Una Evaluación de la Compresión Fractal en Bloques

Andrés Repetto y Claudio Delrieux
Departamento de Ingeniería Eléctrica
Universidad Nacional del Sur
Alem 1253 — Bahía Blanca — ARGENTINA
claudio@acm.org

Palabras Clave: Compresión de Información. Procesamiento de Imágenes. Fractales. Sistemas de Función Iterada.

Resumen

Ante el aumento incesante de la resolución de las imágenes y la utilización masiva de multimedios, día a día el espacio disponible en los dispositivos de almacenamiento se va reduciendo. Del mismo modo, cada vez es mayor la cantidad de imágenes a transmitir a través de las redes y un tamaño excesivo de éstas harían que los costos sean inaceptables. La Compresión Fractal en Bloques (CFB) es relativamente nueva, pero ha evidenciado grandes progresos y concitado el interés de investigadores de todo el ámbito tecnológico y académico. Los buenos resultados obtenidos han hecho que importantes empresas ya la estén utilizando. A modo de ejemplo puede citarse a la Enciclopedia Encarta©, de Microsoft©, cuyas imágenes fueron almacenadas utilizando la tecnología brindada por la compresión fractal.

Existen varios métodos de compresión de imágenes, pero la Compresión Fractal posee características únicas. Éste método se encuentra entre los que codifican la imagen *con pérdida de información*. De todos modos, el error puede llegar a ser muy bajo y como beneficio obtenemos una muy reducida cantidad de bits para caracterizar a la imagen original. El método general de la CFB es muy sencillo ya que se basa en la presunción de que existe redundancia aprovechable en toda imagen. Para ésto la imagen es desarmada en bloques rango y se busca, para cada rango, una transformación de un bloque dominio mayor tal que el dominio transformado se vea similar al rango. La aproximación fractal de la imagen es construída iterando estos mapas sobre una imagen inicial *arbitraria*. La búsqueda de correspondencia entre bloques y la evaluación del error cometido, sin embargo, presenta muchas posibles alternativas desde la búsqueda por fuerza bruta hasta la utilización de algoritmos de optimización diseñados para los métodos numéricos.

No todas las alternativas han sido exhaustivamente exploradas, y la CFB exhibe una gran sensibilidad a cualquier incremento de la eficiencia, produciendo mejores resultados en menor tiempo, o relaciones de compromiso costo–calidad–tiempo más adecuadas. En este trabajo evalúan implementaciones alternativas del método para conocer y comparar sus características principales. La CFB exhibe en algunos casos una buena relación de compromiso entre la compresión, la calidad de la imagen codificada, y el tiempo de compresión y descompresión. Se muestran ejemplos de relaciones de compresión de hasta 100:1 con resultados visualmente aceptables, obtenidos por medio de adecuados ajustes a los parámetros del algoritmo.

Una Evaluación de la Compresión Fractal en Bloques

1 Introducción

Ante el aumento incesante de la resolución de las imágenes y la utilización masiva de multimedios, el espacio de almacenamiento se torna insuficiente, cada vez es mayor la cantidad de imágenes a transmitir a través de las redes y un tamaño excesivo de éstas harían que los costos sean inaceptables. La necesidad de comprimir imágenes se ha vuelto imperiosa. Las tecnologías de compresión pueden dividirse en métodos *sin pérdidas* y métodos *con pérdidas*. En la compresión de imágenes, un método sin pérdidas siempre produce una imagen descomprimida que es idéntica, pixel por pixel, a la imagen original. El problema con esta clase de métodos es que las relaciones de compresión obtenibles son muy pequeñas. Los métodos de compresión con pérdidas pueden lograr relaciones de compresión muy altas. En este grupo se encuentra la Compresión Fractal en Bloques (CFB).

El método general de la CFB es muy sencillo ya que se basa en la presunción de que existe redundancia aprovechable en toda imagen. Para ésto la imagen es desarmada en bloques rango y se busca, para cada rango, una transformación de un bloque dominio mayor tal que el dominio transformado se vea similar al rango. La aproximación fractal de la imagen es construída iterando estos mapas sobre una imagen inicial *arbitraria*. La búsqueda de correspondencia entre bloques y la evaluación del error cometido, sin embargo, presenta muchas posibles alternativas desde la búsqueda por fuerza bruta hasta la utilización de algoritmos de optimización diseñados para los métodos numéricos.

No todas las alternativas han sido exhaustivamente exploradas, y la CFB exhibe una gran sensibilidad a cualquier incremento de la eficiencia, produciendo mejores resultados en menor tiempo, o relaciones de compromiso costo–calidad–tiempo más adecuadas. De ahí la gran cantidad de literatura e implementaciones que han surgido en los últimos cinco años. En este trabajo se busca evaluar un grupo de implementaciones alternativas del método para conocer y comparar sus características principales, y proponer nuevas alternativas que permiten mejorar la eficiencia de los resultados. En la próxima Sección se presenta un repaso del fundamento teórico del método, el cual se basa en los sistemas de función iterada. En la Sección 3 se desarrollan las estrategias de diseño de un sistema CFB, las cuales son implementadas y evauadas en la siguiente Sección. Por último, se discuten las conclusiones y se proponen algunas líneas actualmente investigadas por los autores.

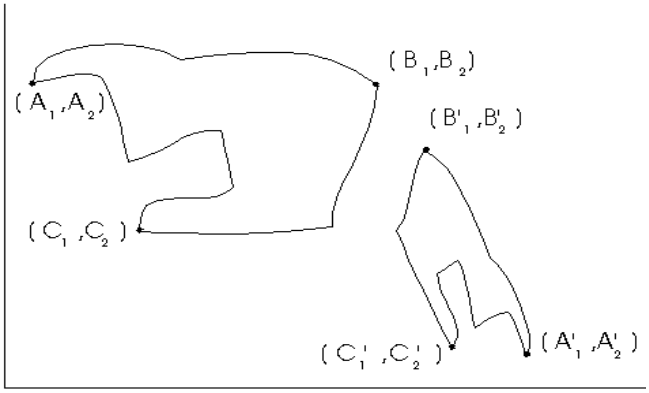

Figura 1: Dos siluetas fijan una transformación afín $W : \Re^2 \to \Re^2$.

2 Sistemas de Función Iterada

La teoría de la CFB se basa en los sistemas de función iterada, por lo cual veremos en esta Sección los resultados teóricos más importantes. Aquellos lectores familiarizados con el tema pueden pasar a la próxima Sección. Un Sistema de Función Iterada (IFS) consiste en una colección de transformaciones afines contractivas que mapea al plano $\Re^2$ sobre sí mismo [2]. Esta colección de transformaciones define un mapa

$$W(\cdot) = \bigcup_{i=1}^{n} w_i(\cdot).$$

El mapa W no es aplicado a todo el plano sino a un subconjunto, es decir, a una colección de puntos del plano. Dado un conjunto inicial S, podemos calcular $w_i(S)$ para cada i, tomar la unión de estos conjuntos y obtener un nuevo conjunto $W(S)$. W es un mapa en el espacio de los subconjuntos del plano. Un teorema importante de Hutchinson [7] es que cuando las transformaciones w_i son contractivas en el plano, entonces W es contractiva en el espacio (cerrado y acotado) de subconjuntos del plano. Las transformaciones contractivas tienen la importante característica que cuando son aplicadas repetidamente, convergen a un punto fijo.

Dos figuras de aproximadamente igual forma pueden ser utilizadas para determinar una transformación afín $w : \Re^2 \to \Re^2$. A modo de ejemplo, tomemos una silueta cualquiera, como la de la Figura 1. Deseamos hallar los números reales a, b, c, d, e y f tales que la transformación de un punto (x, y) de la *Silueta Grande* sea mapeado a la *Silueta Chica*:

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

y que tenga la propiedad de ser w(*Silueta Grande*) aproximadamente igual a *Silueta Chica*. Marcando tres puntos de la silueta grande y determinando sus coordenadas (A_1, A_2), (B_1, B_2) y (C_1, C_2), y lo mismo para los respectivos puntos de la silueta chica (A'_1, A'_2), (B'_1, B'_2) y (C'_1, C'_2), es posible obtener a, b y e para la coordenada x y c, d y f para la coordenada y, resolviendo respectivamente los sistemas de tres ecuaciones lineales:

$$\begin{aligned} A_1 \cdot a + A_2 \cdot b + e &= A'_1 \\ B_1 \cdot a + B_2 \cdot b + e &= B'_1 \\ C_1 \cdot a + C_2 \cdot b + e &= C'_1 \end{aligned} \qquad \begin{aligned} A_1 \cdot c + A_2 \cdot d + f &= A'_2 \\ B_1 \cdot c + B_2 \cdot d + f &= B'_2 \\ C_1 \cdot c + C_2 \cdot d + f &= C'_2 \end{aligned}$$

Una transformación afín está entonces completamente especificada por seis números reales, y es llamada *contractiva* si para todo par de puntos, la distancia entre los mismos se reduce luego de ser aplicada. Un *código IFS bidimensional* consiste en un conjunto de N transformaciones afines $\{w_1, w_2, w_3, \ldots, w_N\}$, donde cada w_i transforma puntos de $\Re^2 \rightarrow \Re^2$. También es requerido un conjunto de probabilidades $\{p_1, p_2, p_3, \ldots, p_N\}$, donde cada $p_i > 0$ y $p_1 + p_2 + p_3 + \ldots + p_N = 1$.

Un algoritmo utilizado en [11] para recuperar la imagen a partir de las transformaciones, comienza con el código IFS $\{w_n, p_n : n = 1, 2, \ldots, N\}$ junto con la ventana de graficación V de resolución $L \times M$. Un arreglo $I[L][M]$ de enteros se asocia a la ventana. También se tiene un número total de iteraciones num, (grande comparado con $L \times M$). En el arreglo se lleva cuenta de las veces que es "visitado" cada pixel. El algoritmo realiza una recorrida aleatoria del siguiente modo: parte de un punto inicial y escoge, al azar, aunque tomando en consideración la probabilidad p_n, un mapa w_n y transforma el punto mediante este mapa, obteniendo las coordnadas de un pixel. Luego incrementa en uno el "contador de visitas" que le corresponde a ese pixel. Después es hecha otra selección aleatoria y el mapa correspondiente se aplica al punto hallado mediante la transformación previa. Este proceso continúa de la misma manera hasta alcanzar el número total de iteraciones. La cantidad de veces que es visitado un pixel depende en forma directa de la medida (o índice de probabilidad) que tenga cada transformación afín.

Para la obtención del código IFS de una figura se parte de una representación inicial aproximada de la silueta S del objeto que se desea modelar. La idea del método es buscar un grupo de mosaicos —copias transformadas de S— que en conjunto cubran la silueta original. El método tiene su sustento matemático en el Teorema del Collage [2], pero hasta ahora no ha podido implementarse en forma automática, debiéndose trabajar con asistencia humana. A partir de la silueta S, son introducidas una a una copias reducidas de la S, de modo de poder ir cubriéndola con sus copias transformadas. Cada copia podrá ser torcida, escalada, rotada o trasladada según convenga para lograr el propósito buscado. Es importante que el tamaño de cada $w_n(S)$ sea menor que el de S para asegurar que cada transformación sea contractiva. Para que el número de transformaciones sea el menor posible, el solapamiento entre las w_n debe ser sólo el necesario. Por otra parte, dejar sin cubrir algún sector significará ausencia de puntos en esa zona cuando la imagen sea reconstruida. Muchas veces este efecto es deseado. En otras palabras, para lograr la codificación se deben determinar un conjunto de transformaciones afines contractivas $\{w_1, w_2, \ldots, w_n\}$ con la siguiente propiedad: La silueta original S y el conjunto

$$\tilde{S} = \bigcup_{n=1}^{N} w_n(S)$$

deben ser visualmente cercanos, siendo el número de transformaciones el menor posible. $w_n(S)$ es llamado el n-ésimo mosaico del collage. Pueden encontrarse más detalles en [11]. En la Figura 2 se muestra un atractor obtenido utilizando el procedimiento descripto.

3 Compresión Fractal en Bloques

Como vimos en la Sección anterior, una clase específica de fractales puede ser recreada por medio de IFS. El problema ahora consiste en que, dada cualquier imagen discreta original especificada como un conjunto de pixels, lograr que una computadora construya

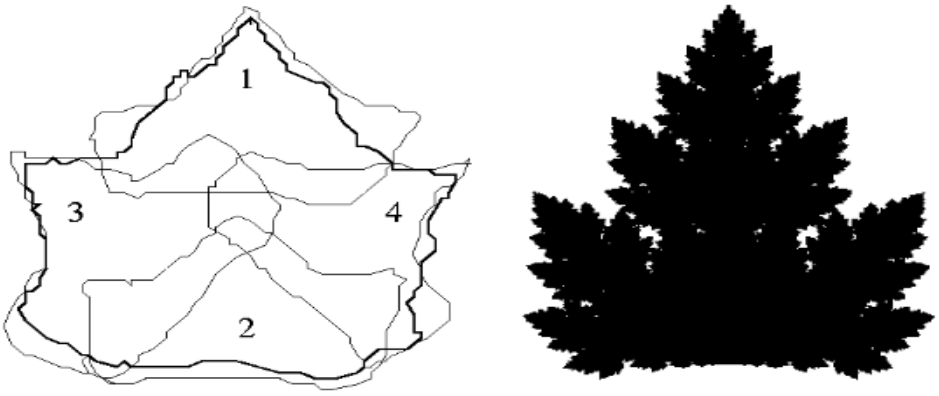

Figura 2: Cubrimiento de una hoja para la obtención de los Códigos IFS y atractor que se logra con ellos.

automáticamente una imagen fractal —la imagen codificada— que sea a la vez próxima a la original y que tenga una representación digital que requiera menos bits que la imagen original. Basándonos en la suposición de que la redundancia de la imagen puede ser eficientemente capturada y aprovechada a través de piezas que sean *auto-transformables* originadas de a bloques, es posible pensar en aproximar una imagen original por una *imagen fractal*, obtenida a partir de un número finito de iteraciones de una serie de transformaciones. La forma general de la compresión fractal se basa en la obtención de un conjunto de transformaciones contractivas para las cuales la imagen original es aproximadamente un punto fijo— el cual, cuando es aplicado iterativamente sobre *cualquier* imagen inicial, produce una secuencia de imágenes que convergen a la aproximación fractal de la imagen original.

Para que el método tenga aplicabilidad, es necesario que funcione en forma automática, sin asistencia humana como sucede con los IFS. Por dicha razón, una de las primeras simplificaciones es buscar un cubrimiento por medio de transformaciones cuya geometría sea lo más sencilla posible. De esa manera se divide la imagen a comprimir en un mosaico de *bloques rango* (o imagen) cuadrados. Para cada bloque rango, se busca un bloque dominio de tamaño mayor el cual, debidamente trasladado y escalado y probablemente rotado un múltiplo de $90°$, se asemeja al bloque rango con un cierto error mínimo. Este tipo de aproximación se denomina *Compresión Fractal en Bloques* (CFB). El código fractal consiste entonces en una descripción de una partición de la imagen y una serie de transformaciones, especificados por un pequeño conjunto de parámetros. En esta Sección veremos la fundamentación teórica de la CFB, la evaluación de la calidad de una aproximación, y las diferentes estrategias de implementación.

Sea (M, d) un espacio métrico de imágenes digitales, donde d es una métrica dada —medida de distorsión, y sea μ_{orig} una imagen original que deseamos codificar. El problema inverso de la teoría de transformaciones iteradas consiste en la construcción de una transformación contractiva de imágenes τ, definida del espacio (M, d) sobre sí mismo, para la cual μ_{orig} es aproximada por un *punto fijo*. Llamaremos $\mathcal{F}$ al conjunto de todas las transformaciones permitidas. Teniendo en cuenta que τ tiene una menor complejidad que la imagen original, se puede ver a τ como a una *imagen codificada con pérdida calidad* respecto de la imagen original μ_{orig}. Luego de cierto número de iteraciones iniciales los

términos de cualquier secuencia iterada de la forma

$$\{\mu_n = \tau^n(\mu_0)\}_{n\geq 0} \tag{1}$$

se aproxima a la imagen inicial, donde μ_0 es *cualquier* imagen inicial. En un espacio de imágenes discretas cuantizadas, la secuencia converge exactamente a una imagen fractal estable. La proximidad de $\tau^n(\mu_0)$ a μ_{orig} está condicionada por la distorsión $d(\mu_{orig}, \tau(\mu_{orig}))$. El hecho de que la redundancia de la imágenes digitales pueda ser capturada a través de bloques auto-transformables, llevó a investigar una clase de transformaciones de imágenes de la forma genérica [5, 9]

$$\forall \mu \in M, \tau(\mu) = \sum_{0\leq i<N} (\tau\mu)_{\rceil R_i} = \sum_{0\leq i<N} \tau_i(\mu_{\rceil D_i}) \tag{2}$$

donde $\mathcal{P} = \{R_i\}_{0\leq i<N}$ denota una partición no superpuesta de la imagen soporte en N *celdas rango* —generalmente bloques cuadrados de diferentes tamaños posibles. El símbolo $\mu_{\rceil R_i}$ indica la restricción de la imagen μ a la celda R_i; la llamamos imagen bloque sobre R_i y escribimos

$$\mu = \sum_{0\leq i<N} \mu_{\rceil R_i} \tag{3}$$

simplemente para indicar que una imagen es la unión de sus restricciones de celdas particionadas. El símbolo τ_i denota una transformación bloque elemental desde la *celda dominio* D_i a la *celda rango* R_i. Para mayor claridad, se considera que $\tau_i = T_i \circ S_i$ es composición de dos transformaciones S_i y T_i llamadas *geométrica* y *másica* (o cromática), respectivamente. La construcción de un código fractal τ para μ_{orig} —una transformación contractiva de imagen de la forma descripta recién bajo la cual μ_{orig} es aproximadamente auto-transformable de a bloques— se hará definiendo las τ_i elementales separada e independientemente unas de otras. Así, la codificación fractal en bloques consiste en encontrar para cada índice de partición i, una transformación τ_i desde una celda dominio D_i a una celda rango R_i tal que el bloque dominio transformado $\tau_i(\mu_{orig\rceil D_i})$ es una aproximación cercana al bloque rango original $\mu_{orig\rceil R_i}$.

Como ya se mencionara, la CFB consiste elaborar un IFS con transformaciones restringidas a celdas dominio y rango cuadrados. El cuadrado soporte S de la imagen digital original μ_{orig} es particionado en celdas cuadradas no superpuestas de dos tamaños diferentes, formando así una *partición cuadrada en dos niveles*, semejante a los *quadtrees*. Las celdas más grandes —de tamaño $B \times B$— serán llamadas *padres*, mientras que las pequeñas —de dimensión $\frac{B}{2} \times \frac{B}{2}$— son llamadas *hijos*. Un padre puede ser particionado en hasta cuatro hijos no superpuestos. Las decisiones sobre los particionamientos de las celdas padres pueden ser hechas adaptativamente durante la codificación. Una partición realizada de esta manera es dependiente de la imagen, permitiendo que el programa que comprime (codificador) utilize bloques mayores para aprovechar variaciones suaves en ciertas áreas de la imagen, y utilize bloques más pequeños para capturar detalles en partes complejas tales como contornos rugosos y texturas finas. La selección de los tamaños y formas para las celdas imagen depende de varios factores. Bloques de imagen pequeños —4×4 y menores— son sencillos de analizar y clasificar geométricamente, permitiendo una rápida evaluación de la distancia inter-bloque, y son además fáciles para codificar con exactitud y conducen a un sistema de encodificación robusto —uno cuya perfomance es constante, incluso cuando las imágenes fuente varíen. Por otra parte, bloques grandes —5×5 y mayores— permiten una mayor explotación de la redundancia en zonas suaves de la imagen y conducen a mayores rangos de compresión.

Para evaluar la calidad de una codificación, debemos determinar una medidia de distorsión entre imágenes digitales a partir de una medida de distorsión inter-bloque. Sea $S(i_0, j_0, B)$ el bloque cuadrado de tamaño $B \times B$, con la esquina izquierda inferior ubicada en la intersección de la fila i_0, columna j_0 de la imagen. Sea μ una imagen $r \times r$ y $\tilde{\mu}$ una aproximación de μ. Sean $\mu_{\restriction S}$ y $\tilde{\mu}_{\restriction S}$ quienes denoten sus restricciones a la celda $S(i_0, j_0, B)$ y $\mu_{i,j}$ el nivel de gris del pixel (i, j). La distorsión media cuadrática L_2 (o error *rms*) entre los bloques imagen $\mu_{\restriction S}$ y $\tilde{\mu}_{\restriction S}$ se define como la raíz cuadrada de la suma sobre la celda S, de las diferencias cuadráticas de los valores de los pixels, i.e.:

$$d_{L_2}(\mu_{\restriction S}, \tilde{\mu}_{\restriction S}) = \left(\sum_{0 \le i,j < B} (\mu_{i_0+i, j_0+j} - \tilde{\mu}_{i_0+i, j_0+j})^2 \right)^{1/2}. \tag{4}$$

Esta medida de la distorsión permite ahora encontrar el mejor bloque dominio para el cual existe una transformación $\tau_i = T_i \circ S_i$ que minimiza el error rms para aproximar un determinado bloque rango.

El procedimiento de compresión comienza entonces tomando uno por uno los bloques rango, y para cada uno de ellos busca un bloque dominio y una transformación que lo aproxime. Supongamos por un momento que la búsqueda de bloques dominio se hace en forma exhaustiva. Para cada uno de ellos es necesario determinar cuál es la transformación que minimiza el error rms. En este caso también podría hacerse una búsqueda exhaustiva, lo cual haría que el método fuese intratable computacionalmente. Por lo tanto se trabaja con un conjunto restringido de transformaciones. La parte geométrica S_i de la transformación mapea los bloques desde la celda dominio $D_i = S(i_d, j_d, D)$ a la celda rango $R_i = S(i_r, j_r, B)$. En el caso simple en que $D = 2B$, el valor de los pixels de la imagen contraída sobre el bloque rango $S(i_r, j_r, B)$ es el valor promedio de cuatro pixels del bloque dominio. Normalmente no podemos esperar que esta simple operación sea adecuada en todos los casos, pero es posible aplicar antes alguna de las siguientes transformaciones (que simplemente traspasan pixels en un bloque rango) llamadas *isometrías*.

1. Reflexión ortogonal del bloque sobre el eje medio vertical ($j = \frac{B-1}{2}$).
2. Reflexión ortogonal del bloque sobre el eje medio horizontal ($i = \frac{B-1}{2}$).
3. Reflexión ortogonal del bloque sobre la primera diagonal ($i = j$).
4. Reflexión ortogonal del bloque sobre la segunda diagonal ($i + j = B - 1$).
5. Rotación alrededor del centro del bloque, en $+90°$.
6. Rotación alrededor del centro del bloque, en $+180°$.
7. Rotación alrededor del centro del bloque, en $-90°$.

Estas transformaciones geométricas son sencillas y eficientes de implementar y, acompañadas por una transformación másica o cromática adecuada, son normalmente suficientemente buenas como para lograr compresión con calidad.

La parte "másica" (o cromática) T_i se encarga de afectar a los valores de luminancia o color de los pixels del bloque. Unas pocas transformaciones simples definidas en el espacio de los bloques imagen discretos que se encuentran en celdas rango son

- Escala de Contraste (α) : $(\sigma\mu)_{i,j} = \alpha\mu_{i,j}$.
- Desplazamiento de Luminancia (β) : $(\tau\mu)_{i,j} = \mu_{i,j} + \beta$.

Debe tenerse en cuenta que la cantidad de bits que se determinen para almacenar estos valores influirá en forma directa tanto en la proximidad que logremos entre el atractor y la imagen original, como en la relación de compresión obtenible. De ahí la importancia de la cuantización de éstas variables.

4 Evaluación de diferentes estrategias de implementación

Una de las primeras implementaciones documentadas es la de Hurd [1], la cual es simple y de alcances muy limitados. Se divide a la imagen a codificar en bloques rango de 4×4 pixels. Los bloques dominio son tomados del doble de este tamaño. Luego recorre todos los dominios y a cada uno, vez por vez, lo lleva al tamaño del rango y le aplica un cambio de escala de luminancia (α) constante. El desplazamiento de luminancia (β) lo calcula como la diferencia del promedio de luminancia del bloque rango y del bloque dominio transformado. Luego elige el menor error rms con cada una de las ocho isometrías. El procedimiento no tiene un buen comportamiento cuando la imagen a codificar posee grandes zonas contiguas blancas y negras. Como las transformaciones geométricas (dominio e isometría) son calculadas correctamente, la silueta de la figura puede ser recuperada, pero la imagen obtenida es mala pues las transformaciones másicas son las que fallan. En la Figura 3 puede verse el proceso de decodificación. En este caso se hicieron hasta veinte iteraciones para mostrar que aumentando este número la imagen no mejora.

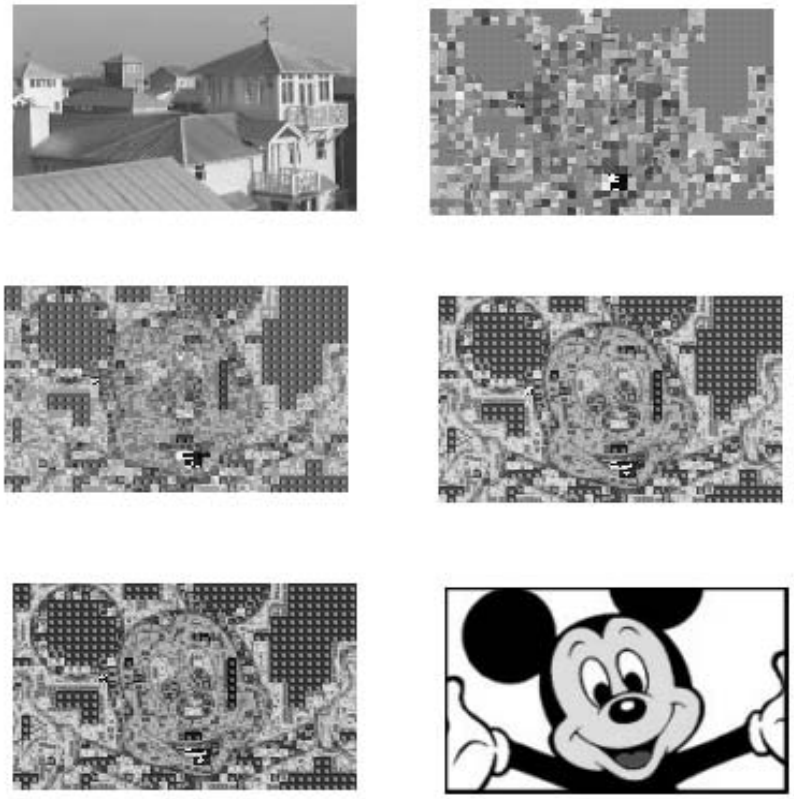

Figura 3: Ejemplo en donde falla la implementación de Hurd. Imagen inicial, 1, 2, 4, 20 iteraciones y la imagen original.

Otro programa que implementa CFB es el realizado por Y. Fisher con propósitos educativos [6]. Este algoritmo trabaja de la siguiente manera. Primero calcula, para cada dominio, la sumatoria y la sumatoria al cuadrado de los valores de gris de cada pixel. Luego estos valores serán utilizados en los cálculos del error medio cuadrático (rms). Como cada dominio es comparado con muchos rangos, es conveniente realizar ésto una sóla vez al comienzo para ahorrar tiempo. Luego el programa divide a cada dominio en cuatro cuadrados a los que ordena por orden de luminancia, para facilitar la determinación de la isometría apropiada. Para realizar la codificación se utiliza el esquema de particionamiento *quadtree* que permite encontrar bloques dominio de tamaño adaptatiavo. Se clasifica al bloque rango según la luminancia de cada uno de sus cuatro partes. Luego se recorren todos los bloques dominio para buscar el del mínimo el error rms. La isometría se obtiene en forma inmediata pues a ambos bloques les fue realizado el pre-ordenamiento por brillo.

Para cada bloque dominio es necesario calcular el α y el β —escala y desplazamiento de luminancia, respectivamente— que hacen mínimo al error rms entre el bloque rango y bloque dominio candidato. α y β deben ser luego cuantizados y con dichos valores se recalcula el error rms. Luego de recorrer todos los dominios, tendremos al que tiene el menor error rms para ese rango en particular. Si el mínimo hallado de ésta forma es suficientemente pequeño o si el nivel de particionamiento quadtree llegó a su límite, se asigna al bloque rango considerado el bloque dominio encontrado. En caso contrario se vuelve a particionar y se sigue buscando recursivemente. El usuario tiene la oportunidad de seleccionar algunos parámetros de la codificación. Entre otros, se puede indicar la cantidad de bits utilizados al cuantificar, indicar el nivel de particionamiento e introducir un valor indicativo del error tolerado. El resultado que puede observarse en la Figura 4 muestra la imagen obtenida por medio de 6133 transformaciones, buscándose la mejor calidad posible. El tiempo de codificación fue de 13'23", lográndose una relación de compresión de 11.07:1. Para resaltar una vez más la independencia del atractor final de la imagen inicial a las que se le aplica las transformaciones, se partió de una imagen completamente negra.

Uno de los aspectos más destacados de la Compresión Fractal es la posibilidad de realizar ampliaciones sin que el efecto de pixelización se ponga de manifiesto. En la Figura 5 se puede ver las imágenes logradas al ampliar 5 veces la figura original y la fractal. Puede verse que en el segundo caso las transformaciones reconstruyen los detalles necesarios.

Una implementación más moderna, y que permite ser modificada para realizar ensayos y optimizaciones, es la de Pulcini y otros [10]. En particular, permite que un usuario programador modifique el algoritmo de búsqueda de dominios, lo cual como veremos termina siendo el factor crítico en la relación calidad–tamaño. Se comienza con la imagen en escala de grises, la cual es subdividida en bloques rango de 4×4 pixels (aunque el tamaño de los bloques rango puede ajustarse). Para cada uno de ellos se busca en la toda la imagen un bloque dominio de 8×8 pixels, el cual siendo *reducido* (al tamaño de 4×4), transformado por una de los 8 isometrías y escalado en luminancia, sea la mejor aproximación del bloque rango original de 4×4. De esta forma, la imagen original es codificada como un conjunto de transformaciones (escala, aplicación de una isometría, escala y desplazamiento de luminancia). En la imagen codificada no hay valores correspondientes a ningún pixel, sino sólo los valores de los parámetros involucrados en las transformaciones aplicadas a cada bloque dominio para la obtención de los bloques rango.

La forma más simple y lenta del encontrar el bloque correspondiente a un bloque rango en particular es ir probando cada bloque dominio. Ésta es la *búsqueda exhaustiva*.

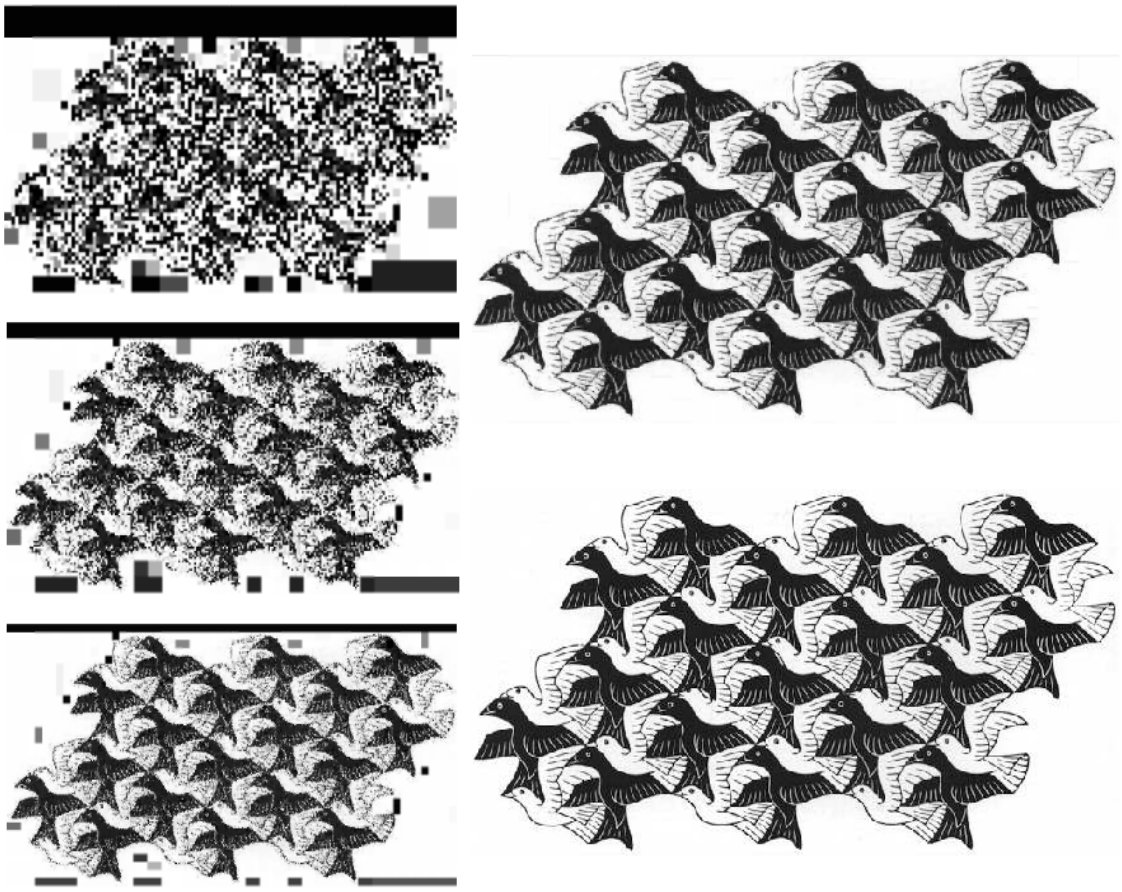

Figura 4: Resultado luego de las iteraciones 1, 2, 3 y 10, y la imagen original.

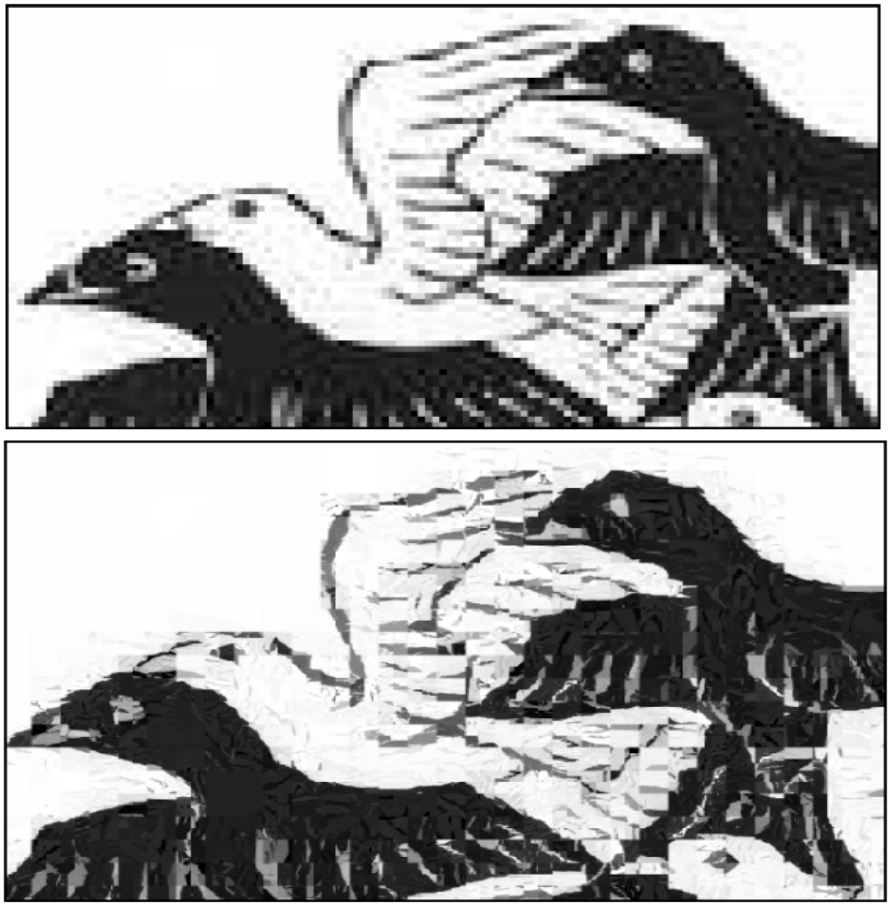

Figura 5: Ampliación de la imagen original y la fractal.

Es posible pensar que se desplaza una ventana cuadrada de 8×8 a lo largo y a lo ancho de la imagen original y que para cada posición de la ventana se aplican las transformaciones y calcula el error rms. Cuando toda la imagen sea explorada, simplemente se selecciona la posición en que la ventana (que identifica la ubicación del bloque dominio) sea la que tiene el mínimo error rms. Sin embargo, como los bloques dominio son contraídos a la mitad de su tamaño original, podemos mover la ventana en pasos de a 2 pixels sin un cambio apreciable para el bloque dominio y de esta manera llevar la búsqueda a la mitad. Es fácil ver que cuanto mayor sea el paso, menor será la búsqueda y mayor la velocidad de compresión. En contrapartida, pasos demasiado grandes llevarán a una búsqueda muy pequeña y a una mala calidad de la imagen decodificada. Ésto permite determinar un *parámetro de paso de dominios*.

Como mejora inmediata a la búsqueda de a pasos podemos pensar lo siguiente: podríamos movernos de a pasos grandes, evaluar el error rms, y cuando obtengamos el mínimo, buscar en las proximidades con un paso menor. Precisamente ésto es lo que hace la *Sub Búsqueda Local*. Los parámetros de Salto sirven para indicar cuántos Saltos de Dominios debe realizar en la búsqueda del mínimo, antes de que la búsqueda se realice con el paso corto en el cuadrado de dimensiones (Salto * Paso de Dominios) pixels. Otra alternativa para reducir el tiempo de la codificación es buscar bloques dominio semejantes sólo en una vecindad cuadrada del bloque rango original. Ésto es lo que hace la *Búsqueda Local*. No sería difícil combinar ésta variante con la anterior.

En este programa se utiliza el espacio cromático YIQ para explotar la diferente sensibilidad que tiene el ojo humano a los canales Y, I, Q. El ojo humano puede percibir transiciones abruptas de brillo mucho más fácilmente que en crominancia. Además el ojo es más sensible a las transiciones en el rango de colores naranja-azul (en donde están los tonos carne) que en el rango púrpura-verde. La Y representa al brillo. Con este canal sólo es posible obtener la imagen en blanco y negro. Los coeficientes de conversión están basados en la respuesta relativa del ojo humano al brillo del rojo, verde y azul. La I corresponde al eje naranja-azul, mientras que la Q se refiere al eje púrpura-verde. Para cada bloque rango se busca al mejor bloque dominio en el canal Y y luego se usan los bloques I, Q que tienen las mismas coordenadas en la imagen. Obviamente no existe para los canales I, Q la *mejor* aproximación. Para los canales I, Q se acepta un mapeo subóptimo. De esta manera sólo se tiene una transformación geométrica (la isometría y la dirección del bloque dominio seleccionado) para las tres componentes YIQ, mientras que se calculan las transformaciones másicas óptimas para cada uno de los tres canales YIQ. Utilizando esta aproximación se ahorra tiempo de compresión y se aumenta la relación de compresión si la comparamos con la forma obvia de codificar separadamente los tres canales.

Podemos ver en la siguiente secuencia de imágenes una foto digitalizada a resolución 512×512 *true-color* (aproximadamente 767 Kbyte en formato .bmp). Con una subbúsqueda local se la codificó con una relación de compresión de 7.11:1. En la Figura 6 se ve el resultado de decodificarla luego de aplicar las transformaciones a una imagen inicial cuadriculada. Luego de 10 iteraciones el resultado es prácticamente indistinguible del original. Luego se buscó una compresión elevada, admitiéndose un empeoramiento en la calidad del atractor. Para ésto se tomaron bloques rango de 16×16 y una búsqueda restringida. El tiempo de codificación necesitado fue de 2'44", consiguiéndose una relación de compresión de 93.24:1. Para la decodificación se requirió de 44" para 10 iteraciones, aunque para este caso con 6 iteraciones (a los 26") la imagen ya estaba estabilizada. El

Figura 6: Iteraciones 2, 3, y 10, junto con la imagen original.

costo de esta compresión es la calidad final. Buscando una solución de compromiso, se tomaron bloques de 8×8. En esta oportunidad los códigos fueron obtenidos en 4'19", comprimiéndose la imagen original en una relación de 23.55:1. El tiempo requerido para la descompresión —tomando 10 iteraciones— fue de 42". De este modo la calidad de la imagen mejoró, pero para ello fue preciso resignar relación de compresión (ver Figura 7). Otras mejoras que están siendo actualmente implementadas se basan en actualizar la imagen que se está iterando cada vez que se encuentra un bloque dominio adecuado. De esa manera es posible mostrar una convergencia más rápida a menores errores rms, resultado que está siendo evaluado.

Figura 7: Se comprimió a 93:1. Luego se mejoró la calidad, con compresión 23:1.

5 Conclusiones

En todo el ámbito de la computación, día a día se utilizan más y mejores imágenes. Ésto requiere grandes espacios de almacenamiento e importantes tiempos cuando se desea transmitirlas. Para reducir los costos, la compresión se ha vuelto una necesidad. Una de las importantes alternativas surgidas es la Compresión Fractal en Bloques. Con un error que puede llegar a ser muy pequeño, se pueden lograr importantes relaciones de compresión. Imágenes comprimidas mediante este método ya están siendo utilizadas en productos comerciales. Aún así, es permanente tema de publicación debido a las numerosas variantes que soporta.

En este trabajo se repasaron los fundamentos de la teoría de sistemas de función iterada (IFS) y el desarrollo de algoritmos de compresión fractal en bloques (CFB). Además se discutieron algunas estrategias de implementación y compararon los resultados de algunos programas disponibles y de determinadas mejoras en los mismos. Como resultado de las evaluaciones, se vio que la calidad, el tiempo y el nivel de compresión logrados tienen entre sí una relación inmediata. Mediante ejemplos se pudo ver cómo, al momento de codificar, se pueden ajustar los parámetros para conseguir resultados diversos. Se obtuvieron niveles de compresión muy altos, con una calidad que si bien es baja, el atractor muestra de manera evidente la escena que representa. De la misma forma, se codificaron imágenes de una calidad superior y con un nivel de compresión interesante. Otra de las características que fue mostrada de este método es la posibilidad de realizar ampliaciones sin que se ponga de manifiesto el efecto de la pixelización. En general, si la ampliación no es excesiva, los resultados obtenidos son excelentes. Esta particularidad de la Compresión Fractal es posible pues lo que se almacena en el archivo comprimido no es la imagen, sino la *forma de obtenerla*.

Una reflexión de gran importancia es que distintas estrategias de obtención del código CFB tienen distintas eficiencias, en tiempo y en relación de compresión. Una estrategia que podría ser más rápida consiste en descomponer la imagen en dominios triangulares, dado que ésto reduce las simetrías y simplifica las transformaciones. Además, todos los

métodos reseñados aquí se basan en iterar la imagen *completa* durante la descompresión. Una de las estrategias que surge como posibilidad para mejorar el tiempo de descompresión es ir utilizando los bloques ya aproximados, de manera de acelerar la convergencia, de una manera similar a las técnicas de relajación en métodos numéricos.

En conclusión, el método ha demostrado ser valioso por sí mismo. En el campo de la Compresión de Imágenes Digitales, la Compresión Fractal está cumpliendo un rol muy importante y, por características que le son propias, está llamada a ocupar un lugar entre las mejores alternativas.

Referencias

[1] L.F. Anson. Fractal Image Compression. *Byte*, pages 195-202, October 1993.

[2] M.F. Barnsley. *Fractals Everywhere*. Academic Press, San Diego, 1988.

[3] M. Barnsley and L. Hurd. *Fractal Image Compression*. AK Peters, Wellesley, Ma., 1992.

[4] Y. Fisher. Fractal image compression. In *SIGGRAPH '92*, 1992. Fractal Course Notes.

[5] Y. Fisher, E.W. Jacobs, and R.D. Boss. Fractal image compression using iterated transforms. In J. Storer, ed., *Image and Text Compression*, pages 35–61. Kluwer, Boston, MA, 1992.

[6] Y. Fisher, editor. *Fractal Image Compression: Theory and Application*. Springer, New York, 1995.

[7] J. E. Hutchinson. Fractals and self similarity. *Indiana Math. Journal*, 3(5):713–747, 1981.

[8] A. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[9] A. Jacquin. Fractal image coding: A review. *Proc. of the IEEE*, 81(10):1451–1465, 1993.

[10] G. Pulcini, V. Verrando, R. Rossi and C. Meloni. *IFS Application Framework (IFSAF)*. "La Sapienza" University, Rome, 1995.

[11] A. Repetto, P. Amaya y C. Delrieux. Síntesis de Imágenes mediante Sistemas de Funciones Iteradas. *XVI CLEI*, Asunción del Paraguay, Septiembre de 1990.