden resumirse en el siguiente algoritmo.

1. Elegir un conjunto arbitrario $B \subset X$ compacto y no vacío
2. Hacer $Z = B$
3. Representar Z
4. Hacer desde $i = 1$ hasta M
   - 4.1. Borrar Z
   - 4.2. Hallar $F(Z) = \bigcup_{i=1}^{N} f_i(Z)$
   - 4.3. Hacer $Z = F(Z)$
   - 4.4. Representar Z
5. Fin

Cuando este algoritmo termine de ejecutarse habremos obtenido $F^M(B)$ que para $M = 10$ nos da, en general, una muy buena aproximación al atractor A.

El SFI asociado al triángulo de Sierpinski de razón $r = 1/2$ construido sobre el triángulo isósceles cuya base y altura coinciden con la base y altura de una ventana 100×100 es $\{f_1, f_2, f_3\}$ donde

$$
\begin{aligned}
f_1 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
f_2 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 50 \\ 1 \end{pmatrix} \\
f_3 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 25 \\ 50 \end{pmatrix}
\end{aligned}
$$

que se puede expresar de forma simplificada como en la tabla 4.1. No debemos fijarnos, por ahora, en la columna marcada como PROB.

Vamos a ocuparnos ahora de la elaboración de un algoritmo aleatorio para la obtención del fractal determinista. Queremos hacer hincapié en el

f	A	B	C	D	E	F	PROB
1	0.5	0	0	0.5	1	1	0.33
2	0.5	0	0	0.5	50	1	0.33
3	0.5	0	0	0.5	25	50	0.33

Cuadro 4.1: Notación simplificada del sistema de funciones iteradas asociado al triángulo de Sierpinski. La columna marcada con PROB no es útil todavía; su significado se discute en el apartado siguiente.

hecho de que el fractal que vamos a generar es un fractal absolutamente determinista (el SFI que genera el fractal está unívocamente determinado) y que la aleatoriedad reside únicamente en el algoritmo que lo genera.

Algoritmo aleatorio

Sea $\{f_1, f_2, \ldots, f_N\}$ un sistema de funciones iteradas planas. Asignamos a cada f_i, $1 \leq i \leq N$, una cierta probabilidad $p_i > 0$ tal que $\sum_{i=1}^{N} p_i = 1$ y realizamos el siguiente proceso iterativo.

Se elige $x_0 \in \mathbf{R}^2$ arbitrario. A continuación se elige aleatoriamente

$$x_1 \in \{f_1(x_0), \ldots, f_N(x_0)\}$$

donde $f_i(x_0)$, $1 \leq i \leq N$, tiene una probabilidad p_i de ser elegido. Análoga e independientemente del paso anterior, se elige aleatoriamente

$$x_2 \in \{f_1(x_1), \ldots, f_N(x_1)\}$$

según la misma distribución de probabilidades. Cuando tenemos construidos $\{x_0, x_1, \ldots, x_p\}$, se determina x_{p+1} mediante el mismo proceso anterior, es decir, eligiendo de manera independiente (de los pasos anteriores) y aleatoria

$$x_{p+1} \in \{f_1(x_p), \ldots, f_N(x_p)\}$$

según la misma distribución de probabilidades. Y así sucesivamente. Entonces *con probabilidad uno*, el conjunto obtenido $\{x_n\}_{n=0}^{\infty} \subset X$ converge en la métrica de Hausdorff al atractor A del SFI en el sentido de que dado $\epsilon > 0$, existe $K = K(\epsilon) \in \mathbf{N}$ tal que

$$\lim_{M \to \infty} d_H(A, \{x_n : K \leq n \leq M\}) < \epsilon$$

De lo anterior se deduce que los puntos del conjunto $\{x_n\}_{n=0}^{\infty}$ que pueden estar a mayor distancia del atractor son los primeros puntos de la sucesión.

Por este motivo, cuando se intenta aproximar el atractor mediante este algoritmo se suelen despreciar los primeros términos (con despreciar los 50 primeros suele bastar).

Si podemos asegurar que el punto inicial considerado pertenece al atractor, $x_0 \in A$, y puesto que las funciones $\{f_i\}_{i=1}^N$ no pueden sacar los puntos del atractor A ($A = \cup_{i=1}^N f_i(A)$), entonces podemos asegurar

$$\{x_0, x_1, \ldots, x_M\} \subset A, \qquad \forall M \in \mathbf{N}$$

y que con probabilidad uno

$$d_H(A, \{x_n\}_{n=0}^\infty) = \lim_{M\to\infty} d_H(A, \{x_0, x_1, \ldots, x_M\}) = 0$$

o, lo que es lo mismo, que con probabilidad uno la sucesión $\{x_n\}_{n=0}^\infty$ es densa en el atractor A:

$$A = adh(\{x_n\}_{n=0}^\infty)$$

Estos resultados están basados en la teoría ergódica y su fundamentación puede estudiarse en el capítulo cuarto de [BAR 93a].

Cuando se quiere aproximar un atractor mediante este algoritmo, nos interesa obtener la mejor aproximación con el menor número de puntos. Si la masa (medida) acumulada en cada $f_i(A)$, $1 \le i \le N$, es aproximadamente la misma, entonces es conveniente elegir $p_i = 1/N$, $1 \le i \le N$. Este es el caso, por ejemplo, del triángulo de Sierpinski ($p_1 = p_2 = p_3 = 1/3$). Si no es así, conviene elegir las probabilidades aproximadamente proporcionales a la cantidad de masa que hay en cada $f_i(A)$.

En este último caso se puede elegir un cierto conjunto $W \subset \mathbf{R}^2$ de área no nula y fácil de calcular y elegir

$$p_i \approx \frac{\text{área}(f_i(W))}{\sum_{j=1}^N \text{área}(f_j(W))}, \qquad 1 \le i \le N$$

de tal forma que $\sum_{i=1}^N p_i = 1$. En el caso particular de que las aplicaciones contractivas sean transformaciones afines, es decir,

$$f_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}, \qquad 1 \le i \le N$$

entonces se puede elegir

$$p_i \approx \frac{|a_i d_i - b_i c_i|}{\sum_{j=1}^N |a_j d_j - b_j c_j|}$$

donde $\approx$ significa *aproximadamente igual* para indicar también que si algún p_i fuese igual a cero, habría que asignarle algún valor pequeño no nulo, por

ejemplo $p_i = 0{,}001$, ya que en caso contrario nunca se aplicaría la transformación correspondiente.

Una redacción más precisa del algoritmo anterior sería la siguiente:

1. Elegir un punto arbitrario $x \in \mathbf{R}^2$
2. Hacer desde $i = 1$ hasta M
   - 2.1. Elegir j aleatoriamente entre $\{1, 2, \ldots, N\}$ con probabilidades $\{p_1, p_2, \ldots, p_N\}$
   - 2.2. Hallar $y = f_j(x)$
   - 2.3. Hacer $x = y$
   - 2.4. Si $i > 50$, representar x
3. Fin

Para $M = 5000$ tendremos, en general, una muy buena aproximación del atractor A.

Un cambio en las probabilidades asociadas al SFI va a producir un cambio en la distribución de los M puntos que se representan, lo que producirá distintos aspectos de sombras sobre el atractor. Esto puede verificarse con el SFI para la obtención del triángulo de Sierpinski con probabilidades $p_1 = 0{,}6$, $p_2 = 0{,}3$, $p_3 = 0{,}1$ mostrado en la tabla 4.2. El conjunto resultante tras la aplicación del algoritmo aleatorio a este SFI puede observarse en la figura 4.6.

f	A	B	C	D	E	F	PROB
1	0.5	0	0	0.5	0	0	0.6
2	0.5	0	0	0.5	0.5	0	0.3
3	0.5	0	0	0.5	0.25	0.5	0.1

Cuadro 4.2: SFI asociado a un triángulo de Sierpinski modificado mediante la variación de las probabilidades asociadas a cada una de sus transformaciones.

4.4. El teorema del collage

¿Se podrán representar todas las imágenes reales mediante fractales? ¿Cómo se podrá hacer, si esto es posible? Vamos a tratar de responder aquí a estas preguntas. Para nosotros una imagen I será un conjunto compacto y no vacío de puntos de $\mathbf{R}^n$, $n = 1, 2, 3$. Sea $I \in \mathcal{H}(\mathbf{R}^n)$ una imagen real y supongamos que existe un SFI $\{f_1, f_2, \ldots, f_N\}$ de razón r tal que $F(I) =$

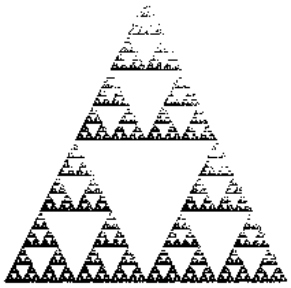


Figura 4.6: Triángulo de Sierpinski obtenido tras la aplicación del algoritmo aleatorio al SFI de la tabla 4.2.

$\bigcup_{i=}^{N} f_i(I)$ está suficientemente próximo a I, es decir,

$$d_H(I, F(I)) \leq \epsilon$$

Entonces si $A \in \mathcal{H}(\mathbf{R}^n)$ es el atractor de este SFI, se tiene, aplicando el teorema del punto fijo, que

$$d_H(A, I) \leq \frac{1}{1-r} d_H(I, F(I)) \leq \frac{\epsilon}{1-r}$$

Es decir, que el atractor A del SFI se aproxima bastante a la imagen real I siempre que $\epsilon > 0$ sea suficientemente pequeño. Tenemos por tanto el siguiente corolario del teorema del punto fijo.

Corolario 4.1 (Teorema del collage) *Sea $I \in \mathcal{H}(\mathbf{R}^n)$ una imagen real y dado $\epsilon > 0$, sea $\{f_1, f_2, \ldots, f_N\}$ un SFI con factor de contractividad r, $0 \leq r < 1$, tal que*

$$d_H(I, F(I)) \leq \epsilon$$

Entonces

$$d_H(A, I) \leq \frac{\epsilon}{1-r}$$

donde A es el atractor del SFI.

A la vista del teorema anterior se puede observar que la aproximación del atractor A a la imagen real I será mejor cuanto más pequeño sea el valor del factor de contractividad r y que esta aproximación no depende del número de aplicaciones que forman el SFI.

La gran importancia de este sencillo resultado estriba en la posibilidad de sustituir la imagen I real por el atractor A, siempre que la aproximación sea

lo suficientemente buena. Si el SFI correspondiente está formado por pocas transformaciones, almacenándolo en lugar de la imagen I habremos obtenido una reducción significativa en el espacio ocupado por la imagen. Esta fue la idea que abrió la investigación en la compresión fractal de imágenes.

Aproximación de imágenes reales mediante SFI

Sea $I \in \mathcal{H}(\mathbf{R}^n)$ una imagen real. El proceso a seguir para aproximarla mediante SFI sería el siguiente:

1. Encontrar aplicaciones contractivas $f_i : \mathbf{R}^n \longrightarrow \mathbf{R}^n$, $1 \leq i \leq N$, tales que
$$d_H\left(I, \bigcup_{i=1}^{N} f_i(I)\right)$$
sea lo más pequeño posible. Sea, por ejemplo, $d_H(I, \bigcup_{i=1}^{N} f_i(I)) \leq \epsilon$. Entonces
$$d_H(A, I) \leq \frac{\epsilon}{1-r}$$
donde A es el atractor del SFI y esta aproximación será mejor cuanto más pequeños sean ϵ y r. Por ello es conveniente elegir transformaciones contractivas de la menor razón posible, independientemente del número de ellas, que puede ser tan grande como se quiera.

2. Generar el atractor A mediante cualquiera de los algoritmos descritos anteriormente.

Una pregunta obvia es por qué no hacer que el SFI F comprima muy ligeramente I con lo que la distancia $d_H(I, F(I))$ será muy pequeña y quizá $d_H(A, I)$ también lo sea. Esto no funcionará porque para tal SFI el término $\frac{1}{1-r}$ será muy grande y no podremos garantizar que $d_H(A, I)$ sea pequeña (de hecho, no lo es).

Una hoja fractal

Sea $I \subset \mathbf{R}^2$ la imagen de la figura 4.7 que vamos a tratar de representar mediante un sistema de funciones iteradas.

Para encontrar el SFI tenemos que descomponer esta imagen I en partes de tal forma que cada una de ellas se pueda obtener a partir de la imagen total mediante una aplicación contractiva (a ser posible afín). Una posible

Figura 4.7: La hoja de helecho que se intentará aproximar mediante un SFI aplicando el teorema del collage.

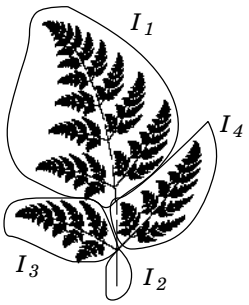


Figura 4.8: Cada una de las cuatro partes de la hoja del helecho aquí indicadas se puede considerar como el resultado de una aplicación contractiva sobre la imagen completa.

descomposición se ilustra en la figura 4.8. En esta descomposición utilizamos 4 partes que llamamos I_i, $1 \leq i \leq 4$, y se cumple que $I = \bigcup_{i=1}^{4} I_i$.

Para hallar las aplicaciones que transforman la imagen total I en I_i, $1 \leq i \leq 4$, tenemos que situar esta imagen en el plano $\mathbf{R}^2$, lo que podemos hacer como se muestra en la figura 4.9, incluyendo I en el cuadrado $[-1/2, 1/2] \times [0, 1]$. De esta forma la imagen queda centrada horizontalmente en el origen y es más fácil operar.

La aplicación f_1 que nos transforma I en I_1 es una homotecia centrada en el origen de razón $3/4$ seguida de un leve giro de ángulo $\pi/32$ y de una

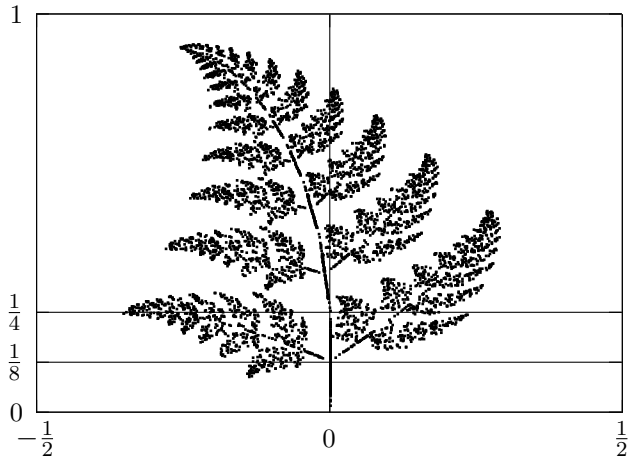


Figura 4.9: Para obtener las aplicaciones contractivas que transforman la imagen completa del helecho en cada una de las partes indicadas en la figura 4.8, tenemos que situar la hoja en el plano $\mathbf{R}^2$. Si la imagen se centra horizontalmente en el origen, las transformaciones se obtienen de manera más comoda.

traslación al punto $(0, 1/4)$. Luego

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{32} & -\operatorname{sen}\frac{\pi}{32} \\ \operatorname{sen}\frac{\pi}{32} & \cos\frac{\pi}{32} \end{pmatrix} \begin{pmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{4} \end{pmatrix}$$

La aplicación f_2 que nos transforma I en I_2 es una homotecia centrada en el origen de razón $1/4$ respecto al eje de ordenadas y cero respecto al de abcisas

$$f_2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

La aplicación f_3 que nos tranforma I en I_3 es una homotecia centrada en el origen de razón $3/10$ respecto al eje de abcisas y $2/5$ respecto al de ordenadas seguida de un giro de $\pi/3$, seguida de una traslación de vector $(0, 1/8)$

$$f_3 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{3} & -\operatorname{sen}\frac{\pi}{3} \\ \operatorname{sen}\frac{\pi}{3} & \cos\frac{\pi}{3} \end{pmatrix} \begin{pmatrix} \frac{3}{10} & 0 \\ 0 & \frac{2}{5} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{8} \end{pmatrix}$$

Por último, la aplicación f_4 que transforma I en I_4 es una homotecia centrada en el origen de razón $3/10$ respecto al eje de abcisas y $1/2$ respecto al de ordenadas seguida de un giro de $-\pi/4$ y seguida de una traslación al

Figura 4.10: Un árbol fractal. El lector puede intentar hallar un SFI que aproxime esta imagen. La solución en [COL 96].

punto $(0, 1/8)$

$$f_4\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\frac{-\pi}{4} & -\operatorname{sen}\frac{-\pi}{4} \\ \operatorname{sen}\frac{-\pi}{4} & \cos\frac{-\pi}{4} \end{pmatrix}\begin{pmatrix} \frac{3}{10} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{8} \end{pmatrix}$$

Después de asignar probabilidades, según los criterios establecidos en la sección anterior, el SFI escrito en forma simplificada sería el de la tabla 4.3, que ejecutado mediante el algoritmo aleatorio y representando 100000 puntos nos daría precisamente la imagen de la figura 4.7.

f	A	B	C	D	E	F	PROB
1	0.746	-0.073	0.073	0.746	0	0.25	0.65
2	0	0	0	0.25	0	0	0.03
3	0.15	-0.344	0.258	0.2	0	0.125	0.14
4	0.212	0.353	-0.212	0.353	0	0.125	0.18

Cuadro 4.3: Aproximación mediante el teorema del collage a la hoja de helecho. Las probabilidades se asignaron en función del área generada por cada transformación.

Como ejercicio el lector puede intentar ahora obtener un SFI que aproxime el árbol mostrado en la figura 4.10. Pista: un posible SFI tiene cinco transformaciones de las cuales dos conforman la parte inferior del tronco.

4.5. Fractales en movimiento

Nos vamos a ocupar aquí de la posibilidad de establecer movimiento en los conjuntos fractales. Puesto que los conjuntos fractales que hemos considerado en este capítulo dependen directamente de una familia de funciones contractivas parece razonable esperar que pequeñas variaciones en estas funciones produzcan pequeñas variaciones en el fractal generado. Si esto fuera así, podríamos producir con una sucesión de fractales muy próximos entre sí un efecto de movimiento.

Supongamos que las aplicaciones contractivas que definen un SFI $\{f_1, f_2, \ldots, f_N\}$ no vienen unívocamente determinadas, sino que están definidas en función de un parámetro $p \in [\alpha, \beta] \subset \mathbf{R}$ del que dependen continuamente. El siguiente teorema determina cómo influyen en el atractor pequeñas variaciones del parámetro p.

Teorema 4.2 *Para cada $p \in [\alpha, \beta] \subset \mathbf{R}$ sea $\{f_1(p), \ldots, f_N(p)\}$ un SFI de razón $r(p)$, $0 \leq r(p) \leq r < 1$, con atractor $A(p) \in \mathcal{H}(\mathbf{R}^n)$. Supongamos que cada transformación $f_i(p)(x) = f_i(p, x)$ es continua para todo x respecto de p en $[\alpha, \beta]$. Entonces el atractor $A(p)$ depende continuamente de $p \in [\alpha, \beta]$.*

La demostración, algo técnica, puede encontrarse en [GUZ 93, p. 201]. Este teorema se puede utilizar para la animación de imágenes. Aunque no entraremos en más detalles, mostraremos un ejemplo.

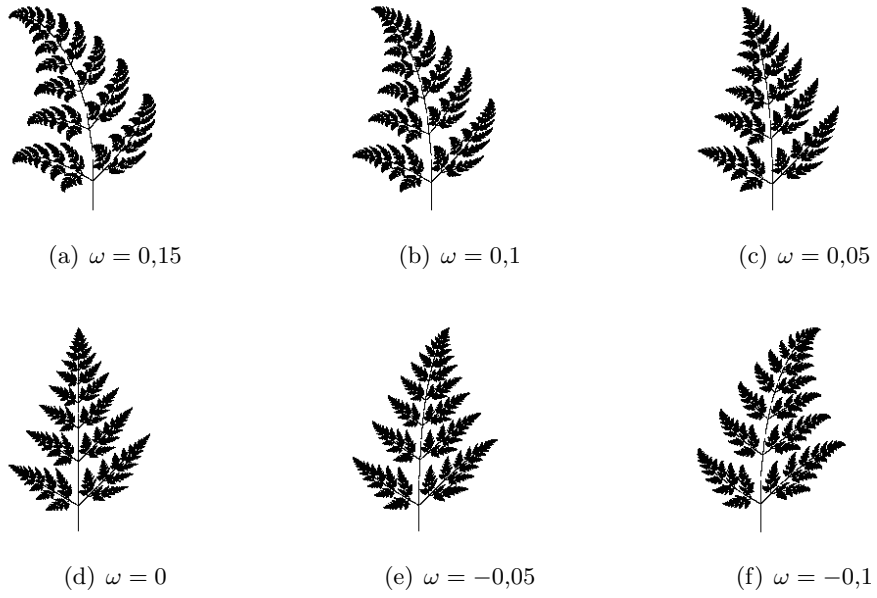
Ejemplo Si en la definición del SFI de la hoja dado en la sección anterior sustituimos la aplicación f_1 por

$$f_1 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\operatorname{sen}\alpha \\ \operatorname{sen}\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{4} \end{pmatrix}$$

entonces $\{f_1(\alpha), f_2, f_3, f_4\}$ es un SFI de razón $3/4$ que depende continuamente del parámetro $\alpha \in [-\pi/2, \pi/2]$. Luego el atractor $A(\alpha)$ dependerá continuamente de α. Al variar continuamente α se obtiene un efecto de movimiento de la hoja. Variando α según los valores

$$\begin{aligned} \alpha_1 &= \operatorname{arc\,sen} 0{,}15 \\ \alpha_2 &= \operatorname{arc\,sen} 0{,}1 \\ \alpha_3 &= \operatorname{arc\,sen} 0{,}05 \\ \alpha_4 &= \operatorname{arc\,sen} 0 \\ \alpha_5 &= \operatorname{arc\,sen} -0{,}05 \\ \alpha_6 &= \operatorname{arc\,sen} -0{,}1 \end{aligned}$$

puede obtenerse la sucesión de imágenes mostrada en la figura 4.11.



(a) $\omega = 0{,}15$ (b) $\omega = 0{,}1$ (c) $\omega = 0{,}05$

(d) $\omega = 0$ (e) $\omega = -0{,}05$ (f) $\omega = -0{,}1$

Figura 4.11: La hoja de helecho agitada por el viento mediante distintos valores del parámetro α. Los valores dados a α son $\alpha = \operatorname{arc\,sen}\omega$ donde ω evoluciona según se indica bajo cada figura. El movimiento de la hoja se puede observar al seguir las imágenes de izquierda a derecha y de arriba a abajo.

4.6. Los conjuntos de Julia como SFI

En la sección 1.3 se introdujeron los conjuntos de Julia y se mostró un algoritmo para su obtención. Al objeto de encontrar el conjunto de Julia asociado a un sistema dinámico complejo cuadrático $f_c(z) = z^2 + c$ para un valor complejo de c fijo y arbitrario, se puede intentar efectuar el proceso inverso al de huida de sus puntos, con lo que nos acercaríamos a él. Es decir, se trataría de iterar la transformación inversa

$$f_c^{-1}(z) = \pm\sqrt{z - c}$$

sobre ciertos conjuntos del plano complejo, hasta que sus órbitas cayeran sobre el conjunto de Julia.

Esto resulta ser cierto en cualquiera de los casos que consideremos, es decir, que si $c = c_1 + c_2\, j$ es un número complejo arbitrario, entonces

$$\lim_{n\to\infty} f_c^{-n}(K) = J(f_c)$$

para cualquier subconjunto no vacío K del plano complejo, donde $J(f_c)$ representa el conjunto de Julia asociado a la función cuadrática de parámetro c y la convergencia se entiende en la métrica de Hausdorff.

La transformación inversa f_c^{-1} consta de dos funciones $g_1(z) = +\sqrt{z-c}$ y $g_2(z) = -\sqrt{z-c}$. Estas funciones se pueden expresar en forma cartesiana como

$$g_1(x,y) = (a,b) \qquad \text{y} \qquad g_2(x,y) = (-a,-b)$$

donde, por cálculos elementales,

$$a = \sqrt{\frac{(x-c_1) + \sqrt{(x-c_1)^2 + (y-c_2)^2}}{2}}$$

$$b = \begin{cases} +\sqrt{\dfrac{\sqrt{(x-c_1)^2 - (y-c_2)^2} - (x-c_1)}{2}}, & \text{si } y \geq c_2 \\ \\ -\sqrt{\dfrac{\sqrt{(x-c_1)^2 - (y-c_2)^2} - (x-c_1)}{2}}, & \text{si } y < c_2 \end{cases}$$

Es decir, asignamos a g_1 la raíz con parte real positiva y a g_2 la opuesta. Entonces, puesto que f_c^{-1} consta de estas dos funciones, g_1 y g_2, se entiende que

$$\begin{aligned} f_c^{-1}(K) &= g_1(K) \cup g_2(K) \\ f_c^{-n}(K) &= f_c^{-1}\left(f_c^{-(n-1)}(K)\right), \quad \text{si } n > 1 \end{aligned}$$

Luego, el proceso de construcción del conjunto de Julia, en el caso cuadrático, resulta ser análogo al proceso de construcción determinista del atractor de un SFI donde las funciones son $\{g_1, g_2\}$. Podemos aplicar entonces el algoritmo determinista, descrito en la página 67, para la determinación del conjunto de Julia asociado a los sistemas cuadráticos.

Capítulo 5

Compresión de imágenes

El volumen de datos electrónicos que maneja la humanidad crece de forma continua. A la enorme cantidad de información en formato electrónico que se genera cada día hay que unir todo el fondo documental de la historia que estará totalmente digitalizado para las primeras décadas del siglo XXI. El tratamiento de esta información origina unos costes de entre los que destacan el coste de almacenamiento y el de transmisión.

La compresión de datos trata de reducir el volumen de la información sin que ésta deje de serlo. En esta obra se trata de la compresión de imágenes mediante técnicas basadas en fractales.

La compresión de imagen puede ser con pérdidas y sin pérdidas. Las técnicas sin pérdidas normalmente no logran reducir el tamaño de las imágenes más allá de su tercera parte, pero son necesarias, ya que algunas imágenes (imágenes médicas o con texto, por ejemplo) se vuelven inservibles si se pierde alguna información en el proceso de descompresión.

Ya que el ojo humano tiene límites, se puede tolerar normalmente alguna pérdida en la imagen al descomprimirla de forma que la imagen restaurada sea una aproximación cercana a la original. En algunos casos las imágenes descomprimidas no parecen tener pérdidas, aunque se haya utilizado alguna técnica con pérdidas en su compresión. Existen en la actualidad varias técnicas para compresión de imágenes con pérdidas: basadas en la cuantización vectorial [SAU 96a], la transformada discreta del coseno [WAL 91], la transformada con wavelets [HIL 94, POL 97] o los sistemas de funciones iteradas [FIS 95, SAU 96a, LU 97].

5.1. Dos pájaros de un tiro

Los grupos de discusión de la red sobre compresión de datos se ven asaltados de tarde en tarde por acaloradas discusiones en torno al anuncio de alguno de sus miembros de un revolucionario mecanismo de compresión sin pérdidas que no sólo permite obtener enormes ratios de compresión sino que, además, lo hace sobre *cualquier entrada*, independientemente de que ésta proceda de un concierto de música celta, del reportaje en vídeo sobre las pasadas vacaciones, de la guía telefónica o de una fuente de datos aleatorios. Un sencillo argumento basta para demostrar que lo anterior es imposible independientemente del método de compresión utilizado. De hecho, es imposible asegurar la compresión, incluso en un bit, de cualquier tipo de programa.

Teorema 5.1 *Es imposible comprimir sin pérdidas todos los archivos de tamaño mayor o igual a N bits para cualquier entero $N \geq 0$.*

DEMOSTRACIÓN Supongamos que existe una técnica capaz de comprimir sin pérdidas *todos* los archivos de tamaño igual o superior a N bits. Comprimamos con este programa todos los 2^N archivos cuyo tamaño es exactamente N bits. Todos estos ficheros comprimidos tienen como máximo $N-1$ bits por lo que habrá como máximo 2^N-1 ficheros comprimidos diferentes (2^{N-1} archivos de tamaño $N-1$, 2^{N-2} de tamaño $N-2$, y así sucesivamente hasta llegar a un fichero de tamaño 0). Con ello al menos dos ficheros de entrada distintos han tenido que ser comprimidos en el mismo archivo de salida. Por lo tanto, la compresión proporcionada por esta técnica no puede ser sin pérdidas. □

Debe tenerse cuidado por otra parte con no caer en las garras de ciertos programas tramposos que circulan por la red. Este tipo de programas no comprimen en absoluto, sino que almacenan los datos originales en ficheros ocultos del disco duro o en *clusters* sin utilizar. Los datos pueden descomprimirse sólo si tales ficheros ocultos no han sido borrados o si los *clusters* no han sido utilizados. Si se copia el fichero comprimido a otra máquina, se obtiene siempre un error de paridad en la estructura del fichero.

Pese a lo anterior, la historia de la compresión de datos es la historia del esfuerzo de numerosas personas que con su creatividad y original forma de abordarla la han llevado a convertirse en una tecnología básica en la actualidad. Un ejemplo en el que se manifiesta cómo una visión diferente de las cosas puede llevar a resultados desconcertantes es el siguiente programa

de autor desconocido que genera las 2400 primeras cifras del número π:[1]

```
#include <stdio.h>

long int a= 10000, b, c= 8400, d, e, f[8401], g;

main () {
    for ( ;b-c; )
        f[b++]= a/5;
    for ( ;
          d= 0, g= c*2;
          c-= 14, printf ("%.4d",e+d/a), e= d%a)
        for (b= c;
             d+= f[b]*a, f[b]= d%--g, d/=g--, --b;
             d*= b)
        ;
}
```

Puede comprobarse que el tamaño de este programa es muy inferior al necesario para codificar la sucesión de tales cifras, incluso admitiendo 4 bits por cifra en la sucesión y 8 bits por carácter en el programa.

Hasta ahora simplemente hemos presentado algunos hechos curiosos en torno a la compresión de datos. En la siguiente sección el enfoque deja atrás lo anecdótico.

5.2. Calidad de la compresión con pérdidas

Cuando se evalúa la calidad de la imagen restaurada tras una compresión con pérdidas se suelen considerar dos cantidades bastante relacionadas. La primera es el *ratio de compresión* que se define como la relación entre el tamaño de la representación original de la imagen y el de su versión codificada.

La otra cantidad a considerar debe medir la calidad de la imagen. La mayor parte de los autores utilizan la *raíz del error cuadrático medio* (RECM) o la *razón señal-ruido máxima* (RSRM). Para imágenes en escala de grises de 8 bits la RSRM se define como

$$RSRM = 20\log_{10}\frac{255}{RECM} = 20\log_{10}\frac{255}{\sqrt{\frac{1}{\#pixels}\sum_{i,j}(\hat{p}_{i,j}-p_{i,j})^2}}$$

donde $p_{i,j}$ y $\hat{p}_{i,j}$ denotan respectivamente las intensidades de los pixels en la

[1] Para más información, puede consultarse el número de octubre de 1994 de *Investigación y Ciencia*. También puede accederse a las páginas del *Concurso internacional de código C ofuscado* en `http://reality.sgi.com/csp/ioccc/index.html`.

imagen original y en su aproximación. La RSRM expresa la relación entre la potencia máxima de la señal y el error y se mide en decibelios.

Si un codificador permite diferentes ratios de compresión para distintas calidades en la codificación, puede ser interesante poder comparar distintos codificadores o los resultados de un único codificador con distintos parámetros. Para ello se recogen en un gráfico varios puntos dados por el par (ratio de compresión,RSRM) de ambos métodos. Al conectar algunos de estos puntos se obtienen las llamadas *curvas razón-distorsión*. Cuanto más alta en el gráfico esté una curva, mejor será el codificador.

Aunque la relación señal-ruido máxima y el ratio de compresión son medidas habituales en la mayor parte de artículos sobre compresión de imagen, no son las únicas medidas existentes. Existe una gran controversia sobre qué métrica es la adecuada para medir el error cometido en la compresión, ya que ninguna de las existentes parece concordar en todos los casos con las consideraciones de un observador humano. Un ejemplo simple y paradigmático es el desplazamiento global en las intensidades de una imagen. Si sumamos 10 a todos los pixels de una imagen, las medidas habituales indicarán un gran error, aunque la imagen obtenida tiene simplemente un poco más de brillo. Por otra parte, el uso del logaritmo en la RSRM exagera las diferencias en el rango de bajas pérdidas y las suprime en el de grandes pérdidas como puede verse en la figura 5.1 donde se muestra la raíz del error cuadrático medio contra la relación señal-ruido máxima.

5.3. Compresión de imágenes en color

Afortunadamente el sistema visual humano sólo utiliza tres canales de color para codificar la información sobre éste que llega al cerebro. Esto significa que los colores pueden simularse mediante la superposición de tres colores primarios, normalmente el rojo, el verde y el azul. Cuando se digitaliza una imagen en color se utilizan tres filtros para extraer las intensidades de cada uno (estas tres intensidades se conocen como *RGB* por las iniciales de los colores en inglés). Al recombinar las tres intensidades, las percibiremos como algún color.

Cualquier método que pueda codificar imágenes monocromo, puede utilizarse para codificar imágenes en color. Pero codificar cada una de las componentes RGB de forma separada no es una opción muy recomendable: el sistema visual humano no es especialmente sensible a la información sobre el color y hay formas para comprimir esta información y sacar partido de esta baja sensibilidad.

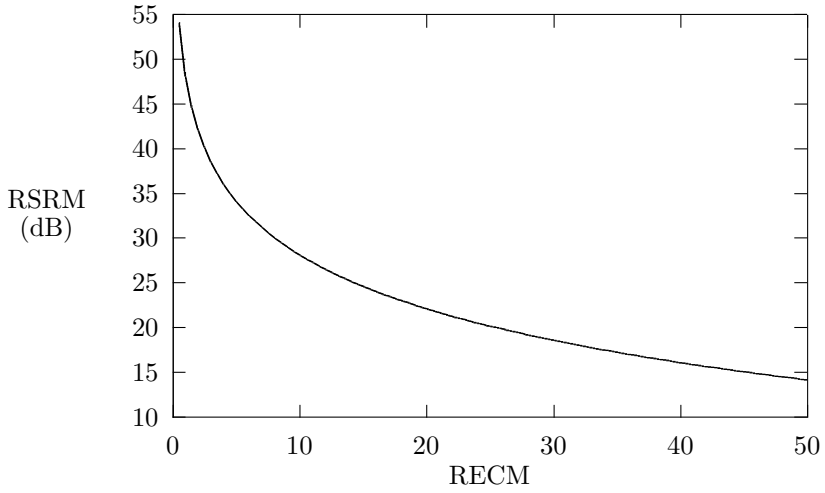


Figura 5.1: Representación de la raíz del error cuadrático medio contra la relación señal-ruido máxima. Puede apreciarse como para pequeños valores de la RECM, la RSRM potencia las diferencias, mientras que su comportamiento es el contrario bajo valores altos de la RECM. En este trabajo se utilizará la RSRM como medida de la distorsión de una imagen debido a su mayor popularidad, pero debe tenerse siempre presente que para ratios de compresión elevados una diferencia pequeña en la RSRM de dos codificaciones de una misma imagen puede significar una gran diferencia en sus calidades.

Dado un triplete (R_i, G_i, B_i) que describe la distribuciónes de color en un pixel i podemos calcular los valores YIQ como

$$\begin{pmatrix} Y_i \\ I_i \\ Q_i \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}596 & -0{,}274 & -0{,}322 \\ 0{,}211 & -0{,}523 & 0{,}312 \end{pmatrix} \begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix}$$

Esta transformación proporciona los tres canales YIQ: la señal Y mide el brillo del color (*luminancia*), la señal I mide el color real (*matiz*) y la señal Q la profundidad del color (*saturación*). Los canales I y Q se denominan también señales de *crominancia*. Los valores RGB originales pueden obtenerse mediante la transformación inversa

$$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} = \begin{pmatrix} 1{,}000 & 0{,}956 & 0{,}621 \\ 1{,}000 & -0{,}273 & -0{,}647 \\ 1{,}000 & -1{,}104 & 1{,}701 \end{pmatrix} \begin{pmatrix} Y_i \\ I_i \\ Q_i \end{pmatrix}$$

Es posible comprimir significativamente las señales I y Q sin ninguna degradación aparente en la calidad de la imagen, ya que el ojo humano no es tan sensible a la información de la crominancia como lo es para la de la luminancia. Esta compresión puede realizarse mediante cuantización o

mediante submuestreo de los valores I y Q de grupos consecutivos de pixels, por ejemplo, tomando el valor medio de cada bloque 4×4 de la imagen.

5.4. Cuantización vectorial

La cuantización es uno de los métodos de compresión con pérdidas más sencillos, aunque no por ello deja de proporcionar resultados aceptables.

La cuantización vectorial es una generalización de la *cuantización escalar*. En la cuantización escalar se representa cada valor mediante un índice a una tabla fija formada por un subconjunto de valores representativos denominado *libro de códigos*. Por ejemplo, si tenemos una secuencia de datos cada uno de ellos de 16 bits y consideramos sólo los 8 bits más significativos de cada elemento, obtendremos una aproximación a los datos originales al perder precisión. En este caso la tabla de códigos estaría formada por todos los números de 16 bits divisibles por 256. El rango del intervalo que separa los dos valores más próximos que se pueden representar mediante esta codificación (en este caso 256) se denomina *cuanto*.

En la *cuantización vectorial* (CV) el índice referencia a un libro de códigos formado no por valores individuales, sino por vectores. Un ejemplo típico es una imagen en color en la que cada pixel viene representado por un triplete con los valores RGB (ver sección 5.3). En la mayor parte de las imágenes tales tripletes no cubren todo el espacio RGB, sino que tienden a concentrarse en determinadas zonas de él. Por ejemplo, una imagen de un bosque tendrá normalmente una gran cantidad de verdes. Podemos, entonces, seleccionar un subconjunto relativamente pequeño (por ejemplo, 256 elementos) de colores representativos (tripletes RGB) y aproximar cada pixel representándolo mediante un índice al color más cercano del libro de códigos.

El diseño de libros de códigos óptimos es muy difícil y en la práctica sólo pueden obtenerse soluciones subóptimas como la proporcionada por la *iteración generalizada de Lloyd*. Además, aunque el enfoque estándar de CV produce buenos resultados, suele hacerlo con libros de códigos de tamaños prohibitivos y de elevados tiempos de cálculo en la mayor parte de las ocasiones. Por esta razón existen muchas variaciones de la CV en las que se utilizan libros de códigos con determinadas estructuras que los hacen computables, aunque reducen el rendimiento en términos de calidad. Uno de estos métodos se conoce como *cuantización vectorial con eliminación de media y ganancia de forma* (CV-EMGF).

Como su nombre sugiere, un vector $R \in \mathbf{R}^n$ que va a ser codificado

mediante CV-EMGF se escribe como

$$R = s \cdot D + o \cdot \mathbf{1}$$

donde $\mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbf{R}^n$ y s, o son escalares. $D = (d_1, d_2, \ldots, d_n)^T$ es un *vector de forma* de media cero y varianza uno, esto es

$$\sum_{i=1}^{n} d_i = 0, \qquad \sum_{i=1}^{n} d_i^2 = 1$$

Con dos libros de códigos escalares para s y o y un libro de códigos vectoriales de vectores de forma el vector de entrada R queda, una vez cuantizado, como

$$R \approx s_{\acute{i}nd_s(R)} D_{\acute{i}nd_D(R)} + o_{\acute{i}nd_o(R)} \cdot \mathbf{1}$$

donde $\acute{i}nd_s(R), \acute{i}nd_D(R)$ e $\acute{i}nd_o(R)$ son índices apropiados generados por el cuantizador. Puede decirse que este esquema codifica de forma separada la media, la desviación estándar y la forma de un vector dado, ya que al considerar los tres libros de códigos simultáneamente se obtiene un libro de códigos conjunto muy grande. Por ejemplo, si el tamaño de los libros de códigos para s, o y D es 32, 128 y 4096, respectivamente, podemos representar de forma exacta un total de 2^{24} vectores.

5.5. El estándar JPEG

Durante los últimos años de la década de los ochenta y los primeros de la de los noventa, un comité mixto ISO y CCITT conocido como JPEG (*Joint Photographic Experts Group*, Grupo Mixto de Expertos Fotográficos) trabajó para establecer el primer estándar internacional para compresión de imágenes estáticas de tono continuo tanto en escala de grises como en color. El denominado *estándar JPEG* incluye dos métodos de compresión básicos, cada uno con varios modos de operación. El estándar especifica un método basado en la *transformada discreta del coseno* (TDC) para compresión con pérdidas y un método predictivo para compresión sin pérdidas. JPEG presenta una técnica simple con pérdidas conocida como método *base*, que es un subconjunto de los otros modos de operación basados en la TDC. El método base ha sido con diferencia el más implementado hasta la fecha y es el que mostraremos aquí.

Modos de operación

El estándar de compresión JPEG especifica los siguientes modos de operación:

Codificación secuencial Cada componente de la imagen se codifica en un barrido de derecha a izquierda y de arriba a abajo.

Codificación progresiva La imagen se codifica en múltiples barridos para poder trabajar con aplicaciones en las que el tiempo de transmisión es largo y se prefiere ver la imagen poco a poco.

Codificación sin pérdidas Para garantizar la recuperación exacta de la imagen original.

Codificación jerárquica La imagen se codifica bajo múltiples resoluciones para que se pueda acceder a versiones en baja resolución sin tener que descomprimir primero la imagen completa en toda su resolución.

A continuación veremos un resumen del método de compresión base del estándar.

Codificación y descodificación

El estándar JPEG funciona mejor cuando se aplica a imágenes naturales. No proporciona resultados muy buenos cuando se aplica a imágenes poco realistas tal como dibujos con líneas. Si se pretende utilizar una imagen para su tratamiento digital, los pequeños errores introducidos por la codificación pueden ocasionar problemas graves en los algoritmos clásicos de tratamiento de imágenes, aunque dichos errores sean invisibles para el ojo humano. Esta es una limitación compartida por todos los esquemas de compresión discutidos en esta obra.

Además las imágenes deben de ser de tono continuo; imágenes con muchos saltos bruscos en sus intensidades no se comprimen bien.

La figura 5.2 muestra la sucesión de etapas claves de los modos de operación basados en la TDC. Ya que aquí nos interesa la compresión de imagen con pérdidas, sólo abordaremos este aspecto del método JPEG. Además, las explicaciones siguientes se harán considerando que las imágenes están en escala de grises. Si la imagen es en color, se puede optar por considerarla como tres imágenes en escala de grises asociadas cada una con las componentes de rojo, verde y azul de la imagen, o bien proceder a la recodificación propuesta en 5.3. Veamos brevemente cada una de las etapas.

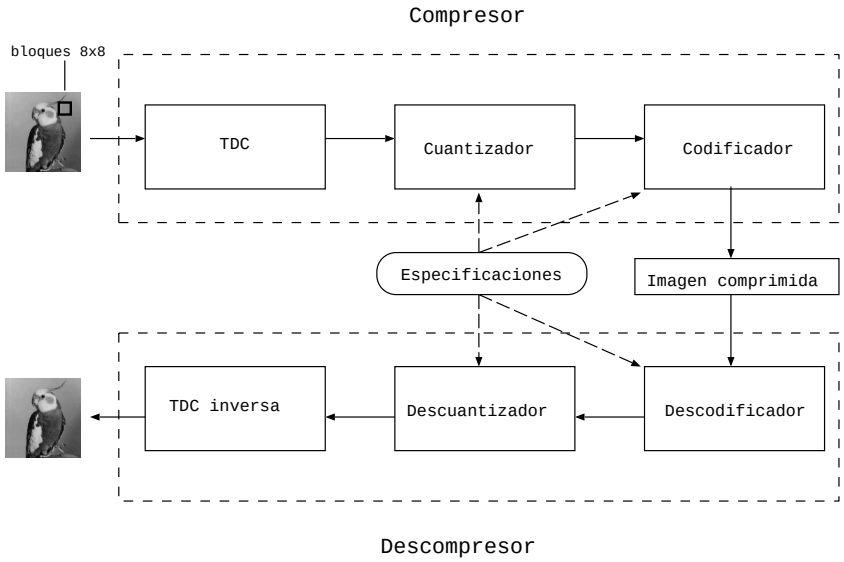


Figura 5.2: Las distintas etapas que conforman el estándar de compresión JPEG tanto en la compresión como en la descompresión. De todas ellas la principal fuente de pérdidas es la etapa de cuantización. La codificación por entropía no introduce ningún tipo de pérdida como tampoco lo haría el cálculo de la transformada del coseno de no ser por la imposibilidad de obtener sus coeficientes con total precisión.

TDC y TDCI

A la entrada del codificador las muestras de la imagen se agrupan en bloques de 8×8 pixels, desplazando sus valores de enteros sin signo en el rango $[0, 2^p - 1]$ a enteros con signo en el rango $[-2^{p-1}, 2^{p-1} - 1]$ e introduciéndolos en la *TDC directa*. A la salida del descodificador, la *TDC inversa* (TDCI) genera bloques de 8×8 muestras para formar la imagen reconstruida.

La TDC está muy relacionada con la *transformada discreta de Fourier*. Cada bloque de 8×8 se considera como una señal discreta función de dos coordenadas espaciales, x e y. La salida de la TDC es un vector de 64 componentes que constituye el espectro de frecuencias de la señal de entrada también de 64 muestras. La ecuación para la TDC 8×8 es

$$F(u,v) = \frac{1}{4} C(u)C(v) \left[\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right]$$

y la correspondiente a la transformada inversa 8×8 es

$$f(x,y) = \frac{1}{4} \left[\sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right]$$

donde

$$\begin{aligned} C(u), C(v) &= \frac{1}{\sqrt{2}} \quad \text{para } u, v = 0 \\ C(u), C(v) &= 1 \quad \text{en otro caso} \end{aligned}$$

El coeficiente con frecuencia cero en ambas dimensiones $F(0,0)$ se denomina coeficiente de *componente continua* y los restantes 63 coeficientes son las *componentes alternas de la señal*. Como las intensidades de una imagen varían normalmente de forma muy lenta de un punto a otro, el paso de aplicar la TDC proporciona la base para conseguir la compresión de datos al concentrar la mayor parte de la señal de entrada en las frecuencias bajas. Para un bloque 8×8 de una imagen típica la mayor parte de las frecuencias tienen amplitud cero o cercana a cero y no necesitan ser codificadas.

En el descodificador la TDCI invierte el paso anterior cogiendo los 64 coeficientes de la TDC (que en ese punto han sido cuantizados, como veremos luego) y reconstruye una señal de 64 puntos. Si la TDC y la TDCI pudieran calcularse con precisión absoluta y si los coeficientes no se cuantizaran, como se explica a continuación, la señal original podría recuperarse exactamente. En un principio, por tanto, la TDC no introduce pérdidas en la codificación, simplemente transforma las muestras a un dominio en el que puedan ser codificadas más eficientemente.

Aunque no entraremos en más detalles, el hecho de que la TDC utilice funciones trascendentes implica que ninguna instrumentación física puede calcularla con precisión absoluta. Esto abre la puerta a numerosos algoritmos que intentan computar la TDC con la mayor precisión posible.

Cuantización

Cada uno de los 64 coeficientes de la TDC se cuantiza uniformemente según una tabla de cuantización de 64 elementos que se proporciona como entrada al codificador. El propósito de la cuantización es conseguir la compresión de los datos al no representar los coeficientes con mayor precisión de la que es necesario para conseguir la calidad de imagen deseada.

Pese a manejar un vector de 64 coeficientes, la cuantización empleada por el método JPEG se puede considerar como una cuantización escalar (y no vectorial) sobre cada componente de forma individual. El cuanto utilizado es mayor para las componentes de frecuencias más altas ya que éstas juegan un papel mucho menos importante que las frecuencias más bajas. La cuantización es la principal fuente de pérdidas en los codificadores basados en la TDC.

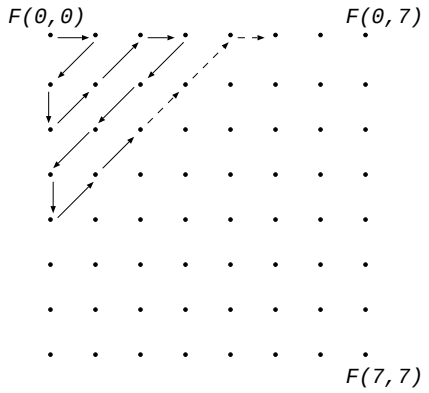


Figura 5.3: La reordenación en zigzag de los coeficientes de la TDC junta los coeficientes de frecuencias altas y facilita su posterior compresión debido a que su valor, al menos en imágenes de tono continuo, será mayoritariamente cero.

La cuantización se define como la división de cada coeficiente de la TDC por su correspondiente cuanto, seguida de un redondeo al entero más cercano:

$$F^Q(u,v) = \text{redondeo}\left[\frac{F(u,v)}{Q(u,v)}\right]$$

donde Q es la *matriz de cuantización* y $Q(u,v)$ es el cuanto a aplicar al elemento de coordenadas (u,v). Aunque no es obligatorio, Q suele ser simétrica. Los cuantos serán pequeños en la parte superior izquierda (bajas frecuencias) y grandes en la inferior derecha (altas frecuencias). Con ello muchos de los coeficientes de frecuencias altas se hacen nulos y son más sencillos de codificar, a la vez que los de bajas frecuencias permanecen casi inalterados. Un cuanto con valor 1 proporciona el resultado más preciso; sin embargo, aún así se producen pérdidas debido a que los coeficientes exactos de la TDC no suelen ser enteros.

El valor de F^Q es normalizado por el tamaño del cuanto. La *descuantización* es la función inversa

$$F^{Q'}(u,v) = F^Q(u,v) \cdot Q(u,v)$$

Ordenación en zigzag

Finalmente, los coeficientes cuantizados se reordenan en la secuencia en zigzag mostrada en la figura 5.3. Este orden facilita la codificación por entropía del próximo paso al situar los coeficientes de baja frecuencia antes de los de frecuencias altas (que son probablemente nulos).

Esto permite un mecanismo adicional de compresión *sin pérdidas*, ya que las largas ristras de ceros pueden reducirse considerablemente si almacenamos únicamente su longitud en lugar de la sucesión completa. Para cada coeficiente no nulo de la TDC se almacena el número de ceros que lo preceden, el número de bits que se necesitan para representar la amplitud del número y la amplitud en sí.

Tras este paso los coeficientes de la TDC cuantizados dejan de verse como tales y, al colocar correlativos los coeficientes de todos los bloques de la imagen, obtenemos una secuencia intermedia de símbolos que serán codificados por el siguiente (y último) paso.

Codificación por entropía

Como último paso de la compresión JPEG se obtiene una reducción adicional del volumen de los datos *sin pérdidas* al codificar los coeficientes cuantizados más compactamente según sus propiedades estadísticas mediante uno de estos dos métodos de *codificación por entropía*: *codificación de Huffman* o *codificación aritmética*. Estas técnicas de codificación se basan en utilizar un número pequeño de bits para aquellos símbolos que aparecen con mayor frecuencia (probabilidad) en un canal, asignando códigos con longitudes mayores que la del símbolo original a los símbolos ocasionales.[2]

5.6. Compresión basada en wavelets

La compresión basada en wavelets constituye uno de los campos de investigación en compresión de señales que mayor atención está recibiendo en los últimos años. Por tratarse de una tecnología nueva con apenas una década de desarrollos es aún demasiado pronto para juzgar su eficiencia, pero los resultados obtenidos hasta ahora son muy prometedores.

Este tipo de compresión se basa en la teoría de los wavelets y más concretamente en la transformada discreta con wavelets. Un conocimiento mínimo de esta teoría es necesario para poder profundizar en esta técnica de compresión. Sin embargo, incluso una introducción al tema no puede realizarse en un par de párrafos por lo que se ha preferido incluirla en un apéndice. El lector puede ahora abordar el estudio del apéndice B o bien conformarse con saber que la compresión basada en wavelets utiliza una transformada que, al igual que la transformada discreta del coseno estudiada en 5.5, proporciona

[2]Un estudio profundo de las técnicas de codificación matemática puede encontrarse en el libro de J. Rifà y Ll. Huguet, *Comunicación digital*, Editorial Masson, 1991.

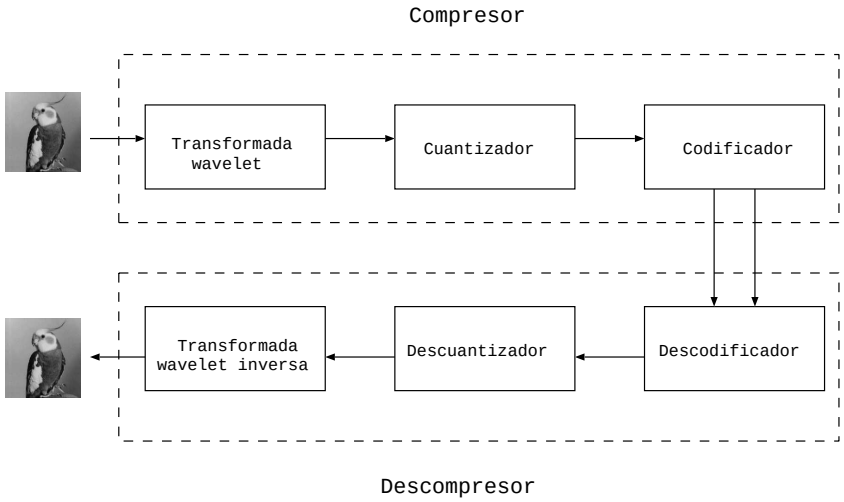


Figura 5.4: El esquema general de las técnicas de compresión basadas en wavelets es muy similar al de las basadas en la transformada discreta del coseno. La transformada con wavelets, sin embargo, parece dotar de mayor potencia a estos métodos con respecto a los basados en la TDC.

un gran número de coeficientes cercanos a cero de los que puede prescindirse sin grandes pérdidas en la calidad de la señal reconstruida.

Se han sugerido en los últimos años un gran número de esquemas de compresión de imágenes basados en wavelets. Todos se ajustan a lo mostrado en la figura 5.4.

Como puede verse en la figura, las etapas de este esquema de compresión son muy similares a las mostradas en la figura 5.2 para el estándar JPEG. La diferencia fundamental está en el tipo de transformada utilizada. En aquel caso se utilizaba la transformada discreta del coseno y aquí la base del método es la transformada discreta con wavelets (TDW).

La TDW sobre la señal bidimensional que es la imagen se lleva a cabo mediante una transformada unidimensional, como la discutida en el apéndice B, aplicada sobre la secuencia obtenida al recorrer por filas o columnas la imagen. Según el tipo de barrido efectuado sobre la imagen distinguiremos entre *TDW horizontal* y *TDW vertical*. Por lo demás, ambas transformaciones son idénticas entre sí e iguales a la TDW.

Para poder capturar mejor la evolución de las intensidades de la imagen, en cada nivel de la codificación subbanda se aplica alternadamente la TDW horizontal o vertical a la subimagen procedente del nivel anterior. En este caso, además, no es necesario iterar el algoritmo de la codificación subbanda

hasta el final. El número de niveles del algoritmo depende de varios factores entre los que se encuentran la cantidad de compresión requerida, el tamaño de la imagen original y la longitud de los filtros utilizados.

Al igual que en la TDC, la codificación elimina (hace cero) aquellos coeficientes con magnitudes pequeñas sin crear una distorsión significativa en la imagen reconstruida. Una forma de eliminar los coeficientes con valores pequeños es aplicando una función umbral

$$T_t(x) = \begin{cases} 0 & \text{si } x < t \\ x & \text{en otro caso} \end{cases}$$

a la matriz de coeficientes de cada nivel. La cantidad de compresión obtenida (y la calidad de la imagen) puede controlarse variando el umbral t. Para obtener mayor compresión pueden cuantizarse los coeficientes no nulos mediante cuantización escalar como la descrita en 5.4.

Para terminar, a las muestras cuantizadas se les aplica algún esquema de compresión sin pérdidas por entropía como los vistos al final de la sección 5.5.

5.7. Compresión fractal

La aplicación de la teoría de los sistemas de funciones iteradas del capítulo 4 a la compresión de imagen se ha convertido en uno de los campos de estudio más fértiles dentro de la codificación de imágenes. Los capítulos anteriores han sentado las bases para un conocimiento profundo de los fundamentos de la geometría fractal. En este capítulo, por otra parte, se han presentado diferentes esquemas de compresión de imagen que compiten en rendimiento con la compresión basada en SFI. El lector puede dar ahora el salto hacia la transformada fractal.

5.8. Comparación de los esquemas de compresión

Es difícil comparar diferentes programas o técnicas de compresión, ya que aparecen problemas en como mínimo dos frentes distintos. En primer lugar los artículos suelen presentar sus resultados sobre imágenes diferentes (incluso aunque a veces parezcan la misma). En segundo lugar, a la hora de mostrar en papel las imágenes comprimidas muchos mecanismos de impresión impiden que puedan apreciarse con claridad determinadas características.

A todo esto hay que unir los problemas derivados de la falta de una medida del error cometido en la compresión que reproduzca adecuadamente la percepción visual humana de la calidad de la imagen, como ya se discutió en el apartado 5.2. Pese a ello, en esta sección estudiaremos, con una pretensión meramente orientativa, los resultados proporcionados por tres programas distintos basados en las principales técnicas de compresión de imagen con pérdidas discutidas en este trabajo: wavelets, JPEG y transformada fractal.

Estos resultados se han tomado de un proyecto de la Universidad de Waterloo[3] que pretendía servir de referente continuo en cuanto a la evaluación de los distintos esquemas de compresión de imagen con pérdidas existentes. Sin embargo, el último estudio que allí aparece data de julio de 1995 lo que lo deja un tanto anticuado. Con todo, los datos de artículos recientes muestran que la situación relativa hoy día entre los distintos métodos es aparentemente similar a la de este estudio, por lo que finalmente lo utilizaremos aquí a falta de uno posterior.

Los programas prototipo considerados para cada técnica son los indicados a continuación:

JPEG Programa `Image Incorporated` 3.1 de Iterated Systems con codificación de Huffman.

Wavelets Programas `Codtree` y `Decdtree` 7.01 de Amir Said y Willian Pearlman con codificación aritmética.

Fractales Programas `Enc` y `Dec` 0.03 de Yuval Fisher [FIS 95] con un árbol cuadricular de 4 niveles y codificación de Huffman.

Se ha indicado el tipo de codificación por entropía utilizado por cada programa, ya que debe tenerse en cuenta que aquellos programas que utilizan codificación aritmética tienen ventaja con respecto a los que no. La codificación aritmética, desafortunadamente, es una tecnología patentada. La codificación de Huffman está ahí para todo el que quiera cogerla.

Antes de mostrar los resultados de la comparativa, debe tenerse muy claro que aquí no se trata de encontrar el *mejor* programa de compresión de imágenes con pérdidas. De momento no existe ningún candidato que destaque lo suficiente sobre el resto y posiblemente nunca existirá, ya que son muchos los factores que entran en juego a la hora de evaluar un programa de compresión y algunos de ellos no tienen un equivalente en otros programas. Por citar algunos:

[3]Las muestras y datos utilizados en esta sección pueden obtenerse directamente a través de la dirección `http://links.uwaterloo.ca/bragzone.base.html`.

(a) JPEG (b) wavelet (c) fractal

Figura 5.5: Una versión generalizada del pasatiempo de descubrir las diferencias. La imagen de la colina con un ratio de compresión de 10 bajo cada una de las tres técnicas consideradas en esta sección.

- ratio de compresión
- calidad de la imagen comprimida
- velocidad de compresión
- velocidad de descompresión
- tipo de imágenes que puede tratar
- patentes y royalties
- portabilidad
- facilidad de uso e instrumentación

Por otra parte, debe evitarse extrapolar hacia el futuro, pues que un determinado codificador sea hoy mejor que otro no quiere decir que lo vaya a seguir siendo mañana. Cada tecnología se encuentra en un estadio de desarrollo diferente: algunas son muy nuevas y aún tienen que ser explotadas; otras son tan maduras que se han desarrollado multitud de mecanismos de optimización para ellas. Lo que sí es cierto es que, por ser JPEG un estándar internacional, muy buena tendrá que ser una técnica alternativa de compresión para lograr desbancarlo.

Ejemplos y resultados

La figura 5.5 muestra la imagen de la colina comprimida con un ratio de compresión de 10 con las tres técnicas a estudio. Puede observarse cómo la

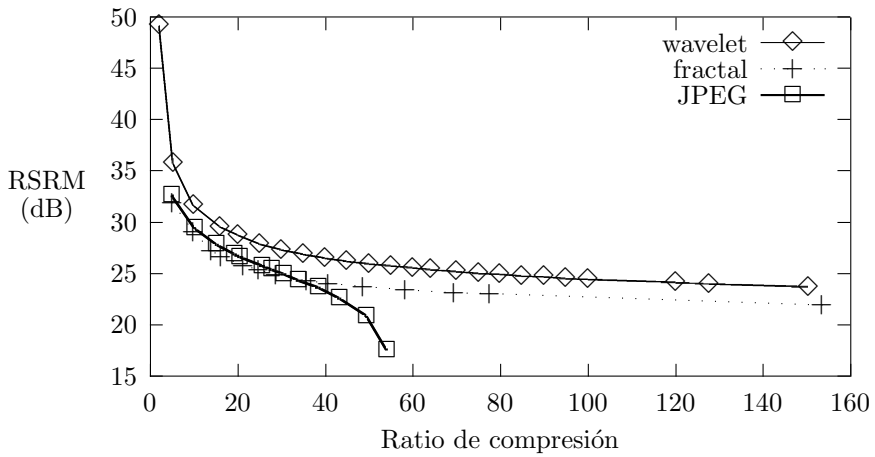


Figura 5.6: Compresión y RSRM obtenidas con cada uno de los tres programas a estudio sobre la imagen de la colina.

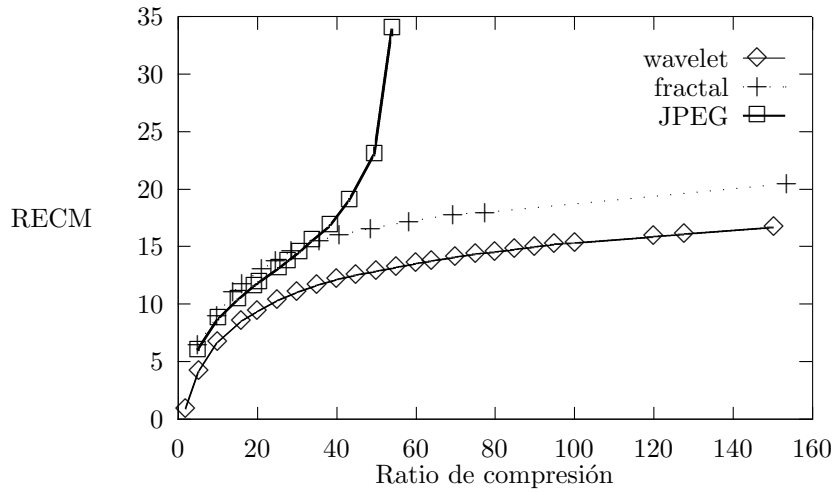


Figura 5.7: Compresión contra RECM en la imagen de la colina. Esta representación es una alternativa a la de la figura 5.6 en la que se utilizaba la RSRM.

calidad visual de las imágenes es totalmente aceptable siendo difícil distinguir una de otra. Esta observación se corrobora en las curvas ratio-distorsión de la gráfica 5.6 donde se aprecia cómo para ratios de compresión bajos (entre 0 y 10) las RSRM proporcionadas por cada técnica son casi iguales. Esta circunstancia aparece también en el resto de curvas ratio-distorsión de esta sección.

(a) JPEG (b) wavelet (c) fractal

Figura 5.8: La imagen del pájaro bajo un ratio de compresión de 40 con cada método.

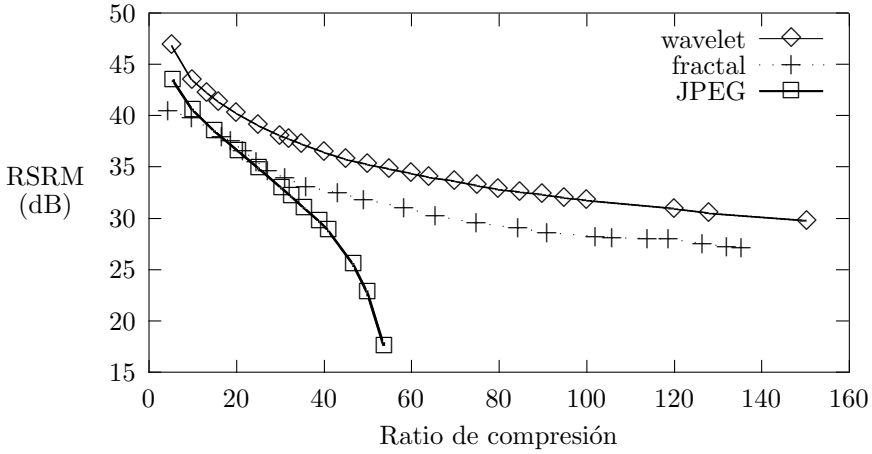


Figura 5.9: Compresión y RSRM obtenidas sobre la imagen del pájaro con cada uno de los tres esquemas de compresión considerados.

La figura 5.6 muestra, además, cómo el estándar JPEG disminuye rápidamente su calidad de compresión al aumentar el ratio de compresión. Esta pérdida de calidad es menos visible en el caso de las compresiones mediante wavelets y fractales,[4] aspecto éste reflejado también en las siguientes curvas ratio-distorsión.

Para que el lector pueda comparar la forma de presentar la información de una curva ratio-distorsión en la que se muestre la RECM con una en la que se considere la RSRM, la figura 5.7 muestra la misma información que

[4]En cualquier caso debe tenerse en cuenta de nuevo que al movernos por valores más bajos de la RSRM, pequeñas diferencias en ésta suponen amplias diferencias de calidad.

(a) JPEG (b) wavelet (c) fractal

Figura 5.10: El puente comprimido con los tres programas a estudio con un ratio de compresión de 32.

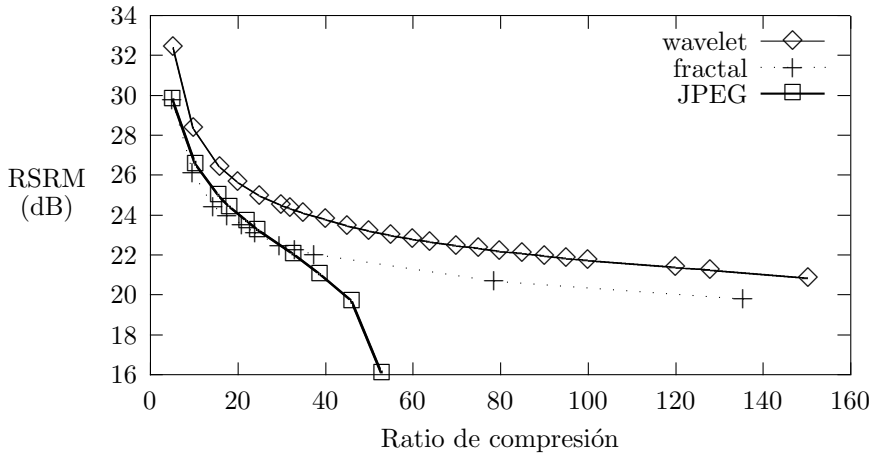


Figura 5.11: Compresión y RSRM de la imagen del puente comprimida según las tres técnicas de esta sección.

la gráfica 5.6 pero considerando la RECM.

La figura 5.8 muestra el resultado de comprimir la imagen del pájaro con un ratio de compresión de 40. En estos niveles la técnica JPEG distorsiona ya claramente la imagen. Los resultados de la compresión fractal son ciertamente aceptables, pese a que se pierde algún detalle en las garras y las plumas de la cabeza. La compresión con wavelets es ligeramente superior y muestra una calidad sorprendente para el elevado nivel de compresión utilizado. Las curvas ratio-distorsión de la figura 5.9 aportan más información acerca de la codificación de la imagen del pájaro. La RSRM se mueve en valores superiores a los mostrados en el caso de la colina debido a que la imagen del pájaro tiene, en general, sus rasgos principales a escalas mayores.

Pese a todas las advertencias realizadas anteriormente, es importante resaltar una vez más que debe evitarse una generalización prematura del comportamiento de los métodos. La figura 5.10 es prueba de ello. En ella se muestran las codificaciones de la imagen del puente con un ratio de compresión de 32. A pesar de ser este ratio inferior al utilizado con el pájaro, el mayor nivel de detalle de la imagen hace que la calidad empeore ostensiblemente. Las curvas ratio-distorsión de esta imagen se muestran en la figura 5.11.

Se ha pretendido mostrar aquí una breve visión del comportamiento de los esquemas de compresión basados en la TDC, en wavelets y en fractales. En los siguientes capítulos se realizará un análisis más detallado centrado en la transformada fractal. Debido a los pobres resultados que suelen obtenerse con cualquiera de estos métodos con imágenes artificiales, todas las muestras de este trabajo serán imágenes naturales.

Capítulo 6

La transformada fractal

Una de las aplicaciones más innovadoras de la geometría fractal y en particular de los sistemas de funciones iteradas es la compresión de imagen. Para proporcionar una visión lo más amplia posible, el tema anterior abordó enfoques alternativos para esta compresión. Aquí, siguiendo particularmente a [FIS 92], [FIS 95], [SAU 96a] y [LU 97], veremos cómo compactar fractalmente una imagen en escala de grises; en 5.3 se explicó cómo generalizar un sistema de este tipo a imágenes en color.

6.1. Historia y fundamentos

En el capítulo 4 se estudió la teoría de los sistemas de funciones iteradas (SFI) desarrollada por Hutchinson y extendida por Barnsley y se sugirió su uso para la compresión de imágenes. La codificación de la hoja de helecho en unos cuantos coeficientes lleva a preguntarse si no será posible obtener codificaciones reducidas similares para cualquier imagen. De hecho, en 1987 Barnsley y sus colegas especularon con ratios de compresión de 10000:1 y con la posibilidad de transmitir vídeo en tiempo real a través de la línea telefónica convencional.

Sin embargo, este enfoque basado en SFI presenta un problema obvio: los fractales que genera un SFI poseen la propiedad de la autosemejanza, es decir, están formados por copias convenientemente transformadas de sí mismos. En el caso de una imagen de un pato, por ejemplo, uno debería poder observar patitos distorsionados por todo su plumaje, lo cual no es evidentemente una suposición muy natural. Los primeros intentos para adaptarse a esta característica de los SFI no produjeron resultados muy alentadores. Así estaban las cosas cuando en 1989 un estudiante de Barnsley, Arnaud Jac-

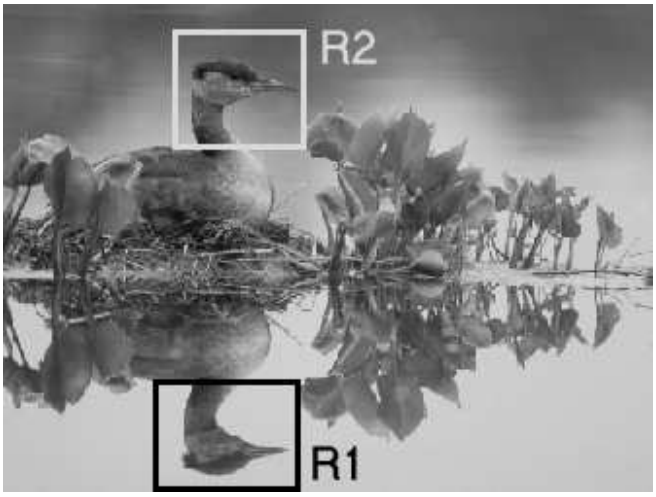


Figura 6.1: La idea clave de Jacquin fue considerar una imagen como formada por copias de partes de sí misma, abandonando el enfoque global de anteriores intentos.

quin, diseñó el primer sistema automático de codificación fractal y dejó a un lado el rígido enfoque basado en SFI globales.

La idea de Jacquin es a primera vista muy simple. En lugar de considerar una imagen como formada por copias de *sí* misma (bajo las transformaciones apropiadas), ahora una imagen estará formada por copias de *partes* de sí misma. En una postal veraniega es difícil que un trozo de una nube se parezca a la postal completa, pero sí que parece posible encontrar otra sección de alguna nube o de otro elemento de la imagen que se parezca al trozo de nube. Las figuras 6.1 y 6.2 muestran regiones de una imagen que son similares a diferentes escalas y bajo una transformación apropiada. El enfoque general consiste en subdividir la imagen mediante una partición (en el caso más sencillo en regiones cuadradas de tamaño fijo) y encontrar para cada región resultante otra parecida en la imagen. Este esquema se conoce como *sistemas de funciones iteradas particionadas* (SFIP) o *locales* (SFIL). Al proceso de obtener el SFIP asociado a una imagen le llamaremos transformada fractal.

Los resultados anteriores significaron el inicio de una prolífica sucesión de investigaciones, que llega hasta nuestros días, para determinar numerosos aspectos todavía abiertos de la transformada fractal entre los que pueden considerarse la forma de particionar la imagen (segmentación), el mecanismo para determinar la correspondencia entre regiones, la extensión a codificación de vídeo o la reducción de la complejidad computacional, principalmente la temporal. Algunas posibilidades para los elementos anteriores se discutirán más adelante.

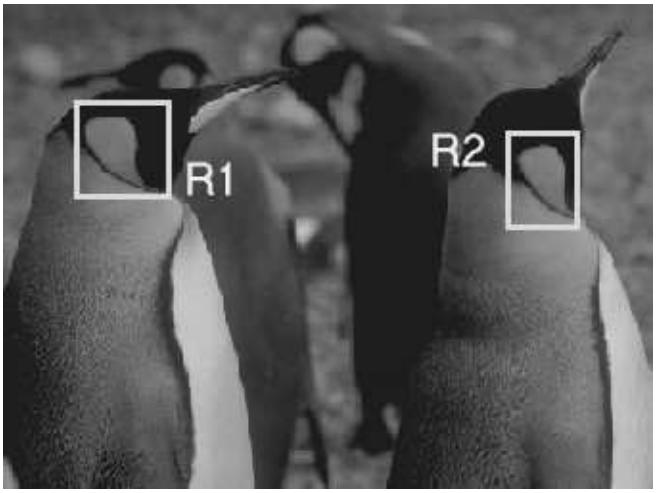


Figura 6.2: Otra muestra de regiones similares en una imagen bajo una transformación apropiada.

6.2. Modelo de imagen

Para poder formalizar los desarrolllos teóricos de este capítulo necesitamos un modelo matemático de imagen en escala de grises. Aunque existen modelos más complejos, para nuestros propósitos trabajaremos con el espacio $\Delta = \{\tau : I^2 \to I\}$ de imágenes cuadradas de lado 1, donde I^2 es el cuadrado unidad con valores en el rango $I = [0, 1]$. La función τ da el nivel de gris de cada punto de la imagen.[1] La generalización a imágenes de otros tamaños es directa.

También necesitaremos en la siguiente sección una métrica que nos proporcione la distancia entre dos imágenes. Aunque en la práctica se consideran métricas como la raíz del error cuadrático medio (discutida en el página 81), es más fácil demostrar las propiedades de contractividad y convergencia con métricas menos elaboradas como la del supremo,

$$d_{sup}(\tau_1, \tau_2) = \sup_{(x,y)\in I^2} |\tau_1(x, y) - \tau_2(x, y)|$$

donde la resta se entiende como resta acotada. En cualquier caso, y aunque no lo demostremos, los resultados posteriores se cumplen con muchas otras métricas, en concreto con el error cuadrático medio.

Un resultado de gran importancia para los desarrollos presentados a continuación, y que aquí no probaremos, es que el espacio (Δ, d_{sup}) es completo.

[1] Sería más apropiado $I^2 \to \mathbf{R}$ para que queden bien definidas las sumas y restas de imágenes, pero mantendremos esta notación.

6.3. Sistemas de funciones iteradas particionadas

Nuestro objetivo es repetir la técnica de generación de fractales mediante SFI, pero desde una perspectiva más amplia. Buscaremos un conjunto de transformaciones $f_1, \ldots, f_n$, las agruparemos bajo una transformación $F = \cup f_i$, mostraremos que bajo ciertas condiciones F es contractiva, deduciremos que tiene un atractor asociado e intentaremos escoger F tal que éste aproxime suficientemente una imagen dada.

Definición 6.1 *Sea X un espacio métrico completo y sea $D_i \subset X$ con $i = 1, \ldots, n$. Un sistema de funciones iteradas particionadas (SFIP)*[2] *es una colección de aplicaciones contractivas $f_i : D_i \to X$ con $i = 1, \ldots, n$.*

Sean $D_1, \ldots, D_n$ y $R_1, \ldots, R_n$ subconjuntos de I^2 que denominaremos dominios y rangos, respectivamente. Es habitual confundir un subconjunto de I^2 con la función definida sobre dicho subconjunto. Por lo tanto, aunque llamemos dominio a D_i y rango a R_i, el dominio y el rango real son los productos cartesianos $D_i \times I$ y $R_i \times I$.

Sea $v_1, \ldots, v_n : I^3 \to I^3$ un conjunto de aplicaciones. Definamos f_i como la restricción

$$f_i = v_i|_{D_i \times I}$$

Las aplicaciones $f_1, \ldots, f_n$ conforman el SFIP. La idea es que el dominio de f_i esté restringido, aunque, aun así, su forma puede ser muy general. La transformación f_i opera sobre τ haciendo $f_i(\tau) = f_i(x, y, \tau(x, y))$ e interpretamos el resultado como una imagen sobre I^2. Para que todo vaya bien debemos imponer la siguiente restricción adicional sobre los f_i:

Definición 6.2 *Se dice que las transformaciones $f_1, \ldots, f_n$ enlosan I^2 si para toda $\tau \in \Delta$,*

$$\bigcup_{i=1}^{n} f_i(\tau) \in \Delta$$

Cuando aplicamos f_i a la parte de la imagen $\tau \cap (D_i \times I)$ sobre D_i el resultado es una subimagen sobre un rango que denominaremos R_i.[3] El en-

[2] Lo bueno sería poder utilizar el teorema del punto fijo (como hicimos con los SFI) para definir un único punto fijo del SFIP. Pero de forma general esto no es posible, ya que al estar restringidos los dominios, el punto inicial es importante: de no tener cuidado nos podemos quedar tras una iteración con un conjunto vacío. No entraremos en más detalles, ya que en el caso que nos ocupa (la codificación de imágenes) no tendremos este problema.

[3] En este caso diremos que D_i cubre R_i.

losado implica que $I^2 = \cup_{i=1}^{n} R_i$. Además, para simplificar nuestra discusión asumiremos que los R_i son disjuntos.

Al imponer la condición de enlosado sobre las transformaciones de un SFIP, estamos cerca de poder codificar una imagen, pero antes debemos determinar la forma de que $F = \cup_{i=1}^{n} f_i$ sea contractiva.

Definición 6.3 *Si $f : \mathbf{R}^3 \to \mathbf{R}^3$ es una aplicación con $f(x, y, z_1) = (x', y', z_1')$ y $f(x, y, z_2) = (x', y', z_2')$, entonces diremos que f es contractiva z si existe un número real positivo $s < 1$ tal que*

$$d(z_1', z_2') \leq s \cdot d(z_1, z_2)$$

donde d es la distancia euclídea habitual, y si, además, x' e y' son independientes de z_1 o de z_2, para todo x, y, z_1, z_2.

Si consideramos transformaciones que presenten contractividad z, es sencillo demostrar la contractividad de un SFIP.

Proposición 6.1 *Si $f_1, \ldots, f_n$ son contractivas z, entonces*

$$F = \bigcup_{i=1}^{n} f_i$$

es contractiva en Δ con la métrica del supremo.

Demostración Sea $s = \max s_i$, donde s_i es la razón de contractividad z de f_i; entonces

$$\begin{aligned}
d_{sup}(F(\tau_1), F(\tau_2)) &= \sup\{\, |F(\tau_1)(x, y) - F(\tau_2)(x, y)| : (x, y) \in I^2\} \\
&= \sup\{\text{componente } z \text{ de } |f_i(x, y, \tau_1(x, y)) \\
&\quad - f_i(x, y, \tau_2(x, y))| : (x, y) \in D_i,\ i = 1, \ldots, n\} \\
&\leq \sup\{s_i\, |\tau_1(x, y) - \tau_2(x, y)| : i = 1, \ldots, n\} \\
&\leq \sup\{s\, |\tau_1(x, y) - \tau_2(x, y)|\} \\
&\leq s \cdot \sup\{|\tau_1(x, y) - \tau_2(x, y)|\} \\
&\leq s \cdot d_{sup}(\tau_1, \tau_2)
\end{aligned}$$

□

Nótese cómo la independencia respecto a z de las coordenadas x e y de f_i se ha utilizado para pasar de la igualdad a la primera desigualdad. Hemos

demostrado que si escogemos f_i de forma que sea contractiva en el eje z, por ejemplo si f_i es de la forma

$$f_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} u_i \\ v_i \\ o_i \end{pmatrix} \tag{6.1}$$

con $s_i < 1$, entonces $F = \cup f_i$ es contractiva bajo d_{sup}. La condición de enlosado nos asegura que al evaluar $F = \cup f_i$ obtendremos una imagen (función) otra vez. Esto es necesario para poder iterar F. La contractividad de F en (Δ,d_{sup}) determina un único punto fijo en Δ al ser (Δ,d_{sup}) un espacio métrico completo (teorema del punto fijo).

La transformación 6.1 puede descomponerse en dos partes diferentes según su forma de actuar sobre la imagen:

- una transformación geométrica

$$g_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

  donde a_i, b_i, c_i, d_i, u_i y v_i representan traslaciones, giros, reflejos y homotecias como los mostrados en 4.2.

- una transformación que controla la escala de grises

$$m_i(z) = s_i \cdot z + o_i$$

  donde s es el escalado del contraste y o el ajuste de brillo.

La transformación geométrica transforma el dominio D_i al tamaño y posición exactos de R_i por lo que la distancia entre los dos bloques en el sentido geométrico es cero.[4] Es la distancia entre los valores de gris la que hay que minimizar.

Ahora ya tenemos las claves para codificar una imagen. Dada una colección de aplicaciones contractivas z formada por $f_1, \ldots, f_n$ que enlosan I^2, sabemos que $F = \cup f_i$ define un atractor τ_A en el espacio de imágenes Δ. A partir de F podemos obtener la imagen asociada de manera similar a como procedimos en el caso de los SFI, esto es, tomando una imagen cualquiera $\tau_0 \in \Delta$ e iterando $F(\tau_0)$, $F(F(\tau_0)) = F^2(\tau_0)$, ... La cuestión inversa, esto es, encontrar un sistema F tal que su atractor asociado aproxime una

[4] En el caso más sencillo $a_i = d_i = 1/2$ y $b_i = c_i = 0$ con lo que los dominios han de tener un tamaño doble al de los rangos y los valores de v_i y u_i simplemente trasladan D_i hasta el bloque ocupado por R_i.

imagen dada τ (la obtención exacta de τ no será en general posible) se puede afrontar considerando el teorema del collage discutido en la página 71, y encontrando, por tanto, dominios $D_1, \ldots, D_n$ y sus correspondientes transformaciones $f_1, \ldots, f_n$ tales que

$$\tau \approx F(\tau) = \bigcup_{i=1}^{n} f_i(\tau)$$

En resumen el proceso de codificación consiste en particionar I^2 en un conjunto de rangos[5] R_i. Para cada R_i se busca un dominio $D_i \subset I^2$ y una transformación $f_i : D_i \times I \to I^3$ tal que $f_i(\tau)$ esté tan cerca de $\tau \cap (R_i \times I)$ como sea posible, esto es, tal que la distancia

$$d(\tau \cap (R_i \times I), f_i(\tau))$$

se minimice.[6] Por supuesto, las f_i que definen F deben elegirse de forma que F sea contractiva.[7]

Ahora podríamos explicar cómo segmentar la imagen y cómo determinar los coeficientes señalados anteriormente para cada f_i. Sin embargo, lo dejaremos para una sección posterior. En la siguiente ofreceremos básicamente gran parte de la información presentada aquí pero desde otro punto de vista. Abandonamos los SFI particionadas para considerar la compresión fractal como una variante de la cuantización vectorial discutida en 5.4.

6.4. Cuantización vectorial y codificación fractal

El esquema básico de la compresión fractal de imágenes es muy similar a la cuantización vectorial con eliminación de media y ganancia de forma presentada en la página 84. La principal diferencia entre ambos es que en la CV se utiliza un libro de códigos fijo que debe estar disponible para el descodificador, mientras que en la codificación fractal el libro de códigos es virtual (no se almacena explícitamente) y está formado por regiones de la imagen original. Lo anterior parece una contradicción, ya que es precisamente labor

[5] Aunque los términos rango y dominio están ampliamente extendidos, otros autores (veáse [LU 97], por ejemplo) prefieren utilizar región destino y región de referencia.

[6] Ya se ha dicho que son posibles muchas otras métricas distintas a la del supremo, que por otra parte no es lo suficientemente precisa al calcular la distancia en base a un único punto. En concreto, aquí podría utilizarse para d el error cuadrático medio como se considerará más adelante.

[7] No entraremos en detalles, pero la condición de que F sea contractiva puede relajarse por la de que sea eventualmente contractiva . Una aplicación f es eventualmente contractiva si existe un valor n tal que f^n es contractiva. Existen versiones del teorema del punto fijo y del teorema del collage para este tipo de aplicaciones [FIS 95, pp. 36 y 52].

del descodificador restituir la imagen original con lo que no tendrá acceso al libro de códigos. Cabe preguntarse, por tanto, cómo puede el descodificador reconstruir la imagen original si ésta se codifica por bloques considerados como copias escaladas de otros bloques de la imagen más bloques con un valor de gris constante.[8]

La respuesta al interrogante anterior debe estar clara para el lector familiarizado con el teorema del punto fijo. Aun así presentamos a continuación la brillante discusión disponible en [SAU 96a].

Ejemplo Vamos a codificar, por simplificar un poco las cosas, un número real en lugar de una región de una imagen, por ejemplo $\pi = 3{,}14159\ldots$ Supongamos que los libros de códigos para la escala y el desplazamiento son $s \in \{0, 0{,}25, 0{,}5, 0{,}75\}$ y $o \in \{0{,}0, 0{,}4, 0{,}8, 1{,}2, 1{,}6, 2{,}0\}$.

El libro de códigos de formas tiene un único número: el propio número π. Si obtenemos todos los posibles valores de $s\pi + o$ con s y o obtenidos de los libros de códigos anteriores, veremos que $s = 0{,}75$ y $o = 0{,}8$ dan la mejor aproximación a π:

$$s \cdot \pi + o = 0{,}75 \cdot \pi + 0{,}8 = 3{,}1561\ldots$$

Con esto, el codificador puede pasar la siguiente información al descodificador: *el número original es aproximadamente 0.75 veces él mismo más 0.8*. Hay muchos números que satisfacen esta descripción. Sin más información el descodificador podría determinar cualquiera de ellos. Sin embargo, uno de ellos es un número x especial, aquél que es *exactamente* 0.75 veces él mismo más 0.8, esto es,

$$x = 0{,}75x + 0{,}8$$

Al resolver esta ecuación obtenemos $x = 3{,}2$, que se interpretaría como el valor descodificado. La resolución de $x = 0{,}75x + 0{,}8$ es sencilla, pero al tratar con imágenes con miles de pixels, el correspondiente sistema de ecuaciones es tan grande que no puede resolverse directamente, sino sólo por iteración. Esto puede comprobarse con nuestro simple ejemplo. Si definimos un operador $T : \mathbf{R} \to \mathbf{R}$ como $Tx = 0{,}75x + 0{,}8$, entonces la información que el codificador pasa al descodificador puede verse como $\pi \approx T\pi$ y debe obtenerse el punto fijo de la ecuación $x = Tx$. Dado un valor inicial arbitrario x_0, aplicamos iterativamente T para obtener

$$x_1 = Tx_0,\ x_2 = Tx_1,\ x_3 = Tx_2, \ldots$$

Por ejemplo, con $x_0 = 0$ obtenemos $x_1 = 0{,}8, x_2 = 1{,}4, x_3 = 1{,}85, \ldots, x_{20} = 3{,}192\ldots$ y $x_{30} = 3{,}1995\ldots$ Esta secuencia converge

[8] El lector que no lo haya hecho ya, debe abordar el estudio de la sección 5.4, especialmente de la parte dedicada a la cuantización vectorial con eliminación de media y ganancia de forma.

al punto fijo 3,2 conocido como atractor del operador T. Este resultado no es coincidencia. Como vimos, si el factor de escalado en valor absoluto es menor que la unidad, esto es, $|s| < 1$, el teorema del punto fijo nos asegura la convergencia al único punto tal que $x = Tx$.

Las propiedades mostradas en el ejemplo anterior también se cumplen en el caso de imágenes. El codificador trabaja de una forma similar a la CV-EMGF. Aquí, sin embargo, el libro de códigos de formas no se proporciona a priori como resultado de algún algoritmo de diseño. En su lugar, el libro de códigos de formas está constituido por bloques extraídos de la misma imagen a codificar. Por lo tanto, estos bloques no se normalizan para que tengan media cero y varianza uno como se hacía en el caso de la CV-EMGF. Cada imagen, en definitiva, posee su propio libro de códigos.

Ejemplo Supongamos que la imagen se segmenta en bloques de 4×4 pixels, lo que hemos denominado antes rangos. Cada rango R debe aproximarse como $R \approx sD + o\mathbf{1}$ donde D es un bloque 4×4 del libro de códigos de formas. Cualquier dominio de tamaño 8×8 de la imagen se reduce mediante submuestreo de sus pixels al tamaño deseado de 4×4 pixels y se añade al libro de códigos. Para una imagen de 512×512 este proceso genera un libro de códigos bastante grande con $(512 - 8 + 1)^2 = 255025$ bloques. Para reducir el número de bloques a una cantidad más manejable pueden considerarse separaciones de más de un pixel entre los dominios como se discutirá en el siguiente capítulo.

El codificador debe resolver por tanto el siguiente problema: para cada rango encontrar la mejor aproximación $R \approx sD + o\mathbf{1}$. A los coeficientes s y o se les denomina escalado y desplazamiento. Para obtener los valores óptimos para s, o y D deben evaluarse en principio todos los bloques D posibles y determinar para cada uno de ellos los mejores coeficientes s y o. En el ejemplo unidimensional anterior con π obtuvimos todas las combinaciones posibles sobre los libros de códigos de s y o y escogimos la mejor. Aunque este enfoque podría aplicarse igualmente con bloques de imágenes, su elevado coste computacional temporal lo hace inviable. Afortunadamente, existen formas de evitar estas evaluaciones y se discutirán en el siguiente apartado.

Los valores obtenidos para s y o se cuantizan, normalmente mediante cuantización escalar uniforme, obteniendo unos valores redondeados $\bar{s}$ y $\bar{o}$. Sea $E(R, D)$ la función que devuelve la diferencia entre dos regiones del mismo tamaño de una imagen. Un codificador fractal básico con bloques de dimensión fija tendría el siguiente esquema genérico:

1. *Segmentación de la imagen.* Dividir la imagen a codificar en bloques de tamaño fijo, por ejemplo, 4×4. Los bloques resultantes se denominan rangos R_i.

2. *Libro de códigos de dominios.* Recorrer la imagen para crear una lista de dominios cuyo tamaño es el doble del de los rangos. Promediando grupos de cuatro pixels, reducir el tamaño de los dominios para que concuerde con el de los rangos.

3. *Búsqueda.* Para cada rango R obtener una aproximación lo más buena posible $R \approx sD + o\mathbf{1}$ siguiendo estos pasos:

   a) Para cada dominio D_i calcular los coeficientes s y o que mejor aproximan a R, cuantizarlos si procede y utilizando los coeficientes cuantizados $\bar{s}$ y $\bar{o}$ calcular el error $E(R, D_i)$.

   b) Entre todos los bloques D_i encontrar aquel D_k con menor error $E(R, D_k) = \min_i E(R, D_i)$.

   c) Mostrar el *código fractal* del bloque R formado por $\bar{s}$, $\bar{o}$ y el índice k que identifica al bloque óptimo D_k.

Como ya hemos visto el conjunto de códigos fractales resultante del algoritmo anterior, denominado *modelo fractal*, no permite al descodificador la obtención inmediata de una aproximación de la imagen original, ya que se trata, en realidad, de la descripción de un operador. Así, como vimos en el ejemplo con π, podemos desarrollar las operaciones indicadas por el modelo fractal sobre *cualquier* imagen inicial τ_0 para obtener una nueva imagen $T\tau_0$. Si repetimos el proceso sobre la nueva imagen de forma iterativa la secuencia resultante

$$\tau_1 = T\tau_0,\ \tau_2 = T\tau_1,\ \tau_3 = T\tau_2, \ldots$$

converge a un atractor que aproximará la imagen original siempre que T sea contractiva, esto es, cuando $|s| < 1$. El teorema del collage motiva la minimización del error $E(R, D_k)$ en el codificador para que la imagen obtenida tras un número suficiente de iteraciones en el descodificador aproxime en la mayor medida posible a la original.

6.5. Obtención de los coeficientes de los códigos fractales

La determinación de unos valores adecuados para el escalado y el desplazamiento es un aspecto crucial para reducir la distorsión de la imagen descodificada. Aunque existen otras aproximaciones, aquí consideraremos

únicamente dos formas de obtener estos coeficientes. La primera [SAU 96a] se basa en el método de los mínimos cuadrados, mientras que la segunda [LU 97] reduce el número de calculos necesarios sacrificando algo de calidad en los valores obtenidos.

Método de los mínimos cuadrados

Consideremos dos bloques R y D con n pixels de intensidades $r_1, r_2, \ldots, r_n$ y $d_1, d_2, \ldots, d_n$. Si trabajamos con el error cuadrático al realizar la selección de los mejores coeficientes, esto es,

$$E(R,D) = \sum_{i=1}^{n} (s \cdot d_i + o - r_i)^2$$

podemos obtener los valores de s y o que minimizan R igualando a cero las derivadas parciales respecto a ambos para obtener:

$$s = \frac{n(\sum_{i=1}^{n} d_i r_i) - (\sum_{i=1}^{n} d_i)(\sum_{i=1}^{n} r_i)}{n\sum_{i=1}^{n} d_i^2 - (\sum_{i=1}^{n} d_i)^2} \tag{6.2}$$

y

$$o = \frac{1}{n}\left(\sum_{i=1}^{n} r_i - s\sum_{i=1}^{n} d_i\right)$$

En ese caso el error cuadrático es

$$E(R,D) = \frac{1}{n}\left[\sum_{i=1}^{n} r_i^2 + s\left(s\sum_{i=1}^{n} d_i^2 - 2\sum_{i=1}^{n} d_i r_i + 2o\sum_{i=1}^{n} d_i\right) + o\left(on - 2\sum_{i=1}^{n} r_i\right)\right]$$

Si el denominador de la ecuación 6.2 es cero, entonces $s = 0$ y $o = \sum_{i=1}^{n} r_i/n$. En este caso no es necesario almacenar la información sobre el dominio pues éste es indiferente.

Método del escalado constante

Aunque muchas de las subexpresiones de las ecuaciones anteriores pueden evaluarse sólo una vez al comienzo del codificador y almacenarse convenientemente en tablas, la obtención de los valores óptimos de s y o no deja de precisar una gran cantidad de operaciones. Una solución subóptima consiste en considerar que el escalado es un valor constante (por ejemplo, $s = 3/4$) y calcular el desplazamiento o que minimice $E(R,D)$ a partir de esta premisa.

En este caso, si $\bar{R}$ y $\bar{D}$ representan las intensidades medias de los bloques R y D, es decir,

$$\bar{R} = \frac{\sum_{i=1}^{n} r_i}{n}$$

y

$$\bar{D} = \frac{\sum_{i=1}^{n} d_i}{n}$$

entonces se toma $o = \bar{R} - \bar{D}$. Aunque de esta forma nos ahorramos tener que almacenar s para cada rango porque su valor es constante, la calidad de la imagen descodificada se resiente, especialmente para ratios altos de compresión.

6.6. Compactación de los códigos fractales

Determinar el cuanto a utilizar para los valores del desplazamiento o y el escalado s de cada código fractal es equivalente a establecer el número de bits con los que se almacenará cada uno de ellos. Los resultados óptimos [FIS 95, p. 63] suelen obtenerse con 5 bits para s y 7 para o, aunque valores de 4 y 6, respectivamente, también proporcionan resultados aceptables. También podría ser factible considerar la codificación adaptativa de estos coeficientes, ya que sus distribuciones presentan una estructura bastante regular [FIS 95, p. 63].

Por otra parte, no es necesario guardar la posición de cada rango R_i, pues éstos pueden ser conocidos por el descodificador si el codificador utiliza algún orden determinado en el almacenamiento de sus códigos fractales asociados (por ejemplo, mediante un barrido por filas). La información sobre el dominio D_i, sin embargo, no es redundante y para referenciarlo se usarán $\lceil \log_2 N \rceil$ bits, donde N es el tamaño de la lista de dominios considerada al codificar.

El codificador necesitará, por tanto, de una rutina que empaquete adecuadamente cada coeficiente según el número de bits requerido. En el otro lado, evidentemente, el descodificador deberá desempaquetar correctamente los datos anteriores.

6.7. Ejemplos

En esta sección se presentan los resultados obtenidos sobre la imagen del pájaro con un compresor desarrollado por el autor mezclando el código sugerido por [LU 97] y [FIS 95] con aportaciones propias. El codificador se basa en el método del escalado constante discutido antes, considerando

Figura 6.3: El pájaro comprimido con rangos de 4×4 con un ratio de compresión de 5.11.

Figura 6.4: El pájaro comprimido con rangos de 8×8 con un ratio de compresión de 19.6.

$s = 3/4$. Además el programa obtiene valores para o en el rango $[-128, 127]$ y los almacena con 8 bits por lo que no hace uso de la cuantización escalar.

La imagen del pájaro tiene dimensiones 256×256 con lo que el número de dominios está limitado entre 0 y $(256^2 - 1) = 65535$ y son necesarios 16 bits para representarlos.[9] En esta imagen, por tanto, son necesarios $16 + 8 = 24$ bits por código fractal. Si consideramos una partición en bloques cuadrados de $n \times n$, con $n = 3, 4, 5, 6, 7, 8, \ldots$, los ratios de compresión serán

[9] En realidad el número de dominios es ligeramente menor y depende del tamaño de los rangos, pero seguirán siendo necesarios 16 bits al menos que los rangos sean mayores de 76×76 pixels.

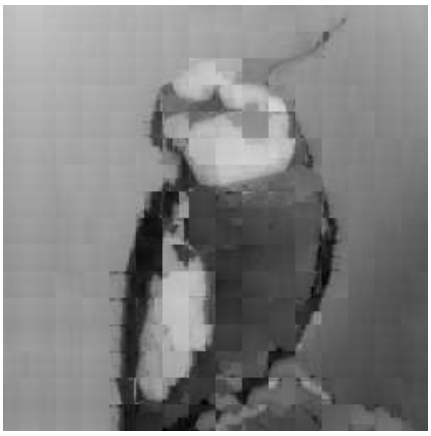


Figura 6.5: El pájaro comprimido con rangos de 16×16 con un ratio de compresión de 77.37.

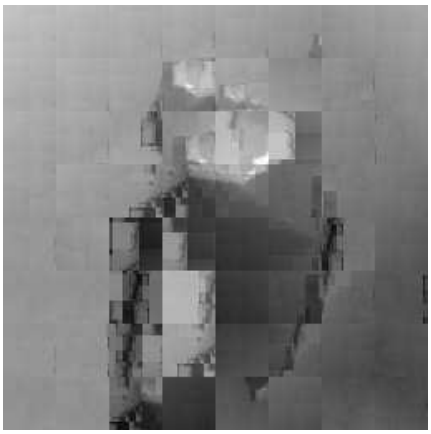


Figura 6.6: El pájaro comprimido con rangos de 32×32 con un ratio de compresión de 293.9.

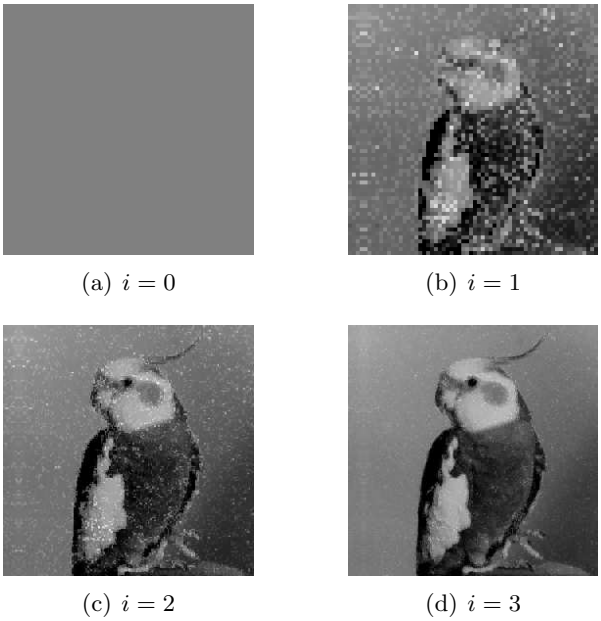
de 3,0, 5,33, 8,33, 12,0, 16,33, 21,33, . . ., respectivamente.

Las figuras de la 6.3 a la 6.6 muestran el resultado de comprimir la imagen del pájaro con distintos tamaños de rangos. La tabla 6.1 presenta los ratios de compresión de cada una de estas figuras junto a las medidas de su distorsión y el tiempo empleado para su codificación.

En la figura 6.7 se muestra la imagen inicial y la obtenida por el descodificador tras distintas iteraciones sobre el modelo fractal. Sin ningún tratamiento adicional al presentado en este capítulo, la convergencia se produce habitualmente en algún lugar entre las 10 y las 20 iteraciones, aunque este

Rangos	Figura	Tiempo (seg)	Compresión	RECM	RSRM
4×4	6.3	241.5	5.11	3.18	38.08
8×8	6.4	159.4	19.6	6.75	31.54
16×16	6.5	126.3	77.37	12.19	26.41
32×32	6.6	100.0	293.9	20.37	21.95

Cuadro 6.1: El efecto del tamaño de bloque de los rangos en el tiempo de compresión, el ratio de compresión y las medidas de distorsión de la imagen. Los tiempos se midieron en una máquina con procesador Pentium II a 233 MHz bajo Linux. Los ratios de compresión no corresponden exactamente a los razonados en el texto debido a la presencia de una cabecera en el fichero comprimido y a algunos bits adicionales incluidos por el codificador al estar diseñado para un esquema más complejo que el comentado en este capítulo.



(a) $i = 0$ (b) $i = 1$

(c) $i = 2$ (d) $i = 3$

Figura 6.7: Resultado de la descodificación de la imagen del pájaro con rangos de 4×4 tras i iteraciones. Como imagen inicial se utiliza una imagen lisa con todos sus pixels con valor de gris constante 128.

valor depende de la imagen y de la complejidad (en el sentido de interdependencias) del modelo fractal generado.

Capítulo 7

Mejoras en la codificación fractal

En el capítulo anterior se describió la transformada fractal en su versión más sencilla y primitiva. Si todo lo conocido sobre la compresión fractal de imágenes se redujera a este esquema básico, no estaríamos hablando de una tecnología prometedora y probablemente estos capítulos no tendrían sentido. Pero el estado actual de la codificación fractal llega mucho más lejos. Durante los últimos años se han publicado numerosos trabajos que superan, en cierta forma, algunos de los múltiples problemas que plantea el enfoque básico de la transformada fractal.

Aquí sólo consideraremos algunas ideas ampliamente difundidas para optimizar en diversos sentidos tanto la compresión como la descompresión fractal. Ampliaciones adicionales pueden encontrarse en la bibliografía, en concreto [SAU 96a] presenta un repaso detallado a muchas de estas innovaciones. En la red [FRE 97] recopila una gran cantidad de material sobre el tema.

7.1. Segmentación de la imagen

La partición de la imagen en rangos cuadrados de tamaño fijo es la forma más sencilla de afrontar el problema de la segmentación de la imagen. Este enfoque, sin embargo, adolece de ser independiente de las características de la imagen considerada. La consideración de una partición adaptativa tiene una gran cantidad de ventajas porque en las imágenes suele haber regiones homogéneas que se pueden cubrir aceptablemente con bloques grandes y regiones con grandes contrastes que precisan bloques más pequeños para

obtener una calidad determinada.

Árboles cuadriculares

La primera idea explorada en el contexto de la codificación fractal fue la considerar dominios cuadrados de distintos tamaños (por ejemplo, de 4, 8 y 16 pixels de ancho). De esta forma surge con naturalidad el uso de árboles cuadriculares en los que cada nodo tiene exactamente cuatro descendientes. Al contrario que en la codificación con bloques de tamaño fijo, en este caso es necesario pasar cierta información sobre el árbol cuadricular subyacente al descodificador.

Con el uso de rangos de tamaño variable es posible diseñar un codificador que proporcione resultados variables. El usuario puede indicar el nivel de primacía de la calidad de la imagen sobre el ratio de compresión mediante un factor de tolerancia del error ε. El codificador va dividiendo recursivamente la imagen hasta que se alcanza este criterio como sigue:

1. Definir una tolerancia ε para el error $E(R, D)$ y un valor máximo y mínimo para el tamaño de los rangos. A continuación dividir la imagen en rangos del tamaño máximo.

2. Crear una pila de rangos e inicializarla metiendo en ella los rangos de tamaño máximo.

3. Mientras la pila no esté vacía, hacer:

   a) Sacar el rango de la cima de la pila y buscar el dominio del tamaño correspondiente que proporcione la mejor aproximación $R \approx sD + o\mathbf{1}$ y el menor error $E(R, D)$.

   b) Si $E(R, D) < \varepsilon$ o si el tamaño del rango es igual al del mínimo tamaño permitido, entonces mostrar el código fractal correspondiente.

   c) Si no, subdividir R en cuatro cuadrantes e introducirlos en la pila.

Variando el valor de ε es posible obtener codificaciones con distintos ratios de compresión y distintos errores respecto a la imagen original. La figura 7.1 representa el árbol cuadricular resultante de dos codificaciones con criterios distintos.

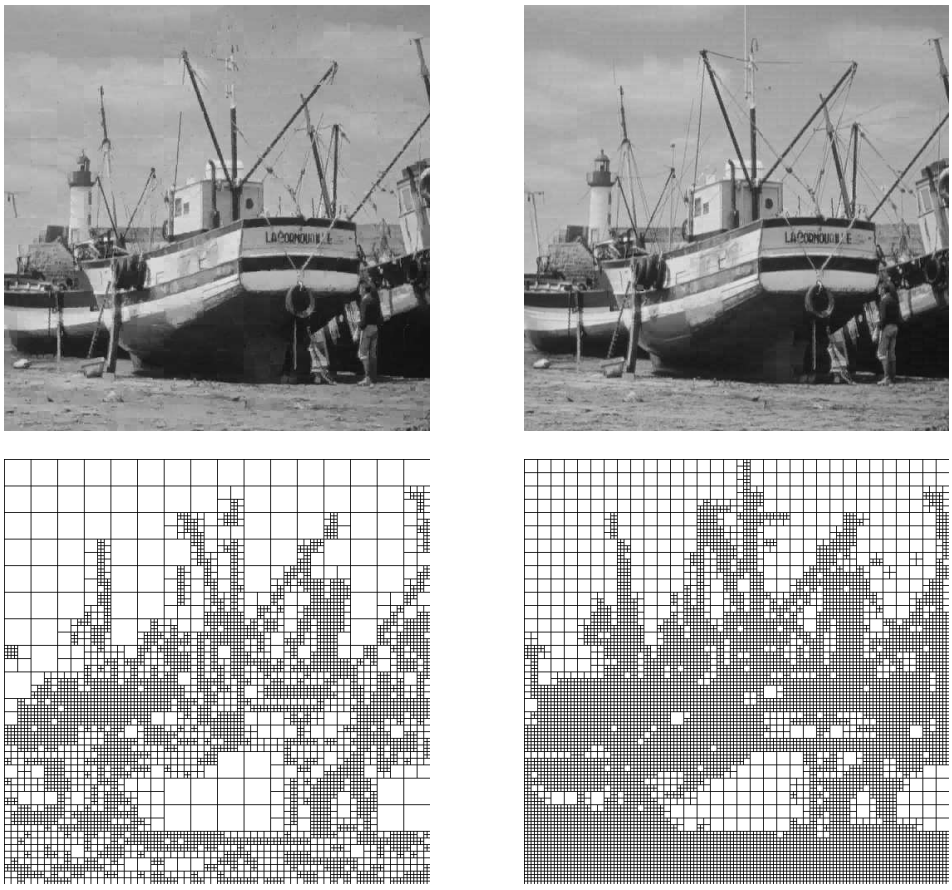


Figura 7.1: Resultados de dos compresiones con árboles cuadriculares. A la izquierda se muestra la imagen del barco con calidad media (30.11 dB) y ratio de compresión medio (13.95). A la derecha la misma imagen con mejor calidad (33.19 dB) y menor ratio de compresión (7.93). Bajo cada una de ellas se muestra su *plantilla* asociada, que refleja la estructura del árbol cuadricular utilizado.

Partición HV

En la segmentación horizontal-vertical [FIS 95, p. 119 y ss.] una imagen rectangular se divide bien horizontal o bien verticalmente para generar dos nuevos rectángulos. La subdivisión se repite de forma recursiva hasta que se alcanza un determinado criterio de tolerancia como en el caso de los árboles cuadriculares. El punto de corte se determina según la uniformidad del bloque a dividir de manera que se evita la restricción de partir la imagen siempre por determinadas posiciones fijas, además de aumentar las posibilidades de que distintos rectángulos posean estructuras similares.

La variedad de formas de los rangos implica una mayor complejidad en el diseño del codificador. Sin embargo, a pesar del incremento de espacio necesario para poder almacenar este tipo de partición, muchos experimentos demuestran que las curvas ratio-distorsión se ven considerablemente mejoradas respecto al uso de árboles cuadriculares.

Partición triangular

Para superar el inconveniente de los dos métodos anteriores al restringir la orientación de las aristas, [DAV 95] propone el uso de triángulos, que se pueden adaptar mucho mejor a las características de la imagen y reducir el efecto de bloque. Mediante el conocido algoritmo de triangulación de Delaunay se va refinando sucesivamente una partición triangular según los valores de gris de sus triángulos. En una fase posterior se reduce el número de códigos fractales obtenidos fusionando pares de triángulos si el cuadrilatero resultante es convexo y si ambos triángulos poseen una distribución de grises similar.

Partición genética

Para determinar una partición adecuada de la imagen se han propuesto también algoritmos basados en computación evolutiva. En [SAU 96b] se considera un rango como un conjunto de pequeños bloques de la imagen conectados. Cada población está formada por N_p *configuraciones*, esto es, N_p particiones cada una con su lista de códigos fractales. En cada paso de evolución se generan σ hijos que heredan las particiones de sus padres, excepto por la fusión de dos rangos vecinos aleatorios. De entre todos los descendientes se seleccionan los mejores para la población de la siguiente generación basándose en el error $E(R, D)$. Una codificación compacta de la estructura resultante no es trivial. Una posibilidad es almacenar el recorrido por el borde de cada rango indicando en cada paso qué dirección se toma (giro a la izquierda, giro a la derecha o seguir recto).

7.2. Transformación geométrica de los dominios

En la página 104 mostramos que la aplicación que transforma un dominio D_i en un rango R_i puede descomponerse en una transformación espacial y en otra que actúa sobre los valores de las intensidades del dominio. Centrándonos ahora en aquélla, cabe preguntarse de qué forma influyen los coeficientes

a_i, b_i, c_i y d_i de la ecuación 6.1 en el resultado de la codificación fractal.[1]

Lo habitual en la práctica totalidad de las implementaciones de algoritmos de compresión fractal es considerar $a_i = d_i = 1/2$ y $b_i = c_i = 0$, lo que lleva al uso de dominios con un tamaño doble al de los rangos. Otra posibilidad es utilizar dominios con tamaño triple al de los rangos mediante los valores $a_i = d_i = 1/3$.

Pero la cuestión realmente importante es si es posible utilizar dominios y rangos del mismo tamaño haciendo $a_i = d_i = 1$. En principio, ninguna de las proposiciones del tema anterior requerían tal condición y esta elección parece la más natural a primera vista. Aunque algunas evaluaciones experimentales [SAU 96a, p. 15] parecen indicar que el error es mayor en este caso, el verdadero problema es que, si no se guarda cuidado, una region puede terminar referenciándose a sí misma lo que va contra el principio de identificar diferentes objetos similares en la imagen. Al utilizar dominios con el doble de tamaño este problema queda automáticamente resuelto. Con todo, si se evitan las autorreferencias (a nivel de una o varias regiones), el uso de dominios y rangos del mismo tamaño [LU 97, p. 128] puede ser beneficioso, ya que suele ser habitual en una imagen encontrar dos objetos similares a la misma escala.

También es una práctica habitual agrandar la lista de dominios incluyendo bloques obtenidos rotando cada dominio valores múltiplos de 90 grados y haciendo esto mismo sobre su versión reflejada (simétrica). Las ocho posibles isometrías resultantes son las mostradas en la tabla 7.1. De esta forma la lista de dominios es ocho veces más grande y es de esperar que mejoren las curvas ratio-distorsión. Debe tenerse en cuenta, por otra parte, que la información sobre la isometría utilizada (3 bits) debe pasarse también al descodificador con lo que se incrementa el número de bits necesarios para cada código fractal. En [SAU 96a, p. 15] se sugiere que la complejidad que la consideración de las isometrías introduce en el algoritmo no está justificada. Entre ambos extremos, [LU 97, p. 125] propone utilizar sólo las cuatro primeras isometrías de la tabla 7.1.

La tabla 7.2 refleja el error obtenido al codificar la imagen del tucán considerando distintos números de isometrías y un árbol cuadricular de dos niveles (8×8 y 4×4). La figura 7.2 muestra cómo queda la imagen del tucán codificada únicamente sobre la transformación identidad. Nótese cómo, al menos en este caso, parece razonable el uso exclusivo de la forma identidad, ya que la enorme ganancia en términos de velocidad compensa la leve pérdida de calidad obtenida.

[1] Los coeficientes u_i y v_i simplemente trasladan el dominio a la posición del rango por lo que no existen alternativas posibles a la hora de escoger sus valores.

Índice	Isometría	Matriz
0	identidad	$\begin{pmatrix} +1/2 & 0 \\ 0 & +1/2 \end{pmatrix}$
1	simetría respecto a x	$\begin{pmatrix} -1/2 & 0 \\ 0 & +1/2 \end{pmatrix}$
2	simetría respecto a y	$\begin{pmatrix} +1/2 & 0 \\ 0 & -1/2 \end{pmatrix}$
3	rotación de π	$\begin{pmatrix} -1/2 & 0 \\ 0 & -1/2 \end{pmatrix}$
4	simetría respecto a la primera diagonal	$\begin{pmatrix} 0 & +1/2 \\ +1/2 & 0 \end{pmatrix}$
5	giro de $\pi/2$	$\begin{pmatrix} 0 & -1/2 \\ +1/2 & 0 \end{pmatrix}$
6	giro de $2\pi/3$	$\begin{pmatrix} 0 & +1/2 \\ -1/2 & 0 \end{pmatrix}$
7	simetría respecto a la segunda diagonal	$\begin{pmatrix} 0 & -1/2 \\ -1/2 & 0 \end{pmatrix}$

Cuadro 7.1: Las ocho isometrías pueden deducirse de las ecuaciones dadas en 4.2. Las transformaciones 1, 2, 4 y 7 son combinaciones de 0, 3, 5 y 6 con una inversión.

Núm. isometrías	Tiempo (seg)	Compresión	RECM	RSRM
8	955.9	13.39	3.38	37.55
4	486.9	13.2	3.44	37.39
1	142.0	14.69	3.69	36.77

Cuadro 7.2: El efecto del número de isometrías consideradas al codificar la imagen del tucán. Para la compresión se utilizó un árbol cuadricular de dos niveles sin clasificación previa de los dominios. El ratio de compresión ligeramente superior de la última fila se debe al hecho de no tener que almacenar la información sobre la isometría asociada a cada código fractal cuando se usa exclusivamente la identidad.

7.3. Postprocesamiento

Al codificar cada rango independientemente no se puede garantizar que las transiciones entre pixels adyacentes situados en los bordes de las regiones

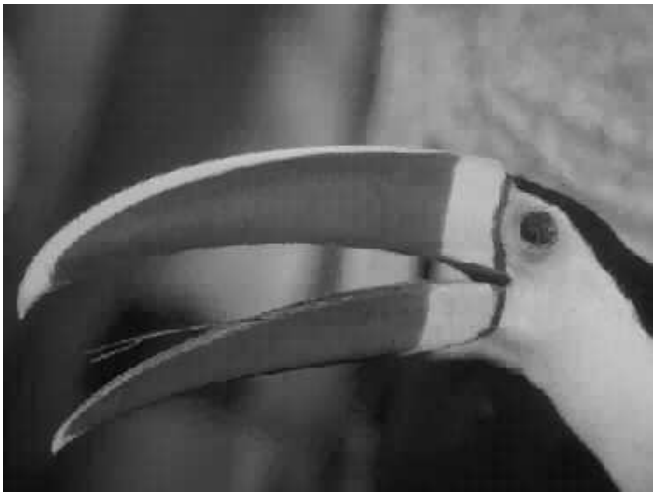


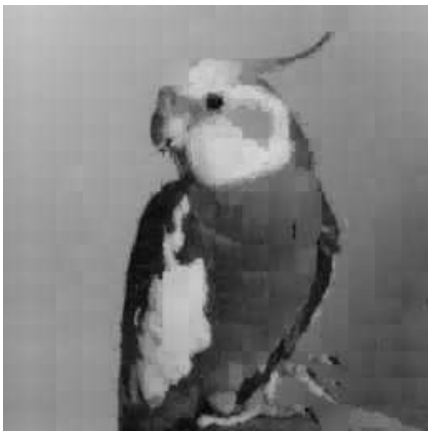
Figura 7.2: El tucán con un ratio de compresión de 14.69 sin utilizar transformaciones adicionales sobre los dominios.

sean suaves. El ojo es sensible a estas discontinuidades, incluso cuando no son muy abruptas. Una manera de reducir la aparición de este efecto de mosaico es postprocesando la imagen. Una posibilidad [FIS 95, p. 61] es modificar, una vez descodificada la imagen, los pixels situados en las fronteras de los rangos mediante un promedio ponderado de sus valores. Si los valores de gris a ambos lados de la frontera son a y b, entonces el valor de a se sutituye por

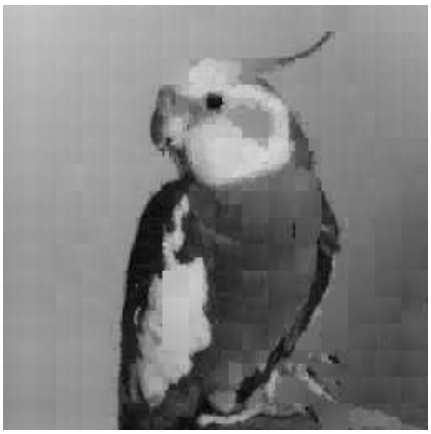
$$\frac{w_1 a + w_2 b}{w_1 + w_2}$$

Aunque son valores totalmente heurísticos, $w_2 \approx (1/3)w_1$ suele proporcionar los mejores resultados. Aun así, la mejora de la calidad es casi imperceptible, como se muestra en la figura 7.3, donde se indica el error sobre una codificación de la imagen del pájaro con y sin este tipo de postprocesamiento. Como se ve la mejora es inferior a 1/2 dB.

Para reducir el efecto de bloque se ha propuesto también [DUG 96] el uso de rangos solapados. La codificación presenta entonces cierta redundancia en las zonas de solapamiento. El descompresor promedia los valores dados por cada rango suavizando el aspecto de la imagen y reduciendo el error al disponer de varios códigos fractales para algunos pixels. Como contrapartida, el rendimiento de la compresión se reduce al tener que tratar más de una vez algunas zonas de la imagen.



(a) RECM=6.44, RSRM=31.95



(b) RECM=6.78, RSRM=31.54

Figura 7.3: Descodificación del pájaro con postproceso (arriba) y sin él (abajo). Los pesos usados en el postprocesamiento fueron $w_1 = 3$ y $w_2 = 1$. El ratio de compresión es de 23.73.

7.4. Clasificación de los dominios

Durante la codificación fractal deben explorarse un gran número de dominios para cada rango de la imagen. Esta exploración es la causa principal del elevado coste temporal del algoritmo. Uno de los caminos más utilizados para reducir este coste es el de la clasificación de los dominios. En los métodos de clasificación los dominios se agrupan independientemente antes del inicio real de la codificación en un número predefinido de clases. Para un rango determinado sólo se busca su dominio asociado en la clase que le corresponde. Existen una gran variedad de algoritmos que siguen esta idea.

A continuación mostramos cuatro por orden creciente de complejidad.

Estructura de dominios

En su trabajo original Jacquin clasificó los dominios según sus características geométricas distinguiendo sólo tres tipos de bloques [SAU 94b]:

Bloques suaves Las intensidades varían levemente a lo largo del bloque.

Bloques con aristas Presentan un cambio abrupto de intensidad, por ejemplo, debido al borde de un objeto de la imagen.

Bloques intermedios Contienen variaciones de intensidad mayores que los bloques suaves, pero sin presentar un gradiente tan pronunciado como en el caso de los bloques con aristas.

Los bloques intermedios son los que cobijan normalmente las texturas existentes en la imagen. Los rangos que se clasifican como bloques suaves pueden aproximarse correctamente a través del desplazamiento o y no es necesario buscar un dominio para ellos (en este caso, $s = 0$). Por lo tanto, en este esquema sólo existen dos clases, en una de las cuales debe buscarse para cada rango que no sea un bloque suave.

Vectores de características

Otra sencilla forma de clasificar los dominios a partir de su estructura, muy ligada a la anterior, se basa en el establecimiento de vectores de características [COL 96]. Consideremos un dominio D cuadrado formado por las intensidades $d_1, d_2, \ldots, d_n$ y definamos su nivel de gris medio como

$$\bar{D} = \frac{1}{n} \sum_{i=1}^{n} d_i$$

Para un rango cuadrado R el nivel medio de gris $\bar{R}$ puede definirse de forma análoga. Consideremos entonces la media anterior de cada bloque y la media de sus cuatro cuadrantes no solapados (izquierda superior, derecha superior, izquierda inferior y derecha inferior) que denotaremos por A_i con $i = 0, 1, 2, 3$. Con estos valores podemos definir un vector de características para dominios $\omega = \{\omega_i,\ i = 0, \ldots, 3\}$ como sigue:

$$\omega_i = \begin{cases} 1 & : \text{ si } A_i > \bar{D} \\ 0 & : \text{ si } A_i \leq \bar{D} \end{cases}$$

y análogamente en el caso de rangos. Lo anterior permite 16 vectores (realmente 15) de estructura distintos. Durante la codificación sólo se realizará la comparación con aquellos dominios que poseen la misma estructura que el rango actual.

Reordenación de cuadrantes

Una clasificación más elaborada [FIS 95, p. 57] divide, al igual que la anterior, los rangos y dominios cuadrados en cuatro cuadrantes. Para cada cuadrante se calcula la intensidad media de los pixels A_i y las varianzas V_i $(i = 0, \ldots, 3)$. Es fácil ver que siempre puede reorientarse (mediante giros y reflejos) cualquier rango o dominio de manera que las intensidades promediadas estén ordenadas de una de las tres formas siguientes:

$$\begin{aligned}
&\text{Clase principal 1: } A_0 \geq A_1 \geq A_2 \geq A_3 \\
&\text{Clase principal 2: } A_0 \geq A_1 \geq A_3 \geq A_2 \\
&\text{Clase principal 3: } A_0 \geq A_3 \geq A_1 \geq A_2
\end{aligned}$$

Una vez reorientado el bloque, existen 24 (factorial de 4) formas diferentes de ordenar las varianzas, que definen 24 subclases para cada clase principal. De esta forma podemos considerar bien 3 clases distintas, bien 72.[2]

Agrupamiento de Heckbert

Aunque la clasificación por reordenación de cuadrantes considera la división de los dominios en un número relativamente grande de clases, éste todavía puede impedir la codificación en entornos en los que la velocidad sea el factor primordial. En este apartado consideraremos un método que permite aumentar casi indefinidamente el número de clases consideradas.

El algoritmo de agrupamiento (*clustering*) de Heckbert [LU 97, pp. 179 y ss.] divide un conjunto de vectores en M grupos (*clusters*) realizando sucesivas divisiones mediante hiperplanos perpendiculares a alguna dirección. Describiéndolo por encima, el algoritmo se basa en elegir el grupo actual con mayor cantidad de vectores, encontrar la dirección en la que el grupo tiene su mayor extensión y dividirlo mediante un hiperplano perpendicular a esa dirección en el valor que hace los tamaños de los dos grupos generados lo

[2] Si el valor del escalado s es negativo, deben reconsiderarse las ordenaciones anteriores. Por lo tanto, cada dominio se clasifica con dos orientaciones, una orientación para valores de s positivos y otra para valores negativos.

más parecidos posible. Este proceso se repite hasta que se obtiene el número de grupos deseados.

Consideremos ahora el conjunto de dominios de la forma $D = d_1, d_2, \ldots, d_n$ de una imagen como un conjunto finito de vectores $\mathbf{x} \in \Omega \in \mathbf{R}^n$ y sea M el número deseado de grupos. El algoritmo de Heckbert define la clasificación a través de una secuencia de $M-1$ divisiones. Cada división se define mediante cuatro parámetros (k, k', i, v), donde el k-ésimo grupo se divide por el i-ésimo hiperplano en el valor v en dos grupos, el grupo k y el grupo k' para $x_i < v$ y $x_i \geq v$, respectivamente. En resumen, el algoritmo divide del k-ésimo grupo al k'-ésimo en la variable i-ésima sobre el valor v. El criterio para determinar las particiones es el siguiente:

Inicialización Inicialmente tenemos un único grupo:

1. Sea $C_0^0 = \Omega$.

Primera división En la primera división $s = 1$:

1. Elegir el grupo $k_1 = 0$.
2. Encontrar la dirección i_1 con mayor extensión, esto es,

$$\max_{\mathbf{x},\mathbf{y} \in C_0^0} \{x_{i_1} - y_{i_1}\} = \max_{0 \leq i < n} \left\{ \max_{\mathbf{x},\mathbf{y} \in C_0^0} \{x_i - y_i\} \right\}$$

3. Encontrar el valor de división por la mitad v_1 para que

$$\left| \text{card}\{\mathbf{x} \in C_0^0 : x_{i_1} < v_1\} - \frac{\text{card}\, C_0^0}{2} \right| = \inf_{v \in \mathbf{R}} \left| \text{card}\{\mathbf{x} \in C_0^0 : x_{i_1} < v\} - \frac{\text{card}\, C_0^0}{2} \right|$$

4. Dividir el grupo en dos:

$$C_0^1 = \{\mathbf{x} \in C_0^0 : x_{i_1} < v_1\} \quad \text{y} \quad C_1^1 = \{\mathbf{x} \in C_0^0 : x_{i_1} \geq v_1\}$$

Resto de divisiones En las siguientes divisiones $s = 2, 3, \ldots, n-1$:

1. Elegir el grupo más grande k_s (si hay más de uno, escoger cualquiera de ellos), esto es,

$$\text{card}\, C_{k_s}^{s-1} = \max_{0 \leq k < s} \text{card}\, C_k^{s-1}$$

2. Encontrar la dirección i_s con la mayor extensión, esto es,

$$\max_{\mathbf{x},\mathbf{y}\in C_{k_s}^{s-1}}\{x_{i_s}-y_{i_s}\}=\max_{0\le i<n}\left\{\max_{\mathbf{x},\mathbf{y}\in C_{k_s}^{s-1}}\{x_i-y_i\}\right\}$$

3. Encontrar el valor de división por la mitad v_s tal que

$$\left|\mathrm{card}\{\mathbf{x}\in C_{k_s}^{s-1}:x_{i_s}<v_s\}-\frac{\mathrm{card}\,C_{k_s}^{s-1}}{2}\right|=\inf_{v\in\mathbf{R}}\left|\mathrm{card}\{\mathbf{x}\in C_{k_s}^{s-1}:x_{i_s}<v\}-\frac{\mathrm{card}\,C_{k_s}^{s-1}}{2}\right|$$

4. Dividir el grupo $C_{k_s}^{s-1}$ en dos:

$$C_{k_s}^{s}=\{\mathbf{x}\in C_{k_s}^{s-1}:x_{i_s}<v_s\}\quad\text{y}\quad C_{s}^{s}=\{\mathbf{x}\in C_{k_s}^{s-1}:x_{i_s}\ge v_s\}$$

y establecer los otros grupos a los mismos que antes:

$$C_k^s=C_k^{s-1},\quad\text{para } 0\le k<s\quad\text{y}\quad k\neq k_s$$

A través de la sucesión de hiperplanos puede conocerse el grupo asociado a cualquier vector $\mathbf{x}$ del conjunto Ω. En concreto, puede obtenerse el grupo de dominios asociado a cada rango de la imagen y realizar la búsqueda exclusivamente sobre ellos.

En la tabla 7.3 puede observarse cómo a partir de cierto punto el incremento del número de grupos no lleva asociado un incremento substancial en el tiempo de codificación. Esto es debido a que la clasificación preliminar de los dominios consume la práctica totalidad del tiempo. En particular, este tiempo representa más del 90 % del total a partir de un número de grupos superior a 256.

7.5. Compresión sustituyente

Una vez empaquetados los bits que definen cada código fractal y almacenados de forma contigua en un fichero puede intentarse su compresión mediante técnicas como las comentadas en la página 90 (entropía, codificación aritmética y otros). Existe, sin embargo, un problema y es que los programas existentes suelen operar a nivel de bytes con lo que pueden no apreciar una regularidad existente en los datos si el tamaño de cada código fractal no es múltiplo de 8 bits. Parece necesario, por tanto, adaptar los algoritmos para que ajusten su dinámica a la de los códigos fractales.

Núm. Grupos	Tiempo (seg)	RECM	RSRM
1	1393.0	3.14	38.18
2	771.6	3.37	37.58
4	419.0	3.64	36.9
8	238.7	3.89	36.33
16	147.9	4.23	35.59
32	101.6	4.54	34.99
64	79.4	4.76	34.59
128	67.9	5.07	34.02
256	62.4	5.30	33.64
512	59.6	5.64	33.1
1024	58.6	5.85	32.78
2048	58.5	6.25	32.21

Cuadro 7.3: Efecto del número de grupos del algoritmo de Heckbert sobre la imagen de las flores. Los tiempos corresponden a un sistema Pentium II 233 MHz con Linux. Se utilizaron rangos de 4×4 y cuatro isometrías. El ratio de compresión resultante fue de 4.7.

En concreto, distintos ficheros de códigos fractales comprimidos mediante el algoritmo LZ77 de Lempel-Ziv (programa `gzip`) no lograban reducir el tamaño de los ficheros más allá del 6 %.

7.6. Independencia de la resolución

Una característica particular de las imágenes comprimidas fractalmente, y que distingue especialmente a esta técnica de las otras técnicas de compresión, es el denominado *zoom* o ampliación fractal. Las imágenes codificadas mediante la transformada fractal se describen exclusivamente como el punto fijo de un determinado operador y no presentan referencias a ninguna escala particular de la imagen ni a ningún tamaño determinado en pixels. Por ello, un código fractal puede descodificarse a cualquier resolución, generando detalles en todas las escalas. Este es uno de los aspectos que, pese a algunas críticas, justifica considerar este método como fractal.

Evidentemente los detalles generados al descodificar una imagen a escalas mayores son artificiales y no tienen mucho que ver con los existentes en el mundo real cuando fue tomada, pero debido al carácter autorreferencial del modelo, estos detalles son relativamente consistentes con la apariencia global de los objetos y presentan una estructura más natural que las imágenes ampliadas simplemente por duplicación de pixels o interpolación.

Figura 7.4: A la izquierda parte de la imagen del barco descodificada a cuatro veces su tamaño y a la derecha la imagen original agrandada 4 veces mediante la técnica de duplicación.

Si tenemos un código fractal asociado a un rango R_i de tamaño $n \times n$ con origen en las coordenadas (x_i, y_i) y perteneciente a una imagen de tamaño $\bar{x} \times \bar{y}$, el código fractal correspondiente para generar una imagen de tamaño $2\bar{x} \times 2\bar{y}$ con el rango R_i de tamaño $2n \times 2n$ se obtiene simplemente sustituyendo el origen del rango por $(2x_i, 2y_i)$.

La figura 7.4 muestra las diferencias entre la forma clásica de ampliación y el *zoom* fractal con el faro que aparece a lo lejos en la imagen del barco.

La independencia de la resolución se ha utilizado algunas veces como

justificación de ciertos resultados muy cercanos a la *publicidad engañosa*. Supongamos que codificamos una determinada imagen, obtenemos un ratio de compresión de 12 y descodificamos la imagen comprimida con un factor de ampliación de 2, es decir, a un tamaño doble del de la original. Algunos autores justifican el siguiente razonamiento: la imagen descodificada tiene 4 veces más pixels que la original, por lo que su ratio de compresión es de $12 \cdot 4 = 48$. El lector ya está prevenido.

7.7. Mejora de la resolución

Muy ligado a la técnica de la sección anterior se encuentra la mejora de resolución [LU 97, p. 218]. Mediante esta técnica una imagen de poca resolución puede codificarse fractalmente y descodificarse a continuación a mayor resolución generando una versión mejorada.

7.8. Aceleración de la compresión

En esta sección veremos algunas ideas para acelerar la compresión fractal mediante técnicas que no utilizan clasificación de dominios.

Búsqueda local

Ocurre a menudo que el dominio con que se empareja un rango se encuentra relativamente cerca de éste. Esta situación es comprensible, ya que la distribución de los objetos por una imagen suele determinar que regiones similares se encuentren cercanas al pertenecer ambas al mismo objeto. Esta hipótesis, sin embargo, no tiene que ocurrir (y de hecho no ocurre) en todas las imágenes, ni siquiera en todos los rangos de una determinada imagen. Sin embargo, esto da la idea de utilizar un algoritmo de compresión que favorezca los códigos fractales con dominios locales al rango correspondiente.

El algoritmo de compresión que incorpore esta idea [LU 97, pp. 117 – 122] puede realizar una búsqueda local sobre un reducido número de dominios cercanos al rango actual, además de sobre la lista completa de domimios. Las ventajas de este enfoque es que permite utilizar muchos menos bits para referenciar al dominio óptimo si éste se encontró en la búsqueda local.[3] Es

[3]Puede añadirse a cada código fractal un bit que indique si la información sobre el dominio es la de un dominio local o global. Si una gran cantidad de referencias a lo largo de la imagen son locales, se reducirá el ratio de compresión sin ninguna pérdida de calidad.

más, llegado el caso puede desestimarse la realización de la búsqueda global si el mejor dominio encontrado localmente proporciona un error aceptable. Es posible también cierta ganancia en velocidad, ya que [LU 97] sugiere el uso exclusivo de la forma identidad de la tabla 7.1 para la búsqueda local.

Búsqueda del vecino más cercano

El tiempo de la codificación en la compresión fractal de imágenes es considerablemente elevado. Si el número de dominios manejados es N, entonces el tiempo invertido en la búsqueda para cada rango es lineal con N, es decir, $O(N)$. Las técnicas de clasificación de dominios vistas en 7.4 pretenden reducir este tiempo mediante la división en grupos de los dominios según determinadas características. De esta forma en cada búsqueda sólo se evalúan los dominios de un determinado grupo. Sin embargo, este enfoque sólo reduce el factor de proporcionalidad de la complejidad, que sigue siendo $O(N)$.

En [SAU 95] se demuestra que el problema de encontrar pares óptimos de rangos y dominios es equivalente al de búsqueda del vecino más cercano en un espacio euclídeo adecuado formado por vectores de características de los dominios y de los rangos. El problema de encontrar el vecino más cercano puede ser resuelto con las estructuras de datos y los algoritmos adecuados en $O(\log N)$, con lo que se obtendrá una clara ventaja sobre la búsqueda lineal, mayor cuanto mayor sea el número de dominios N.

Consideremos cada rango R_i como un vector $R_i \in \mathbf{R}^n$ y, análogamente, cada dominio D_i (convenientemente submuestreado) como un vector del mismo espacio, $D_i \in \mathbf{R}^n$. Sea $\mathbf{e} = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \mathbf{R}^n$ un vector unitario con el mismo número de componentes y definamos

$$\phi(\mathbf{x}) = \frac{\mathbf{x} - \langle \mathbf{x}, \mathbf{e} \rangle \mathbf{e}}{\|\mathbf{x} - \langle \mathbf{x}, \mathbf{e} \rangle \mathbf{e}\|}$$

donde $\mathbf{x} \in \mathbf{R}^n$ y $\langle \cdot, \cdot \rangle$ denota el producto escalar en $\mathbf{R}^n$ con lo que $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

Ahora puede demostrarse que la minimización del error cuadrático medio $E(R, D_i)$, con $i = 1, \ldots, N$, es equivalente a la búsqueda del vecino más cercano de $\phi(R) \in \mathbf{R}^n$ en el conjunto de los $2N$ vectores $\pm\phi(D_i) \in \mathbf{R}^n$. Como ya se ha dicho, este problema puede resolverse con coste $O(\log N)$.

A la hora de realizar una implementación práctica del algoritmo deben tenerse en cuenta algunas cosas:

- No pueden obviarse los requerimientos espaciales de manejar los vecto-

res de características al trabajar con listas de miles de dominios. Para limitar este espacio pueden submuestrearse los dominios y los rangos a una dimensión moderada, lo que implica que el método anterior sólo podrá dar resultados aproximados, pero no exactos.

Otra alternativa para reducir la complejidad espacial es considerar en el algoritmo de búsqueda del vecino más cercano índices de rangos y dominios en lugar de los vectores en sí. Con este enfoque las componentes de intensidad de cada bloque se obtienen accediendo directamente a la imagen mediante una función que devuelva el bloque asociado a un determinado índice.

- Para un determinado rango no todos los dominios de la lista de dominios son *admisibles*, en el sentido de que el factor de escalado s obtenido mediante el método de los mínimos cuadrados puede no garantizar la contractividad de la transformación (en general, si $|s| \geq 1$). Para considerar esta posibilidad, no se realiza la búsqueda del vecino más cercano, sino de los k vecinos más cercanos,[4] quedándonos con aquél que nos asegure la convergencia del modelo.

Distancia entre dominios

En la página 107 se señaló que una forma de reducir el número de regiones de la lista de dominios es considerando que los orígenes de dominios consecutivos (en el orden establecido con un barrido por líneas) no están situados en pixels adyacentes, sino que existe una separación $l > 1$ entre ellos. De esta manera el número de comparaciones se reduce por un factor l y con ello el tiempo necesario para la codificación, aunque es de esperar que a costa de una pérdida de la calidad al existir el riesgo de *saltarnos* alguna región importante.

La tabla 7.4 cuantifica estas pérdidas sobre la imagen del perro pastor en la que valores pequeños de l no deberían alterar excesivamente la distorsión de la imagen codificada. La codificación usa exclusivamente el esquema básico del capítulo anterior. La figura 7.5 muestra la imagen codificada con dominios con orígenes separados 128 pixels. Debe tenerse cuidado con generalizar estos resultados, ya que debido al gran parecido de todas las regiones de esta imagen era de esperar poca influencia del valor de l.[5] En general, el valor de l es mucho más importante en la mayor parte de las imágenes y no debe mantenerse muy por encima de 4.

[4] El problema de los k vecinos más cercanos también es resoluble en tiempo logarítmico.

[5] Aun así los resultados son espectaculares: el tiempo de codificación logra reducirse a la centésima parte con una reducción de 5 dB.

Separación	Tiempo (seg)	RECM	RSRM
1	78.9	3.32	37.69
2	38.8	3.47	37.31
3	25.8	3.62	36.97
4	19.9	3.69	36.79
5	15.5	3.9	36.32
6	13.0	3.87	36.39
7	11.2	3.86	36.39
8	9.9	3.92	36.27
16	5.0	4.2	35.67
32	2.7	4.55	34.98
64	1.4	5.12	33.95
128	0.7	5.87	32.75

Cuadro 7.4: Efecto de la separación entre los origenes de los dominios con la imagen del perro pastor. Los tiempos corresponden a un sistema Pentium II 233 MHz con Linux. Se utilizaron rangos de 4×4 y el ratio de compresión resultante fue de 5.81.

Figura 7.5: Considerando sólo 256 dominios la imagen del perro pastor se codifica tan bien como se muestra aquí en tan sólo 0.7 segundos. El ratio de compresión es 5.87 y la RSRM 32.75.

7.9. Aceleración de la descompresión

Aunque la diferencia entre los tiempos necesarios para la codificación y la descodificación fractal es abismal (en beneficio de esta última), en apli-

caciones en las que la velocidad de descodificación es vital (por ejemplo, vídeo en tiempo real) se hace necesario encontrar métodos más rápidos que el clásico. Se han propuesto distintos métodos [SAU 96a, p. 45] que incluso evitan en parte la iteración del esquema tradicional.

Encadenamiento de pixels

Una de las técnicas de descodificación rápida [LU 97, p. 207] es la conocida como encadenamiento de pixels, aplicable cuando el ajuste del tamaño de los dominios al de los rangos se realiza mediante submuestreo y no mediante el promediado que hemos considerado hasta ahora.

Aunque no vamos a entrar en demasiados detalles, el método parte del hecho de que para cada pixel de la imagen existe un único pixel, llamado *pixel de dominio* asociado, que llega hasta él mediante la transformación fractal correspondiente. Si conociésemos el valor de este pixel de dominio, el valor correspondiente del pixel destino podría derivarse de un solo paso. Considerando este pixel de dominio como un nuevo pixel destino, podemos encontrar para él otro nuevo pixel de dominio. Siguiendo este procedimiento, podemos construir una cadena de pixels. Esta cadena se detiene cuando choca con un punto que ya está en la cadena o con un pixel de valor conocido que ya fue calculado en una cadena anterior.

Si el pixel es conocido, podemos generar los valores de todos los pixels de la cadena siguiéndola hacia atrás y aplicando los ajustes de brillo y contraste definidos por cada código fractal.

En el otro caso, el valor del último pixel de la cadena se obtiene saltando hacia atrás un número suficiente de pixels, estableciendo el valor del pixel final al gris 128 y deshaciendo el camino hasta dar valor al último pixel de la cadena. Desde este momento podemos dar valor al resto de pixels de la cadena como antes. Si $s = 3/4$, entonces basta saltar 20 pixels hacia atrás para que el error cometido sobre el último pixel de la cadena sea despreciable.

7.10. Enfoques híbridos

La unión hace la fuerza y son los enfoques híbridos los que en la actualidad acaparan las líneas más prometedoras de investigación [LU 97, pp. 41 – 45], especialmente los que aunan de diversas formas la teoría fractal con los wavelets.

Los aspectos analizados en este capítulo no cubren en absoluto todas las

alternativas al enfoque básico de codificación fractal y se han presentado como pequeña muestra de la increible evolución experimentada por estos desarrollos en apenas una década. Los próximos años verán, sin duda, la aparición de innovadoras técnicas fractales que mejorarán nuestra visión del mundo en todas sus escalas.

Apéndice A

Medida de conjuntos

A menudo resulta necesario determinar el tamaño de un fractal para poder establecer su similitud con algún otro. Existen distintos números asociados con los fractales que nos permiten compararlos. Estos números, denominados normalmente *dimensiones fractales*, son un intento de cuantificar nuestra idea subjetiva acerca de la densidad con que un fractal ocupa el espacio al que pertenece.

La más importante de tales dimensiones es la *dimensión de Hausdorff*, aunque su dificultad de instrumentación en la práctica lleva a considerar desarrollos alternativos. Aun así, el camino hasta la obtención de una medida satisfactoria pasó por el rechazo de ciertas métricas que se mostraban impotentes con los conjuntos fractales a pesar de su consolidada eficacia con la mayor parte de los conjuntos clásicos. De todo esto se hablará en este apéndice. Los conceptos a continuación presentados pueden accederse también a través de [GUZ 93] y [BAR 93a].

A.1. La medida de Lebesgue

La medida de conjuntos de puntos en la recta, en el plano o en el espacio tiene tras de sí una larga historia. Ya Euclides consideró el cálculo de áreas de rectángulos y triángulos. Para otras regiones limitadas por líneas rectas se procedía a su *triangulación*. Si la región está limitada por curvas, la triangulación no es posible y se hace necesario recurrir entonces al método de exahusción propuesto por Arquímedes y que es, básicamente, una versión del concepto actual de paso al límite en el que se van considerando sucesivos triángulos de área cada vez menor para rellenar adecuadamente la región.

Costó dos mil años abandonar la tradición griega de la triangulación que requería grandes dosis de ingenio y habilidad para cada caso particular. En la década de 1890, Peano y Jordan consideraron un método basado en enlosados que, aun siendo mucho mejor que el de triangulación, producía resultados erróneos con cierto tipo de conjuntos.

No estaba claro cómo había que proceder para encontrar una definición satisfactoria de área hasta que Emile Borel sugirió la idea de usar enlosados formados por infinitos rectángulos.

La idea de Borel fue desarrollada por Henri Lebesgue llegándose así a la definición de lo que hoy día se conoce como *medida de Lebesgue* y que generaliza la idea de longitud, área y volumen en la recta, plano y espacio, respectivamente. Exponemos la definición en $\mathbf{R}^n$, aunque puede ser más sencillo pensar en lo siguiente que $n = 2$.

Consideremos la familia $\Re$ de los hipercubos (rectángulos) R de $\mathbf{R}^n$ de la forma

$$[a_1, b_1) \times \cdots \times [a_n, b_n) = R$$

Definimos el volumen $V(R)$ de un rectángulo R como

$$V(R) = (b_1 - a_1) \ldots (b_n - a_n)$$

Dado ahora $E \subset \mathbf{R}^n$ consideraremos todos los recubrimientos (enlosados) $\{R_i\}$ de E, es decir, tales que $E = \bigcup_{i=1}^{\infty} R_i$ y definimos

$$L^n(E) = \text{ínf} \left\{ \sum_{i=1}^{\infty} V(R_i) : \{R_i\} \text{ es un recubrimiento de } E \right\}$$

Pueden demostrarse las siguientes proposiciones:

a) $L^n(\emptyset) = 0$

b) $L^n(A) \leq L^n(B)$ si $A \subset B$

c) para cualquier familia numerable $\{E_k\}$, $k = 1, 2, \ldots$, de subconjuntos de $\mathbf{R}^n$ se verifica
$$L^n \left(\bigcup_{k=1}^{\infty} E_k \right) \leq \sum_{k=1}^{\infty} L^n(E_k)$$
propiedad denominada *subaditividad*.

A L^n se le denomina *medida exterior de Lebesgue*. Sin embargo, no es cierto siempre que dado un E tal que $E \subset \mathbf{R}^n$ se verifique

$$L^n(E - B) + L^n(E \cap B) = L^n(E) \quad \text{para todo } B \subset \mathbf{R}^n$$

es decir, que la medida de un conjunto sea siempre igual a la suma de las medidas de dos partes complementarias. Cuando dicha igualdad es cierta para todo $B \subset \mathbf{R}^n$ diremos que el conjunto E es *medible Lebesgue*, hecho que afortunadamente ocurría para los conjuntos más usuales hasta que los fractales comenzaron a ser conjuntos usuales.

Si E es medible Lebesgue, $L^n(E)$ es una medida del conjunto E que se llama *medida de Lebesgue*. La clase de los conjuntos medibles Lebesgue abarca a una clase de conjuntos muy amplia.

A.2. Problema del área

Si bien el problema de medir el área de una región plana queda razonablemente resuelto con la definición de la medida de Lebesgue, el problema de medir el área de una superficie no contenida en un plano presenta serias dificultades, debido a la rigidez que impone el tener que recubrir bien con rectángulos, bien con cubos. Si la superficie es suave (diferenciable), el problema lo resolvía la geometría diferencial que Gauss había iniciado en el primer tercio del siglo XIX, pero en el caso de una superficie menos regular no se disponía de una opción alternativa.

Por si fuera poco, la insuficiencia de la medida de Lebesgue se hizo aún más evidente a comienzos del siglo XX con la aparición de los primeros conjuntos fractales.

A.3. Dimensión

Hay varios conceptos matemáticos diferentes que responden al nombre de dimensión de un conjunto geométrico. Uno de ellos, el de *dimensión topológica*, hace alusión a la forma de ocupar el espacio que tiene el conjunto. Así, tanto a una curva diferenciable, como a la curva de Koch, o a la de Peano, se les asigna dimensión topológica igual a uno, y a un punto, a los puntos racionales de la recta real y al conjunto de Cantor se les asigna dimensión cero.

Pero este tipo de dimensiones plantean un serio problema pues resultan ser *poco finas* al otorgar la misma dimensión a un único punto que a un conjunto no numerable de puntos como es el conjunto de Cantor. La medida de Lebesgue $L^n(E)$ proporciona análogas consecuencias al considerar siempre dimensiones enteras $n = 1, 2, 3 \ldots$ Este planteamiento llevó a tomar en consideración la introducción de categorías intermedias o dimensiones

fraccionarias.

A.4. Dimensión de homotecia

Una manera de asignar una dimensión fraccionaria a un conjunto parte de una propiedad de la dimensión que sirve a Platón en su diálogo *Menón* para demostrar la doctrina de la reminiscencia a partir de la creencia mítica en la preexistencia y transmigración del alma. Sócrates mediante el método mayéutico consigue que un esclavo iletrado de Menón *recupere* de sí mismo el conocimiento acerca de cómo cuando el lado de un cuadrado se duplica se obtiene una figura equivalente a cuatro, y no a dos, de los cuadrados iniciales. Si en lugar de un cuadrado se tratara de un cubo, el factor de multiplicación sería de ocho y no de cuatro.

Nos encontramos, por tanto, con un rango distintivo de cada dimensión. Precisándolo más, supongamos que una figura de dimensión entera d puede ser descompuesta en n copias a escala r de sí misma (piénsese en el ejemplo del cuadrado de Socrates donde $d = 2$, $r = 1/2$ y $n = 4$). Entonces es fácil ver que $n = (1/r)^d$ o tomando logaritmos que

$$d = \frac{\log n}{\log(1/r)} = \frac{\log n}{-\log r}$$

Si tomamos esta fórmula como definición del valor de la dimensión de cualquier figura que pueda ser descompuesta en copias a escala de sí misma, obtenemos una manera cómoda de asignar una dimensión a algunos conjuntos fractales clásicos. El conjunto de Cantor E, por ejemplo, puede descomponerse en dos copias de sí mismo a escala $1/3$ o en cuatro copias de sí mismo a escala $1/9$ lo que nos da

$$\dim(E) = \frac{\log 2}{\log 3} = \frac{\log 4}{\log 9} = 0{,}63092975\ldots$$

Este método es de muy fácil aplicación, pero exige que el conjunto a medir pueda ser descompuesto en copias de sí mismo a escala, lo cual es una propiedad muy específica de ciertos conjuntos. Sin embargo, la idea puede aprovecharse para calcular la dimensión de cualquier tipo de conjunto.

A.5. Medida de Haussdorf

Dado un conjunto $E \subset \mathbf{R}^n$ y un número real positivo δ, diremos que una familia $\{A_i\}$, $i = 1, 2, \ldots$, de subconjuntos de $\mathbf{R}^n$ es un recubrimiento-δ de

E, si la unión de tales conjuntos contiene a E, es decir,

$$E = \bigcup_{i=1}^{\infty} A_i$$

y el diámetro de cada uno de los miembros del recubrimiento es menor o igual que δ

$$|A_i| \leq \delta \quad \forall\, i = 1, 2, \ldots$$

Dado un conjunto $E \subset \mathbf{R}^n$ y un número real $s > 0$, definimos

$$H^s_\delta(E) = \text{ínf} \left\{ \sum_{i=1}^{\infty} |A_i|^s : \{A_i\} \text{ es recubrimiento-}\delta \text{ de } E \right\}$$

número[1] que mide el *tamaño-s* del conjunto E, es decir, ignorando las irregularidades de E que tienen tamaño menor que δ.

Si hacemos tender δ a cero, iremos apreciando irregularidades de tamaño cada vez menor. Además, si $\delta \to 0$, $H^s_\delta(E)$ aumenta, pues es un ínfimo tomado cada vez sobre una clase más restringida de recubrimientos y, por tanto, existe el límite

$$H^s(E) = \lim_{\delta \to 0} H^s_\delta(E)$$

que puede ser finito o infinito. Al número $H^s(E)$ se le conoce como *medida s-dimensional de Hausdorff*.

Puede demostrarse que la definición es equivalente si se supone que los recubrimientos están formados por abiertos, por cerrados o por convexos, debido a que tales restricciones no alteran las sumas de los diámetros de los conjuntos del mismo.

Puede también demostrarse que $H^s(E)$ tiene las propiedades que se exigen para ser una medida y que, en general, si $s = n$ es entero y E es medible Lebesgue, entonces $H^n(E)$ es proporcional a la medida de Lebesgue n-dimensional $L^n(E)$.

A.6. Dimensión de Hausdorff

La medida s–dimensional de Hausdorff como función de s tiene un comportamiento especial. Su rango está formado por uno, dos o tres valores. Estos posibles valores son cero, un número finito e infinito.

[1] Para una definición precisa de los conceptos de ínfimo y supremo puede consultarse la página 38.

Teorema A.1 *Sea n un número entero positivo y sea E un subconjunto acotado de $\mathbf{R}^n$. Sea $H^s(E)$ la medida s–dimensional de Hausdorff tal como se definió más arriba con $0 \leq s < \infty$. Entonces existe un único número real $0 \leq D_H \leq n$ tal que*

$$H^s(E) = \begin{cases} \infty & si\, s < D_H \\ 0 & si\, s > D_H \end{cases}$$

El único número real D_H que cumple el teorema anterior se conoce como la *dimensión de Hausdorff* del conjunto E y se escribe también como $D_H(E)$. Evidentemente, los conjuntos clásicos conservan su dimensión clásica bajo D_H, pero los conjuntos fractales están ahora mucho mejor caracterizados y podemos intentar dar una definición precisa de ellos.

Es en exceso simplista afirmar que un fractal es aquel conjunto con dimensión fraccionaria. Algunos fractales tienen dimensión entera, por lo que la afirmación anterior es incluso errónea. Un fractal sería cualquier conjunto cuya dimensión de Hausdorff sea mayor estrictamente que su dimensión topológica. Con ello incluso la curva de Hilbert se consideraría fractal, ya que su dimensión de Hausdorff es 2 y su dimensión topológica es 1.

A.7. Dimensión fractal

Aunque el concepto de dimensión de Hausdorff es esencial en la geometría fractal, su definición es relativamente compleja y en la práctica se utilizan otras definiciones de dimensión que resultan, además, de gran valor a la hora de determinar empíricamente la dimensión de series de datos obtenidas del mundo real, y que suelen coincidir con la dimensión de Hausdorff en los casos más interesantes. La más extendida dentro de esta categoría es la denominada *dimensión fractal*. En este apartado restringiremos nuestro estudio a la dimensión de conjuntos compactos.

Definición A.1 *Sea $A \in \mathcal{H}(\mathbf{R}^n)$ un conjunto compacto y no vacío de $\mathbf{R}^n$. Sea $\mathcal{N}(A, \epsilon)$ el menor número de bolas cerradas de diámetro $\epsilon > 0$ necesarias para cubrir A. Si existe*

$$D = \lim_{\epsilon \to 0} \left\{ \frac{\log \mathcal{N}(A, \epsilon)}{\log(1/\epsilon)} \right\}$$

entonces D se denomina dimensión fractal de A. Se escribirá también $D = D(A)$ para indicar que A tiene dimensión fractal D.

Ejemplo Consideremos la curva de Koch discutida en la página 8. Se necesita una bola de diametro 1 para cubrir todo el conjunto, 4 bolas de diámetro 1/3, 16 bolas de diámetro 1/9, 64 bolas de diámetro 1/27... En general, son necesarias 4^n bolas de diámetro $(1/3)^n$ para cubrir la curva de Koch. Su dimensión fractal será, por tanto,

$$D = \lim_{n\to\infty} \frac{\log 4^n}{\log 3^n} = \frac{\log 4}{\log 3} = 1{,}261859507\ldots$$

lo que indica que la curva de Koch está más cerca de ser una curva que un área (nótese que para que el diámetro tienda a cero, n ha de tender a infinito).

Ya hemos dicho que en muchas ocasiones la dimensión fractal y la dimensión de Hausdorff coinciden. Más concretamente puede demostrarse el siguiente teorema.

Teorema A.2 *Sea n un entero positivo y sea A un subconjunto de $\mathbf{R}^n$. Si $D(A)$ denota la dimensión fractal de A y $D_H(A)$ la dimensión de Hausdorff de A, entonces*

$$0 \leq D_H(A) \leq D(A) \leq n$$

El siguiente teorema simplifica aún más el cálculo de la dimensión fractal al permitir la sustitución de la variable continua ϵ por una variable discreta.

Teorema A.3 (Teorema de recuento por cajas) *Sea $A \in \mathcal{H}(\mathbf{R}^n)$ un conjunto compacto y no vacío de $\mathbf{R}^n$. Cubramos $\mathbf{R}^n$ mediante cajas cuadradas cerradas con lados de longitud $(1/2)^m$. Sea $\mathcal{N}_m(A)$ el número de cajas de lado de longitud $(1/2)^m$ que intersectan con A. Si*

$$D = \lim_{m\to\infty} \left\{ \frac{\log \mathcal{N}_m(A)}{\log 2^m} \right\}$$

entonces A tiene dimensión fractal D.

La aplicación del teorema es tan sencilla como situar una malla sobre el conjunto a medir y contar el número de cajas en las que hay algún punto del conjunto. Calculemos mediante este método la dimensión fractal del triángulo de Sierpinski.

Ejemplo Consideremos el triángulo de Sierpinski S de la figura A.1. Puede verse como $\mathcal{N}_1(S) = 3$, $\mathcal{N}_2(S) = 9$, $\mathcal{N}_3(S) = 27$ y, en general, $\mathcal{N}_n(S) = 3^n$ para $n = 1, 2, \ldots$

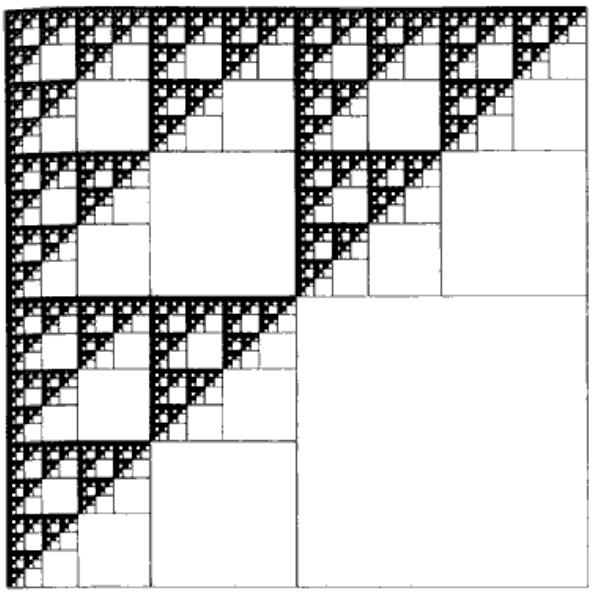


Figura A.1: Se necesitan 3^n cajas cerradas de lado $(1/2)^n$ para cubrir el triángulo de Sierpinski. Se deduce por tanto que su dimensión fractal es $\log 3/\log 2$. Figura tomada de [BAR 93a].

El teorema A.3 implica que

$$\begin{aligned} D(S) &= \lim_{n\to\infty}\left\{\frac{\log \mathcal{N}_n(A)}{\log 2^n}\right\} \\ &= \lim_{n\to\infty}\left\{\frac{\log 3^n}{\log 2^n}\right\} = \frac{\log 3}{\log 2} = 1{,}584962501\ldots \end{aligned}$$

Aunque es posible utilizar los métodos analíticos anteriores para el cálculo de la dimensión sobre cualquier tipo de fractal, la dificultad de hacerlo puede ser casi insuperable. Aunque la curva de Koch o el triángulo de Sierpinski se domestican fácilmente, conjuntos como los de Julia o el de Mandelbrot, vistos en el capítulo 1, son otra historia. Si existen técnicas analíticas para calcular su dimensión, todavía no han sido descubiertas. No obstante, es posible utilizar técnicas experimentales. La más sencilla de tales técnicas consiste en registrar varios valores de $\mathcal{N}_m(A)$ y representar los resultados en un gráfico con $\log \mathcal{N}_m(A)$ en el eje vertical y $\log 2^m$ en el eje horizontal. La pendiente de la curva de regresión que mejor se adapte a los puntos representados será una aproximación de la dimensión fractal del objeto.

Las definiciones dadas en este apéndice para la dimensión no cubren la multitud de distintas definiciones existentes hoy día. Aun así pueden considerarse un buen punto de partida para estudios más profundos.

Ante el asombro de Menón por los conocimientos demostrados por su siervo, Sócrates le explica cómo *"éste se ha de comportar de la misma manera con cualquier geometría y con todas las demás disciplinas"*. Quizá de haber escrito hoy día sus *Diálogos*, Platón habría hecho que el esclavo razonara a continuación la evolución a diferentes escalas de algún conjunto insigne de la geometría fractal.

Apéndice B

La teoría de los wavelets

Aunque la transformada de Fourier es probablemente la transformada más utilizada, no es la única. Existen multitud de transformadas distintas, cada una con sus ventajas e inconvenientes, usadas en ingeniería y matemáticas: la transformada de Hilbert, la transformada de Fourier a corto plazo, las distribuciones de Wigner o la transformada con wavelets, por poner algunos ejemplos.

Debido a su utilidad para la compresión de señales, nos centraremos aquí en la transformada con wavelets, principalmente siguiendo a [POL 97]. Aunque se mostrarán algunos de los conceptos más importantes, es necesario que el lector conozca los fundamentos de la transformada de Fourier y de la representación de señales en el dominio de la frecuencia.[1]

B.1. Limitaciones de la transformada de Fourier

La transformada de Fourier (TF) descompone una señal en funciones exponenciales complejas de diferentes frecuencias:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi jft}dt \qquad \text{(transformada de Fourier)}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{-2\pi jft}df \qquad \text{(transformada inversa de Fourier)}$$

donde t es el tiempo, f la frecuencia, x denota la señal en el dominio del tiempo y X es la señal en el dominio de la frecuencia.

[1] Existen multitud de referencias sobre la transformada de Fourier por lo que cualquier trabajo sobre tratamiento de señales debería servir. Un buen libro, centrado en las señales discretas, es *Discrete-Time Signal Processing* de Alan V. Oppenheim y Ronald W. Schafer, Prentice–Hall, 1989.

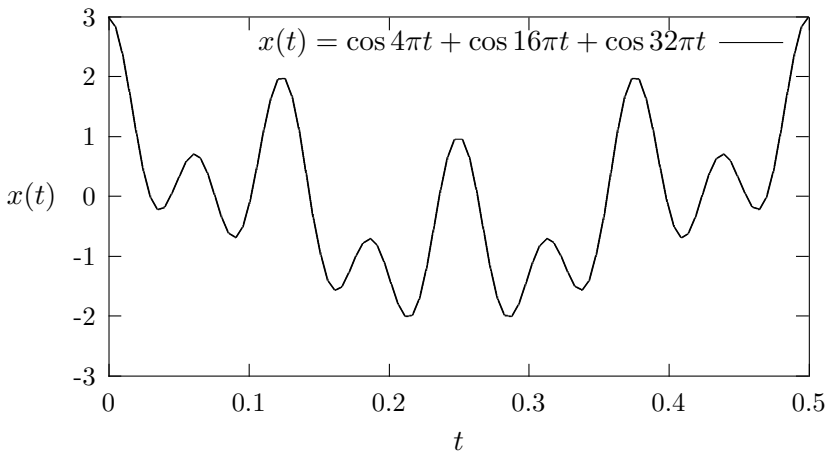


Figura B.1: La señal $\cos 4\pi t + \cos 16\pi t + \cos 32\pi t$ es estacionaria en el sentido de que sus componentes frecuenciales se distribuyen a lo largo de toda la señal.

Según la ecuación de Euler

$$e^{j\alpha} = \cos\alpha + j \cdot \operatorname{sen}\alpha$$

por lo que la ecuación de la transformada de Fourier multiplica la señal original por una expresión compleja formada por senos y cosenos de frecuencia f y posteriormente integra este producto. Si el resultado de esta integración para un cierto valor de f es grande, diremos que la señal $x(t)$ tiene una componente espectral dominante en la frecuencia f.

A pesar de su enorme utilidad, la TF presenta ciertas limitaciones cuando se aplica a señales no estacionarias.[2] Supongamos, por ejemplo, que tenemos dos señales diferentes, ambas con las mismas componentes espectrales, pero con una diferencia: una de las señales tiene tres componentes frecuenciales en todo el tiempo (puede tratarse, por ejemplo, de la señal mostrada en la figura B.1) y la otra tiene las mismas tres componentes frecuenciales, pero en tiempos distintos (tal como la señal mostrada en la figura B.2). Aunque las señales son completamente diferentes, su TF es la misma.

Cuando se necesita la localización en el tiempo de las componentes espectrales por estar tratando una señal no estacionaria, es necesaria una transformada que nos dé la representación tiempo-frecuencia de la señal.

Una de tales transformadas es la transformada de Fourier a corto plazo,

[2]En las señales estacionarias todas las componentes de frecuencia existentes en la señal aparecen a lo largo de toda su duración; en las señales no estacionarias, por otra parte, la frecuencia cambia constantemente a lo largo del tiempo.

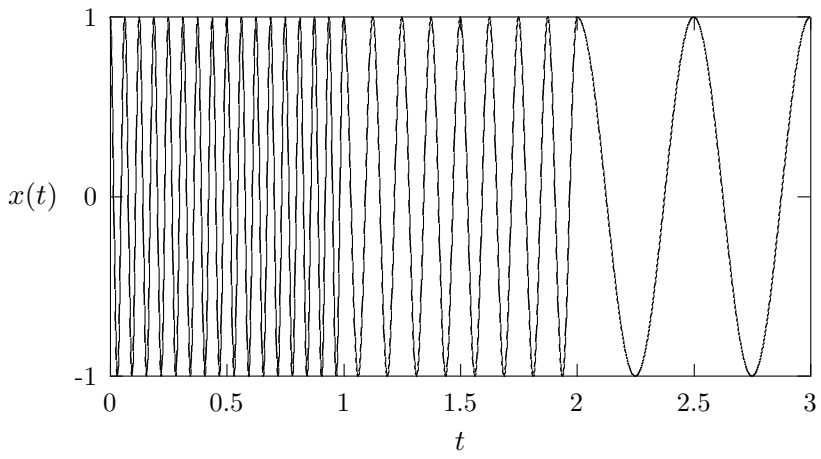


Figura B.2: La señal mostrada no es estacionaria debido a que se trata de la concatenación de tres ondas finitas de distintas frecuencias que no se solapan en el tiempo. En este caso la señal está definida como $\cos 32\pi t$ en el intervalo [0,1], como $\cos 16\pi t$ en [1,2] y como $\cos 4\pi t$ en el intervalo [2,3].

que nos proporciona una representación de la señal en tiempo y frecuencia simultáneamente. La transformada con wavelets se desarrolló para superar algunos problemas de resolución de la transformada de Fourier a corto plazo.

B.2. La transformada de Fourier a corto plazo

Para superar el problema de la TF con señales no estacionarias, podemos considerar que una señal no estacionaria es estacionaria localmente, es decir, en pequeños intervalos de tiempo. Este enfoque llevó a la formulación de la denominada *transformada de Fourier a corto plazo* (TFCP). En ella la señal se divide en segmentos lo suficientemente pequeños como para que pueda asumirse un comportamiento estacionario en ellos. Con este fin se elige una ventana $w(t)$ de ancho finito que se va deslizando sobre la señal:

$$TFCP_x^w(t', f) = \int_t [x(t) \cdot w^*(t - t')] \cdot e^{-2\pi jft} dt$$

El término $w^*(t - t')$ representa el conjugado de la función ventana desplazada t' unidades de tiempo a la derecha. Como puede verse, la TFCP no es más que la TF de la señal previamente multiplicada por una función ventana. Para cada t' y f se obtiene un coeficiente.

Por lo tanto, estamos obteniendo una representación real de la señal en

tiempo y frecuencia. La TFCP es bidimensional (en un eje aparece la frecuencia y en el otro el tiempo) o tridimensional, si consideramos la amplitud.

Sin embargo, hay un problema nada despreciable en la formulación anterior. Un problema que tiene sus raíces en el *principio de incertidumbre de Heisenberg*.

El principio de incertidumbre, formulado originalmente por Heisenberg, afirma que no pueden conocerse simultáneamente el momento y la posición de una partícula en movimiento. Aplicándolo a nuestro estudio, no podemos conocer qué componentes espectrales existen en un determinado instante de tiempo. Lo más que podemos hacer es investigar qué componentes espectrales existen en un cierto intervalo de tiempo, lo cual es un problema de *resolución*.[3]

En la TF no existe problema de resolución en el dominio de la frecuencia, es decir, sabemos *exactamente* qué frecuencias existen en la señal; de manera similar, no existe un problema de resolución en el dominio del tiempo, ya que conocemos el valor de la señal en cada instante de tiempo. Sin embargo, la resolución temporal en la TF y la resolución en frecuencias en el dominio del tiempo son nulas, ya que no tenemos información sobre ellas. Lo que da la resolución perfecta en frecuencias es el hecho de que la ventana utilizada en la TF, la función $e^{-2\pi jft}$, tiene duración infinita. En la TFCP la ventana es de longitud finita, por lo que perdemos la resolución en frecuencias perfecta y obtenemos una resolución en frecuencias más pobre a costa de una mejor resolución temporal.

Nuestro dilema puede esquematizarse como: ventana estrecha $\Longrightarrow$ buena resolución temporal y resolución en frecuencias pobre; ventana ancha $\Longrightarrow$ buena resolución en frecuencias y resolución temporal pobre. La figura B.3 muestra la TFCP de una señal, similar a la de la figura B.2 pero con cuatro frecuencias distintas en lugar de tres, en la que se utilizó una ventana estrecha.[4] Puede verse cómo en el dominio de la frecuencia cada pico cubre un rango de frecuencias y no un único valor. En el dominio del tiempo, sin embargo, los cuatro picos están bien separados unos de otros.

[3]El principio de incertidumbre afirma que no es posible reducir arbitrariamente a la vez la resolución en el tiempo (Δx) y la resolución en frecuencia ($\Delta\omega$) debido a que su producto está acotado inferiormente por la desigualdad de Heisenberg

$$\Delta x\, \Delta\omega \geq \frac{1}{2}$$

Esta desigualdad implica que debemos sacrificar la resolución en el tiempo por la resolución en frecuencias o viceversa.

[4]Por motivos que ahora no entraremos a analizar, la TFCP es siempre simétrica. Puede, por tanto, descartarse la mitad de la transformada sin que se pierda ninguna información.

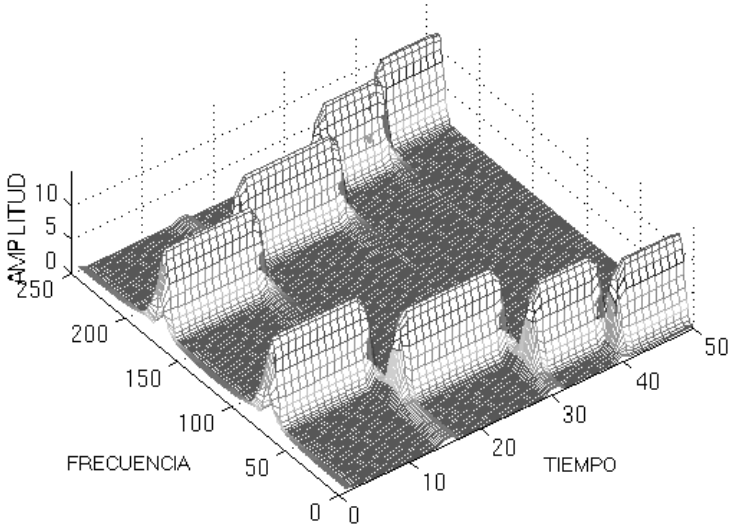


Figura B.3: La transformada de Fourier a corto plazo de una señal realizada con una ventana estrecha proporciona una muy buena resolución temporal, pero una resolución en frecuencias relativamente pobre. Figura tomada de [POL 97].

La transformada con wavelets resuelve hasta cierto punto el dilema. La TFCP da una resolución fija durante todo el tiempo, mientras que la transformada con wavelets da una resolución variable basándose en que las frecuencias altas se resuelven mejor en el tiempo y las frecuencias bajas se resuelven mejor en la frecuencia.

B.3. Análisis multirresolución

Aunque los problemas de resolución en tiempo y frecuencia son el resultado de un fenómeno físico (el principio de incertidumbre de Heisenberg) y existen independientemente de la transformada utilizada, es posible analizar una señal mediante un enfoque alternativo denominado *análisis multirresolución*. El análisis multirresolución analiza la señal con diferentes resoluciones en diferentes frecuencias. A diferencia de la TFCP, cada componente espectral no se trata de la misma forma.

El análisis multirresolución está diseñado para dar una buena resolución temporal y una resolución en frecuencias pobre para las frecuencias altas y una buena resolución en frecuencias junto a una resolución temporal pobre para las frecuencias bajas. Este enfoque tiene especial sentido cuando la

señal a analizar tiene componentes frecuenciales altas durante breves periodos de tiempo y componentes frecuenciales bajas durante periodos largos. Afortunadamente, en la práctica, casi todas las señales son de este tipo.

La transformada con wavelets basa su funcionamiento en el análisis multirresolución.

B.4. La transformada continua con wavelets

La transformada continua con wavelets fue desarrollada a comienzos de los años ochenta como alternativa a la TFCP para superar el problema de la resolución. Aunque guarda un cierto parecido con la TFCP, en el caso de la transformada continua con wavelets el ancho de la ventana va cambiando conforme se calcula la transformada para cada componente espectral. La transformada continua con wavelets (TCW) se define como

$$TCW_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s} \right) dt$$

donde τ es el parámetro de traslación, s es el parámetro de escala y $\psi(t)$ es la función de transformación conocida como *onda madre* o *wavelet madre*.

El término *wavelet*[5] significa en inglés *onda pequeña*. Lo de *onda* se refiere al hecho de que esta función es oscilatoria. El término *madre* se refiere a que las ventanas con distintos anchos que se utilizan en la transformada se derivan de una única función. En otras palabras, la onda madre es un prototipo para generar el resto de ventanas.

La traslación τ está relacionada, como en la TFCP, con el desplazamiento de la ventana por la señal. Aquí, sin embargo, no hay un parámetro para la frecuencia. En su lugar se utiliza el parámetro de escala s que se define como la inversa de la frecuencia.

El parámetro de escala es similar a la escala utilizada en los mapas. Al igual que en los mapas, una escala alta corresponde a una vista global de la señal sin mucho detalle y una escala baja corresponde a una vista detallada. De igual forma, en términos de frecuencia, las frecuencias bajas (escalas altas) corresponden a una información global de la señal, mientras que las frecuencias altas (escalas bajas) corresponden a una información detallada de la señal que dura normalmente un breve instante de tiempo.

[5]Este es el único término no traducido al castellano en esta obra, lo cual se debe a la dificultad de encontrar una traducción agradable y al poco tiempo de vida de esta transformada, que impide la existencia de un equivalente ampliamente aceptado en castellano.

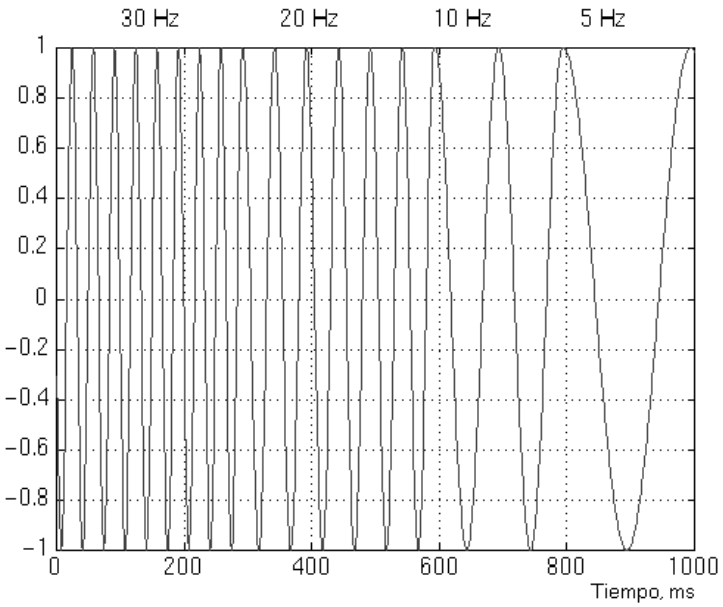


Figura B.4: Señal no estacionaria utilizada para mostrar el aspecto de la transformada continua con wavelets. La frecuencia de la señal va decreciendo conforme aumenta el tiempo. Su transformada se muestra en la figura B.5. Figura tomada de [POL 97].

Matemáticamente hablando, el escalado comprime o dilata una señal. Las escalas altas correspoden a señales dilatadas o expandidas y las escalas bajas a señales comprimidas. Por tanto, todas las ventanas utilizadas en la TCW son versiones desplazadas y dilatadas (o comprimidas) de la onda madre.

La multiplicación por $1/\sqrt{|s|}$ en la expresión anterior se utiliza para normalizar las energías en cada escala. La transformada con wavelets proporciona, en definitiva, un punto del plano escala-desplazamiento para cada escala y para cada instante de tiempo.

Veamos ahora un ejemplo del aspecto de una transformada con wavelets. La figura B.5 es la TCW de la señal no estacionaria de la figura B.4. Recordemos que la escala debe interpretarse como la inversa de la frecuencia por lo que las zonas de la gráfica de la figura B.5 con escala cercana a cero corresponden realmente a las frecuencias más altas. En cualquier caso, los valores de los ejes no deben tenerse en cuenta, ya que están normalizados, pero sí puede observarse cómo se refleja que las frecuencias más altas de la señal (bajas escalas) aparecen primero y cómo, conforme nos desplazamos por el eje del desplazamiento, la señal presenta frecuencias cada vez más bajas (escalas altas), lo cual está acorde con la representación de la señal de

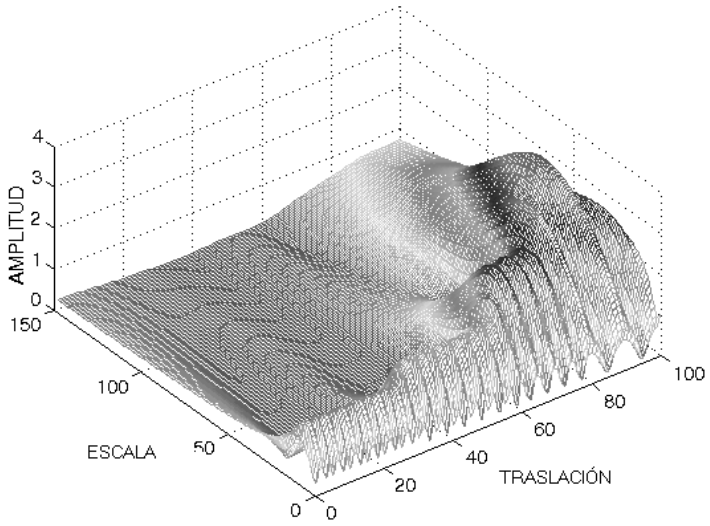


Figura B.5: TCW de la señal de la figura B.4. La transformada refleja la presencia de las frecuencias a lo largo de la señal. Conforme se aumenta el tiempo (desplazamiento), aparecen más componentes en las escalas altas, es decir, en las frecuencias bajas. Figura tomada de [POL 97].

la figura B.4.

Una de las funciones más utilizadas para la onda madre es la función de sombrero mejicano que se define como la segunda derivada de la gaussiana

$$w(t) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{\frac{-t^2}{2\sigma^2}}$$

que es

$$\psi(t) = \frac{1}{\sqrt{2\pi}\,\sigma^3} \left(e^{\frac{-t^2}{2\sigma^2}} \left(\frac{t^2}{\sigma^2} - 1 \right) \right)$$

y que aparece representada en la figura B.6.

Resolución en tiempo y frecuencia

Para ver cómo puede interpretarse la resolución en tiempo y frecuencia, podemos observar la figura B.7. Cada caja de la figura corresponde a un valor de la transformada con wavelets en el plano tiempo-frecuencia. Nótese que las cajas tienen área no nula, lo que implica que no puede conocerse el valor de un punto particular en el plano tiempo-frecuencia. Todos los

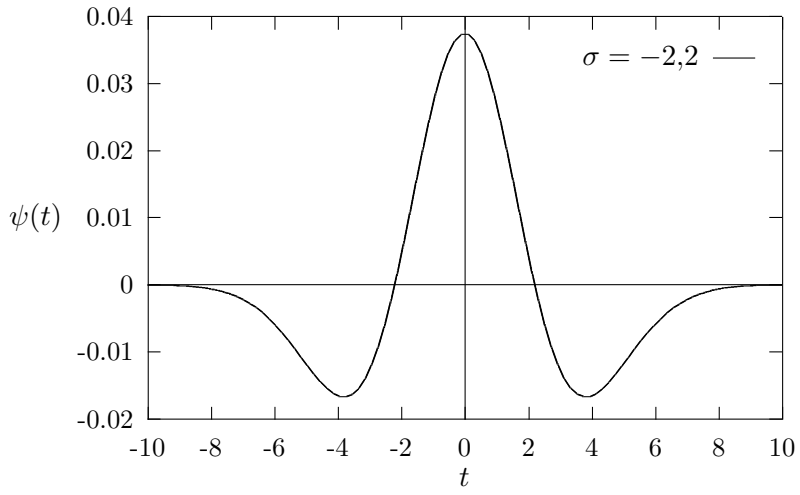


Figura B.6: Una de las ondas madre utilizada con mayor frecuencia en el cálculo de la TCW es la función de sombrero mejicano.

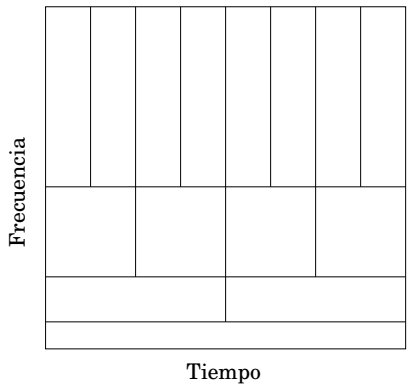


Figura B.7: Este diagrama es habitual a la hora de analizar la forma de la resolución en tiempo y frecuencia proporcionada por la transformada con wavelets. El principio de incertidumbre obliga a que el área de todas las cajas sea la misma, pero la transformada con wavelets varía sus dimensiones para atender de distinta manera a las distintas frecuencias de la señal.

puntos de este plano que caen en una caja se representan por un valor de la transformada con wavelets.

Aunque el alto y ancho de las cajas cambia, el área es constante. Es decir, cada caja representa una parte igual del plano tiempo-frecuencia, pero con diferentes proporciones de tiempo y frecuencia. Esta área no puede reducirse debido al principio de incertidumbre de Heisenberg. Véase cómo para frecuencias bajas las cajas son menos altas (lo que corresponde a mejores re-

soluciones en frecuencias, ya que hay menos ambigüedad para determinar el valor exacto de la frecuencia), pero tienen una anchura mayor (lo que corresponde a una resolución temporal más pobre, ya que hay mayor ambigüedad para determinar el valor del tiempo exacto). Para frecuencias mayores decrece el ancho de las cajas, es decir, mejora la resolución temporal, y aumenta su altura, con lo que se empobrece la resolución en frecuencias.

La transformada inversa

La TCW es una transformada reversible siempre que se cumpla la ecuación B.1 dada más abajo. Afortunadamente, éste no es un requerimiento muy restrictivo. La reconstrucción es posible mediante

$$x(t) = \frac{1}{c_\psi^2} \int_s \int_\tau \Psi_x^\psi(\tau, s) \frac{1}{s^2} \psi\left(\frac{t-\tau}{s}\right) d\tau \, ds$$

donde c_ψ es una constante que depende de la onda madre utilizada. El éxito en la reconstrucción depende de que esta constante, llamada *constante de admisibilidad*, satisfaga la siguiente *condición de admisibilidad*:

$$c_\psi = \sqrt{2\pi \int_{-\infty}^{\infty} \frac{\left|\hat{\psi}(\xi)\right|^2}{|\xi|} d\xi} \quad < \infty \tag{B.1}$$

donde $\hat{\psi}(\xi)$ es la tranformada de Fourier de $\psi(t)$. La ecuación B.1 implica que $\hat{\psi}(0) = 0$ que, por una propiedad de la TF, es equivalente a

$$\int \psi(t) dt = 0$$

Esta última ecuación no es muy restrictiva, ya que hay muchas funciones que se pueden utilizar como onda madre cuya integral es cero. Para ello la función debe ser oscilatoria. La función sombrero mejicano de la página 152 es una de las muchas que cumple la condición de admisibilidad.

B.5. La transformada discreta con wavelets

La idea de la transformada discreta con wavelets (TDW) es la misma que en la transformada continua. Utilizando filtros digitales se obtiene una representación tiempo-escala (frecuencia) de una señal digital. La TCW se calculaba variando la escala de la ventana de análisis, desplazando la ventana en el tiempo, multiplicando por la señal e integrando sobre todo el tiempo.

En el caso discreto se utilizan filtros de diferentes frecuencias de corte para analizar la señal a diferentes escalas. La señal se pasa por una serie de filtros de paso alto para analizar las frecuencias altas y por una serie de filtros de paso bajo para analizar las frecuencias bajas. La resolución de la señal se cambia mediante operaciones de filtrado y la escala se cambia con operaciones de submuestreo y supermuestro.

Submuestrear una señal es reducir la tasa de muestreo, eliminando algunas muestras de la señal. Por ejemplo, para submuestrear por dos una señal puede eliminarse una de cada dos muestras consecutivas. El submuestreo por un factor n reduce n veces el número de muestras de la señal.

Supermuestrear una señal es incrementar la tasa de muestreo de la señal, añadiéndole nuevas muestras. Por ejemplo, para supermuestrear por dos una señal puede añadirse entre cada dos muestras una nueva, normalmente un cero o un valor interpolado. El supermuestreo por un factor n incrementa el número de muestras de la señal por un factor n.

En lo siguiente representaremos una secuencia por $x(n)$ donde n es un entero. Además, debe tenerse en cuenta que en las señales discretas la frecuencia se expresa en términos de radianes y que el ancho de banda de cualquier secuencia es de π rad/s.

Veamos ahora cómo actúa la TDW. En primer lugar la señal original $x(n)$ se pasa por un filtro de media banda de paso alto $g(n)$ y por uno de paso bajo $h(n)$. El filtrado de una secuencia por un filtro digital de respuesta impulsiva $h(n)$ se expresa matemáticamente como la convolución

$$\begin{aligned} y(n) &= x(n) * h(n) \\ &= \sum_k x(k)h(n-k) \\ &= \sum_k h(k)x(n-k) \end{aligned}$$

donde $y(n)$ es la secuencia a la salida del filtro.

Tras el filtrado pueden eliminarse la mitad de las muestras según el principio de Nyquist, ya que la máxima frecuencia de la señal (frecuencia de corte) es ahora de $\pi/2$ rad/s y no de π rad/s. Puede, por tanto, submuestrearse la señal por dos tomando una muestra sí y otra no.

La descomposición anterior reduce a la mitad la resolución temporal, ya que la señal completa está caracterizada ahora por la mitad de las muestras. Sin embargo, la resolución en frecuencias se ha duplicado, ya que el ancho de banda de la señal es la mitad del ancho de banda anterior.

Este procedimiento, también conocido como *codificación subbanda*, se

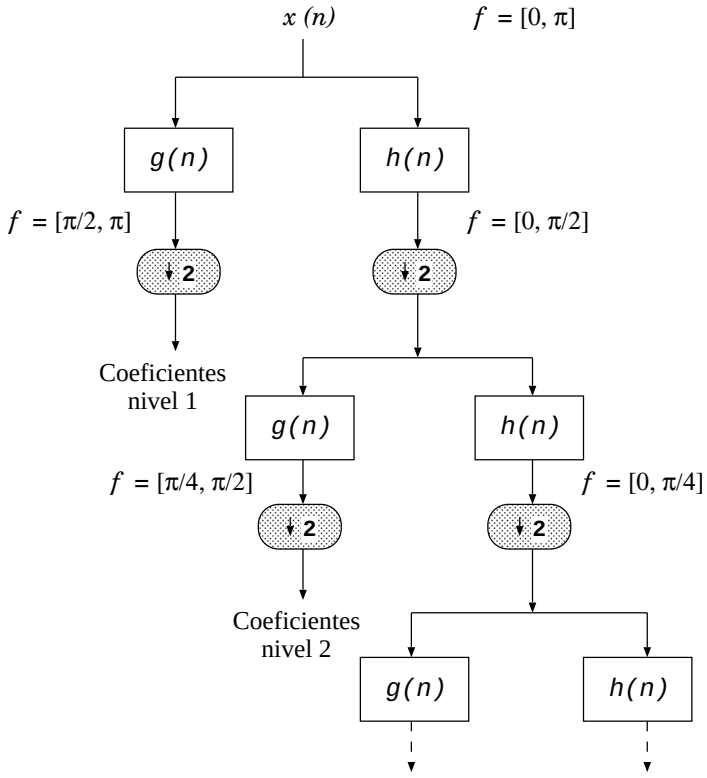


Figura B.8: El algoritmo de codificación subbanda puede verse como la aplicación de una cascada de filtros de paso de media banda seguidos de submuestreadores por dos. El ancho espectral de la señal en cada nivel se representa mediante f.

repite sucesivamente sobre la salida del filtro de paso bajo hasta obtener una secuencia de longitud dos. Los primeros pasos del procedimiento se muestran en la figura B.8.

Finalmente, la TDW de la señal original se obtiene concatenando todos los coeficientes desde el primer nivel de descomposición. La TDW tendrá el mismo número de coeficientes que la señal original.

Las frecuencias más destacadas de la señal original aparecerán con amplitudes grandes en la región de la transformada que incluya esas frecuencias en particular. La diferencia entre esta transformada y la transformada discreta de Fourier es que no hemos perdido la localización temporal de esas frecuencias. Sin embargo, la resolución de esta localización temporal dependerá del nivel en que aparezca la frecuencia. Si la información principal de la señal aparece en las frecuencias altas, como ocurre muy a menudo, la localización temporal de estas frecuencias será más precisa, ya que vienen caracterizadas por un mayor número de muestras. Si la información principal de la señal

aparece sólo en las frecuencias bajas, su localización temporal no será muy precisa, ya que se utilizan pocas muestras para representar la señal en esas frecuencias. Este procedimiento ofrece, en definitiva, una buena resolución temporal en las frecuencias altas y una buena resolución en frecuencias para frecuencias bajas. Como ya hemos dicho, este comportamiento es adecuado para tratar la mayor parte de las señales encontradas en la práctica.

Las bandas de frecuencias que no sean muy prominentes en la señal original tendrán amplitudes muy bajas y podremos prescindir de esa parte de la TDW casi sin perder información. Esta es la idea base en todos los sistemas de compresión basado en wavelets como el discutido en el capítulo 5.

Nótese que, debido a los sucesivos submuestreos por dos, la longitud de la señal debe ser potencia de dos o, al menos, un múltiplo de una potencia de dos, para que este esquema sea eficiente.

Filtros utilizados en la TDW

Una propiedad importante de la TDW es la relación entre las respuestas impulsivas de los filtros de paso alto y paso bajo. Ambos están relacionados por

$$g(L-1-n) = (-1)^n\, h(n) \tag{B.2}$$

donde $g(n)$ es el filtro de paso alto, $h(n)$ el filtro de paso bajo y L es la longitud del filtro. Las operaciones de filtrado y submuestreo de la transformada pueden expresarse como

$$y_{alta}(n) = \sum_k x(k) \cdot g(-k+2n) \tag{B.3}$$

$$y_{baja}(n) = \sum_k x(k) \cdot h(-k+2n) \tag{B.4}$$

Si los filtros utilizados cumplen la condición B.2, la transformada inversa es muy sencilla y se limita a reconstruir la señal original siguiendo el procedimiento explicado anteriormente en orden inverso. Las señales de cada nivel se supermuestrean por dos, se pasan por los filtros de análisis $\tilde{g}(n)$ y $\tilde{h}(n)$ (de paso alto y paso bajo, respectivamente) y, finalmente, se suman. Los filtros de análisis y los de síntesis son idénticos entre sí, excepto por estar invertidos uno respecto al otro:

$$\begin{aligned} \tilde{g}(n) &= g(-n) \\ \tilde{h}(n) &= h(-n) \end{aligned}$$

Por lo tanto, la reconstrucción de la señal es para cada nivel

$$x(n) = \sum_k [y_{alta}(k) \cdot g(-n+2k) + y_{baja}(k) \cdot h(-n+2k)] \tag{B.5}$$

Las ecuaciones B.3, B.4 y B.5 son la piedra angular de todo codificador/descodificador basado en wavelets.

El desarrollo anterior se llevó a cabo suponiendo que los filtros eran ideales. Aun así, aunque no es posible diseñar filtros ideales, sí pueden encontrarse filtros que permitan una reconstrucción perfecta bajo ciertas condiciones. Unos de los más utilizados son los *filtros de Daubechies*, cuyos coeficientes $h(n)$ son

$$\begin{aligned} h_0 &= 0{,}332670552950 \\ h_1 &= 0{,}806891509311 \\ h_2 &= 0{,}459877502118 \\ h_3 &= -0{,}135011020010 \\ h_4 &= -0{,}085441273882 \\ h_5 &= 0{,}035226291882 \end{aligned}$$

Los coeficientes de los otros filtros pueden obtenerse a partir de éstos mediante las expresiones dadas anteriormente.

Ya se ha comentado la utilidad de la TDW para la compresión de señales. Su aplicación detallada a la compresión de imágenes se discute en la sección 5.6.

Apéndice C

Imágenes originales

Este apéndice muestra el original de las imágenes utilizadas a lo largo de la obra. Para cada una se indica brevemente los aspectos esenciales que una buena codificación debería preservar. Evidentemente, con independencia de la técnica de compresión utilizada, estos detalles se irán perdiendo conforme aumente el ratio de compresión.

Figura C.1: Aparecen numerosas zonas de baja resolución y escaso contraste, especialmente en la zona del follaje. Las vallas del puente generan un gradiente alto en sus bordes. El tamaño de la imagen es 256 $\times$ 256.

Figura C.2: El cuerpo del pájaro muestra una textura agradable rodeada por cortes abruptos en las fronteras entre las plumas blancas y negras y entre el cuerpo y el fondo. Las garras, el pico y los ojos son estructuras cruciales para una codificación detallada. El fondo aumenta su luminosidad gradualmente desde la esquina inferior derecha a la superior izquierda, pero sin ser completamente liso. Tamaño 256 $\times$ 256.

Figura C.3: Similar a la figura C.1. Destaca la presencia de texturas de escala media así como la ausencia de grandes áreas uniformes. La variedad cromática de las fachadas de las casas debe capturarse adecuadamente. Por otra parte, la persona que sube por la cuesta podría fácilmente confundirse con el fondo. Tamaño 256 $\times$ 256.

Figura C.4: Similar a la figura C.2. Se trata de una imagen con numerosas transiciones suaves, aunque es importante capturar correctamente los cambios en el color del plumaje y del pico. Tamaño 372 $\times$ 279.

Figura C.5: Presenta numerosos detalles a baja escala, como los mástiles o las texturas de los barcos, sobre un cielo suave. Las letras que configuran el nombre del barco deberían ser atrapadas adecuadamente. Tamaño 512 × 512.

Figura C.6: Una de las imágenes más sencillas de codificar fractalmente debido a la similitud entre todas las regiones de la imagen. Tamaño 256 × 256.

Figura C.7: Dos planos bien diferenciados. En primer plano las flores con escaso contraste. Detrás, las hojas presentan detalles a menor escala. Tamaño 460 × 345.

Bibliografía

[ALI 91] M. Ali, T. G. Clarkson, "Fractal image compression". *Proc. 1st Seminar on Information Technology and its Applications (ITA'91)*, Markfield Conf. Centre, Leicester, U.K., 29 Sept., 1991. También disponible vía [FRE 97].

Un artículo de los primeros tiempos de la compresión fractal de imágenes en el que se muestran algunas técnicas (sin experimentos que las evalúen) para intentar resolver el problema inverso, esto es, obtener automáticamente el SFI que genera una imagen cualquiera. El texto propone expresar las transformaciones que conforman un SFI en forma compleja de manera que el problema se traslade al dominio de los momentos complejos. Se sugiere, entonces, el método de recocido simulado (*simulated annealing*) para encontrar el SFI más cercano a una imagen o a un segmento de ella.

[BAR 93a] M. Barnsley, *Fractals Everywhere*, Second Edition, Academic Press, 1993.

La enciclopedia de los sistemas de funciones iteradas y obra fundamental para el desarrollo de la compresión fractal. Aunque no se enfrenta decididamente al tema de la codificación de imágenes (aspecto éste presentado en posteriores obras del autor), expone exhaustivamente la teoría y las implicaciones prácticas de los SFI desde la topología de los espacios métricos, el teorema del collage y la dimensión fractal hasta los conjuntos de Julia o de Mandelbrot (desde la perspectiva de los SFI) y los SFI recurrentes.

[BAR 93b] J. Barrallo Calonge, *Geometría fractal: algorítmica y representación*, Anaya Multimedia, 1993.

El principal mérito de este libro es la enorme cantidad de programas en C que incluye. Casi sin matemáticas logra introducir prácticamente todas las facetas de estudio actuales sobre los fractales. El capítulo 13

discute los sistemas de funciones iteradas y el capítulo 14 la generación de curvas fractales mediante gramáticas.

[BLA 94] J. Blanc–Talon, "Inference of grammars from fractal sets: the inner structure". En *Grammatical Inference*, L. Erlbaum Assoc. Publisher, 1994, Simon Lucas Ed.

Los sistemas D0L son uno de los mecanismos más sencillos para generar estructuras fractales además de poseer propiedades topológicas bien conocidas. El artículo repasa la situación del momento de la generación sintáctica de fractales, principalmente a través del estudio de los sistemas D0L y de las gramáticas independientes del contexto. A continuación presenta un método para el problema inverso, es decir, la inferencia de sistemas D0L a partir de conjuntos de datos con distribución fractal y su generalización a la inferencia de gramáticas independientes del contexto.

[CIE 97] L. Ciepliński, C. Jędrzejek, T. Major, "Acceleration of a fractal image compression by fast nearest-neighbor search". *Fractals*, vol. 5, supplementary issue, 1997.

Sigue la idea de [SAU 95] de acelerar la compresión mediante la búsqueda multidimensional del vecino más cercano en un espacio de proyecciones. Los autores proponen y estudian empíricamente el algoritmo de *eliminación parcial de la distorsión* para la búsqueda. También se sugieren algunas mejoras adicionales como buscar sólo en la mitad del espacio de proyecciones o postprocesar la imagen mediante el suavizado de los bordes de las regiones destino.

[COL 96] J. Colvin, "Iterated function systems and fractal image compression". Disponible en `http://hamnetcenter.com/jeffc/fractal.html`.[1]

Una introducción muy sencilla a los SFI y a la codificación fractal de imágenes mediante SFI particionados. Un breve tutorial recomendado para una primera aproximación al tema.

[DAV 95] F. Davoine, J. Svensson, J.-M. Chassery, "A mixed triangular and quadrilateral partition for fractal image coding". *IEEE Int. Conf. on Image Processing (ICIP'95)*. También disponible vía [FRE 97].

Presenta un esquema para la segmentación de una imagen, basado en

[1] Debe tenerse en cuenta que las referencias disponibles en la red existían cuando se preparó este trabajo: en el momento de acceder a ellas pueden haber cambiado de lugar o incluso desaparecido.

triángulos y cuadrilateros, y los correspondientes algoritmos de compresión y descompresión fractal. Los triángulos obtenidos mediante triangulación de Delaunay permiten reducir el efecto de bloque en la imagen, pero, debido a que su elevado número puede afectar al ratio de compresión excesivamente, se procede a continuación a agrupar triángulos vecinos para formar cuadrilateros convexos.

[DED 98] Ľ. Dedera, J. Chmúrny, "A parallel approach to image decoding in the fractal image block coding scheme". Disponible vía [FRE 97].

Artículo sencillo que propone un modelo alternativo para el algoritmo de descodificación de imágenes comprimidas mediante SFI particionados. Este nuevo modelo utiliza una especie de red neuronal recurrente que necesita el mismo número de iteraciones que el algoritmo iterativo clásico. Este enfoque, sin embargo, facilita la paralelización del proceso.

[DUG 96] J.-L. Dugelay, E. Polidori, S. Roche, "Iterated function systems for still image processing". *Institut EURECOM*. Disponible en `http://www.eurecom.fr/~image`.

Breve artículo que propone dos líneas de desarrollo bien diferenciadas. Por una parte sugiere el uso de rangos no disjuntos en la codificación de imágenes para mejorar el zoom fractal clásico. Por otro lado se estudia un control jerárquico de acceso a imágenes codificadas mediante la transformada fractal.

[FIS 92] Y. Fisher, "Fractal Image Compression". *Siggraph'92 Course Notes*. Disponible también vía `http://inls.ucsd.edu/y/Fractals`.

Una revisión sencilla sobre los matices básicos de la compresión fractal de imágenes. Tras analizar los fundamentos de los SFI con ayuda de la metáfora de la fotocopiadora de reducciones múltiples, analiza el algoritmo mínimo de codificación de imágenes en escala de grises mediante SFI particionados. Posteriormente se amplía este esquema proponiendo la obtención de rangos mediante partición con árboles cuadriculares, partición HV o partición triangular.

[FIS 95] Y. Fisher (ed.), *Fractal Image Compression: Theory and Application*, Springer Verlag, 1995.

Una recopilación de artículos de diversos autores acerca de la codificación fractal. Es una de las referencias indispensables sobre el tema. Los tres primeros capítulos pueden considerarse una introducción pormenorizada a los SFI, a los SFI recurrentes, a los SFI particionados y

a la compresión fractal de imágenes, especialmente mediante árboles cuadriculares. Siguen varios artículos en los que se presentan distintas formas de aproximación al tema como pueden ser la codificación basada en arquetipos, en resultados del álgebra lineal, en cuantización vectorial o en autómatas finitos ponderados. Los apéndices son casi tan interesantes como el resto del libro; en ellos aparece código de ejemplo en C para un compresor/descompresor basado en árboles cuadriculares, así como aspectos todavía abiertos o poco investigados de la compresión fractal. Estos últimos constituyen una valiosa fuente para nuevos proyectos.

[FRE 97] Una gran colección de artículos sobre compresión fractal de imágenes puede encontrarse en `ftp://ftp.informatik.uni-freiburg.de/documents/papers/fractal`.

Este es uno de los puntos más interesantes para encontrar referencias actualizadas sobre el tema; cada mes se incorporan varios artículos en formato *postscript*. Además, se dispone de una extensa bibliografía con muchos más artículos de los disponibles en línea.

[FRI 94] C. Frigaard, J. Gade, T. Hemmingsen, T. Sand, "Image compression based on a fractal theory". Institute for Electronic Systems, Aalborg University, Denmark, 1994. También disponible vía [FRE 97].

Revisa el esquema original de compresión fractal propuesto por Jacquin, ampliando y mejorando la rutina de clasificación de bloques al asociar a cada rango un vector de características continuo y comparar cada rango únicamente con aquellos dominios cuyos vectores asociados están próximos en el espacio de características. El texto termina ofreciendo resultados comparativos entre el algoritmo propuesto y el estándar JPEG.

[GAI 97] Jean-loup Gailly, "Frecuently asked questions about compression". *FAQ de comp.compression*, Sept. 1997. Disponible en la dirección `ftp://rtfm.mit.edu/pub/usenet/news.answers/compression-faq`.

La respuesta a las preguntas más habituales sobre compresión de datos. Entre muchos otros temas aborda la compresión fractal de imágenes.

[GHA 96] M. Gharavi-Alkhansari, T. S. Huang, "Generalized image coding using fractal-based methods". *PCS'94*, Sacramento, Sept. 94. También disponible vía [FRE 97].

Un método distinto al básico de compresión fractal que convierte a

éste en un caso particular de aquél. Cada bloque se aproxima mediante una combinación lineal de bloques escogidos de un banco de bloques base. Este banco está formado por *a)* un conjunto de bloques tomados de la misma imagen que el bloque a aproximar (con la misma escala o bien resultantes de submuestrear la imagen) y *b)* un conjunto de bloques fijos suministrados inicialmente al codificador. El artículo muestra métodos para escoger los bloques del banco de bloques de manera que disminuya el error del collage.

[GUZ 93] M. de Guzmán, M. A. Martín, M. Morán, M. Reyes, *Estructuras fractales y sus aplicaciones*, Labor, 1993.

El primer libro español que aborda el tema a un nivel matemático profundo. El texto está escrito con un lenguaje claro y abierto y contiene varios programas en BASIC para explorar algunas estructuras fractales. El capítulo 2 discute ampliamente la dimensión de Hausdorff, el capítulo 4 aborda la teoría de conjuntos autosemejantes de Hutchinson y el capítulo 5 se centra totalmente en los sistemas de funciones iteradas, incluidos el teorema del collage y la generación de animaciones mediante SFI.

[HIL 94] M. L. Hilton, B. D. Jawerth, A. Sengupta, "Compressing still and moving images with wavelets". *Multimedia Systems*, Vol. 2, No. 3. También disponible en `ftp://ftp.math.scarolina.edu:21/pub/wavelet/papers/varia/tutorial`.

Explica de manera sencilla la transformada con wavelets y su aplicación a la compresión de imágenes y vídeo. Pese a su aparente sencillez contiene información suficiente para poder desarrollar un codificador de andar por casa. Además, el artículo muestra algunos resultados de la comparación entre el estándar JPEG y varios esquemas de codificación con wavelets.

[KOM 95] J. Kominek, "Algorithm for fast fractal image compression". *Proceedings of SPIE*, Volume 2419, 1995. También disponible vía [FRE 97].

Este texto propone una técnica para reducir la complejidad temporal de la compresión fractal. La idea básica es normalizar las intensidades de los pixels de los rangos para que tengan media y varianza fija y utilizar, entonces, un árbol r (una extensión de los árboles b) para acceder eficientemente a los dominios estructuralmente cercanos a cada rango.

[LU 97] N. Lu, *Fractal Imaging*, Academic Press, 1997.

Una revisión acutualizada y didáctica de la incursión fractal en la

compresión de imágenes. Con claridad no exenta de rigor aborda el modelo fractal de una imagen básico ampliándolo posteriormente mediante el estudio detallado de técnicas de partición, transformaciones espaciales y de intensidad, búsqueda mediante clasificación o descompresión rápida. El texto dedica también capítulos a la compresión con wavelets o transformada del coseno, compresión de vídeo o imágenes en color y codificación mediante entropía. Aunque contiene abundante código en C, éste no es totalmente funcional debido a la presencia de errores, tipográficos normalmente aunque no siempre. De cualquier forma el estudio detallado de los desarrollos teóricos permite encontrar con cierta facilidad estos fallos. Se deja leer y se aprende bastante.

[LUT 95] E. Lutton, J. Lévy Véhel, G. Cretin, P. Glevarec, C. Roll, "Mixed IFS: resolution of the inverse problem using genetic programming". *Institut National de Recherche en Informatique et en Automatique*. Disponible en `http://www-syntim.inria.fr/fractales`.

Se enfrenta a la resolución del problema inverso para SFI con un enfoque muy distinto al de [ALI 91]. El artículo considera SFI no afines (SFI mixtos) con distintos operadores y constantes, representando cada transformación mediante un árbol. La búsqueda del SFI mixto que mejor aproxima un determinado conjunto se lleva a cabo mediante un esquema de programación genética sobre estos árboles. Estudia, además, el ajuste de los distintos parámetros del algoritmo evolutivo y la forma de asegurar la contractividad del sistema obtenido.

[POL 97] R. Polikar, "The wavelet tutorial". Disponible en `http://www.public.iastate.edu/~rpolikar/WAVELETS/WTpart1.html`.

Un tutorial en línea sobre la teoría de los wavelets cargado de imágenes muy útiles para comprender intuitivamente sus fundamentos. Describe con un enfoque muy ameno y educativo la noción del espectro de frecuencias de una señal, la transformada de Fourier clásica y a corto plazo y la transformada con wavelets tanto continua como discreta.

[RUH 97] M. Ruhl, H. Hartenstein, D. Saupe, "Adaptative partitionings for fractal image compression". *IEEE Int. Conf. on Image Processing (ICIP'97)*. También disponible vía [FRE 97].

Repasa brevemente el algoritmo evolutivo de segmentación presentado en [SAU 96b] y entra en algunos detalles para una implementación eficiente no discutidos allí. El artículo continúa con nuevos resultados y su confrontación con los obtenidos mediante particiones HV.

[SAU 94a] D. Saupe, R. Hamzaoui, "A guided tour of the fractal image compression literature". Technical Report 58, *Institut für Informatik*, July 94. Una primera versión aparece en las notas del SIGGRAPH'94. También disponible vía [FRE 97].

Este artículo presenta un repaso cronológico a la evolución de la compresión fractal de imágenes mediante el análisis de la mayor parte de los trabajos sobre el tema aparecidos hasta julio de 1994, fecha de edición del artículo. Los trabajos más relevantes se presentan, además, mediante sus abstracts originales. Finalmente, se incluye una extensa bibliografía, la mayor publicada hasta ese momento y la principal contribución del artículo según los autores.

[SAU 94b] D. Saupe, R. Hamzaoui, "Complexity reduction methods for fractal image compression". *IMA Conf. Proc. on Image Processing; Mathematical Methods and Applications*, Sept. 94, J. M. Blackledge (ed.), Oxford University Press, 1997. También disponible vía [FRE 97].

Recorre distintos métodos para la aceleración de la búsqueda en la codificación fractal como la clasificación por reordenación de cuadrantes, la búsqueda del vecino más cercano o distintas técnicas de agrupamiento adaptativo. Incluye una pequeña comparación entre todos los métodos analizados.

[SAU 95] D. Saupe, "Accelerating fractal image compression by multidimensional nearest neighbor search". *Proceedings DCC'95 Data Compression Conference*, J. A. Storer, M. Cohn (eds.), IEEE Computer Society Press, March 1995. También disponible vía [FRE 97].

La fase más costosa de la compresión fractal de imágenes es la búsqueda de la región de referencia más parecida (en términos de escalado y desplazamiento) entre todas las posibles regiones de referencia de la imagen. Si el número de éstas es N, el tiempo de cada búsqueda lineal es $O(N)$. El uso de un algoritmo de clasificación (*clustering*) inicial únicamente reduce el factor de proporcionalidad de la complejidad, pero no su orden. El artículo demuestra que la búsqueda secuencial sobre las regiones de referencia (o una clase de ellas) puede sustituirse por una búsqueda multidimensional del vecino más cercano, tarea ésta que puede llevarse a cabo con varios algoritmos de coste $O(\log N)$. El autor muestra resultados prometedores de una instrumentación mediante *arboles kd*.

[SAU 96a] D. Saupe, R. Hamzaoui, H. Hartenstein, "Fractal image compression: an introductory overview". En *Fractal Models for Ima-*

ge Synthesis, Compression, and Analysis, D. Saupe, J. Hart (eds.), ACM SIGGRAPH'96 Course Notes. También disponible vía [FRE 97].

El artículo de cabecera de la compresión fractal de imágenes, además de ser una fuente indispensable junto a [FIS 92] para una primera aproximación al tema. Analiza de forma excelente la compresión fractal a través de su similitud con la cuantización vectorial con eliminación de media y ganancia de forma. Además, aborda temas avanzados como la reducción del coste temporal, la partición adaptativa o métodos híbridos, haciendo un recorrido por la literatura existente. Imprescindible.

[SAU 96b] D. Saupe, M. Ruhl, "Evolutionary fractal image compression". ICIP-96 *IEEE International Conference on Image Processing*, Lausanne, Sept. 1996. También disponible vía [FRE 97].

Este artículo muestra cómo particionar una imagen en rangos mediante computación evolutiva. El algoritmo se inicia con una partición en *bloques atómicos* (por ejemplo, bloques de 4×4) y sucesivas generaciones unen pares de ellos para obtener un nuevo rango. Se discuten también cuatro métodos para codificar eficientemente la partición finalmente obtenida y se presentan resultados que demuestran que el algoritmo evolutivo supone una gran mejora con ratios de compresión altos sobre el tradicional esquema basado en árboles cuadriculares.

[SAU 97] D. Saupe, R. Hamzaoui, "A Bibliography on Fractal Image Compression". Disponible en [FRE 97].

La bibliografía (sólo la bibliografía) aparecida en [SAU 94a] se actualiza regularmente en este documento para incorporar los últimos trabajos sobre compresión fractal de imágenes. Aunque no es una bibliografía comentada, es una buena referencia para seguirle la pista a los numerosos trabajos que se van publicando sobre el tema.

[SVR 95] E. R. Vrscay, "A hitchhiker's guide to fractal-based function approximation and image compression". Disponible en `http://links.uwaterloo.ca/hitchiker.html`.

La mayor parte de esta introducción está dedicada a presentar los SFI sobre funciones $u : [0,1] \longrightarrow \mathbf{R}^+$ en lugar de sobre conjuntos, esto es, sobre *imágenes unidimensionales* en escala de grises y no sobre las imágenes habituales (bidimensionales). Así se presenta, principalmente, la aplicación de los SFI a la aproximación de funciones, aunque los resultados se extrapolan en los últimos párrafos de manera directa a la compresión de imágenes.

[TAY 97] M. C. Taylor, "Frecuently asked questions about fractals". *FAQ de sci.fractals*, Sept. 1997. Disponible en `ftp://rtfm.mit.edu/pub/usenet/news.answers/sci/fractals-faq`.

La respuesta a las preguntas más habituales sobre fractales. Su valor estriba en el gran número de referencias bibliográficas y de Internet que aporta. Entre otros temas aborda los sistemas de funciones iteradas y la compresión fractal.

[WAL 91] G. K. Wallace, "The JPEG still picture compression standard". *Communications of the ACM*, April 1991. También disponible en `ftp://ftp.uu.net/graphics/jpeg/wallace.ps.gz`.

Aunque no pretende sustituir la necesidad de acceder a la especificación ISO completa a la hora de realizar una implementación, es una excelente introducción al estándar JPEG para compresión de imágenes. Además de las nociones básicas sobre la compresión mediante la transformada discreta del coseno, aborda detalladamente los cuatro modos de operación del estándar: codificación secuencial, progresiva, sin pérdidas y jerárquica.

Índice alfabético

Vocabulario bilingüe (inglés-español)

A

adherence
adherencia
admissibility constant
constante de admisibilidad
affine transformation
transformación afín
alphabet
alfabeto
arithmetic coding
codificación aritmética
attractor
atractor ,
axiom
axioma

B

baseline method
método base
bifurcation diagram
diagrama de bifurcación
bounded set
conjunto acotado
box counting theorem
teorema de recuento por cajas
brightness adjustment
ajuste de brillo

C

Cantor set
conjunto de Cantor
cardioid
cardioide
Cauchy sequence
sucesión de Cauchy
chaotic dynamical system
sistema dinámico caótico
chrominance
crominancia
closed ball
bola cerrada
cluster
grupo
clustering algorithm
algoritmo de agrupamiento
codebook
libro de códigos
collage theorem
teorema del collage
compact set
conjunto compacto
compactness
compacidad
complete metric space
espacio métrico completo ,
completeness
completitud
compression ratio
ratio de compresión
continuous mapping
aplicación continua
contractive map
aplicación contractiva
contractive mapping fixed-point theorem

teorema del punto fijo
contractivity factor
factor de contractividad
razón de contractividad
contrast adjustment
ajuste de contraste
convergent sequence
sucesión convergente

D

D0L system
sistema D0L
dequantization
descuantización
destination region
región de destino
discrete cosine transform
transformada discreta del coseno
discrete wavelet transform
transformada discreta con wavelets
distance
distancia
domain
dominio

E

entropy coding
codificación por entropía
escape time algorithm
algoritmo de tiempo de escape
eventually contractive map
aplicación eventualmente cnotractiva

F

Feigenbaum's constant
constante de Feigenbaum
Feigenbaum's number
constante de Feigenbaum
fixed point
punto fijo
fractal dimension
dimensión fractal
fractal imaging model
modelo de imagen fractal
fractal transform
transformada fractal

H

Hausdorff dimension
dimensión de Hausdorff
Hausdorff metric
distancia de Hausdorff
high pass filter
filtro de paso alto
hue
matiz
Huffman coding
codificación de Huffman

I

IFS
SFI, *véase* iterated function system
image distortion rate
ratio de distorsión de la imagen
image template
plantilla de la imagen
infimum
ínfimo
intensity component
componente de intensidad
interior
conjunto interior
invariant function
función invariante
inverse discrete cosine transform
transformada discreta del coseno inversa
iterated function system
sistema de funciones iteradas

J

Julia set
conjunto de Julia

K

Koch curve
curva de Koch

L

L system
sistema L
language
lenguaje
Lebesgue measure
medida de Lebesgue
losless compression
compresión sin pérdidas
lossy compression
compresión con pérdidas
low pass filter
filtro de paso bajo
luminance
luminancia

M

maximum
máximo
metric
métrica
metric space
espacio métrico
minimum
mínimo
mother wavelet
onda madre
multiresolution analysis
análisis multirresolución

O

open ball
bola abierta
open set condition
condición de abierto

P

partitioned iterated function system
sistema de funciones iteradas particionadas
peak-to-peak signal-to-noise ratio
razón señal-ruido máxima
period-dubling
duplicación de periodo
pixel chaining
encadenamiento de pixels
PSNR
RSRM, *véase* peak-to-peak signal-to-noise ratio

Q

quadtree partition
partición mediante árboles cuadriculares
quantization
cuantización
quantization step
cuanto

R

range
rango
rate-distorsion curves
curvas razón-distorsión
recurrent iterated function system
sistema de funciones iteradas recurrente
reference region
región de referencia
resolution
resolución
rms error

RECM, *véase* root mean square error

root mean square error
raíz del error cuadrático medio

S

saturation
saturación

scalar quantization
cuantización escalar

scale
escala

scale invariance
invarianza de escala

self-reference
autorreferencia

self-similarity
autosemejanza
autosimilitud

short time Fourier transform
transformada de Fourier a corto plazo

Sierpinski triangle
triángulo de Sierpinski

similitude
semejanza

spatial component
componente espacial

strange attractor
atractor extraño

string
cadena

subband coding
codificación subbanda

subsample
submuestreo

supremum
supremo

T

threshold
umbral

tiling
enlosado

topological dimension
dimensión topológica

U

uncertainty principle
principio de incertidumbre

upsample
supermuestreo

V

vector quantization
cuantización vectorial

VQ
CV, *véase* vector quantization