COMPRESIÓN FRACTAL DE IMÁGENES FIJAS Y SECUENCIAS DE IMÁGENES UTILIZANDO ALGORITMOS GENÉTICOS

LUCíA VENCES SALCEDO

Trabajo de Investigación

Para Obtener el Título de

Maestro en

CIENCIAS DE LA COMPUTACIÓN

Presentado a la Facultad del

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Mayo 1994

[redacted]

nín

Abstract

In this thesis we present a method to encode still images by finding an Iterated Function System (IFS) that describes an approximation to the image by using Genetic Algorithms (GA). With this method, the time needed to get an IFS is reduced in about a 50% less than with other techniques.

The genetic algorithm is used to find an IFS for a given image. The search is restricted to IFS with a fixed number of maps, and a contractivity factor also fixed.

To improve the convergence for this problem, it was used a genetic algorithm with diploid chromosomes.

Resumen

En este trabajo se presenta un método de compresión fractal imágenes fijas usando algoritmos genéticos. Con este procedimiento, el tiempo que se necesita para encontrar una codificación adecuada para una imagen dada se reduce hasta en un 50% en comparación con el tiempo que requieren otros métodos de compresión fractal de imágenes.

El algoritmo genético se usa para encontrar un Sistema de Funciones Iteradas que codifique una imagen dada. Para la solución de este problema, el espacio de búsqueda del algoritmo genético se restringe a los Sistemas de Funciones Iteradas con un factor de contracción fijo y con una cantidad también fija de transformaciones contractivas.

Para la solución de este problema, se usó un algoritmo genético con cromosomas diploides a fin de obtener una más rápida convergencia.

Contenido

Lista de Tablas

Lista de Figuras

Capítulo 1

Compresión de Imágenes

1.1 Introducción

Los volúmenes de información que se manejan hoy en día son enormes y cada vez más demandan mayor capacidad en dispositivos de almacenamiento y tiempo para su transmisión. Esto sucede especialmente cuando se trata de imágenes digitalizadas, sean fijas o secuencias de éstas, dada la reciente explosión de aplicaciones que hacen uso de ellas como las de multimedia, CD-ROM, sistemas de teleconferencias y video-telefonía, archivos de fotografías y video en computadora, televisión de alta definición, sistemas satelitales y de cable, discos láser digitales y cámaras electrónicas, entre otras.

En particular, cuando se trata de aplicaciones de video, es importante tener un acceso aleatorio rápido para la manipulación y procesamiento de una imagen, además de lograr una buena calidad cuando es transmitida o reproducida.

Una imagen digitalizada está representada por una matriz de valores, en la que un valor es una colección de parámetros que describen los atributos de un pixel en la imagen. Estos atributos pueden ser color, brillo, su vector normal, por ejemplo. Las dimensiones de esta matriz de pixeles y la información que debe tener cada uno de éstos dependen de la calidad de la imagen que se desea producir. Ya que el

ojo humano es capaz de procesar imágenes con grandes volúmenes de información, se necesitan muchos pixeles para guardar esas imágenes, aún cuando éstas sean en blanco y negro y de una resolución mediana.

El espacio de almacenamiento que se necesita para guardar las imágenes de 512×512 pixeles de un segundo de video es de 7.5Mb, suponiendo que cada pixel ocupa 8 bits de información y dado que en un segundo se presentan 30 cuadros. Por otro lado, el tiempo que se emplearía para transmitir las imágenes de un segundo de video es también excesivo. Este ejemplo hace ver la necesidad de buscar formas de comprimir eficientemente los datos que describen una imagen para almacenarlas y transmitirlas de forma eficaz.

Figura 1.1: Las imágenes digitalizadas requieren de gran espacio de almacenamiento además del tiempo que es necesario para transmitirse; sin embargo, los datos que contiene una imagen, en general, son muy redundantes. Debido a esta redundancia, es posible aplicar una compresión.

Además, la cantidad de espacio de almacenamiento y de tiempo de transmisión requeridos por estas imágenes incide directamente en su costo. Es por ésto que la compresión de imágenes, y de información en general, está siendo de relevancia actualmente para el desarrollo de esta amplia gama de aplicaciones.

La compresión de imágenes es posible debido a que la mayoría de los datos son redundantes (como se puede observar en la figura 1.1) y se pueden quitar para

comprimir y volverlos a obtener al recuperar la imagen. Otro factor importante para la compresión de imágenes es el hecho de que el ojo humano no es lo suficientemente sensible a cierta pérdida de información.

Entre los métodos de compresión de imágenes se encuentra el de compresión fractal que fue desarrollado originalmente por Michael F. Barnsley [Bar88, BaSl88, BaHu92, HGB92] basado en la teoría de funciones iteradas de la geometría fractal. Este método presenta, entre otras ventajas, razones de compresión bastante buenas (entre 10:1 y 80:1) y que la imagen descodificada puede generarse a cualquier escala creando el detalle adicional que el original no tiene. Esto significa que la imagen descodificada realmente no tiene un tamaño natural ya que no importa a qué escala se descomprima. Sin embargo, tiene como desventaja que requiere mucho tiempo de cómputo para obtener una codificación adecuada, aún cuando la decodificación es relativamente rápida.

En imágenes fijas, podemos aplicar una compresión explotando la redundancia en los datos. A esta redundancia se le conoce como coherencia o correlación espacial. En una secuencia de imágenes, además de la correlación espacial, tenemos correlación temporal entre un cuadro y el cuadro inmediato siguiente; ésto quiere decir que, las diferencias entre un cuadro y el inmediato siguiente son muy pequeñas.

1.2 Almacenamiento de Imágenes

Cuando almacenamos una imagen, en principio, guardamos un arreglo bidimensional en el que cada valor representa datos asociados a un pixel en la imagen. Este valor puede ser binario en el caso de imágenes en blanco y negro o una colección de tres números representando las intensidades de rojo, verde y azul para cada pixel en una imagen a color.

Además, un pixel puede tener otra información asociada como un valor de α-buffer [1], un valor de z-buffer [2] o una tripleta de números indicando la normal a la superficie dibujada en ese pixel. Por lo tanto, una imagen consiste en una colección de canales, cada uno de los cuales tiene asociada cierta información específica sobre los pixeles de la imagen.

Por razones prácticas, normalmente, cada uno de los canales se guardan separadamente al representar la imagen. Sin embargo, existen algunos métodos de compresión para los que resulta más conveniente considerar la imagen como una matriz de datos. Tal es el caso de los esquemas de funciones iteradas.

1.3 Métodos de Compresión sin Pérdida de Información

Como hemos mencionado, la información sobre una imagen se almacena en distintos canales de datos. Si al presentar una imagen, se espera que ésta se encuentre descrita sólo en determinados canales, lo mejor es que la imagen sea codificada en cada uno de ellos. Generalmente, ésto significa asignar mayor espacio de almacenamiento a la imagen, por lo que, normalmente se busca alguna manera de comprimir toda esta información.

Cuando una imagen tiene pocos colores y éstos se repiten constantemente, puede crearse una tabla con ellos y usar un solo canal que sea un índice de ella. Sin embargo, si la imagen tiene demasiados colores realmente no se logra gran compresión en los datos. Para que verdaderamente sea útil crear una tabla, la cantidad de colores debe ser menor a la mitad de pixeles que requiere la imagen.

Si la imagen presenta gran redundancia, entonces existe otro método de compresión que podemos aplicar además del anterior. Este procedimiento, conocido como *run-length encoding*, consiste en sustituir una secuencia repetida de valores

[1] El valor de α-buffer está asociado a la transparencia del pixel.
[2] Un valor de z-buffer representa la profundidad del pixel en la imagen.

por dos parámetros: el primero indicando la cantidad de veces que se repite el segundo. Esta es una técnica muy sencilla que, aunque depende de los datos mismos, puede reducir el tamaño de un archivo entre un 50% y 75%.

1.3.1 Codificación de Huffman

Esta codificación parte del hecho de que, si cada símbolo s_i tiene una probabilidad de ocurrencia P_i, el contenido de información medio por símbolo es

$$-\sum_{i=1}^{N} P_i \log_2 P_i$$

donde N es el número de símbolos en el alfabeto. Esta cantidad se le conoce como entropía por símbolo.

El algoritmo de Huffman hace una lista de los símbolos de entrada ordenados ascendentemente de acuerdo a la probabilidad de ocurrencia que tiene cada uno de ellos. En seguida, toma los dos símbolos con menor probabilidad en la lista y con ellos forma un árbol de dos ramas, una con etiqueta 0 y la otra con etiqueta 1. Este árbol toma como valor de probabilidad la suma de probabilidades de sus hojas y, de acuerdo con este valor, se inserta en la lista en el lugar que le corresponda. Después, se vuelve a examinar la lista para formar un nuevo árbol de dos ramas con los dos miembros de menor probabilidad que se etiquetarán con 0 y 1 respectivamente. Éste último paso se repite hasta que se tenga construido el árbol completo, es decir, hasta agotar la lista.

En la figura 1.2 puede verse un ejemplo de la construcción de un árbol de Huffman, que corresponde al código

$$\begin{aligned} A &\rightarrow 00 \\ B &\rightarrow 0100 \\ C &\rightarrow 0101 \\ D &\rightarrow 011 \\ E &\rightarrow 1 \end{aligned}$$

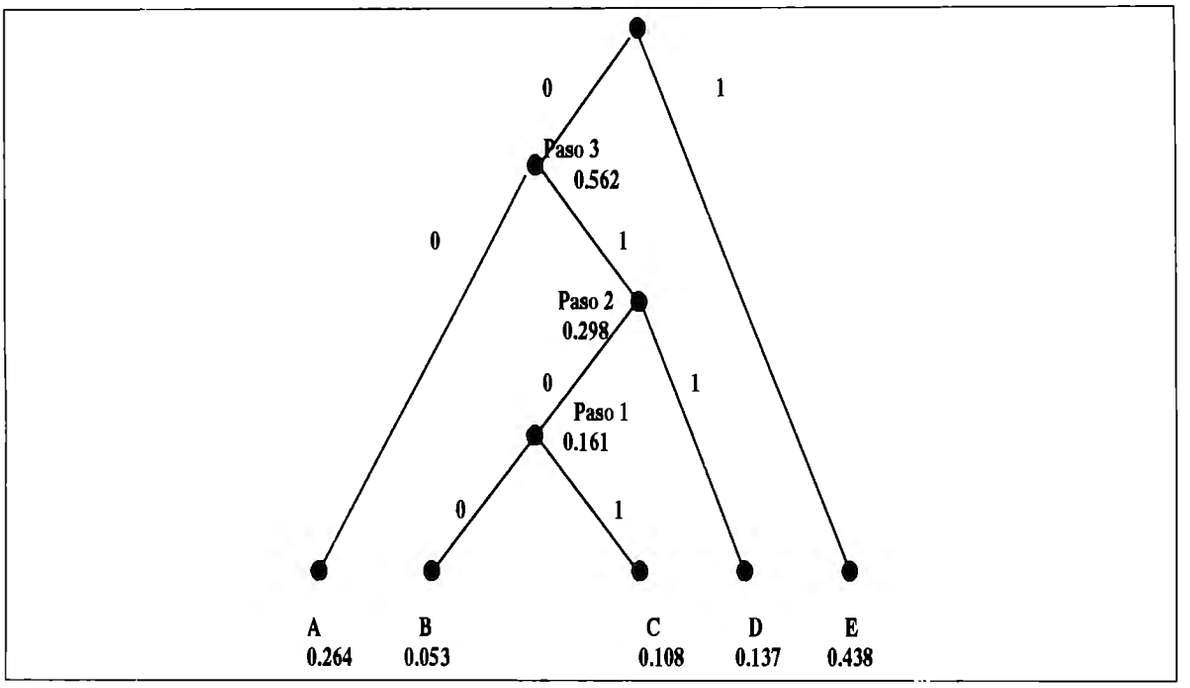

Figura 1.2: Ejemplo de la construcción de un árbol de Huffman.

1.4 Métodos de Compresión con Pérdida de Información

Los algoritmos de *run-length encoding* y de Huffman tienen la característica de que la información reconstruida es idéntica a la original; es decir, que no pierden en absoluto información, por lo que las razones de compresión que logran en imágenes no son muy grandes (en general, no son superiores a 3:1). Estas técnicas sólo se usan en aplicaciones en las que la fidelidad es de mayor importancia y son muy sensitivas a variaciones como las imágenes médicas.

Por el contrario, en aplicaciones comerciales e industriales en las que es más importante ahorrar espacio de almacenamiento, tiempo de transmisión y ancho de banda en comunicaciones, se prefieren algoritmos de compresión que aprovechen las debilidades de la visión humana. El ojo humano es más receptivo al detalle fino en señales de luminancia que en las señales de color, por lo que algunos sistemas muestrean las señales de luminancia a una mayor resolución espacial y se le asignan

más bits a éstas que a las señales de crominancia. También, el ojo es menos sensible a la energía con alta frecuencia espacial que a la que tiene baja frecuencia. Esta deficiencia se aprovecha codificando los coeficientes de alta frecuencia con menor cantidad de bits que los coeficientes de baja frecuencia. Estas técnicas, por lo tanto, logran mayores proporciones de compresión en imágenes que las anteriores.

1.4.1 JPEG

Para desarrollar estándares de compresión de imágenes, en 1987 las organizaciones *International Standards Organization* (ISO) e *International Telegraph and Telephone Consultative Committe* (CCITT) reunieron a un grupo de expertos en codificación de imágenes que se llamó *Joint Photographic Experts Group* (JPEG). El objetivo de este grupo consistía en establecer un algoritmo que favoreciera la mayor variedad de servicios de comunicación de imágenes como fuera posible con un solo estándar.

La estructura del algoritmo JPEG tiene tres partes [Wal89, ARA91] : (1) un sistema que usa la información de la imagen ordenada en líneas, (2) un conjunto de opciones que extienden las capacidades del sistema anterior, y (3) la posibilidad de codificar sin pérdida de información de manera independiente de las anteriores.

El sistema basado en líneas es la parte más simple del algoritmo propuesto por JPEG. Este sistema consta de técnicas que son muy conocidas como la Transformada Discreta del Coseno, la cuantización uniforme y la codificación de Huffman. Además, hace uso también de algoritmos que proporcionan mayor compresión en imágenes. El sistema está construido de tal forma que permite que una imagen codificada sea transmitida secuencialmente, y sus pixeles sean reconstruidos por el decodificador siguiendo el orden de un cuadriculado.

El sistema extendido proporciona un conjunto de posibilidades adicionales a las del sistema basado en líneas, que pueden ser utilizadas de forma individual o combinándolas. En estos están incluidas las codificaciones aritméticas [3].

[3]Las codificaciones aritméticas mapean una secuencia de símbolos del alfabeto de entrada a un punto del intervalo unitario

El algoritmo de compresión propuesto por JPEG (que se ilustra en la figura 1.3) consiste primeramente en dividir la imagen en bloques no superpuestos de 8×8 pixeles. A cada uno de estos bloques se le aplica la transformada discreta del coseno, de donde se obtienen 64 coeficientes representan el contenido de frecuencia del bloque. El primero de ellos mide la energía de la frecuencia cero o el término de corriente directa (dc). Los 63 restantes son los coeficientes de corriente alterna. Estos coeficientes son cuantizados a un conjunto finito de valores, ordenados en un arreglo unidimensional siguiendo un camino en zigzag (como se ilustra en la figura 1.4) y codificados usando el método de Huffman o una representación aritmética.

1.4.2 La Transformada Discreta del Coseno

La transformada discreta del coseno (DCT) es una variante de la transformada de Fourier que toma sólo la parte real de los datos [Pet92]. Esta transformada toma un bloque de $n \times n$ pixeles como un vector en un espacio de n^2 dimensiones. Cada uno de estos bloques se transforma de manera separada. Los n^2 valores de los pixeles s_{xy} son transformados a través de la transformada directa discreta del coseno (FDCT) en n^2 nuevas coordenadas S_{uv}, de manera que,

$$S_{uv} = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} s_{xy} \cos \frac{(2x+1)u\pi}{16} \cos \frac{2y+1)v\pi}{16},$$

$$0 \leq u, v \leq 7.$$

Los valores s_{xy} pueden ser recuperados aplicando la transformada inversa (IDCT).

$$s_{xy} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)S_{uv} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16},$$

$$0 \leq x, y \leq 7.$$

1.4.3 Descomposición de Paquetes de Onda

Una técnica de compresión de imágenes que se ha valido de la relación funcional entre la intensidad y la frecuencia de las ondas de luz en la percepción del ojo humano es

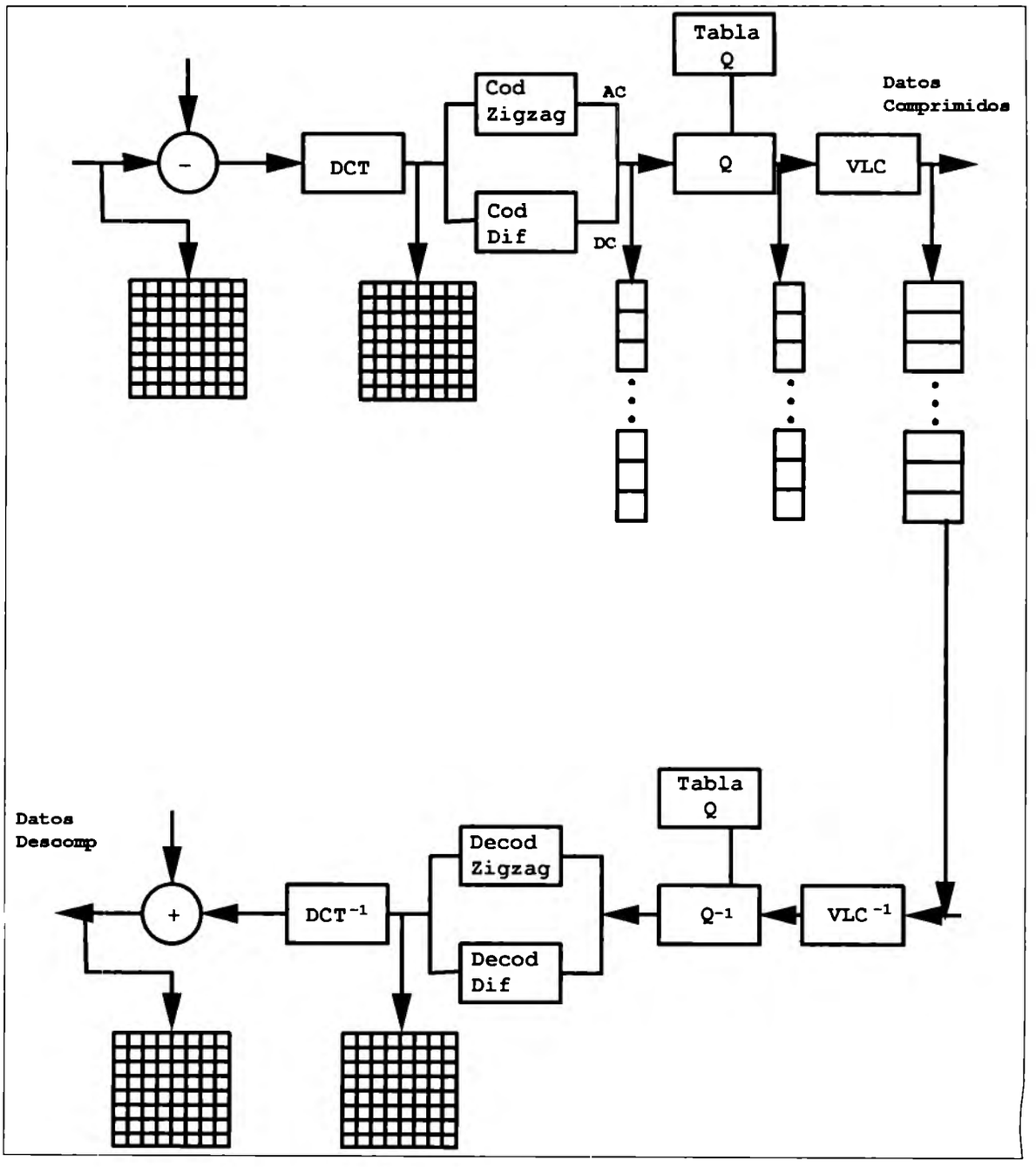

Figura 1.3: Codificación y decodificación propuesta por JPEG.

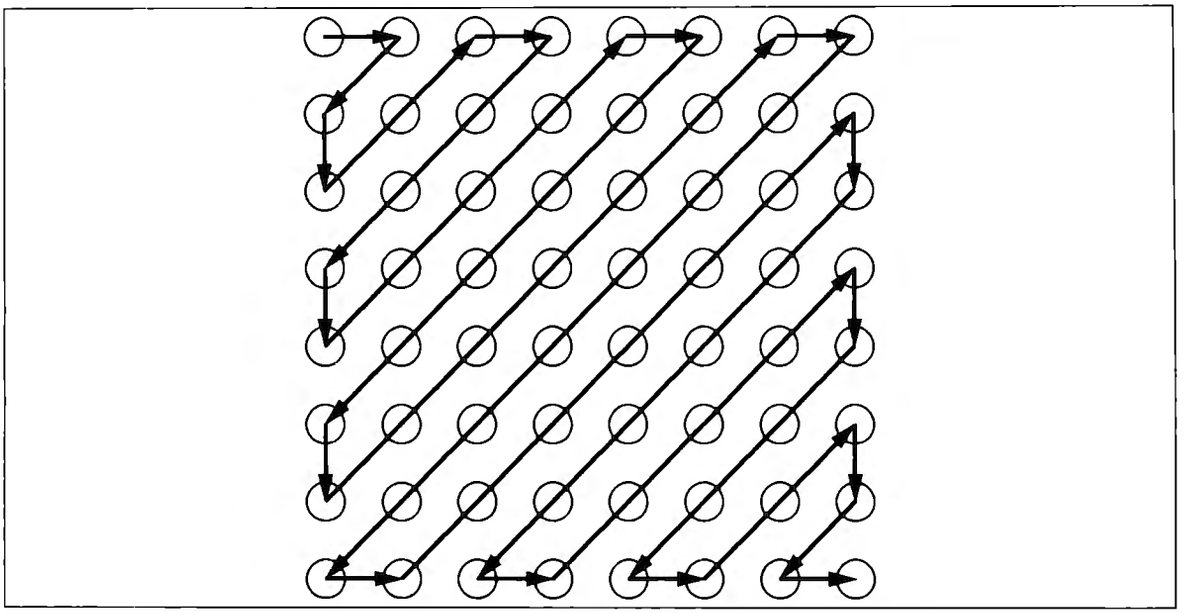

Figura 1.4: Los coeficientes resultantes de la DCT son ordenados en zigzag en la compresión que propone JPEG.

la descomposición de paquetes de onda (wavelets) [DJL91, LK90].

La descomposición de paquetes de onda de una función f definida en $\Re^n$ está dada por

$$f = \sum_{k \geq 0, j=(j_1,\ldots,j_n) \in Z^n} c_{j,k}\phi_{j,k}, \tag{1.1}$$

donde los coeficientes $c_{i,j}$ dependen de la función f y las funciones

$$\phi_{j,k}(x) = \phi(2^k(x - j/2^k))$$

son dilataciones y traslaciones de una función ϕ llamada *wavelet.*

Dada una descomposición en paquetes de onda de una función f, que es la que se quiere comprimir (como en la ecuación 1.1), esta técnica toma los coeficientes $c_{j,k}$ y, con ellos, calcula los coeficientes cuantizados $\tilde{c}_{j,k}$. Entonces, la función comprimida toma la siguiente forma:

$$\tilde{f} = \sum \tilde{c}_{j,k}\phi_{j,k} \tag{1.2}$$

Los coeficientes $\tilde{c}_{j,k}$ se escogen de manera que, dado un $\epsilon > 0$, se satisfaga la condición,

$$\|(c_{j,k} - \tilde{c}_{j,k})\phi_{j,k}\|_{\mathcal{L}^2} \leq \epsilon \tag{1.3}$$

donde

$$\|\mu, \hat{\mu}\|_{\mathcal{L}^2} = \sum_{0 \leq i,j \leq B} (\mu_{ij} - \hat{\mu}_{ij})^2.$$

1.4.4 Programación Genética

Por otro lado, Koza [Koz92] propuso usar programación genética para convertir imágenes a color en expresiones simbólicas de Lisp. De esta manera, demostró que imágenes a color con una gran cantidad de pixeles puede ser representada en esta forma y transmitida usando un ancho de banda pequeño.

La programación genética es una técnica relativamente reciente que ha demostrado ser una herramienta efectiva en la solución de algunas aplicaciones concretas de Generación Automática de Programas. La programación genética sigue el camino de los algoritmos genéticos (los cuales describiremos en el capítulo 3) para buscar en un espacio de posibles programas el que mejor puede realizar una tarea dada.

Para la programación genética, un programa se representa como un árbol de parseo cuyos nodos son procedimientos, funciones, variables y constantes. Los subárboles de un nodo representan los argumentos del procedimiento o funciones de éstos. Ya que en la sintaxis del lenguaje Lisp, una subrutina se representa básicamente por su árbol de parseo, este lenguaje facilita la representación de estos árboles como expresiones simbólicas.

En la programación genética, el operador básico es el de cruzamiento. Este operador trabaja de la siguiente forma: Primero, selecciona aleatoriamente un nodo de cada uno de los dos árboles sobre los que va a operar. Entonces, toma, del primer árbol, el subárbol que tiene como raíz el nodo seleccionado y lo reemplaza en el segundo árbol por el subárbol con raíz en su nodo seleccionado. Este proceso se ilustra en la figura 1.5.

Para la compresión de imágenes usando programación genética, Koza realiza una *regresión simbólica* sobre un arreglo bidimensional de datos que representan el color de los pixeles de la imagen en cuestión. El objetivo de usar la regresión simbólica es encontrar un programa, o expresión simbólica de Lisp, que represente una imagen dada, de forma exacta o aproximada; de manera que, esta representación resulte ser más compacta que la imagen original.

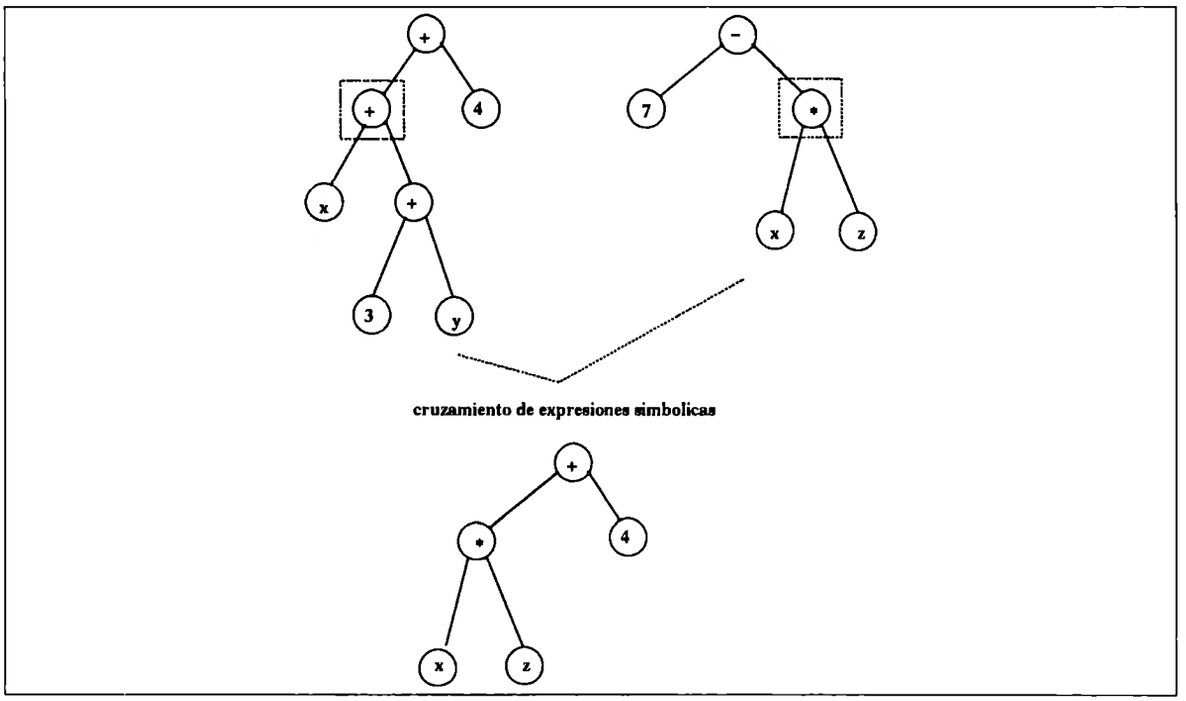

Figura 1.5: Operador de cruzamiento de árboles de expresiones simbólicas.

1.4.5 Compresión de Secuencias de Imágenes

Los métodos de compresión de secuencias de imágenes explotan las correlaciones espacial y temporal que existen entre cuadros consecutivos, como se ilustra en la figura 1.7. La compresión en el espacio es conocida como *intra-cuadro*, mientras que la compresión en el tiempo es llamada *inter-cuadro*.

Por otro lado, el grupo MPEG presentó otro algoritmo para compresión de video. El grupo *Moving Picture Expert Group* (MPEG) es parte de la organización ISO-TEC JTC1/SC2/WG11 [ARA91, LeG91]. Esta organización es responsable de la

estandarización de sistemas de audio y video.

En esta técnica, la compresión espacial se realiza usando la DCT sobre cuadros de 8×8 pixeles y aplicándoles una cuantización a los coeficientes resultantes (como se hace en JPEG, sección 1.4.1).

Para lograr una compresión inter-cuadro, se escoge un paraámetro M, que indica que uno de cada M cuadros será comprimido espacialmente y los cuadros restantes serán obtenidos mediante una interpolación. Este parámetro M se escoge de acuerdo a la calidad y proporción de compresión que se desee obtener.

Para videotelefonía, CCITT propuso la técnica híbrida H.261 que combina las codificaciones que aplican transformaciones, como la DCT, y una codificación predictiva; ésto es, una codificación en la que un bloque del cuadro actual se predice por un bloque del cuadro anterior.

Esta técnica se encuentra ilustrada en la figura 1.6. El algoritmo comienza aplicando una compresión intra-cuadro al primero de una secuencia. Como se hace en JPEG, a cada bloque de 8×8 pixeles se le aplica la transformada discreta del coseno y se cuantizan sus coeficientes. En este punto, se tienen dos caminos: en el primero, a éstos últimos coeficientes se les aplica una codificación en la que no se pierde información y una corrección de error. En la segunda alternativa, se aplica una transformación inversa de la cuantización y de la DCT, para recuperar la información anterior y poder aplicar una compresión inter-cuadro. Esta compresión inter-cuadro usa una codificación predictiva.

Estas dos técnicas de compresión de video son usadas ampliamente en la actualidad ya que la calidad que logran, su costo y las características de sus algoritmos han hecho posible que sean implementadas en hardware.

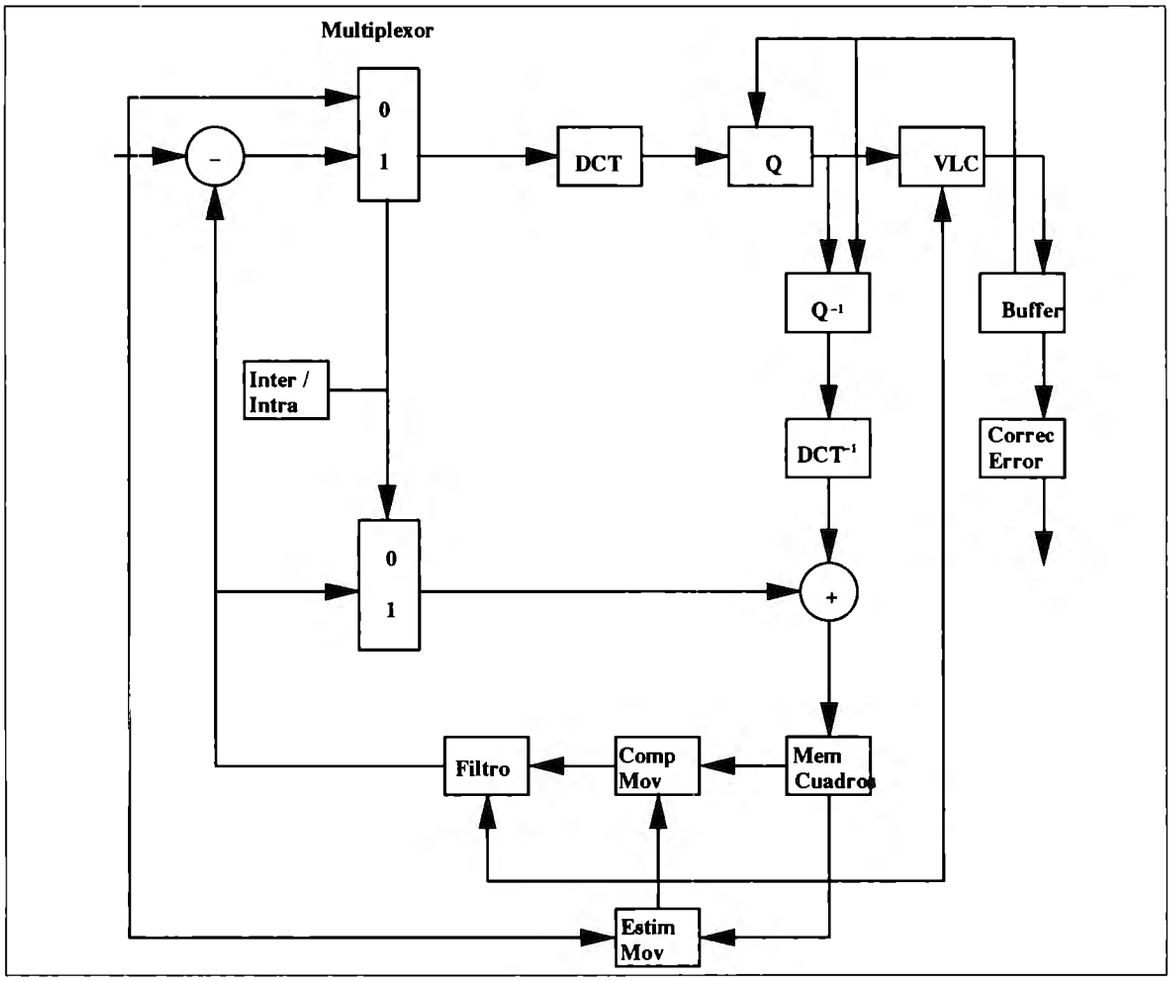

Figura 1.6: Codificación de video propuesta por CCITT.

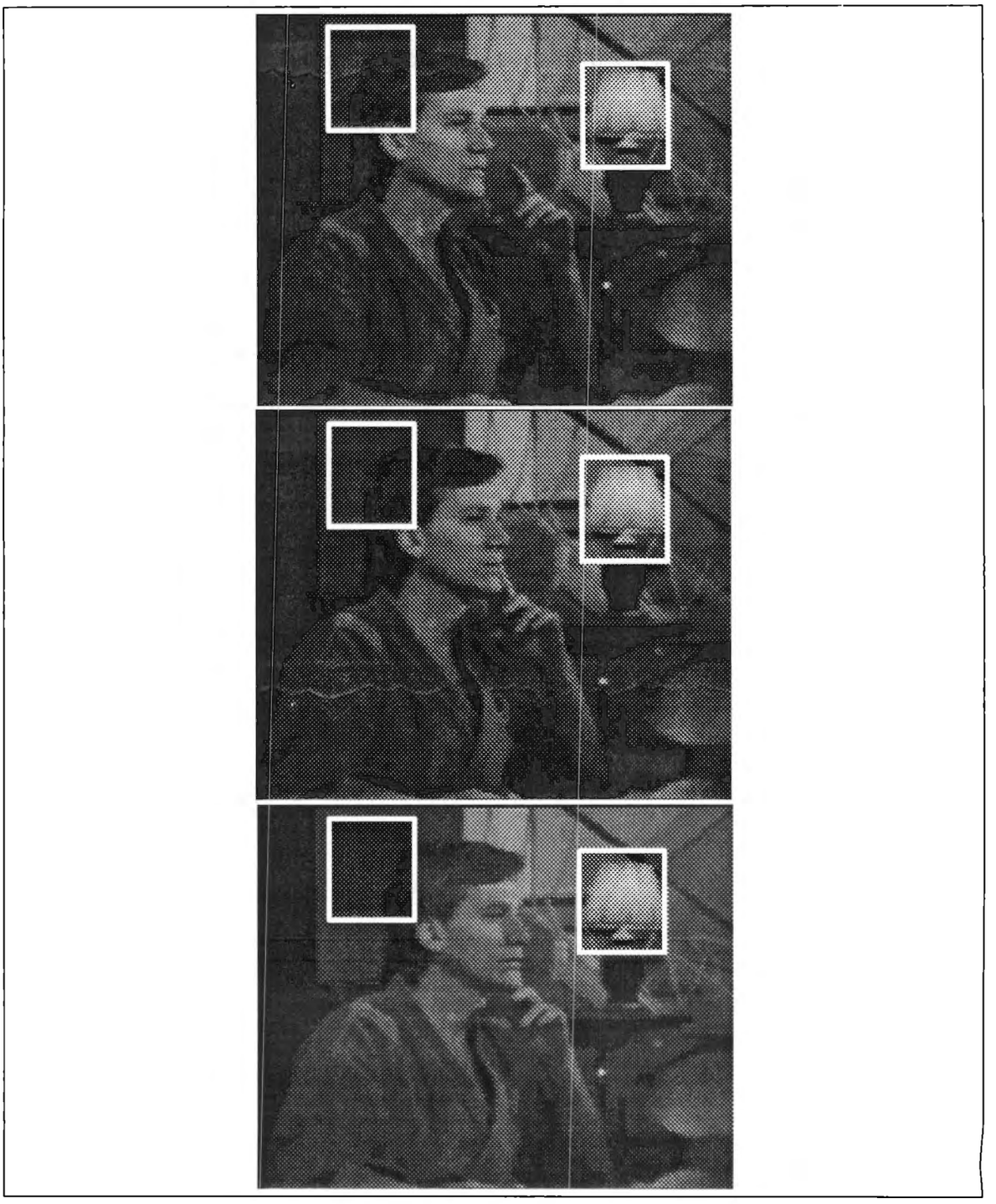

Figura 1.7: Tres cuadros consecutivos de un video. En los tres cuadros, la pantalla de la lámpara permanece igual, mientras que la cabeza de la persona se va moviendo.

Capítulo 2

Compresión Fractal de Imágenes

2.1 Conjuntos Fractales

En el pasado, las matemáticas se habían concentrado en conjuntos y funciones para los que el cálculo podía ser aplicado. Conjuntos y funciones *irregulares* no se consideraban de interés. Sin embargo, esta actitud ha cambiado recientemente, y se considera que estos conjuntos y funciones irregulares describen mejor los fenómenos que se encuentran en la naturaleza.

El término *fractal* fue usado originalmente por Benoit B. Mandelbrot en 1977 [Man93] para una geometría que describe patrones irregulares, fragmentados, *más cercanos a las formas de la naturaleza;* de manera que la geometría fractal proporcionó una herramienta para el estudio de tales figuras.

Mandelbrot propuso la siguiente definición tentativa de un fractal:

Un fractal es por definición un conjunto para el que su dimensión de Hausdorff-Besicovitch excede estrictamente a su dimensión topológica [Man93].

En este contexto, la dimensión topológica es siempre entera. (La dimensión de Hausdorff-Besicovitch se definirá más adelante, en la definición 2.4.)

En realidad, esta definición, aunque es formal, no es completa, pues excluye muchos fractales que se presentan en fenómenos físicos.

Un ejemplo clásico de fractal es el conjunto del tercio medio de Cantor, cuya construcción se ilustra en la figura 2.1. Este conjunto se forma tomando como base el intervalo unitario al que se le aplica un serie de sustracciones. Sea E_0 el intervalo $[0,1]$. Sea E_1 el conjunto E_0 al que se le ha quitado el tercio medio; ésto es, $E_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$. Para obtener el conjunto E_2, cada uno de los intervalos que forman E_1 se le quita su tercio medio, de manera que $E_2 = [0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{1}{3}] \cup [\frac{2}{3}, \frac{7}{9}] \cup [\frac{8}{9}, 1]$. Siguiendo de esta forma, obtenemos E_k al restar a cada intervalo de E_{k-1} su tercio medio. Por lo tanto, E_k consistirá de 2^k intervalos de longitud 3^{-k}. El conjunto del tercio medio de Cantor F, matemáticamente queda descrito por:

$$F = \bigcap_{k=0}^{\infty} E_k.$$

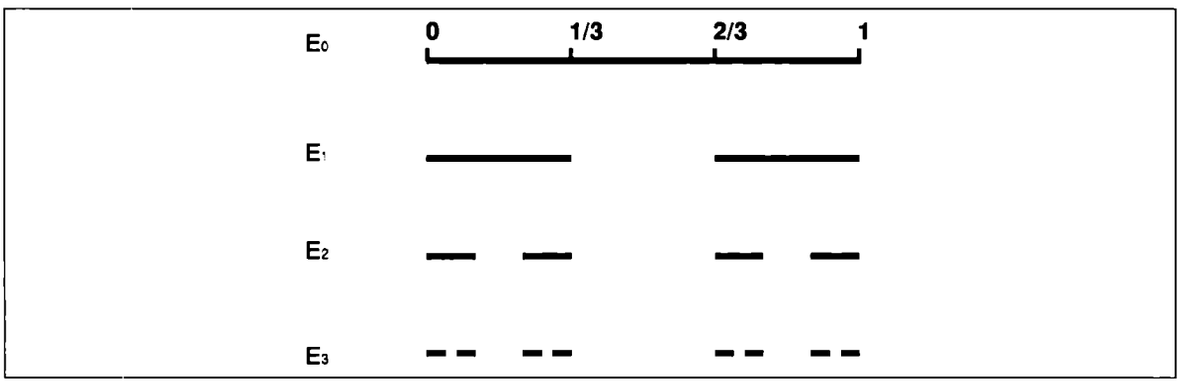

Figura 2.1: Construcción del conjunto de tercio medio de Cantor

Entre las características que podemos identificar en este conjunto y que son comunes en muchos fractales [Fal90b], se encuentran:

1. F es auto similar. Es claro que F consiste de copias de él mismo a diferentes escalas. Por ejemplo, los intervalos de E_1 son iguales a los intervalos de E_0 con un factor de escala de $\frac{1}{3}$.

2. F contiene detalle a cualquier escala. Esta propiedad se le conoce como *estructura fina*; entre mayor sea la escala a la que se examine, el conjunto aparentará tener más espacios vacíos.

3. Aún cuando la estructura de F es compleja, su descripción es muy sencilla.

4. F se obtiene mediante la aplicación de un procedimiento recursivo.

5. La geometría de F no puede describirse en términos de la geometría clásica; ésto es, no es el lugar geométrico de un conjunto de puntos que son solución de una ecuación.

6. No es sencillo describir el lugar geométrico de F.

7. A pesar de ser un conjunto infinito no numerable, el tamaño de F no se determina mediante las medidas usuales como longitud.

Un segundo ejemplo es la curva de von Koch. Si E_0 un segmento de línea de longitud unitaria, sea E_1 los cuatro segmentos que se obtienen al quitar el tercio medio de E_0 y reemplazarlo por otros dos del triángulo equilátero que tendría como base el segmento que se quitó. E_2 se construye aplicando a cada uno de sus segmentos, el mismo procedimiento que se empleó en E_1. Entonces, E_k se obtiene de reemplazar el tercio medio de cada segmento de línea recta de E_{k-1} por los otros dos lados del triángulo equilátero.

Cuando k es suficientemente grande, las curvas E_{k-1} y E_k sólo difieren en el detalle fino. Al tender k a infinito, la sucesión de curvas E_K se aproxima a su curva límite F, que se conoce como *curva de von Koch.*

Esta curva tiene varias características similares a las enunciadas para el conjunto de Cantor. Aún cuando parece razonable llamar a F una curva, es demasiado irregular como para tener tangentes en el sentido clásico. Por otro lado, es fácil ver que la longitud de la E_k es $\left(\frac{4}{3}\right)^k$, que tiende a infinito cuando k tiende a infinito; lo que significa que F tiene longitud infinita. Además, el área que ocupa F en el plano es cero, por lo que ni la longitud de F ni su área son descripciones útiles del *tamaño* de F. La dimensión de Hausdorff-Besicovitch nos da una idea del tamaño de un conjunto fractal.

Para comprender el término *conjunto fractal* es necesario definir primeramente lo que es una *dimensión fractal.* En realidad, hay distintas dimensiones fractales; sin embargo, la definición de Hausdorff es la más antigua y, probablemente, la más importante.

Definición 2.1 *Sea $U \subseteq \Re^n$ un conjunto no vacío. El diámetro de U se define por*

$$|U| = \sup\{|x - y| : x, y \in U\}.$$

Ésto quiere decir que el diámetro $|U|$ de un conjunto U es la distancia (euclidiana) más grande que hay entre cualquier par de puntos del conjunto.

Definición 2.2 *Sea $F \subset \Re^n$ y $\{U_i \subset \Re^n\}$ una colección contable de conjuntos tales que para cada i*

$$0 < |U_i| \leq \delta$$

y

$$F \subset \bigcup_{i=1}^{\infty} U_i.$$

Entonces decimos que $\{U_i\}$ es una δ-cubierta de F.

Definición 2.3 *Supongamos que $F \subseteq \Re^n$ y c es un número no negativo. Para cualquier $\delta > 0$, se define $\mathcal{H}^c_\delta(F)$ como,*

$$\mathcal{H}^c_\delta(F) = \inf\{\sum_{i=1}^{\infty} |U_i|^c\}$$

donde el ínfimo es sobre todas las familias $\{U_i\}$ contables de δ-cubiertas de F. Además,

$$\mathcal{H}^c(F) = \lim_{\delta \to 0} \mathcal{H}^c_\delta(F).$$

$\mathcal{H}^c(F)$ se conoce como la medida de Hausdorff c-dimensional *de F.*

Definición 2.4 *La* dimensión de Hausdorff-Besicovitch *se define entonces como*

$$dim_H F = \inf\{c : \mathcal{H}^c(F) = 0\} = \sup\{c : \mathcal{H}^c(F) = \infty\}$$

de manera que

$$\mathcal{H}^c(F) = \begin{cases} \infty & c < dim_H F \\ 0 & c > dim_H F \end{cases}$$

Si $c = dim_H F$, entonces $\mathcal{H}^c(F)$ puede ser 0 o infinito, o

$$0 < \mathcal{H}^c(F) < \infty.$$

Para el conjunto del tercio medio de Cantor, $c = \log 2/\log 3 = 0.6309\ldots$ entonces $dim_H F = c$ y $\frac{1}{2} \leq \mathcal{H}^c(F) \leq 1$.

2.2 Sistemas de Funciones Iteradas

Muchos conjuntos fractales se construyen de conjuntos que, en cierto sentido, son similares al conjunto en su totalidad. Por ejemplo, el conjunto de Cantor E es la unión de dos conjuntos similares al conjunto mismo. Los conjuntos $E \cap [0, \frac{1}{3}]$ y $E \cap [\frac{2}{3}, 1]$ son similares al conjunto E con un factor de escala de $\frac{1}{3}$; $E \cap [0, \frac{1}{9}]$, $E \cap [\frac{2}{9}, \frac{1}{3}]$, $E \cap [\frac{2}{3}, \frac{7}{9}]$ y $E \cap [\frac{8}{9}, 1]$ son conjuntos similares a E con un factor de escala de $\frac{1}{9}$.

La autosimilaridad de un conjunto no es una propiedad exclusiva de los fractales, sin embargo se usa con frecuencia para su definición [Fal90]. El trabajo de Hutchinson [Hut81], analizó formalmente, por primera vez, esta propiedad en conjuntos fractales. Nosotros comenzamos este estudio con una serie de definiciones, para en seguida enunciar el *teorema del Collage*, que es fundamental para la compresión fractal de imágenes.

Definición 2.5 *Sea $\mathcal{S}$ la clase de todos los subconjuntos compactos no vacíos de $D \subset \Re^n$. Definimos el δ-cuerpo paralelo de $A \subseteq D$, A_δ, como*

$$A_\delta = \{x \in D : |x - a| \leq \delta, a \in A\}.$$

Definición 2.6 *Una transformación afín* $W : \Re^n \to \Re^n$ *es una transformación de la forma* $W(x) = T(x) + b$ *donde* T *es una transformación lineal sobre* $\Re^n$ *y* b *es un vector en* $\Re^n$.

Es claro que una transformación afín combina traslación, rotación, dilatación (o contracción) y reflexión. Un ejemplo de una transformación afín se ilustra en la figura 2.2.

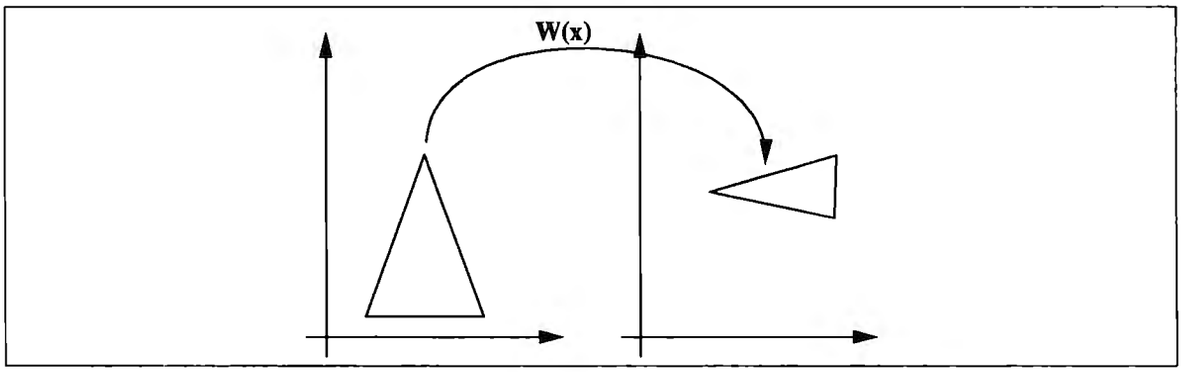

Figura 2.2: Ejemplo de una transformación afín.

Definición 2.7 *Se dice que una transformación afín* $W : D \to D, D \subseteq \Re^n$, *es una contracción sobre* D *si, para una medida* d *de* $\Re^n$, *existe un número* $c, 0 < c < 1$, *tal que*

$$d(W(x), W(y)) \leq c \cdot d(x, y)$$

para cualesquiera $x, y \in D$. *Si*

$$d(W(x), W(y)) = c \cdot d(x, y),$$

entonces llamamos a W *una similaridad.*

Definición 2.8 *Una familia* $\{\omega_1, \ldots, \omega_m\}$ *de contracciones se le llama Sistema o Esquema de Funciones Iteradas (IFS).*

Definición 2.9 *Si un subconjunto* F *de* D *es tal que para un IFS*

$$\{\omega_1, \ldots, \omega_m\},$$

$$F = \bigcup_{i=1}^{m} \omega_i(F), \tag{2.1}$$

se dice que F es invariante para ese sistema. Si F es invariante bajo una colección de similaridades, entonces se dice que F es un conjunto auto similar.

Un resultado interesante de la geometría fractal es el hecho de que, dado un esquema de funciones iteradas $\{\omega_1, \ldots, \omega_m\}$, existe un único conjunto F tal que, la ecuación 2.1 se cumple.

A esta F se le conoce como el atractor del sistema. Además, si E es un subconjunto compacto no vacío tal que $\omega_i(E) \subset E$ y definimos la transformación

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

y escribimos la k-ésima iteración de W como $W^k(E)$, de manera que

$$W^0(E) = E,$$

$$W^k(E) = W(W^{k-1}(E))$$

para $k \geq 1$. Entonces tenemos que

$$F = \bigcap_{i=1}^{\infty} W^k(E).$$

Formalmente, esta afirmación se enuncia de la siguiente manera.

Teorema 2.1 *Sean $\{\omega_1, \ldots, \omega_m\}$ contracciones sobre $D \subset \Re^n$ tales que*

$$|\omega_i(x) - \omega_i(y)| \leq c_i|x - y|, \quad \forall x, y \in D$$

y $c_i < 1$ para cada i. Entonces existe un único conjunto no vacío compacto F que es invariante a la familia de contracciones $\{\omega_i\}$, ésto es, F satisface que

$$F = \bigcup_{i=1}^{m} \omega_i(F). \tag{2.2}$$

Aún más, si definimos una transformación W sobre la clase $\mathcal{W}$ de conjuntos compactos no vacíos mediante

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

y escribimos W^k como la k-ésima iteración de W, dada por

$$W^0(E) = E,$$

$$W^k(E) = W(W^{k-1}(E)), \quad k > 1,$$

entonces

$$F = \bigcap_{k=1}^{\infty} W^k(E) \tag{2.3}$$

para cualquier conjunto $E \in \mathcal{W}$ tal que $\omega_i(E) \subset E$ para cada i.

La demostración de este teorema puede verse en el apéndice, A teorema A.1.

De hecho, lo que afirma este teorema es que la sucesión $W^k(E)$ converge al atractor del sistema para un conjunto E arbitrario, como el ejemplo que muestran las figuras 2.3 y 2.4. Este es un hecho interesante dado que, una familia pequeña de contracciones puede determinar imágenes visualmente muy complicadas. Ésto sugiere que, si una imagen puede ser representada por un IFS, entonces puede ser almacenada y transmitida de una manera muy eficiente. Es suficiente con almacenar y/o transmitir el IFS de la imagen para poder reproducirla.

Dado un IFS, computacionalmente resulta muy sencillo obtener la imagen codificada por este esquema. La dificultad principal de esta técnica es el problema inverso que consiste en encontrar un IFS que caracterice una imagen dada.

Definición 2.10 *La* métrica o distancia de Hausdorff *d, se define sobre una colección de conjuntos compactos, no vacíos de $\Re^n$, como*

$$d(E, F) = \inf\{\delta | E \subset F_\delta \wedge F \subset E_\delta\}$$

para $E, F \in \Re^n$.

El teorema 2.1 podría expresarse, usando la distancia de Hausdorff, como que $W^k(E)$ converge a F para cualquier conjunto inicial $E \in \mathcal{W}$, de forma que

$$d(W^k(E), F) \to 0$$

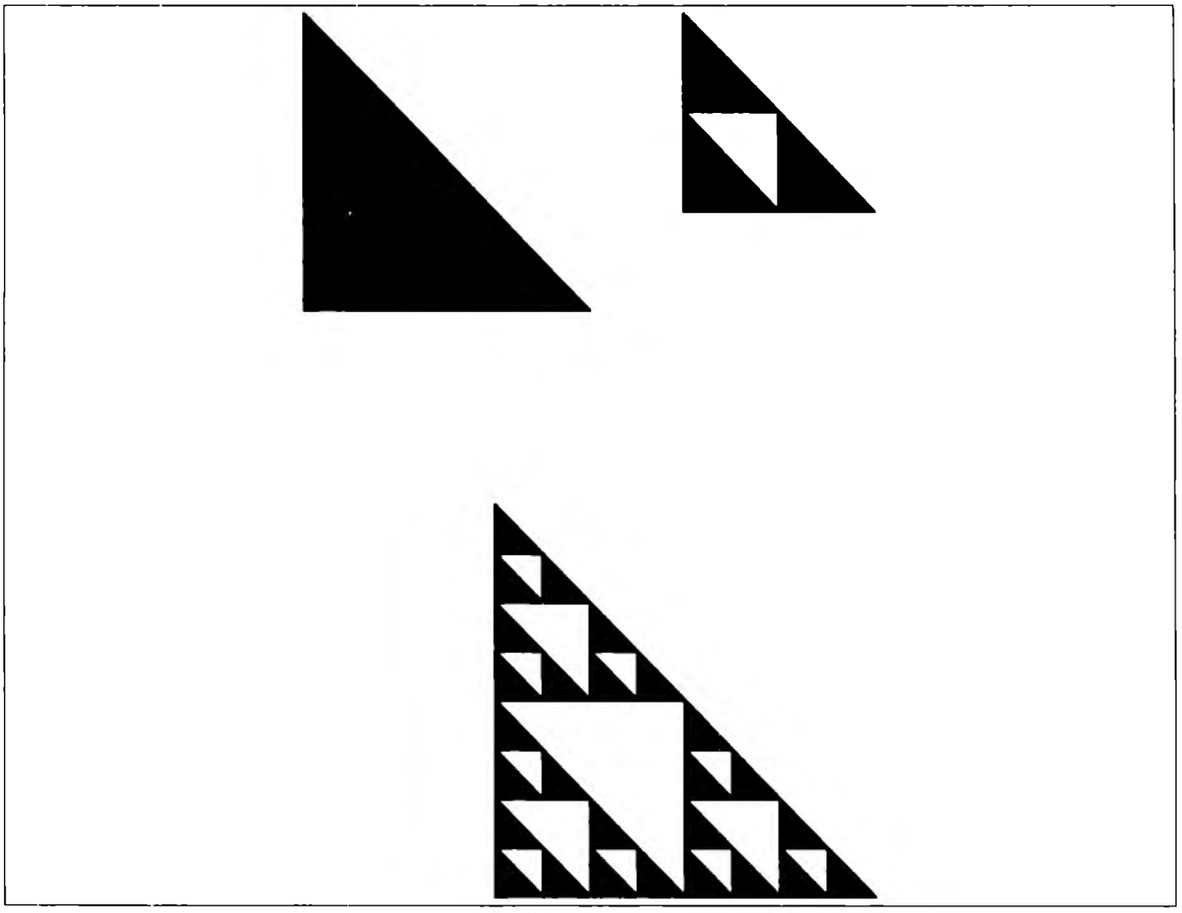

Figura 2.3: Triángulo de Sierpinski. Esta imagen está formada de partes similares a ella misma.

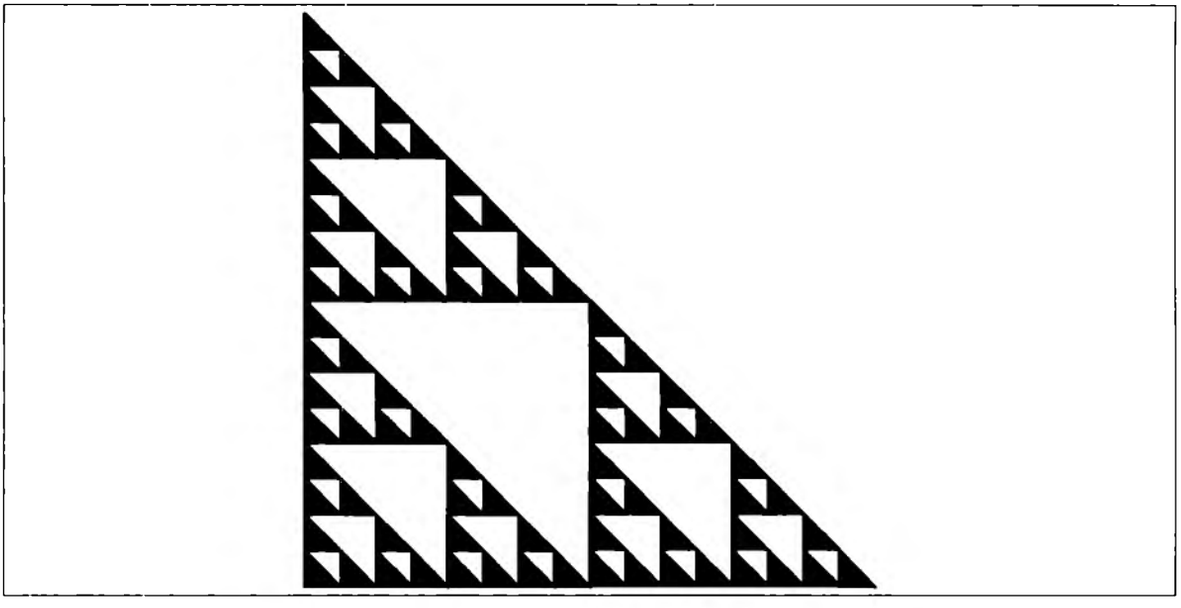

Figura 2.4: La aplicación de tres transformaciones afines deja invariante al triángulo.

cuando

$$k \to \infty,$$

donde d es la distancia de Hausdorff.

El teorema conocido como del *Collage* da una idea de qué tan bien un conjunto puede aproximar al atractor de un IFS.

Teorema 2.2 (Teorema del Collage) *[Bar88] Sean*

$$\{\omega_1, \ldots, \omega_m\}$$

contracciones sobre $\Re^n$ *de manera que*

$$|\omega_i(x) - \omega_i(y)| \leq c|x - y|, \quad \forall x, y, \in \Re^n \wedge \forall i$$

donde $c < 1$. *Sea* $E \subset \Re^n$ *un subconjunto compacto no vacío. Entonces*

$$d(E, F) \leq \frac{1}{(1-c)} d(E, \bigcup_{i=1}^{m} \omega_i(E)) \tag{2.4}$$

donde F *es el conjunto invariante de* $\{\omega_i\}$ *y* d *la distancia de Hausdorff.*

La demostración de este teorema puede consultarse en el apéndice A, teorema A.2.

Esto significa, por la parte derecha de la ecuación 2.4, que la distancia de Hausdorff entre una imagen y la transformación de esta imagen multiplicada por un factor igual a $\frac{1}{1-c}$, constituye una cota superior de la distancia entre el atractor del sistema y la imagen misma.

Una consecuencia inmediata de este teorema es el siguiente corolario.

Corolario 2.1 *Sea* $E \subset \Re^n$, *un conjunto compacto no vacío. Dado un* $\delta > 0$, *existe un conjunto de similaridades contractivas* $\{\omega_1, \ldots, \omega_m\}$ *tales que su conjunto invariante* F *satisface que* $d(E, F) < \delta$.

La demostración de este corolario puede consultarse en el apéndice A, corolario A.1.

De acuerdo con estos resultados, una imagen arbitraria puede aproximarse, tanto como se quiera, mediante otra que es atractor de un conjunto de transformaciones contractivas. Una vez determinado el conjunto de transformaciones $\{\omega_i\}$, es suficiente conservar éste para poder recuperar la imagen. Para guardar las transformaciones, se pueden guardar sólo los coeficientes de las transformaciones, que en general, requerirán un menor espacio, que codificar todos los canales de todos los pixeles de la imagen.

Teniendo un conjunto de transformaciones $\{\omega_i\}$, el recuperar la imagen que éstas codifican es muy sencillo y directo, ya que sólo hay que aplicar iterativamente las similaridades a un conjunto arbitrario para reproducirla. Ésto significa que se puede tomar como conjunto inicial una imagen cualquiera y aplicarle repetidamente la codificación que se ha guardado para así obtener la original. La aplicación reiterada de las transformaciones es computacionalmente directa y su complejidad temporal es lineal, lo que significa que descomprimir una imagen es muy rápido. En los experimentos que se realizaron, fue suficiente con aplicar entre dieciséis y treinta y dos iteraciones a una imagen cualquiera para recuperar la imagen original.

Sin embargo, la cantidad de transformaciones necesarias para aproximar una imagen dada, prefijando el error máximo δ, puede ser enorme. Además, éstos resultados no nos dicen cómo se pueden determinar las similaridades ω_i, ni cuántas de ellas son necesarias.

La estrategia que usan los métodos de compresión fractal de imágenes es distinta. En vez de fijar una distancia máxima entre la imagen original y su aproximación fractal, lo que se hace es fijar la cantidad transformaciones que codificarán la imagen, y, mediante algún algoritmo de optimización, se minimiza la distancia entre ésta y su aproximación.

2.3 Sistemas de Funciones Iteradas Locales

Con estas ideas, Barnsley y sus colaboradores comenzaron a trabajar en la compresión fractal de imágenes. Pero aún no lograban tener un procedimiento que automáticamente buscara un IFS para una imagen cualquiera.

Fue uno de los alumnos de Barnsley, Jaquin [Jaq89, Jaq92] quien logró automatizar la compresión introduciendo el concepto de *transformada fractal.* La teoría de transformación fractal trata básicamente de los *sistemas de funciones iteradas locales*, que se abrevian como *LIFS.*

Un LIFS

$$\{\omega_1, \ldots, \omega_m\}$$

es un IFS para el que el dominio de cada ω_i puede ser un subconjunto del conjunto en cuestión.

Una transformada fractal es un IFS cuyo código hace referencia al dominio en el que está definido. [BaHu92]. Aunque un LIFS no es idéntico a un IFS, la mayoría de los resultados que se aplican a éstos son ciertos también para aquellos.

A continuación, se definirá rigurosamente los LIFS para, posteriormente, hacer ver cómo con ellos se pueden construir métodos para la compresión fractal automática.

Definición 2.11 *Sea $D \subseteq \Re^n$ un conjunto no vacío compacto con una métrica d y sea $R \subset D$ no vacío. Sea $\omega : R \to D$ y $0 < c < 1$. Si*

$$d(\omega(x), \omega(y)) \leq c \cdot d(x, y) \quad \forall x, y \in R,$$

entonces ω se conoce como contracción local en D y la constante c se le llama factor de contracción de ω.

Definición 2.12 *Sea $D \subseteq \Re^n$ un conjunto no vacío compacto con una métrica d y sea $\omega_i : R_i \to D$ una contracción local enD, con factor de contractividad*

$$0 \leq c_i < 1, \quad i = 0, \ldots, m.$$

Entonces

$$\{w_i : R_i \to D, \quad i = 0, \ldots, m\}$$

se le llama sistema de funciones iterados locales *(LIFS). La constante* $c = max\{c_i\}$ *se conoce como el factor de contracción del LIFS.*

Definición 2.13 *Sea* $\mathcal{S} = \{E \subseteq \Re^n : E \subseteq D\}$, *entonces se define el operador* $W_{local} : \mathcal{S} \to \mathcal{S}$ *de manera que*

$$W_{local}(B) = \bigcup_{i=1}^{m} \omega_i(R_i \cap B), \quad \forall B \in \mathcal{S}.$$

Definición 2.14 *Se dice que* $A \subseteq D$ *es un atractor o conjunto invariante del LIFS si*

$$W_{local}(A) = A.$$

Sea $\{w_i : R_i \to D, \quad i = 0, \ldots, m\}$ un LIFS tal que los conjuntos R_i son compactos. Definimos una sucesión de subconjuntos de D, $\{A_n : n : 0, 1, \ldots\}$, como

$$\begin{aligned} A_0 &= D, \\ A_n &= \bigcup_{i=1}^{m} \omega_i(R_i \cap A_{n-1}), \quad n = 1, 2, 3, \ldots \end{aligned}$$

Es claro que $\{A_n : n : 0, 1, \ldots\}$ es una sucesión decreciente de conjuntos compactos.

Si existe un conjunto compacto $A \subset D$ no vacío tal que

$$\lim_{n \to \infty} A_n = A$$

y

$$A = \bigcup_{i=1}^{m} \omega_i(R_i \cap A) = W_{local}(A),$$

entonces A es el máximo atractor del LIFS.

Barnsley [BaHu92] supone que, sin pérdida de generalidad, el teorema del Collage 2.2 se puede aplicar a LIFS (aunque no lo demuestra). Es decir que si se tiene $\epsilon > 0$ arbitrario y una imagen E que se divide en cuadros no superpuestos

$$D_i, \quad i = 1, 2, \ldots, N,$$

hay que encontrar un conjunto de transformaciones afines

$$\omega_i : R_i \rightarrow E$$

con un factor de contracción c_i tales que

$$\omega_i(R_i) = D_i$$

y

$$d(\omega_i(R_i \cap E), D_i) < \epsilon, \quad i = 1, 2, \ldots, N,$$

donde d es una métrica en $\Re^n$.

2.4 Compresión Fractal de Imágenes

Una consecuencia inmediata del teorema 2.2 es que, un conjunto cualquiera puede ser aproximado arbitrariamente por un conjunto auto similar con una tolerancia o error aceptable; ésto es que, dado un valor $\epsilon > 0$ y una imagen E descrita en $\Re^n$, existen similaridades contractivas $\{\omega_1, \ldots, \omega_m\}$ con F como su conjunto invariante que satisface que $d(E, F) < \epsilon$.

Esto significa que el problema de determinar el IFS $\{\omega_1, \ldots, \omega_m\}$ cuyo atractor F aproxima a una imagen E dada es equivalente a minimizar la distancia

$$d(E, \bigcup_{i=1}^{m} \omega_i(E)) \tag{2.5}$$

con lo que, el problema inverso se traduce en un problema de minimización.

2.4.1 Método de Búsqueda Exhaustiva

Estas ideas fueron explotadas originalmente por M. F. Barnsley para comprimir imágenes usando el camino que marcan las LIFS.

El método que usó para determinar un LIFS consiste en dividir la imagen que se va a comprimir en un conjunto de bloques no superpuestos $D = \{D_i\}$ de $B \times B$ pixeles

llamados bloques del dominio y en un conjunto de bloques $R = \{R_i\}$ de $2B \times 2B$ pixeles, llamados bloques del rango (éstos últimos posiblemente superpuestos). El algoritmo busca un ω_j tal que

$$d(\omega_j(R_k), D_i)$$

sea mínima.

Éste es en realidad un algoritmo de fuerza bruta, ya que para cada D_i, se calcula la distancia $d(\omega_j(R_k), D_i), \forall j, k$ y aquel ω_j que tenga la mínima distancia es el que se guarda.

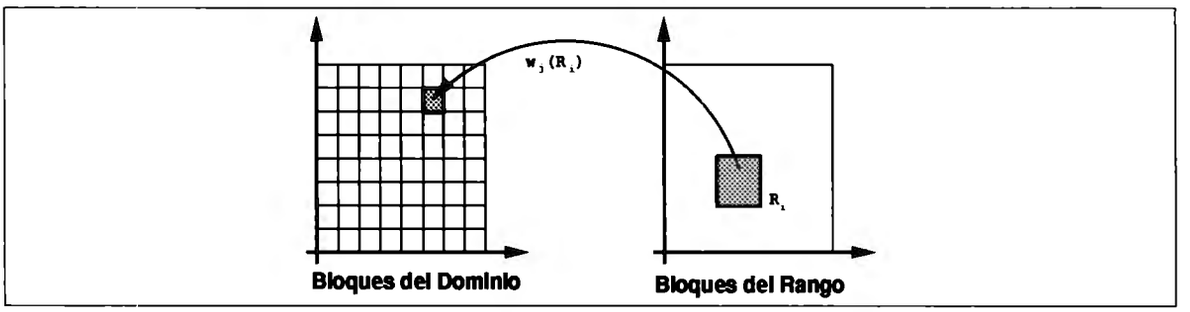

Figura 2.5: Algoritmo de compresión de Barnsley

Una característica interesante de este método de compresión es que la resolución de la imagen resultante es independiente en el sentido de que las dimensiones de la imagen descomprimida no deben ser necesariamente las de la imagen original ya que las transformaciones que codifican la imagen transforman una imagen cualquiera en una imagen que aproxima a la original.

Si la imagen se descomprime a un tamaño mayor que el de la original, el IFS genera el detalle adicional de manera realista.

El algoritmo de Barnsley trabaja bien cuando se ha paralelizado, o bien, implementado en hardware; pero, en general, la complejidad temporal del algoritmo es muy alta; es decir, que el tiempo requerido para comprimir una imagen es grande.

En los últimos años se han hecho algunos otros trabajos sobre el tema y se han propuesto otras alternativas para la solución de este problema de la complejidad temporal.

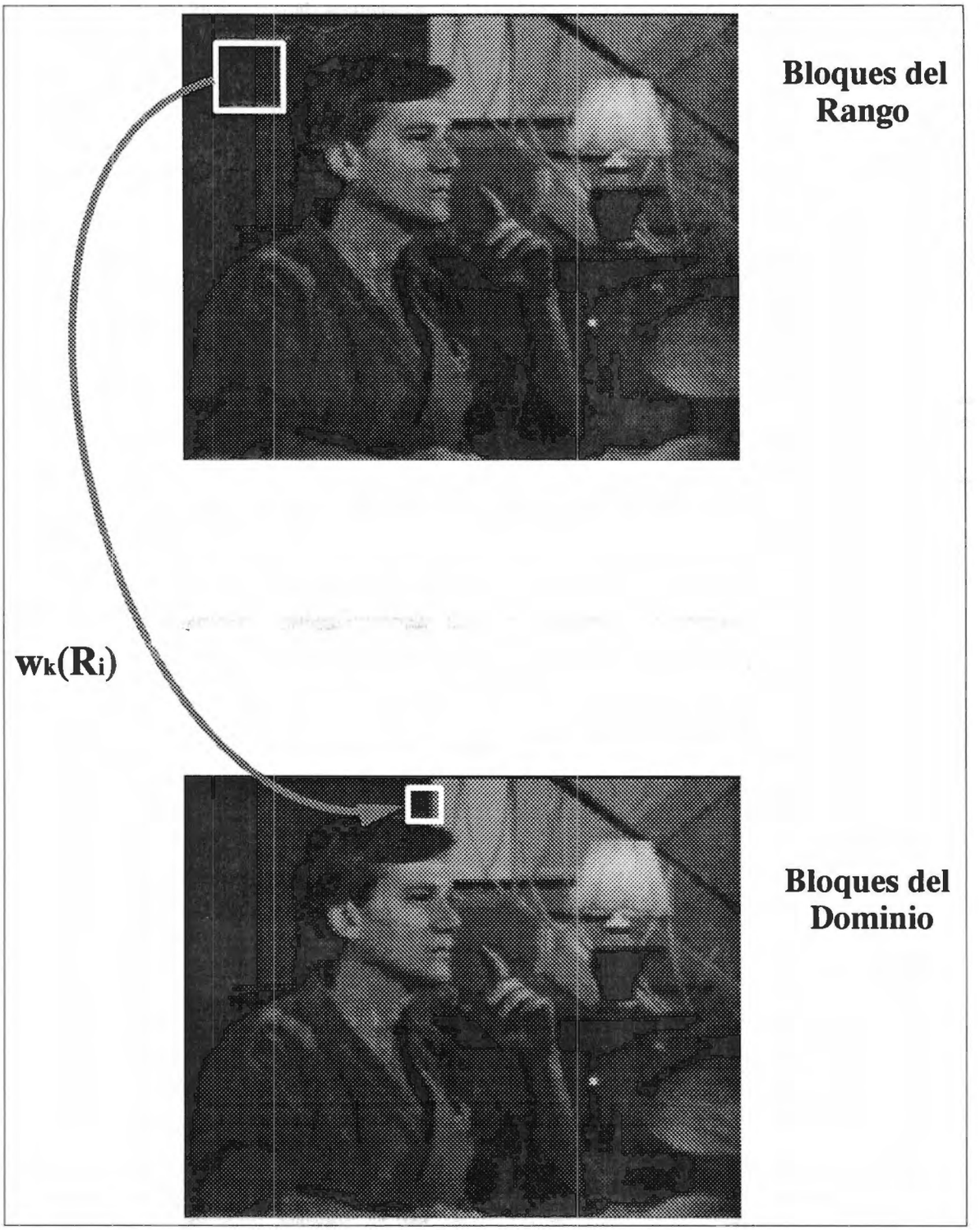

Figura 2.6: El problema de la compresión fractal de imágenes se traduce a encontrar un bloque del rango R_i, para cada bloque del dominio D_i, y una transformación ω_i tal que $d(D_i, \omega(R_i))$ sea mínima.

Jaquin [Jaq92], propuso clasificar los bloques del rango en un conjunto de categorías y realizar la búsqueda del mejor bloque del rango para un bloque dado del dominio sólo entre aquellos de la misma categoría. Con ésto, ya no se realiza una búsqueda exhaustiva para cada bloque.

Para el caso de imágenes en escalas de grises, los bloques se clasifican en los que son uniformes (shade), los que contienen un borde (edge) y los que tienen una categoría entre los dos anteriores (midrange). Con esta clasificación, la búsqueda de los bloques del rango se limita a aquellos que tienen las misma categoría, reduciendo de esta manera el universo de búsqueda, de manera que el algoritmo trabaja más rápido.

Las imágenes pueden presentar, por un lado, regiones que son difíciles de cubrir mediante bloques de un tamaño prefijado y, por otro lado, pude haber regiones en la misma imagen que podrían cubrirse con bloques de un tamaño mayor. Para resolver este problema, Fisher [Fis92] propuso usar *quadtrees* para dividir la imagen. Ésto es, escoger un tamaño de bloque de $B \times B$ y si, de acuerdo con un criterio de partición, debe usarse una partición más fina, el bloque en cuestión se dividirá en cuatro bloques de igual tamaño $B/2 \times B/2$.

Sin embargo, para Fisher, la partición por quadtrees tiene la debilidad de no poder hacer la clasificación de los bloques del dominio. El mismo Fisher propone que se pueden hacer particiones HV. En este tipo de particiones, una imagen rectangular se va dividiendo horizontal o verticalmente, según vaya siendo conveniente. Esta manera de dividir una imagen es más flexible que la anterior, ya que la posición donde se corta la imagen no es fija.

Otro esquema de partición que sugiere Fisher es mediante triángulos. Una imagen rectangular puede descomponerse en dos triángulos si se corta por alguna de sus diagonales, y cada una de estas partes, se puede seguir dividiendo en triángulos de manera recursiva. De esta manera, dice Fisher, se pueden disminuir las imperfecciones que generan las otras particiones por cubrir las imágenes de manera estrictamente horizontal y vertical. Sin embargo, no se ha desarrollado lo suficiente

este último procedimiento.

2.4.2 Métodos de Momentos

Otra estrategia que se ha utilizado para el problema inverso se basa en los momentos de un IFS. Un IFS genera una medida invariante y los momentos de una medida son a su vez invariantes a ella; es decir, que dos IFS tienen la misma medida sí y sólo sí éstas tienen los mismos momentos.

Definición 2.15 *Una medida μ sobre un conjunto $K \subset \Re^n$ se define como un mapeo*

$$\mu : \{X | X \subset K\} \rightarrow [0, \infty]$$

tal que

$$\mu(\emptyset) = 0$$

y

$$\mu(\bigcup_{i=1}^{\infty}(E_i) \leq \sum_{i=1}^{\infty} \mu(E_i),$$
$$E_i \subset X$$

Definición 2.16 *El soporte de una medida μ, $supp(\mu)$, se define como el conjunto cerrado más pequeño $S \subset \Re^n$ tal que*

$$\int_S f d\mu = 0$$

para cualquier función f tal que $f(x) = 0, \forall x \in S$. Ésto se expresa como $S = supp(\mu)$.

Definición 2.17 *Sea $(K, d), K \subseteq \Re^n$ un espacio métrico compacto con métrica d, $W = \{\omega_i\}$ un conjunto de contracciones continuas sobre K, ésto es,*

$$W = \{\omega_i : K \rightarrow K\}$$

y un conjunto

$$p = \{0 \leq p_i \leq 1 : \sum_i p_i = 1\}$$

tales que a cada ω_i le corresponde un valor p_i. Si $X \subseteq K$ es un espacio de medidas de Borel normalizadas sobre K, entonces, se define el operador de Markov $M : X \to X$ como

$$M(\nu) = \sum_i p_i \nu \circ \omega_i^{-1}.$$

Existe una única medida $\mu \in X$, llamada la medida invariante, tal que

$$M\mu = \mu.$$

Aún más, si F es el atractor del IFS (K, d), entonces $supp(\mu) = F$.

Es conveniente señalar que, el atractor de un IFS es independiente del conjunto p [Hut81].

Definición 2.18 *Si $\{K, w, p\}$ es un IFS en $\Re^n$, con atractor F, definimos los momentos de potencias de las medidas invariantes asociadas por las integrales de Lebesgue*

$$g_{i_1 i_2 \ldots i_n} = \int_F x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n} d\mu,$$

$$g_{00\ldots0} = \int_F d\mu = 1.$$

La base del método de momentos se encuentra en el hecho de que dada una sucesión infinita de reales positivos $g_{i_1 i_2 \ldots i_n}$ que satisfacen un conjunto de condiciones positivas, existe una única medida μ, tal que tiene como momentos a $g_{i_1 i_2 \ldots i_n}$.

Para aplicar los momentos en el problema fractal inverso, generalmente se busca que un cierto número finito de momentos de una medida μ de un IFS coincidan con los momentos de un conjunto predeterminado ν.

Un procedimiento que se propuso fue el de hacer coincidir un número finito de momentos para encontrar los parámetros adecuados para el IFS. Sin embargo, existen grandes problemas con la no linealidad de las ecuaciones implicadas, pues, en general, los métodos que se usan para resolverlas son muy inestables [Vrs90].

Vrscay y Roehrig [Vrs90], sugirieron minimizar la distancia entre los momentos de la medida invariante del IFS y los momentos de la imagen.

Por ejemplo, si consideramos un IFS $\{\omega_k : K \subseteq \Re \to \Re\}$, entonces,

$$\omega_k(x) = c_k x + a_k,$$

$$j = 1, 2, \ldots, N.$$

De esta forma, una contracción ω_k queda determinada por los parámetros c_k, a_k, independientemente de la probabilidad p_i que tengan asociada.

Si E es la imagen dada y

$$G^j_{i_1 i_2 \ldots i_n} = \int_E x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n} d\mu,$$

son los momentos de la imagen y los momentos

$$g^j_{i_1 i_2 \ldots i_n} = g^j_{i_1 i_2 \ldots i_n}(c, a, p),$$

$$c = c_k, \quad a = a_k, \quad p = p_k$$

corresponden a una medida invariante de un IFS con un número fijo N de transformaciones y M es el número de momentos G^j que se van a analizar, entonces la función que debe minimizarse es la de las sumas parciales del cuadrado de la distancia Euclidiana en el espacio de momentos,

$$D^N_M(\Pi) = \sum_{j=1}^{M} (g^j(s, a, p) - G^j)^2$$

Es claro que, para poder utilizar esta técnica, es necesario conocer un gran número de momentos G^j de la imagen I que se desea comprimir; situación que en la práctica no se da y los valores de los momentos con que pudiera disponerse no son muy exactos.

Planteado como una optimización, el método de los momentos para encontrar un IFS cuyo atractor sea una aproximación de una imagen I dada puede atacarse con distintas técnicas que existen para esta familia de problemas. Vrscay et al. [Vrs90] utilizaron métodos de gradiente y algoritmos genéticos. Los resultados obtenidos señalan que, en general, esta estrategia no funciona muy bien si no se conocen los momentos de la imagen.

2.4.3 Método de Monro/Dudbridge

En 1992, Monro y Dudbridge, presentaron un trabajo con en el que resuelven el problema fractal inverso de otra manera [MoDu92].

Consideran un sistema de contracciones en $\Re^2$,

$$W = \{\omega_i : i = 1, \ldots, m\} \tag{2.6}$$

donde

$$\omega_i(x, y) = \begin{bmatrix} a_{11}^{(i)} & a_{12}^{(i)} \\ a_{21}^{(i)} & a_{22}^{(i)} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1^{(i)} \\ b_2^{(i)} \end{bmatrix} \tag{2.7}$$

Como cada ω_i es una contracción, entonces existe un único conjunto F, que satisface la condición de invarianza de la ecuación 2.2 del teorema 2.1.

$$F = \bigcup_{i=1}^{m} \omega_i(F). \tag{2.8}$$

Una función $f : A \rightarrow \Re, \quad A \subseteq \Re^2$, se le llama invariante si existe un conjunto de funciones contractivas $\{v_k : A \times \Re \rightarrow \Re, \quad k = 1, \ldots, m\}$ tales que $\forall x \in A$,

$$f(x) = V \circ f(x) = v_k(\omega_k^{-1}(x), f(\omega_i^{-1}(x)))$$

donde $k = \max\{i : x \in \omega_i(A)\}$.

Se define *la medida de la raíz cuadrada media* sobre funciones de F como

$$d_{rms}(f, g) = \left\{ \frac{\int_F (f - g)^2 dL}{\int_F dL} \right\}^{1/2} \tag{2.9}$$

donde L es la medida de Lesbegue en $\Re^2$.

En estos términos, el teorema del Collage 2.2 quedaría expresado de la siguiente forma:

$$d_{rms}(f, g) \leq \frac{1}{1 - c} d_{rms}(g, \bigcup_{i=1}^{m} \omega_i(g)) \tag{2.10}$$

donde f es una aproximación invariante de la función g.

Para obtener una aproximación por mínimos cuadrados, se minimiza el lado derecho de la ecuación anterior 2.10 con respecto las variables $a_{ij}^{(k)}$ de la ecuación 2.7.

La solución que se encuentre para $a_{ij}^{(k)}$ es la aproximación fractal que estamos buscando.

2.4.4 Compresión Fractal de Video

En 1992, Barnsley publicó un método en el que aplica compresión fractal a imágenes de video ([HGB92]).

Primero, toma una imagen y la divide en bloques cuadrados (bloques del dominio) y encuentra un conjunto de transformaciones contractivas como lo hace para imágenes fijas (sección 2.4.1). A estas transformaciones, les agrega un parámetro de traslación, A_i.

El primer cuadro del video se codifica usando únicamente las funciones iteradas. Para los siguientes cuadros, utiliza las funciones iteradas y además el parámetro de traslación.

Si T_1 es la transformación usada para el primer cuadro F_1, su descompresión F_1' se obtiene

$$F_1' = T_1(F_1) \approx F_1$$

y para el segundo cuadro F_2,

$$F_2' = T_2(F_1') \approx F_2.$$

Los siguientes cuadros se obtienen aplicando la transformación de ese cuadro a la imagen previa, ésto es,

$$F_n' = T_n(F_{n-1}') \approx F_n$$

2.4.5 Conclusión

Los teoremas 2.1, 2.2 y el corolario 2.1 hicieron ver que es posible resolver el problema fractal inverso para codificar imágenes usando transformaciones afines. Este problema se planteó como un problema de minimización de la distancia en la ecuación 2.4, con lo que se empezó a vislumbrar como un método de compresión.

M. F. Barnsley fue quien comenzó a buscar alguna forma de llevar a la práctica estas ideas. En los inicios, las codificaciones se buscaban de manera interactiva (véase por ejemplo [BaSl88]). Jaquin fue el primero en lograr un mecanismo de obtener las transformaciones contractivas introduciendo los LIFS. Aunque el procedimiento de Jaquin permitió obtener de manera automática una codificación a una imagen dada, el tiempo que consumía el proceso fue excesivo, ya que su algoritmo realiza un búsqueda exhaustiva [Jaq89, Jaq92].

Vrscay propuso solucionar el problema fractal inverso usando los momentos de potencias de las medidas invariantes de las transformaciones, pero resultó impráctico ya que hay que conocer una gran cantidad de estos momentos para lograr buenos resultados [Vrs90].

Para disminuir el tiempo de la búsqueda exhaustiva, Jaquin propuso clasificar los bloque de la imagen original y realizar la búsqueda sólo entre aquellos bloques de las misma categoría [Jaq92].

Fisher [Fis92] propuso que la imagen se podría dividir usando otros esquemas como los quadtrees, particiones triangulares o HV. Usando estas técnicas, dice Fisher, se podría mejorar la calidad de la imagen resultante.

Monro y Dudbridge, por su lado estudiaron la posibilidad de plantear un sistema de ecuaciones en términos de los coeficientes de las transformaciones a fin de obtener una codificación apropiada [MoDu92, Mon93].

Dado que el problema fractal inverso puede plantearse como uno de optimización, Shonkwiler et al., decidieron estudiar la posibilidad de atacarlo usando algoritmos genéticos [SMD91]. Usando algoritmos genéticos, lograron resolver este problema en

una dimensión.

Capítulo 3

Algoritmos Genéticos

Los algoritmos genéticos son algoritmos de optimización y aprendizaje basados en los principios de la evolución natural que han probado ser muy eficientes en la búsqueda de aproximaciones a óptimos globales en espacios grandes y complejos en tiempos relativamente cortos [Gol89] .

El paradigma bajo el que trabajan los algoritmos genéticos consiste en emular los operadores naturales de procesamiento de información, tales como reproducción y selección de poblaciones para lograr algoritmos de optimización robustos y eficientes.

En 1975, Holland presentó a los algoritmos genéticos como resultado de un trabajo cuyos objetivos eran los de explicar rigurosamente los procesos de adaptación natural y diseñar sistemas artificiales que simulen estos mecanismos. Dentro de este contexto de adaptación natural, presenta los AG como algoritmos para la optimización de parámetros [Gol89].

3.1 Descripción General

El principio bajo el cual se desarrolló esta metodología se basa en la teoría de que la evolución es un proceso eficiente que busca soluciones óptimas a las situaciones que le presenta el medio ambiente [Sou92]. Una de las características importantes de los AG, es que éstos exploran espacios complejos y de dimensiones grandes en busca de

la solución del problema que se le presenta.

El tipo de problemas que se resuelven con esta estrategia se plantean, de forma general, como:

Dada una función $f : D \rightarrow R$, encontrar un elemento $d \in D$ que optimiza la función f.

Un algoritmo genético tiene entonces como componentes básicos:

1. Un conjunto de operadores genéticos que proporcionen un mecanismo para reproducir la información de uno o más elementos del espacio de solución que produzcan nuevos elementos mejor adaptados y así resolver el problema que se plantea. Generalmente se usa un operador binario de cruzamiento y un operador unitario de mutación. Cada uno de ellos tiene, además, una probabilidad de ser aplicado sobre la población (P_m es la probabilidad de mutación y P_c la probabilidad de cruzamiento).

2. Una representación apropiada del espacio de solución para que los operadores genéticos trabajen sobre ese espacio. Las representaciones más comúnmente utilizadas son aquellas con una longitud fija de parámetros; sin embargo puede utilizarse cualquier representación que pueda ser manejado por los operadores genéticos. Cada elemento $d \in D$ es representado mediante un mapeo g tal que

$$d = g(x_1, x_2, \ldots, x_l).$$

3. Una función de evaluación f (función objetivo) definida en el espacio solución que es la función que hay que optimizar. Esta función debe ser tal que los valores que asigne reflejen que tan bien un elemento de la población se adapta a la solución deseada.

4. Un procedimiento de inicialización que seleccione aleatoriamente un conjunto de elementos del espacio de solución como población inicial con la que comenzará a trabajar el AG.

5. Un conjunto de parámetros que determinan el proceso del AG, tales como el tamaño de población, *psize,* y la probabilidad asociada a cada uno de los operadores genéticos (por ejemplo, la probabilidad de mutación P_m y la probabilidad de cruzamiento P_c), así como un criterio de terminación.

Para la representación del espacio de solución D, a la que se hace referencia en el punto 2, se define un alfabeto Σ con cardinalidad k. Cada uno de los puntos del espacio de solución $d \in D$ se representa mediante una cadena de longitud l. Es decir que, para cada $d \in D$ se asocia una cadena

$$X = x_1 x_2 \ldots x_l, x_i \in \Sigma,$$

$$i = 1, \ldots, l.$$

A esta cadena, se le conoce como cromosoma y a cada uno de los x_i que forman el cromosoma se le llama gen.

Entonces, la función g que se menciona en este punto 2 está definida como

$$g : \Sigma^l \rightarrow D.$$

De esta manera, cada cromosoma $X = x_1 x_2 \ldots x_l$ de la población se evalúa mediante $f(g(X)) = f(g(x_1 x_2 \ldots x_l))$.

Los operadores genéticos del punto 1 trabajan sobre cadenas definidas en Σ^l. El operador de mutación queda como un mapeo m definido sobre Σ^l, ésto es,

$$m : \Sigma^l \rightarrow \Sigma^l;$$

mientras que, el operador de cruzamiento Cr se define como

$$Cr : \Sigma^l \times \Sigma^l \rightarrow \Sigma^l \times \Sigma^l.$$

Haciendo uso de estos componentes básicos, un AG comienza a trabajar como sigue:

1. Con el procedimiento de inicialización, crea aleatoriamente una población inicial y evalúa cada elemento de la población. De acuerdo con la evaluación que recibe, cada elemento de la población se le asigna una probabilidad ps_i de ser seleccionado para reproducirse en la siguiente generación.

2. Usando esta probabilidad ps_i, los operadores genéticos seleccionan a algunos miembros de la población para trabajar sobre ellos y así obtener nuevos individuos.

3. Aquellos elementos de la población que hayan resultado peor evaluados, son reemplazados por los nuevos individuos.

4. El algoritmo continúa trabajando hasta que se cumpla con algún criterio de terminación.

3.2 El teorema de los Esquemas

Uno de los aspectos importantes de un algoritmo de optimización que trabaja sobre espacios grandes y complejos es que debe existir un equilibrio entre la exploración de nuevas regiones en el espacio de solución y el aprovechamiento de aquellas regiones que ya han sido explorados y que resultaron buenas.

Holland propuso el *teorema de los esquemas* como una forma de explicar por qué el proceso de evolución es un mecanismo de optimización robusto y eficiente; y para demostrar que este equilibrio se cumple implícitamente en los AG.

Para la noción de esquema, se agrega un símbolo # al alfabeto Σ. Cuando en una cadena aparece el símbolo # en la i-ésima posición , significa que el elemento del alfabeto que ocupe esa posición carece de importancia.

Si se fijan k posiciones de una cadena de longitud $l > k$, de manera que las posiciones restantes tengan #, se forma un hiperplano de D de k-ésimo orden. Cada uno de los puntos de este hiperplano es llamado un esquema.

Un esquema define un hipercubo que contiene todas las tuplas que coinciden en las k posiciones fijas.

Si denotamos el número de tuplas de la presente población en un tiempo t que pertenecen a un esquema H mediante $M(H,t)$, entonces la cantidad de instancias de un esquema H en la siguiente población, $M(H,t+1)$ está dado por

$$M(H,t+1) = \frac{M(H,t)f_H(t)}{< f(t) >}$$

donde

$$f_H(t) = \frac{\sum\limits_{X_i \in H} f(g(X_i))}{\sum\limits_{X_i \in H} 1}$$

es el valor promedio del esquema en la población actual y

$$< f(t) >= \frac{1}{psize} \sum_{i=1}^{psize} f(X_i)$$

es el valor promedio de f en todas las tuplas de la población, X_i.

El crecimiento de un esquema particular después de t generaciones tiene un comportamiento exponencial que está dado por

$$M(H,t) = M(H,0)\left[\frac{< f(t) > +c < f(t) >}{< f(t) >}\right]^t = M(H,0)(1+c)^t$$

para una constante c.

La longitud $\delta(H)$ de un esquema H se define como la distancia entre las dos posiciones fijas más distantes en la tupla. Por lo que la probabilidad de que una operación de cruzamiento destruya un esquema H dado es la probabilidad de que el punto de cruce caiga en una de las posiciones que especifican el esquema. Esta probabilidad es igual a

$$\frac{P_c \delta(H)}{l-1},$$

donde P_c es la probabilidad de cruzamiento.

El orden de un esquema $o(H)$, denota el número de posiciones fijas en un esquema. La probabilidad de destruir un esquema debido a un evento de mutación es

$$o(H)^{P_m} \approx P_m o(H),$$

$$P_m \ll 1,$$

donde P_m es la probabilidad de mutación.

Estas ecuaciones determinan una cota inferior del crecimiento que puede tener un esquema debido a la mutación, el cruzamiento y la selección.

$$M(H,t+1) = M(H,t)\frac{f_H(t)}{< f(t) >}\left[1 - P_c\frac{\delta(H)}{l-1} - P_m o(H)\right] \tag{3.1}$$

A esta cota, Holland la llamó el teorema de los esquemas.

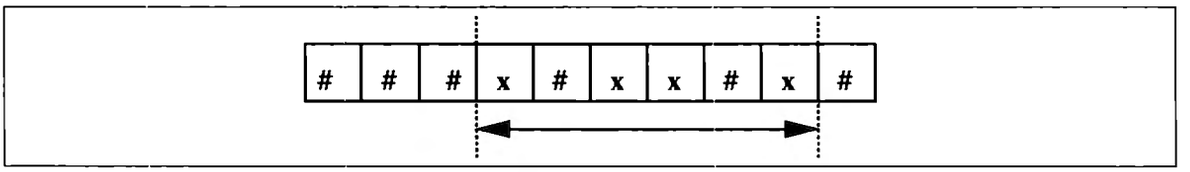

Figura 3.1: Esquema de cuatro dimensiones. La longitud de este esquema es $\delta(H) = 6$ y su orden $o(H) = 4$.

El teorema de los esquemas establece entonces que, con cada población que genera el AG, se va obteniendo mayor información sobre comportamiento del espacio de solución como resultado del crecimiento o decrecimiento de los esquemas. Como cada punto en el espacio de solución está representado por una l-tupla, que puede estar instanciada por 2^l esquemas diferentes, cuanto más detallada sea una representación, más esquemas son examinados a la vez. Este hecho se conoce como paralelismo implícito y es fundamental en los algoritmos genéticos, ya que con él se logra una rápida convergencia.

3.3 Modelos Alternativos de Algoritmos Genéticos

El algoritmo genético simple puede modificarse para mejorar su desempeño así como para que se ajuste mejor a un problema en particular. En esta sección, describimos las modificaciones que usamos para nuestra implementación.

La literatura sobre algoritmos genéticos ha tomado muchos de los términos que usa la genética natural y los han hecho corresponder con los conceptos que se manejan

en esta área.

En los sistemas naturales, un paquete genético es llamado genotipo y un organismo que ha sido formado por la interacción de un paquete genético con su ambiente se le llama fenotipo. En los sistemas artificiales, las cadenas, como hemos visto, codifican puntos en el espacio de solución.

Se dice también que, los cromosomas están compuestos por genes. Cada gen puede tomar valores particulares que se conocen como alelos.

3.3.1 Modelos Diploides y Dominancia

El modelo más simple de genotipo que se encuentra en la naturaleza es el de cromosomas haploides en el que una cadena contiene toda la información que es importante para el problema que se está tratando.

En el modelo diploide, un genotipo trae consigo uno o más pares de cromosomas, cada uno de los cuales contiene información sobre las mismas funciones o actividades que debe desempeñar y es necesario contar con un mecanismo de que decida cuál de los dos valores se debe tomar.

Una manera de resolver este conflicto es mediante un operador genético llamado *dominancia*. Este operador protege la información genética que ha sido útil en el pasado para que no desaparezca.

Al introducir la dominancia y la diploidía a un algoritmo genético, la ecuación 3.1 para el crecimiento de un esquema sigue siendo válida si el efecto de aquellos se refleja en el promedio de evaluación que tiene el esquema, $< f_H(t) >$.

3.3.2 Reescalamiento Lineal

Cuando un AG comienza a ejecutarse, la mayoría de los miembros de la población tienen una evaluación pobre por lo que el proceso de cruzamiento debe fomentar la combinación de la información genética con que se cuenta, favoreciendo a los mejores individuos. Si se produjera una pérdida inicial de la diversidad, podría provocarse

una convergencia prematura, sin que se haya dado oportunidad a examinar otras posibilidades.

Por otro lado, cuando el proceso está por terminar, los elementos de la población tienen una evaluación alta, lo que puede provocar que la búsqueda degenere en una caminata aleatoria, sin que se exploren las pequeñas diferencias que existen entre los miembros de la población.

Para evitar este tipo de problemas, en la implementación del AG, usamos un esquema de reescalamiento de la función objetivo. Estos esquemas favorecen la minimización de las diferencias entre los individuos de generaciones iniciales y contrastan las diferencias en las últimas generaciones.

El modelo de reescalamiento más sencillo es el lineal:

$$f' = af + b \tag{3.2}$$

donde a, b se escogen de manera que,

$$< f' >=< f > \tag{3.3}$$

$$f'_{\mathbf{max}} = C < f' > \tag{3.4}$$

donde $f'_{\mathbf{max}}$ es el máximo valor que toma la función escalada.

Al parámetro C se le conoce como constante de reescalamiento y se asocia con el número esperado de copias del mejor elemento de la población presente, que se quiere que tenga la siguiente población.

Las ecuaciones 3.2, 3.3 y 3.4 nos conducen al sistema:

$$a < f > +b =< f >$$

$$af_{max} + b = C < f >$$

que tiene como solución:

$$a = \frac{(C-1) < f >}{f_{max} - < f >}$$

$$b = \frac{< f > f_{max} - C < f >^2}{f_{max} - < f >}$$

3.3.3 Selección Determinista

El número de veces que un miembro de la población puede ser seleccionado para la reproducción no está determinado de forma unívoca por su probabilidad de selección. Este número de veces puede variar y, en ocasiones, toma valores muy alejados de su valor esperado.

Por esta razón, se ha propuesto el uso de un criterio determinista de selección. De acuerdo con este esquema, para cada elemento X_i de la población se calcula su probabilidad de ser seleccionado,

$$ps_i = \frac{f(X_i)}{\sum_{j=1}^{psize} f(X_j)}.$$

El valor esperado del número de selecciones para cada cromosoma X_i, de la población es

$$e_i = ps_i \cdot psize.$$

Entonces, a cada elemento se le asigna un número de selecciones igual a la parte entera de e_i y se ordena decrecientemente la población de acuerdo con la parte fraccionaria de los valores e_i.

Capítulo 4

Compresión Fractal de Imágenes mediante AG

Los algoritmos genéticos han sido usados para resolver los problemas de optimización relacionados con la compresión fractal de imágenes.

Vrscay [Vrs90] aplicó los algoritmos genéticos para la minimización de la distancia entre las funciones de momentos así como para el problema inverso del teorema del Collage. Sin embargo, en las conclusiones de su trabajo, Vrscay señala que con la aplicación de algoritmos genéticos en el segundo caso obtuvo mejores resultados; además de que el algoritmo es paralelizable, lo que acelera su convergencia.

Esta técnica también se puede usar para encontrar directamente un IFS cuyo atractor sea cercano a la imagen que queremos comprimir generando poblaciones de IFS. El operador de mutación, en este caso, introduce pequeños cambios aleatorios a los coeficientes de las transformaciones.

En 1991, Shonkwiler, Mendivil y Deliu [SMD91] presentaron un trabajo en el que emplearon algoritmos genéticos para el problema fractal inverso en una dimensión obteniendo buenos resultados.

En este capítulo, describimos las estructuras básicas que usamos para resolver el problema inverso de la compresión fractal de imágenes, así como los operadores

genéticos que empleamos.

4.1 Cromosomas

La imagen que se quiere comprimir se divide, de la misma manera que hace Barnsley [Bar88], en un conjunto de bloques no superpuestos del dominio $D = \{D_i\}$ de $B \times B$ pixeles y en bloques del rango $R = \{R_i\}$ de $2B \times 2B$. Esto significa que, el factor de contracción que usamos para el IFS $W = \{\omega_i\}$, lo fijamos en $\frac{1}{2}$.

El objetivo del algoritmo genético consiste, entonces, en determinar un conjunto de transformaciones afines $W = \{\omega_i\}$, tal que la distancia de la ecuación 2.5 sea mínima. Esto quiere decir que, el espacio de solución de nuestro problema se encuentra en el espacio de transformaciones afines con factor de contracción igual a $\frac{1}{2}$.

La razón de usar este factor de contracción se explica al observar la ecuación 2.4 del Teorema del Collage 2.2. En este caso, estamos utilizando $c = \frac{1}{2}$, por lo que tenemos,

$$d(E, F) \leq 2d(E, \bigcup_{i=1}^{m}(E, \omega_i(E))). \tag{4.1}$$

Puesto que vamos a estimar la distancia entre la imagen E y el atractor del IFS F calculando la distancia, $d(E, \bigcup_{i=1}^{m}(E, \omega_i(E)))$, es conveniente que el factor $\frac{1}{1-c}$ sea pequeño. Si el factor de contracción fuera mayor, $\frac{1}{2} < c < 1$, entonces la expresión del lado derecho de esta ecuación tendría un factor mayor a 2.

Por otro lado, si usamos un valor de c menor a $\frac{1}{2}$, ésto es, $0 < c < \frac{1}{2}$, la cantidad de transformaciones necesarias m para que

$$E \subseteq \bigcup_{i=1}^{m} \omega_i(R_i)$$

tendría que ser mayor. Es decir que, serían necesarias más transformaciones ω_i para cubrir a la imagen E con la imagen de éstas.

Para la codificación de los genes del AG, se usaron los parámetros que definen una transformación afín ω_i, de suerte que un cromosoma representa un IFS, $W = \{\omega_i\}$, y cada gen, una transformación afín, ω_i. Este esquema se ilustra en la figura 4.1.

Como la imagen que se pretende codificar se divide en bloques del dominio no superpuestos de $B \times B$, la longitud de los cromosomas queda determinada por el tamaño de la imagen. Esta longitud es igual a la cantidad de transformaciones afines ω_i, (una por cada bloque de $B \times B$) multiplicada por el número de parámetros que se necesita para especificar cada mapeo.

Los parámetros que se usaron para las transformaciones afines fueron los mismos con los que trabajó Jaquin [Jaq92]: las coordenadas x, y del correspondiente bloque R_i, la isometría aplicada al bloque y un factor de brillo.

Las transformaciones isométricas que se usaron fueron las ocho isometrías canónicas de un bloque cuadrado, que se encuentran resumidas en la tabla 4.1.

Si w es el ancho de la imagen que se va a comprimir y h la altura de la misma, entonces, el espacio de solución cuando se trata de comprimir una imagen en niveles de grises es:

$$[0, w-1] \times [0, h-1] \times [0, 7] \times [0, 255],$$

ya que,

$$0 \leq x \leq w-1,$$

$$0 \leq y \leq h-1,$$

$$0 \leq isom \leq 7,$$

$$0 \leq brillo \leq 255.$$

Es claro que, en general, el espacio de búsqueda en el que se encuentra la solución es grande. Si se tratara de una imagen de 256×256 pixeles, usando una búsqueda exhaustiva sería necesario examinar alrededor de 2^{27} posibilidades. Es por eso que éste método es tan lento para comprimir una imagen. Sin embargo, usando algoritmos genéticos es posible disminuir el tiempo empleado para la compresión.

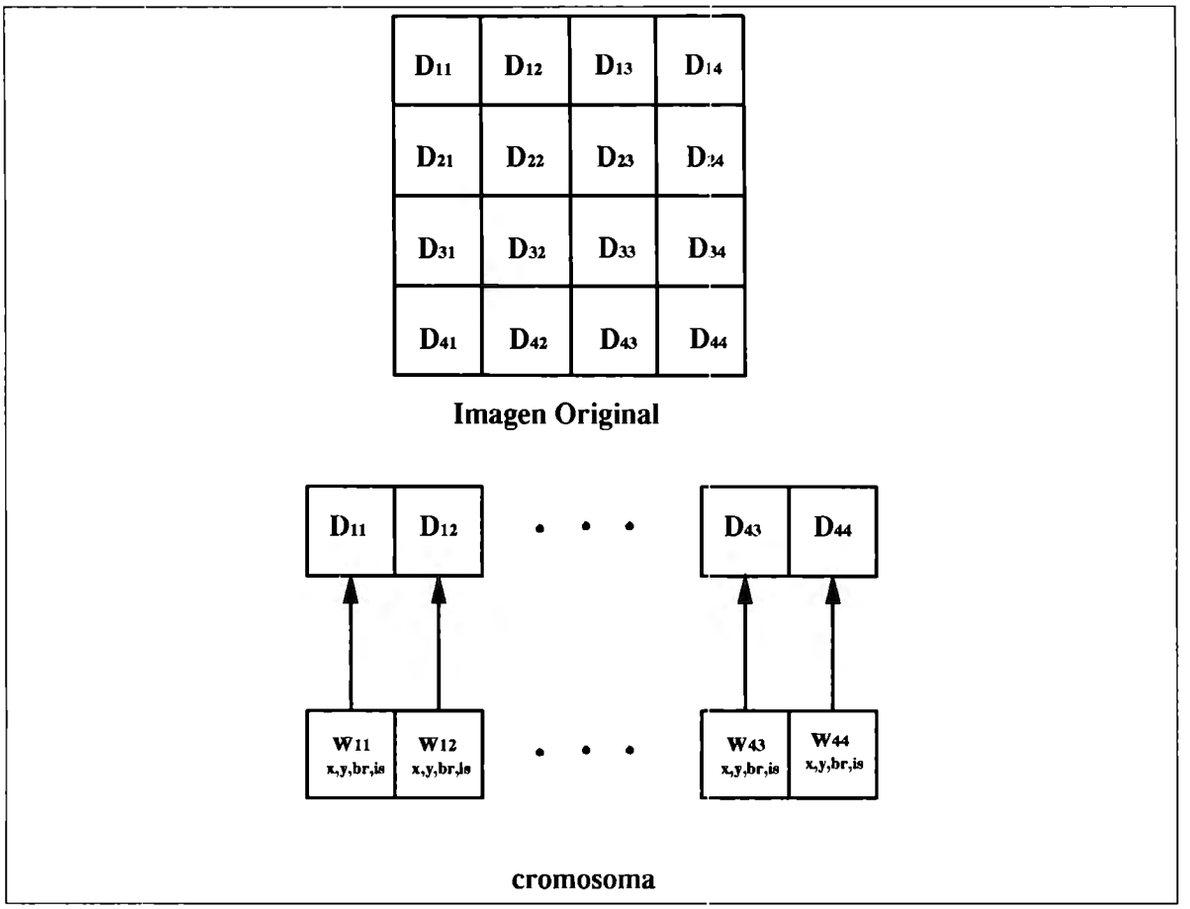

Figura 4.1: Estructura de los cromosomas. La imagen original se divide en bloques $B \times B$. A cada uno los bloques le corresponde una transformación ω_i. El conjunto de todas estas transformaciones forma un IFS. Para el algoritmo genético, cada IFS es un cromosoma.

#	Isometría	Fórmula
0	Identidad	$(i_0\mu)_{i,j} = \mu_{i,j}$
1	Reflexión Ortogonal (eje vertical)	$(i_1\mu)_{i,j} = \mu_{i,B-1-j}$
2	Reflexión Ortogonal (eje horizontal)	$(i_2\mu)_{i,j} = \mu_{B-1-i,j}$
3	Reflexión Ortogonal (primera diagonal) $(i = j)$	$(i_3\mu)_{i,j} = \mu_{j,i}$
4	Reflexión Ortogonal (segunda diagonal) $(i + j = B - 1)$	$(i_4\mu)_{i,j} = \mu_{B-1-j,B-1-j}$
5	Rotación (centro del bloque) $+90°$	$(i_5\mu)_{i,j} = \mu_{j,B-1-i}$
6	Rotación (centro del bloque) $+180°$	$(i_6\mu)_{i,j} = \mu_{B-1-i,B-1-j}$
7	Rotación (centro del bloque) $-90°$	$(i_7\mu)_{i,j} = \mu_{B-1-i,j}$

Tabla 4.1: Transformaciones Isométricas

El algoritmo genético que se desarrolló usa una población de IFS con un número fijo de transformaciones, por lo que la razón de compresión que se logra es constante.

De hecho, como se usó el mismo esquema de codificación que Barnsley, la razón de compresión que se logra es la misma que él obtiene.

4.2 Función de Evaluación

El problema inverso de las transformaciones fractales consiste en obtener un conjunto de transformaciones, tales que, aplicando esas transformaciones a una imagen codificada mediante ellas, se obtenga una aproximación lo más cercana posible a la imagen original.

El teorema del Collage (teorema 2.2), nos permite tener una idea de que tan cerca está una imagen E del atractor F de un IFS W. Pero además, por otro lado, observando la parte derecha de la ecuación 2.4, este teorema nos dice también que, al aplicar el IFS W a la imagen E que se quiere comprimir, y calculando la distancia que hay entre la imagen resultante y la imagen original, $d(E, W(E))$, se está estimando la distancia entre el atractor del sistema y la imagen original $d(E, F)$, ya que la distancia $d(E, W(E))$ constituye una cota superior de la primera distancia, $d(E, F)$. Esto quiere decir que, podemos calcular la proximidad o lejanía del atractor F a la imagen original E, calculando $d(E, W(E))$.

Tomando este hecho en cuenta, la función de evaluación que usa el algoritmo genético, toma cada elemento de la población que representa un IFS, lo aplica a la imagen que se está codificando y calcula la distancia entre la imagen original y la imagen transformada. La evaluación que recibe ese cromosoma es el inverso de la distancia entre la imagen original y la imagen original transformada (véase figura 4.2).

La función de distancia d, en realidad puede ser cualquier medida del espacio de imágenes. En la realización de este trabajo se probaron dos medidas:

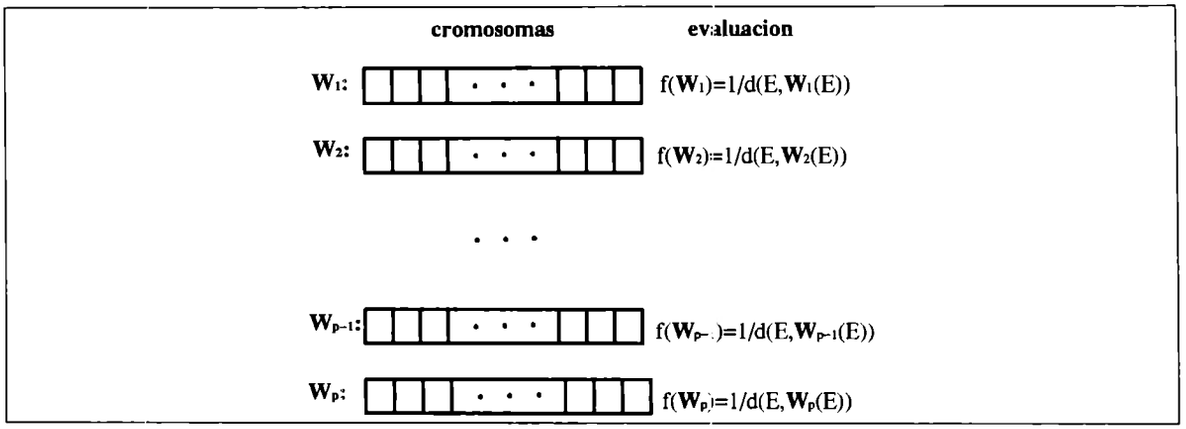

Figura 4.2: La función de evaluación calcula la distancia de cada IFS aplicado a la imagen y la imagen. El inverso de esta distancia es la evaluación que obtiene el cromosoma.

$$\mathcal{L}^1(\mu, \hat{\mu}) = \sum_{0 \leq i,j \leq B} |\mu_{ij} - \hat{\mu}_{ij}| \tag{4.2}$$

y

$$\mathcal{L}^2(\mu, \hat{\mu}) = \sum_{0 \leq i,j \leq B} (\mu_{ij} - \hat{\mu}_{ij})^2. \tag{4.3}$$

Sin embargo, resultó ser mejor medida la segunda de éstas.

4.3 Operadores Genéticos

Los operadores genéticos que se usaron fueron dos: un operador de cruzamiento y un operador de mutación.

4.3.1 El Operador de Cruzamiento Simple

El operador de cruzamiento trabaja seleccionando dos elementos de la población como padres de nuevos individuos. Esta selección se hace de acuerdo con la probabilidad que se asigna a cada elemento como resultado de la evaluación que se hizo de la población.

La función de evaluación que se usó para medir la distancia entre dos imágenes es la métrica $\mathcal{L}^2$, definida en la ecuación 4.3.

Dado que todos los cromosomas de la población tienen el mismo tamaño, el punto de cruzamiento puede ser cualquiera.

Cuando se usó el algoritmo genético simple, en el que los cromosomas son haploides, los dos padres combinan las dos partes en que quedan divididas para crear dos nuevos individuos: uno cuyo primera parte corresponde a la primera parte del primer padre, y su segunda parte, la toma de la segunda parte del segundo padre; y el segundo, se construye de manera opuesta. Este operador de cruzamiento simple se ilustra en la figura 4.3.

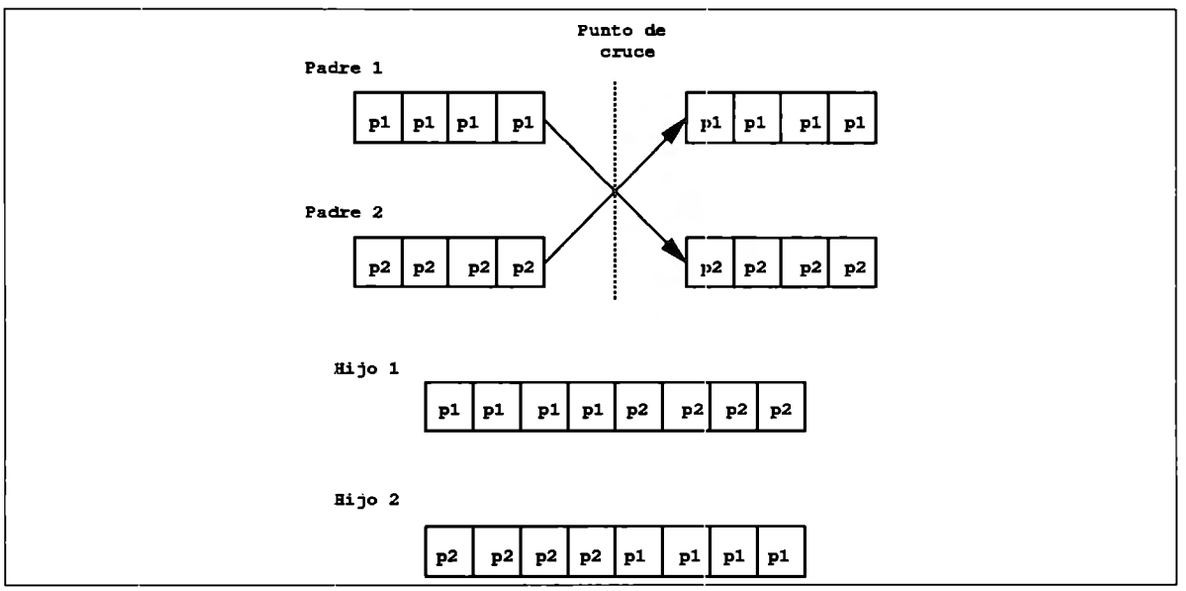

Figura 4.3: Operador de cruzamiento de un AG con cromosomas haploides.

4.3.2 Cromosomas Diploides

Con el objeto de obtener una convergencia más rápida, el operador de cruzamiento fue modificado para tratar a los cromosomas dentro de un modelo de diploidía (sección 3.3.1).

Ya que cada uno de los mapeos que forman el cromosoma corresponde a un determinado bloque del dominio D_i, la evaluación que obtiene este mapeo es independiente de la evaluación del resto de los genes que forman el cromosoma.

Para aprovechar este hecho, se modificó el operador de cruzamiento, de manera que, una vez seleccionados dos padres, de éstos se obtenga un nuevo individuo que esté formado por los mejores genes de cada padre. La manera como trabaja este operador se muestra en la figura 4.4.

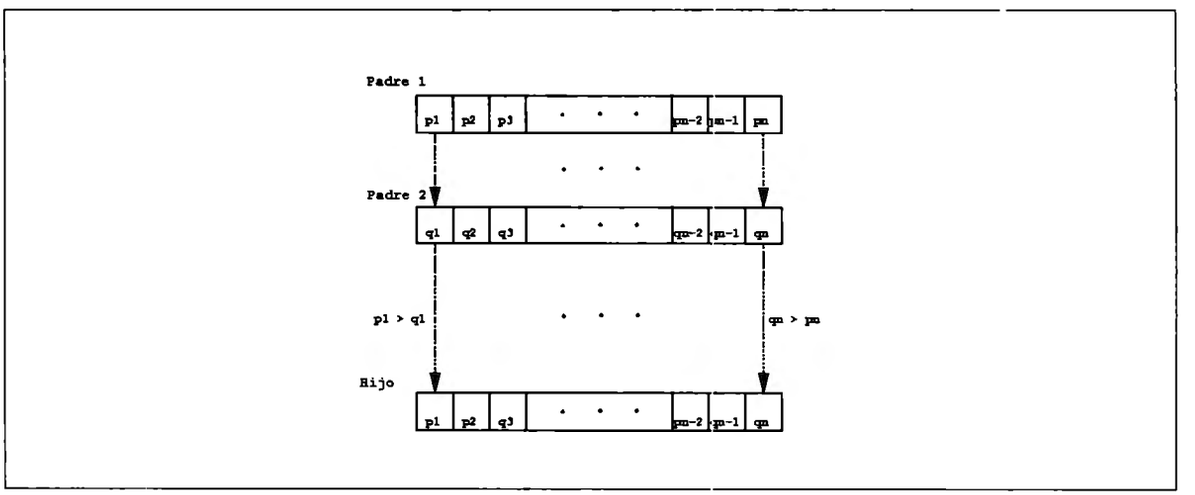

Figura 4.4: Operador de cruzamiento modificado para manejar cromosomas diploides ($p_i > q_i$ significa que el bloque i es aproximado de mejor manera por el mapeo p_i que por el mapeo q_i).

En los experimentos que se realizaron, se observó que la convergencia se aceleró significativamente con este operador de cruzamiento modificado.

4.3.3 Operador de Mutación

Se probaron dos operadores de mutación.

El primer operador, escoge aleatoriamente, el parámetro del gen en que se va a aplicar la mutación. Una vez hecho ésto, selecciona un nuevo valor para ese parámetro, dentro del espacio que le corresponde a ese parámetro. Es decir, por ejemplo, que si se va a aplicar una mutación sobre el parámetro x, se escoge un nuevo valor entero que esté entre 0 y el ancho de la imagen.

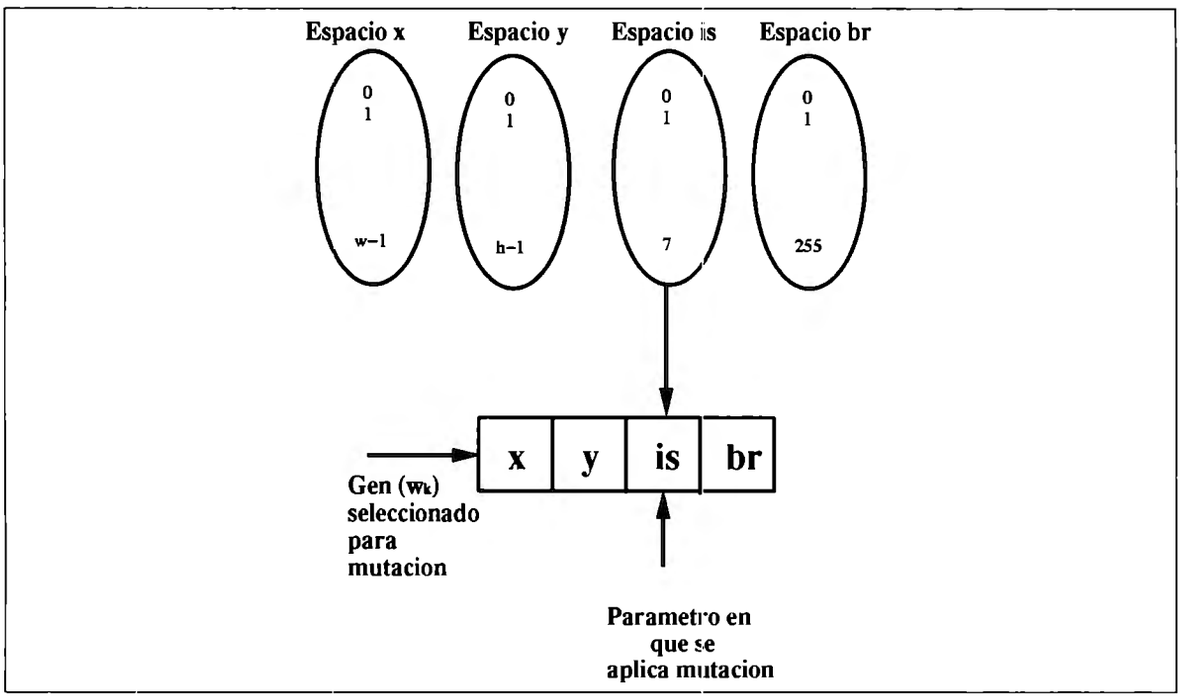

Figura 4.5: Primer operador de mutación: Se elige aleatoriamente un parámetro que se sustituye por un nuevo valor dentro del rango correspondiente. (x,y son las coordenadas de la esquina inferior izquierda del bloque del rango correspondiente, is es la isometría y br el factor de brillo que se aplica.)

El segundo operador de mutación, en vez de escoger valores al azar en los espacios correspondientes a los parámetros, produce pequeños cambios alrededor de los valores previos que tienen los parámetros x, y. Este operador genera una pequeña variación Δ que se suma al parámetro en que opera la mutación.

Ambos operadores de mutación son aplicados con un probabilidad P_m a cualquiera de los genes que forman un cromosoma.

4.4 Implementación

Con el objeto de tener un administración eficiente de la memoria que se usa, fue empleada una pila. Dentro de esta pila se almacenan las localidades de memoria de los individuos de la población actual que obtuvieron una evaluación pobre. Cuando el operador de cruzamiento genera nuevos elementos, éstos ocupan el espacio de aquellos que fueron puestos en la pila. De esta manera, los cromosomas con una adaptación pobre son desechados de la población y el tamaño de ésta se mantiene siempre constante.

Nuestra implementación del algoritmo genético usa también un esquema de reescalamiento lineal (sección 3.3.2) además de un criterio de selección determinista (sección 3.3.3).

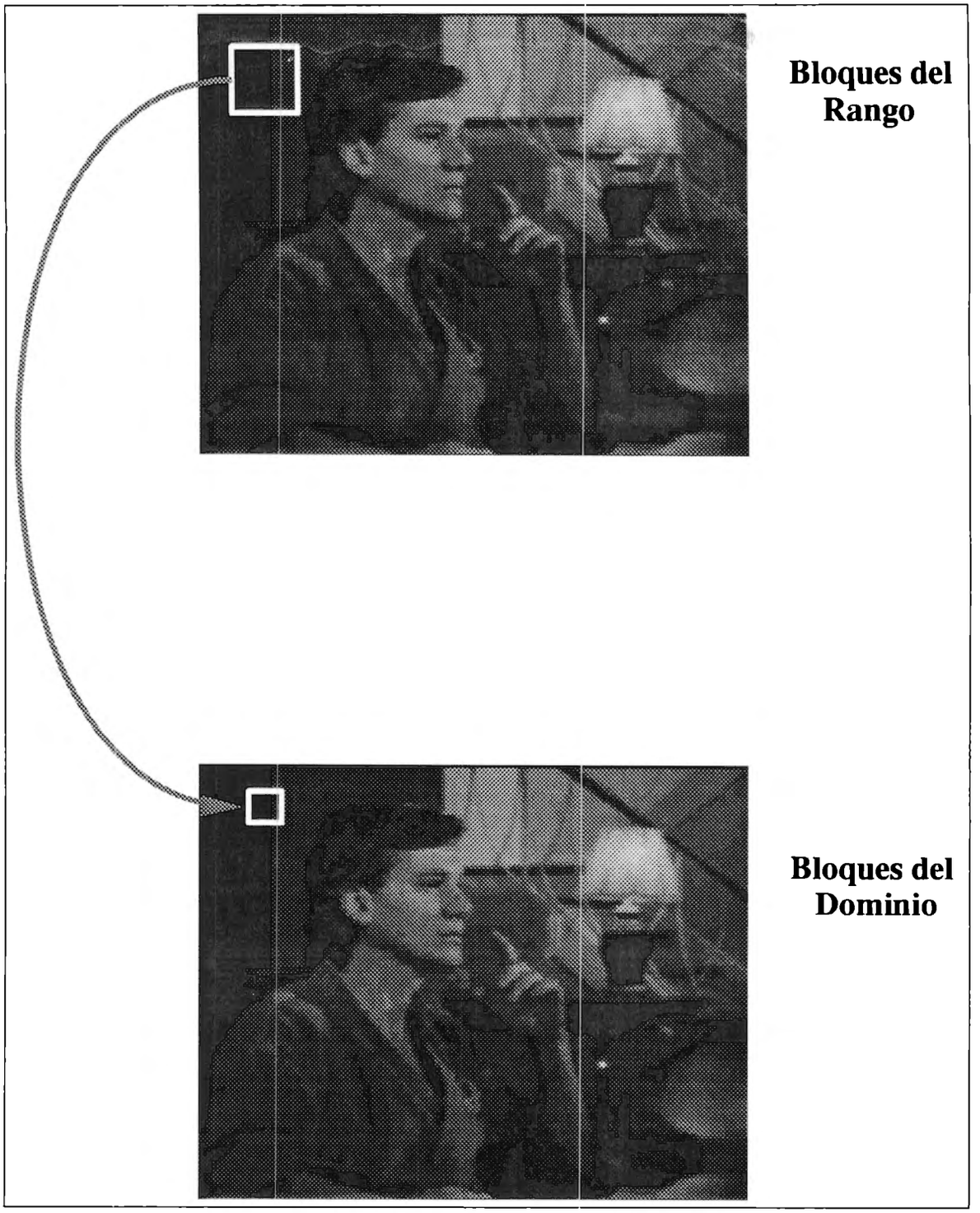

Figura 4.6: El algoritmo genético busca la mejor transformación de un bloque del rango a cada bloque del dominio de la imagen.

Capítulo 5

Resultados

Todas las pruebas que describimos en este capítulo fueron ejecutadas en una máquina IBM-RS6000, modelo 375.

En las pruebas que se realizaron, se usó una imagen en tonos de grises de 296×240 pixeles. Tanto para el algoritmo de búsqueda exhaustiva como para los distintos modelos de algoritmos genéticos que se implementaron, se dividió la imagen en bloques del dominio de 8×8 pixeles. Esto significa que, la cantidad de mapas que se necesitaron para la imagen en todos los algoritmos fue la misma, por lo que la compresión lograda fue igual con todos los procedimientos.

Las diferencias en las imágenes resultantes de los distintos algoritmos fueron en calidad y tiempo empleado. La medida de error se tomó como la distancia entre la imagen original y la imagen resultante.

5.1 Resultados en Imágenes Fijas

Primero, se corrieron pruebas usando el algoritmo genético de cromosomas haploides. El algoritmo corrió con un número fijo de generaciones usando dos métricas como función de evaluación, $\mathcal{L}^1$ y $\mathcal{L}^2$, para poder determinar cuál métrica guía mejor al algoritmo genético.

La imagen que se obtuvo con la distancia $\mathcal{L}^1$ estaba muy alejada de la imagen que

se quería comprimir. En cambio, con la distancia $\mathcal{L}^2$ (ecuación 4.3), el algoritmo obtuvo una imagen más próxima a la imagen objetivo, por lo que se decidió usar esta distancia.

Con el algoritmo genético de cromosomas haploides, se corrieron varias pruebas usando diferentes tamaños de población: 512, 256, 128 y 64. Esta pruebas se hicieron con el fin de poder determinar un tamaño de población adecuado.

El principal problema de esta técnica de compresión fractal, hasta la fecha, ha sido el tiempo excesivo que es necesario para comprimir una sola imagen. Por un lado, entre más grande se escogiera el tamaño de población, ésto significaba mayor consumo de tiempo en la evaluación de esa población. Pero, por otro, escoger una población muy pequeña significa retardar la convergencia del algoritmo, que también implica mayor tiempo para obtener una imagen aceptable. Por lo tanto, en este punto, era importante seleccionar un tamaño de población conveniente.

En la gráfica de la figura 5.1 se presenta el comportamiento del algoritmo en cada caso. La convergencia del algoritmo para las poblaciones de 512, 256 y 128 individuos no cambia significativamente. En cambio, cuando el tamaño de la población es de 64 miembros, la rapidez de la convergencia se frena un poco. Por esta razón, se consideró que un tamaño de población adecuado para las pruebas era 128 individuos en una población.

De esta misma figura, se puede ver que la convergencia de este algoritmo es muy lenta. Si se usa para este algoritmo un tiempo de corrida similar al que emplea el algoritmo de búsqueda exhaustiva de Barnsley, y se comparan los resultados de cada uno, vemos que el error del algoritmo genético simple es bastante mayor que el error que produce la búsqueda exhaustiva.

Además, hay que tener en cuenta que el error que produce el algoritmo de búsqueda exhaustiva es el mínimo error que se puede cometer, ya que la codificación que se usa es semejante en todos los casos.

Las siguientes pruebas que se hicieron fueron modificando el modelo de cromosomas para introducir un esquema de dominancia, como el mencionado en la sección

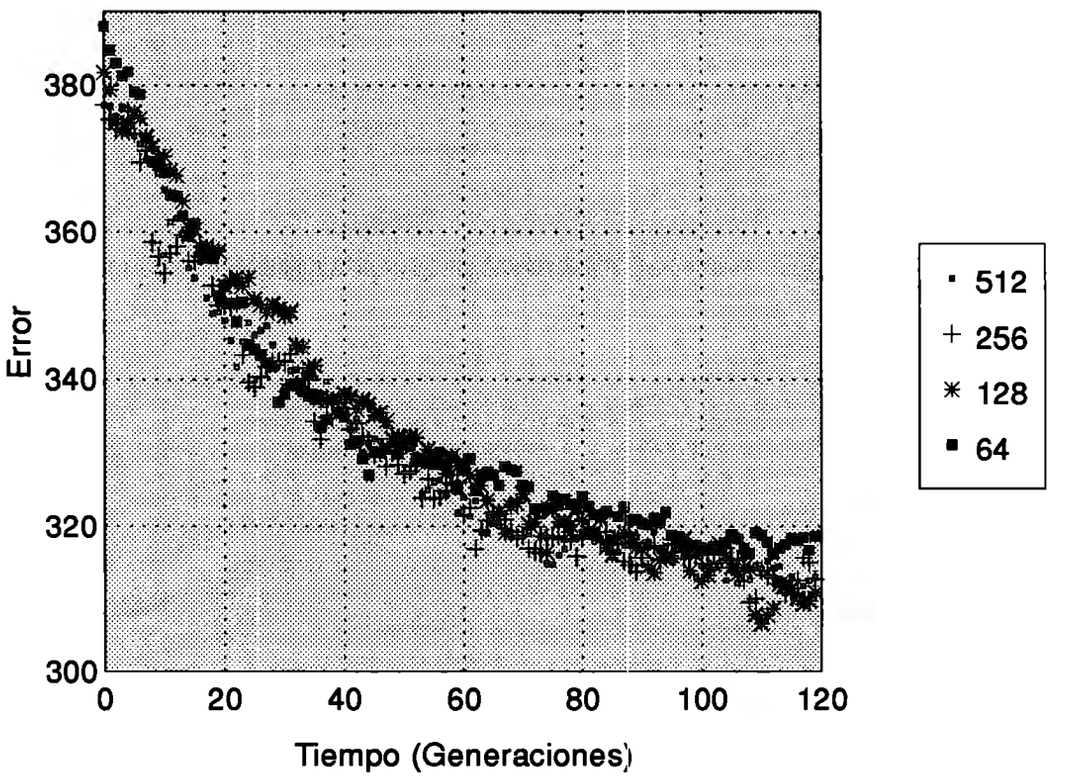

Figura 5.1: Comparación de la convergencia del Algoritmo Genético Simple usando distintos tamaños de población: 512, 256, 128 y 64 individuos.

4.3.2. Haciendo esta modificación, se logró acelerar la convergencia significativamente, como se puede apreciar en la gráfica de la figura 5.2.

Usando tiempos de ejecución similares al tiempo empleado por el algoritmo de Barnsley, este último algoritmo logra disminuir notoriamente el error que produce el algoritmo de cromosomas haploides.

Sin embargo, para obtener una imagen de calidad comparable con la lograda por la búsqueda exhaustiva, este último procedimiento emplea un tiempo menor que el algoritmo de Barnsley y que el algoritmo de cromosomas haploides.

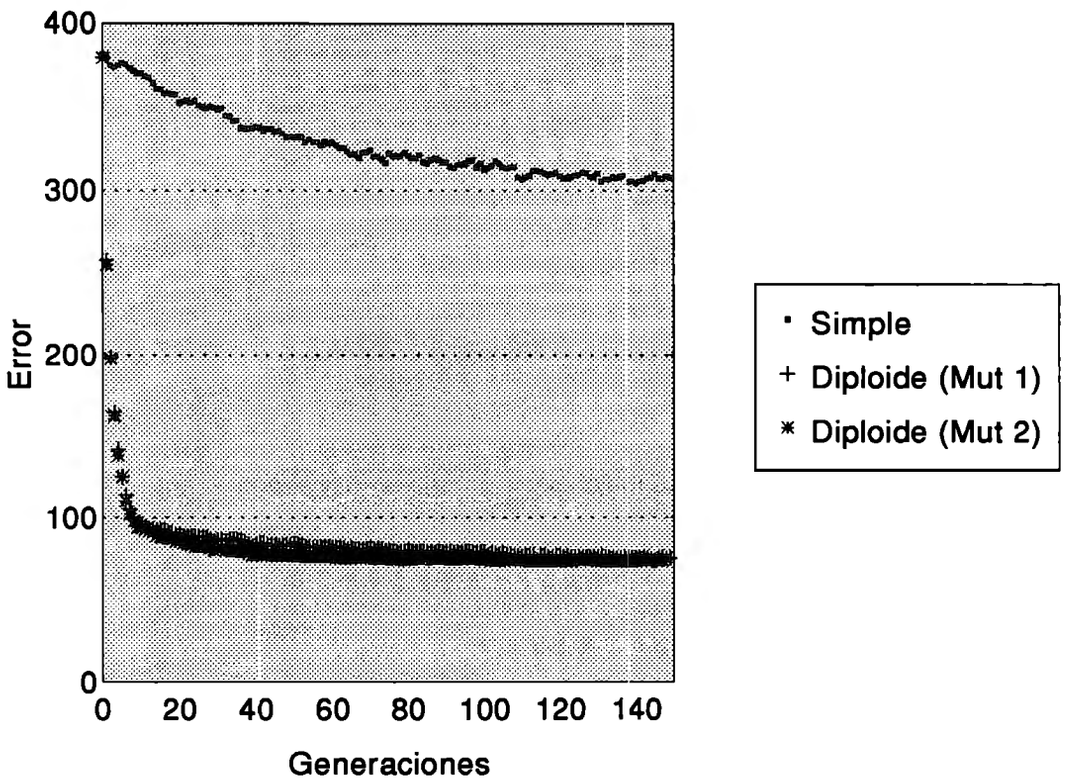

Figura 5.2: Comparación entre la convergencia de los distintos algoritmos.

En la tabla 5.1 se compara el tiempo y el error que producen los distintos métodos de algoritmos genéticos que se implementaron.

Dado que nuestro espacio de búsqueda es el mismo que el que usa el algoritmo

Tiempo (min)	*Barnsley*	*AG Haploide*	*AG Diploide*	*Mutación Modificado*
33	- -	358.03	82.33	80.37
82.5	- -	342.39	76.98	69.64
115	- -	321.48	71.94	66.01
247.5	- -	315.70	68.78	65.48
330	40.29	312.74	66.85	65.45

Tabla 5.1: Tiempo y Error en los distintos métodos.

de Barnsley, no es posible que los algoritmos genéticos puedan encontrar una mejor solución que la de Barnsley. Sin embargo, sí podemos encontrar aproximaciones aceptables y que se encuentran cerca de las imágenes obtenidas por aquel procedimiento, en un tiempo menor; ésto es, alrededor de una tercera parte del tiempo que requiere una búsqueda exhaustiva.

Pero además, si se desea una imagen de mejor calidad, es posible dejar correr el algoritmo genético por más tiempo. Mientras que en el otro caso, es necesario esperar hasta que termine la ejecución del algoritmo para poder tener una imagen.

Las siguientes pruebas que se hicieron, fueron con el segundo operador de mutación, produciendo cambios aleatorios pequeños alrededor de los valores previos de los genes. Cuando se usó este operador en una sola imagen, se observó que no obtenía mejores resultados que el primer operador. Sin embargo, al aplicar el algoritmo a secuencias de imágenes, este operador sí mejoró los resultados del primero.

En la figura 5.2 observamos el comportamiento de los distintos algoritmos genéticos. De la figura se puede ver cómo desciende rápidamente el error de los algoritmos genéticos con cromosomas diploides, mientras que con los cromosomas haploides la convergencia es muy lenta. Además, se observa que la curva del error del segundo operador de mutación tiene un comportamiento semejante al del primer operador.

En la figura 5.3, se presentan las imágenes obtenidas por los distintos algoritmos que se utilizaron.

En las figuras 5.4, 5.5 se observa cómo va cambiando la solución que obtiene

Figura 5.3: Imagen original (superior izquierda) comparada con las imágenes obtenidas por el algoritmo de Barnsley (superior derecha) y los algoritmos genéticos con cromosomas diploides (en la parte inferior, el de la izquierda con el primer operador de mutación y el de la derecha con el segundo).

el algoritmo genético en distintas generaciones. Como es normal en estos procedimientos, puede suceder que el mejor cromosoma de una generación anterior tiene una evaluación mayor que el mejor cromosoma de una generación posterior. Por esta razón, es conveniente llevar una bitácora en la que se guarde la mejor solución encontrada a lo largo de toda la ejecución del algoritmo. De esta forma, se garantiza que se conservará la mejor aproximación encontrada.

5.2 Resultados en Secuencias de Imágenes

Se corrieron pruebas en secuencias de imágenes usando los dos operadores de mutación.

Los algoritmos toman el primer cuadro de la serie, inicializan una población aleatoriamente y comienzan a buscar un IFS para esta imagen. Al encontrar una solución para el primer cuadro, la guardan y usan la última población de éste como población inicial del segundo cuadro y comienzan a buscar una solución para el nuevo cuadro. Los algoritmos siguen trabajando de manera semejante para los siguientes cuadros hasta terminar la secuencia.

El error que obtenía la población inicial del segundo cuadro y de los siguientes, era menor que el error de la población inicial para el primer cuadro. Sin embargo, la pendiente de la curva del error de estos cuadros no es tan pronunciada como la del primero, ya que tenemos una población inicial demasiado especializada. Por consiguiente, la convergencia es más lenta.

Este hecho lo observamos en las figuras 5.8, 5.9. Aunque, el segundo operador de mutación obtiene mejores resultados que el primero, no se alcanza a disminuir el error de los siguientes cuadros (segundo, tercero y cuarto) de la misma manera que con el primer cuadro.

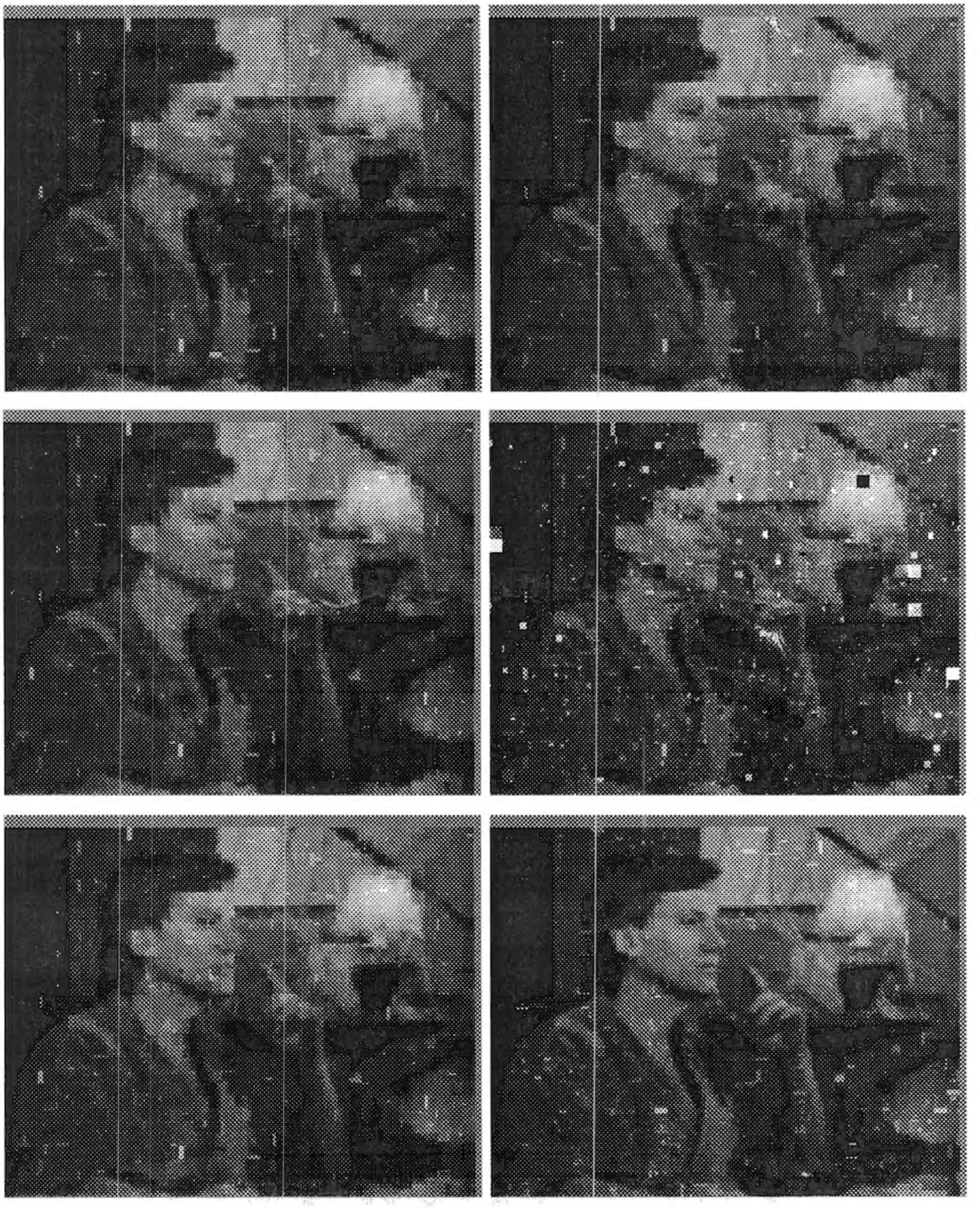

Figura 5.4: Imágenes obtenidas por el algoritmo genético con mutación modificada en las generaciones 0, 6, 12, 18, 24, 30.

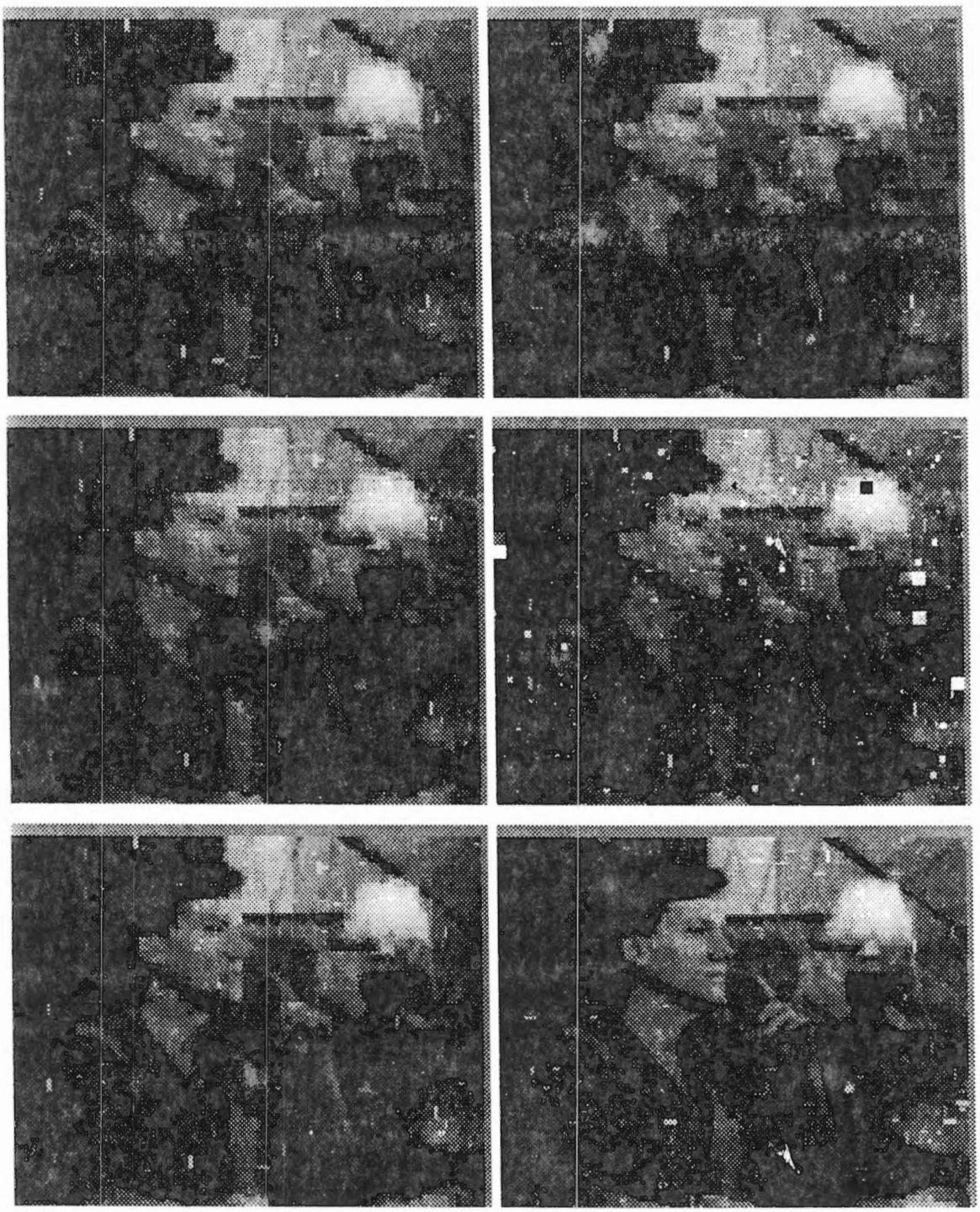

Figura 5.5: Imágenes obtenidas por el algoritmo genético en las generaciones 36, 42, 48, 54, 60 y por búsqueda exhaustiva.

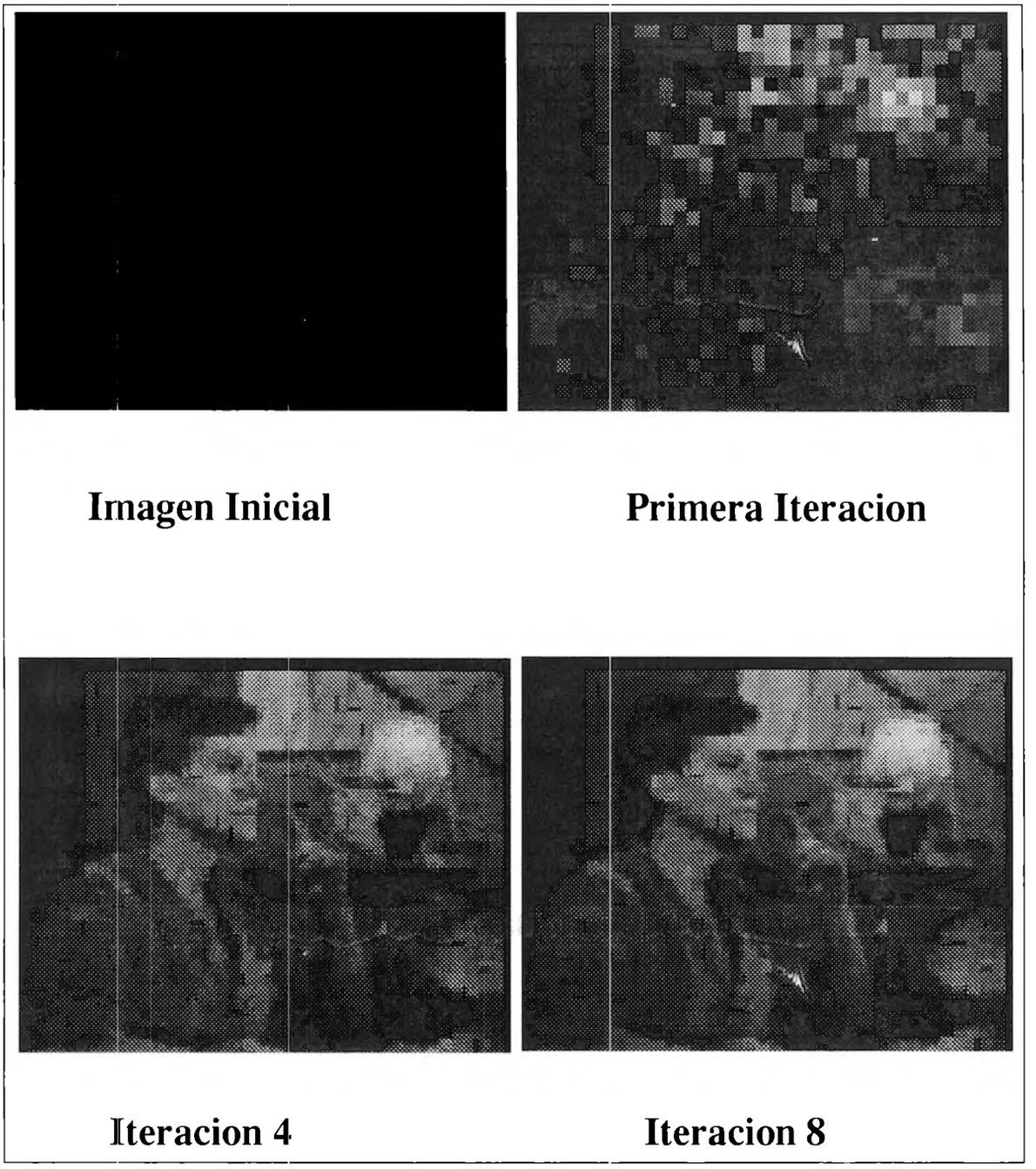

Figura 5.6: Descompresión de una imagen

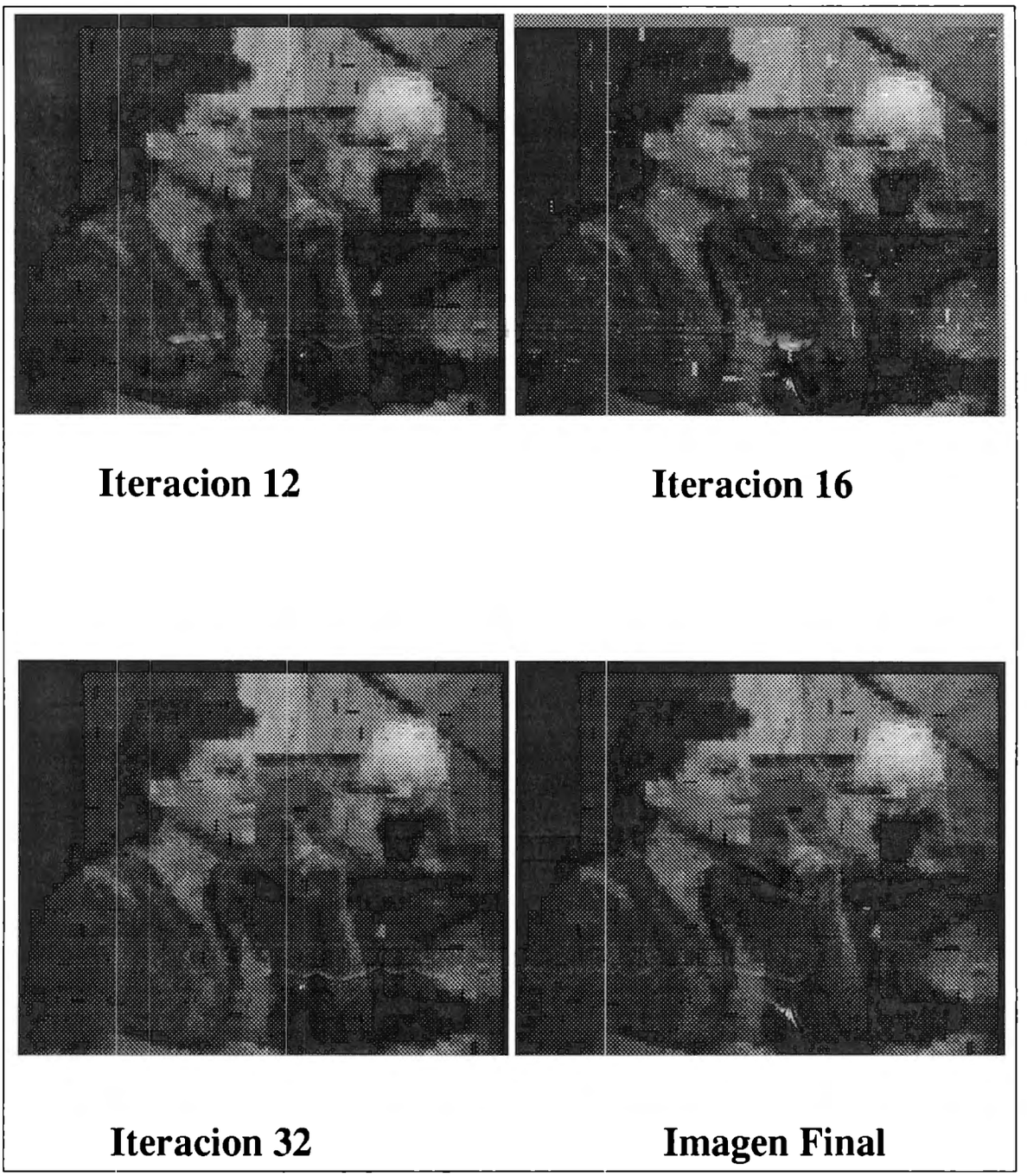

Figura 5.7: Descompresión de una imagen

Figura 5.8: Secuencia de cuatro cuadros consecutivos de video, tomando como población inicial de un cuadro, la última población del cuadro inmediato anterior.

Figura 5.9: Secuencia de cuatro cuadros consecutivos de video, tomando como población inicial de un cuadro, la última población del cuadro inmediato anterior (b).

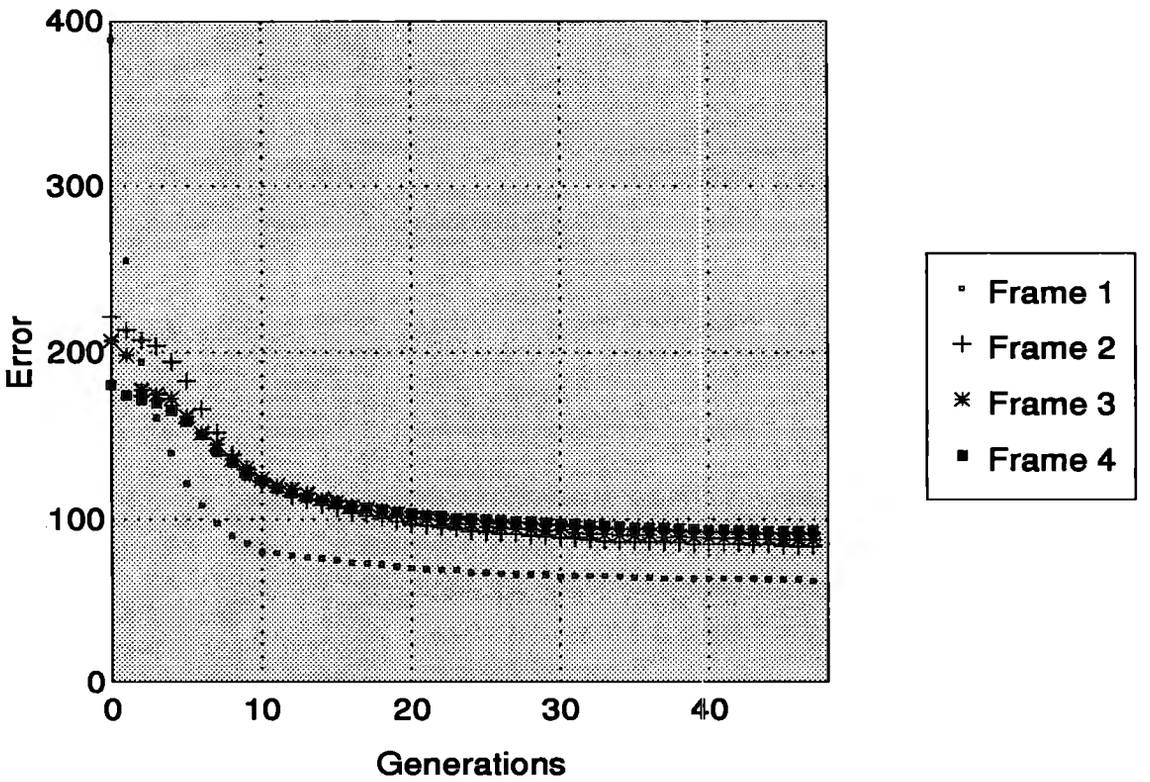

Figura 5.10: Comportamiento del error en un secuencia de cuatro cuadros consecutivos de video, tomando como población inicial de un cuadro, la última población del cuadro inmediato anterior (Primer operador de mutación).

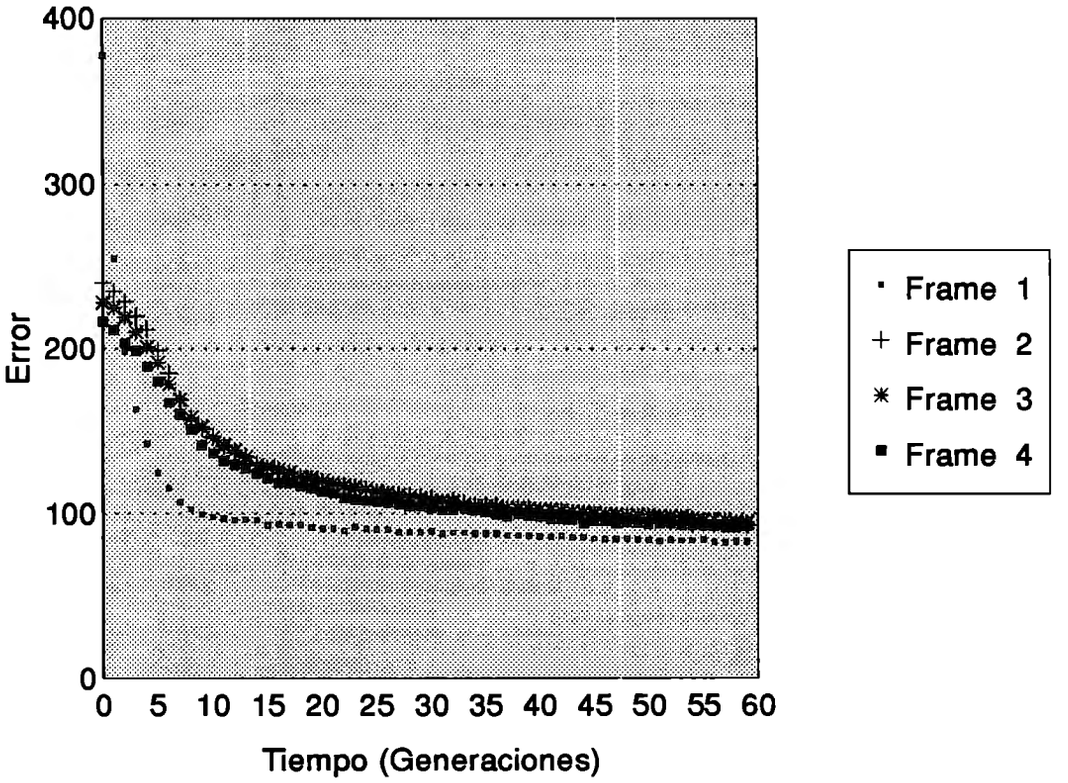

Figura 5.11: Comportamiento del error con el algoritmo genético de cromosomas diploides y el segundo operador de mutación de una secuencia de cuatro cuadros consecutivos de video.

Capítulo 6

Conclusiones

En este trabajo, se presentaron distintos modelos de algoritmos genéticos para la solución del problema de compresión fractal de imágenes. Aunque se pueden encontrar trabajos anteriores que resuelven este problema, estas soluciones tienen una complejidad temporal muy grande.

Para la solución de este tipo de problemas, los algoritmos genéticos han probado ser una herramienta muy poderosa; particularmente cuando el espacio de solución es muy grande y complejo. Tal es el caso del problema fractal inverso para la compresión de imágenes.

Se presentaron dos modelos de algoritmos genéticos: un modelo de cromosomas haploides y un modelo de cromosomas diploides. Se encontró que los modelos de cromosomas diploides obtienen más rápidamente una solución aceptable que los primeros modelos.

Aunque la razón de compresión que logra este algoritmo es similar a la de otras técnicas propuestas, el tiempo que emplea este algoritmo no solo es menor al del algoritmo genético con cromosomas haploides, sino también es menor hasta en un 30% que un algoritmo de búsqueda exhaustiva, como el Barnsley, Jaquin, Fisher u otros.

Para secuencias de imágenes, se esperaba tener una convergencia todavía más

rápida que la observada para una sola imagen. Pero como la última población de la imagen anterior estaba ya especializada, es decir, los elementos de la población estaban concentrados en un punto, y en realidad la convergencia fué más lenta.

Aún cuando los resultados obtenidos en el caso de una secuencia de imágenes no son los esperados, sí se logró reducir el tiempo empleado para la compresión de una sola imagen, lo que implica también una reducción del tiempo de compresión de una secuencia.

Sin embargo, hay que considerar que los resultados obtenidos mejoran los reportados en trabajos previos. Existen otros modelos evolutivos que no fueron examinados en este trabajo, que pudieran llevar estos resultados más adelante; como por ejemplo, los modelos de regresión simbólica de Koza. Por lo que debe seguirse profundizando este estudio.

En este problema de la compresión fractal de imágenes, es necesario tomar en cuenta que, en el método que se proponga para comprimir imágenes tiene que haber un equilibrio entre el tiempo que tarde la compresión y la calidad que se logre. Para mejorar la calidad de la imagen puede tomarse una tamaño de bloque más pequeño, pero ésto aumentaría la complejidad temporal y reduciría la razón de compresión. Un camino que podría tomarse puede ser el examinar otros tipos particiones como las que propone Fisher (sección 2.4.1): quadtrees, particiones HV o particiones triangulares. El algoritmo de compresión, en este caso, debe ser capaz de identificar aquellos bloques que tengan un borde y dividirlos en bloques más pequños para mejorar la fidelidad de esas partes de la imagen.

Una vez obtenido un IFS para la imagen propuesta, se puede aplicar algún algoritmo de compresión sin pérdida de información a esta codificación, como los presentados en la sección 1.3, con el fin de hacer aún más compacta la representación de la misma.

Por otro lado, los algoritmos genéticos son procedimientos que pueden paralelizarse fácilmente [BiBr93]. En particular, el algoritmo que se presentó en este trabajo podría ser paralelizado para acelerar su rapidez.

Apéndice A

Demostraciones Fundamentales

Teorema A.1 *(Teorema 2.1)*

Sean $\{\omega_1, \ldots, \omega_m\}$ *contracciones sobre* $D \subset \Re^n$ *tales que*

$$|\omega_i(x) - \omega_i(y)| \leq c_i|x - y|, \ \forall x, y \in D$$

y $c_i < 1$ *para cada* i. *Entonces existe un único conjunto no vacío compacto* F *que es invariante a la familia de contracciones*$\{\omega_i\}$, *esto es,* F *satisface que*

$$F = \bigcup_{i=1}^{m} \omega_i(F).$$

Aún más, si definimos una transformación W *sobre la clase* $\mathcal{W}$ *de conjuntos compactos no vacíos mediante*

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

y escribimos W^k *como la k-ésima iteración de* W, *dada por*

$$W^0(E) = E,$$

$$W^k(E) = W(W^{k-1}(E)), k > 1,$$

entonces

$$F = \bigcap_{k=1}^{\infty} W^k(E)$$

para cualquier conjunto $E \in \mathcal{W}$ *tal que* $\omega_i(E) \subset E$ *para cada* i.

Demostración.

Sea $E \in \mathcal{W}$, tal que $\omega_i(E) \subset E, \forall i$. Entonces,

$$W^k(E) \subset W^{k-1}(E),$$

por lo que $W^k(E)$ es una sucesión decreciente de conjuntos compactos no vacíos y, por el teorema de intersección de Cantor, su intersección,

$$F = \bigcap_{k=1}^{\infty} W^k(E),$$

es no vacía.

Dado que, $W^k(E)$ es una sucesión decreciente, entonces,

$$W^k(F) = F,$$

ésto es, F es invariante.

Para demostrar la unicidad, sean $A, B \in \mathcal{W}$. Entonces,

$$d(W(A), W(B)) = d(\bigcup_{k=1}^{m} \omega_i(A), \bigcup_{k=1}^{m} \omega_i(B)) \leq \max_{1 \leq i \leq m} d(\omega_i(A), \omega_i(B))$$

Si escogemos δ tal que

$$\omega_i(B) \subset (\omega_i(A))_\delta,$$

para cada i, entonces,

$$\bigcup_{i=1}^{m} \omega_i(B) \subset (\bigcup_{i=1}^{m} \omega_i(A))_\delta$$

Como ω_i son contracciones,

$$d(W(A), W(B)) \leq (\max_{1 \leq i \leq m} c_i) d(A, B).$$

Si A, B son conjuntos invariantes, entonces $W(A) = A, W(B) = B$ y $d(A, B) = 0$ y, por lo tanto, $A = B$. □

Teorema A.2 (Teorema del Collage) *(Teorema 2.2)*

Sean

$$\{\omega_1, \ldots, \omega_m\}$$

contracciones sobre $\Re^n$ *de manera que*

$$|\omega_i(x) - \omega_i(y)| \leq c|x - y|, \forall x, y \in \Re^n, \forall i = 1, \ldots, m$$

donde $c < 1$. *Sea* $E \subset \Re^n$ *un subconjunto compacto no vacío. Entonces*

$$d(E, F) \leq \frac{1}{(1-c)} d(E, \cup_{i=1}^{m} \omega_i(E)) \tag{A.1}$$

donde F *es un conjunto invariante de* $\{\omega_i\}$ *y* d *la distancia de Hausdorff (definición 2.10).*

Demostración.

Aplicando la desigualdad triangular de la métrica de Haussdorf y el teorema A.1, tenemos que

$$\begin{aligned} d(E, F) &\leq d(E, \bigcup_{i=1}^{m} \omega_i(E)) + d(\bigcup_{i=1}^{m} \omega_i(E), F) \\ &\leq d(E, \bigcup_{i=1}^{m} \omega_i(E)) + d(\bigcup_{i=1}^{m} \omega_i(E), \bigcup_{i=1}^{m} \omega_i(F)) \\ &\leq d(E, \bigcup_{i=1}^{m} \omega_i(E)) + cd(E, F), \end{aligned}$$

de donde se sigue inmediatamente la conclusión. □

Corolario A.1 *Sea $E \subset \Re^n$, un conjunto compacto no vacío. Dado un $\delta > 0$, existe un conjunto de similaridades contractivas $\{\omega_1, \ldots, \omega_m\}$ tales que su conjunto invariante F satisface que $d(E, F) < \delta$.*

Demostración.

Sean $B_1, \ldots, B_m$ una colección de n-esferas que cubren a E con sus centros en E y radio menor $\delta/4$. Entonces,

$$E \subset \bigcup_{i=1}^{m} B_i \subset E_{\delta/4}$$

donde $E_{\delta/4}$ es la $\delta/4$-cubierta paralela de E.

Sea ω_i una similaridad contractiva con razón de contracción menor a $1/2$. Entonces,

$$\omega_i(E) \subset B_i \subset (\omega_i(E))_{\delta/2},$$

lo que implica que,

$$\bigcup_{i=1}^{m} (\omega_i(E)) \subset E_{\delta/4}$$

y

$$E \subset \bigcup_{i=1}^{m} (\omega_i(E))_{\delta/2}.$$

Por definición de la distancia de Haussdorf, tenemos que

$$d(E, \bigcup_{i=1}^{m} \omega_i(E)) < \delta/2.$$

De la ecuación A.1, tenemos que

$$d(E, F) < \delta$$

donde F es el conjunto invariante de $\{\omega_i\}$. □

Bibliografía

[ARA91] Ang, Peng H., Peter A. Ruetz, David Auld, *Video compression makes big gains,* IEEE Spectrum, October, 1991.

[Bar88] Barnsley, Michael, *Fractals Everywhere*, Academic Press, San Diego (1988).

[BaSl88] Barnsley, M.F., Alan D. Sloan, *A Better Way to Compress Images*, BYTE, January, 1988.

[BaHu92] Barnsley, Michael, Lyman P. Hurd, *Fractal Image Compression*, AK Peters, Ltd (1992).

[BiBr93] Bianchini, Ricardo, Christopher Brown, *Parallel Genetic Algorithms on Distributed-Memory Architectures*, The University of Rochester, Computer Science Department, Technical Report 436, (May 1993).

[CuSi91] Culik, Karel, Simant Dube, *Using Fractal Geometry for Image Compression*, DCC '91, Data Compression Conference, p. 459. IEEE, 1991.

[DJL91] DeVore, Roland A., Bjorn Jawerth, Bradley J. Lucier, *Data Compression using Wavelets: error, smoothness, and quantization*, IEEE (1991).

[Fal90] Falconer, K.J. *The Geometry of Fractal Sets*, Cambridge University Press, Great Bretain (1990).

[Fal90b] Falconer,K.J. *Fractal Geometry. Mathematical Foundations and Applications*, John Wiley and Sons, Inc, Great Bretain (1990).

[Fed88] Feder, Jens. *Fractals* Plenum Press, New York (1988).

[Fis92] Fisher,Yuval, *Fractal Image Compression*, SIGGRAPH 92, Course Notes.

[FFH90] Foley, Van Dam, Feiner, Hughes, *Computer Graphics. Principles and Practice*, Addison Wesley, 1990.

[Gol89] Goldberg E.David, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Massachusetts (1989).

[HGB92] Hurd, L.P., M.A. Gustavus, M.F. Barnsley, *Fractal Video Compression*, Digest of Papers, COMPCON Spring 1992, 37th IEEE Computer Society International Conference, p.41-2, 1992.

[Hut81] Hutchinson, Joseph, *Fractals and self-similarity*, Indiana Univ. Math. J. Vol. 30, pp. 713-747 (1981).

[HKR93] Huttenlocher, Daniel P., Gragory A. Klanderman, William J. Rucklindge, *Comparing Images Using the Hausdorff Distance*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15 No.9, September, 1993.

[Jaq89] Jaquin, A.E. *A fractal theory of iterated Markov operators with applications to digital image coding*, Ph.D. Thesis, Georgia Tech (1989).

[Jaq92] Jacquin, Arnaud E, *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations*, IEEE Transactions on Signal Processing, March 1992.

[Koz92] Koza John R., *Genetic Programming, On the Programming of Computers by Means of Natural Selection.* MIT Press, Massachusetts (1992).

[LeG91] Le Gall, Didier J. The MPEG Video Compression Standard IEEE (1991).

[LK90] Lewis, A. S. , Knowles, G. *Video Compression using 3D Wavelet Transforms,* Electronic Letters, IEEE, March, 1990.

[Lin91] Lindley, Craig A., *Practical Image Processing in C*, John Wiley and Sons (1991).

[Man93] Mandelbrot, Benoit B., *The Fractal Geometry of Nature*, W.H. Freeman and Co.,(1993,3Ed).

[MaMo92] Mazel, David, Monson Hayes, *Using Itereted Function Systems to Model Discrete Sequences*, IEEE- Transactions on Signal Processing 40, July 92.

[MoDu92] Monro, D.M., F. Dudbridge, *Fractal Block Coding of Images*, IEEE Electronic Letters, 21st May 1992, Vol.28, No.11.

[Mon93] Monro, D.M. *Class of Fractal Transforms*, IEEE Electronic Letters, 18th February, 1993, Vol.29, No.4

[NaSm93] Nadler, Morton, Eric P. Smith, *Pattern Recognition Engineering*, John Wiley and Sons (1993).

[NiLa93] Nicolas, Henri, Claude Labit, *Motion and Illumination Variation Estimation Using a hierarchy of Models: Application to Image Sequence Coding*, IRISA, Publication Interne No. 742, France, June, 1993.

[PaCl89] Papadopoulos, C.A., T.G. Clarkson, *Data Compression Of Moving Images Using Geometric Transformations*, International Conference on Image Processing and its Applications IEEE (1989).

[Pet92] Perajan, Eric *Digital Video Coding Techniques for US High-Definition TV,* IEEE (1992).

[SaLa89] Safavian, S. Rasoul, David A. Landgrebe, *Use of Robust Estimators in Parametric Classifiers*, IEEE (1989).

[SMD91] Shonkwiler, R, F. Mendivil, A. Deliu, *Genetic Algorithms for the 1-D Fractal Inverse Problem*, Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego (1991).

[Sou92] Soucek, Branko and the IRIS Group, *Dynamic, Genetic, and Chaotic Programming* John Wiley and Sons, Inc (1992).

[ViFa90] Viéville, Olivier Faugeras, *Feed-Forward Recovery of Motion and Structure from a Sequence of 2D-Lines Matches*, Proceedings. Third International Conference on Computer Vision, IEEE (1990).

[VrRo89] Vrscay, Edward, Roehrig, *Computers and Mathematics*, Springer Verlag (1989).

[Vrs90] Vrscay, Edward R. *Moment and Collage Methods of the Inverse Problem of Factal Construction with Iterated Function Systems*, Proceedings of 1st IFIP Conference on Fractals, FRACTAL 90, Lisbon (June 6-8, 1990).

[Wal89] Wallace, Gregory K. *The ISO/CCITT Standard for Continuos-Tone Image Compression*, IEEE, 1989.