# Compresión de Imágenes vía auto-similitud

Diego F. Fonseca. V. *

**Resumen.** Dada una imagen, por conveniencia en escala de grises, existen regiones que son similares a otras regiones de la misma imagen bajo ciertas transformaciones, la similitud es determinada por una adecuada medida para comparar imágenes, esta medida también induce una noción de cercanía entre imágenes, con esta propiedad en mente se pretende construir una aplicación contractiva en el contexto de las imágenes cuyo punto fijo, que también sera una imagen, este cercana a la imagen dada inicialmente, los datos que permiten describir la aplicación contractiva constituyen la compresión de la imagen inicial.

**Palabras clave:** Auto-similitud, compresión, optimización.

## 1 Introducción

La idea de la compresión a través de la caracterización de propiedades fractales en las imágenes naturales tiene su desarrollo ligado al desarrollo de las ciencias computacionales, especialmente 1995 es un año importante en la historia de el estudio de este tema, en ese año se expusieron precisos trabajos en el tema de compresión de imágenes, [2] es uno de dichos trabajos; los resultados acerca de este tema se han incrementado debido al aumento de la capacidad y velocidad de los computadores ademas de la facilidad para obtener un computador con buenas características para llevar a cabo experimentos numéricos eficientemente, por ejemplo, en [1] para abordar la medida del grado de auto-similitud de una imagen se evidencia que dicho estudio es solo posible gracias a los computadores.
Precisamente, el método que expondremos en el presente articulo es conocido como *compresión fractal*, esta es una técnica que usa la auto-similitud de una imagen para representarla sin necesidad de tomar toda la imagen, es decir, existe partes de una imagen que después de aplicarle adecuadas transformaciones afines (traslaciones y rotaciones), de contraste y de brillo son similares a otras partes de la misma imagen, si particionamos la imagen en bloques disjuntos cada uno de estos bloques determina una transformación adecuada que básicamente lo que hace es traer la región de la imagen que es similar al bloque dado, cada una de estas transformaciones sera contractiva y al unirlas para determinar una transformación global esta también sera contractiva, la cual tendrá un punto fijo que estará cerca a la imagen inicial, es decir, ese punto fijo es una estimación de la imagen inicial, de modo que basta con tener en cuenta tan solo los datos que determina la transformación global, la cantidad de dichos datos es mucho menor a los datos que determinan la imagen inicial. Esta idea es descrita en varios artículos como [1] y [2], pero en ellos no se muestra un algoritmo especifico, únicamente ilustran la idea, por esta razón en este ariticulo intentamos describir un algoritmo concreto y para tal fin evidenciaremos que en este proceso esta inmerso un problema de *optimización convexa* el cual abordaremos mediante el *Método de Barrera* donde en cada iteración usaremos el *Método de Newton modificado con Backtracking.*

Es importante tener en cuenta que la existencia del punto fijo de la transformación descrita es un consecuencia del Teorema del punto fijo de Banach:

**Teorema 1.** ***(Banach fixed-point)*** *Sea $(X, d)$ un espacio métrico completo no vació, y $T : X \to X$ una contracción, i.e, $d(T(x), T(y)) < \lambda d(x, y)$ con $0 \leq \lambda < 1$ para todo $x, y$ en $X$. entonces existe un único punto $p$ en $X$ tal que $T(p) = p$, $p$ es llamado punto fijo.*

* Universidad de los Andes,Bogotá, Colombia.

Para garantizar que el punto fijo de la transformación esta cerca a la imagen inicial dada se debe garantizar que la imagen inicial esta lo más cerca posible a la primera iteración de la imagen inicial bajo dicha transformación, la noción de distancia va ligado a la de similitud que es una cuestión que abordaremos posteriormente, lo que acabamos de decir es una consecuencia de un corolario del Teorema 1 conocido como *Collage Theorem*:

**Corolario 1.** ***(Collage theorem)*** *En términos del Teorema del punto fijo de Banach,*

$$d(x,p) \leq \frac{1}{1-\lambda} d(x, T(x))$$

*for all* $x \in X$.

Estos dos últimos resultados fundamentan la idea de la compresión fractal, a lo largo del texto especificaremos quien es $X$ y $d$ en el contexto de las imágenes.

## 2 El espacio de la imágenes en escala de grises

En esta sección definiremos el espacio y la métrica en lo que resta de este articulo. En primer lugar, definiremos formalmente el espacio de las imágenes de tamaño $n \times n$ en escala de grises (i.e. en blanco y negro), este articulo se restringe a imágenes cuadradas.

**Definición 1.** *El conjunto de* ***las imágenes de tamaño*** $n \times n$ ***en escala de grises****, denotado por* $F_n$*, es definido como el conjunto*

$$F_n := \{A \in M_n(\mathbb{R})^n \mid 0 \leq A(i,j) \leq 255 \text{ para cada } (i,j)\}$$

*donde* $M_n(\mathbb{R})^n$ *es el conjunto de las matrices de* $n \times n$.

Note que en la definición anterior se esta permitiendo que $A(i,j)$ tome valores no necesariamente enteros, aunque realmente $A(i,j)$ debería tomar solo valores enteros, esta consideración puede hacerse gracias a que computacionalmente esto no tiene mayores efectos, por ejemplo, si $A$ es una imagen con algunos $A(i,j)$ no enteros, entonces al mostrar la imagen que determina $A$ en un computador este aproxima $A(i,j)$ al entero mas cercano, visualmente esto no influye.

Ahora, vamos a inducir dos métricas en el conjunto $F_n$, una ayudara a definir la otra, la ultima es con la que se desarrollan los resultados de este articulo. En primer lugar, sabemos que $M_n(\mathbb{R})$ es un espacio vectorial de dimensión $n^2$, entonces podemos considerar el isomorfismo canónico $\Phi$ dado por

$$\begin{gathered} \Phi : M_n(\mathbb{R}) \to \mathbb{R}^{n^2} \\ \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \end{gathered} \tag{1}$$

donde los $v_i$ son vectores columna. Si consideramos a $d_{eu}$ como la métrica euclidiana de $\mathbb{R}^{n^2}$ entonces esta define una métrica en $M_n(\mathbb{R})$ denotada por $d_n$ y definida para $A, B \in M_n(\mathbb{R})$ como

$$d_n(A,B) := d_{eu}(\Phi(A), \Phi(B)).$$

Es fácil ver que $d_n$ es una métrica, en realidad es consecuencia de la inyectividad de $\Phi$, más aun, esto implica que $M_n(\mathbb{R})$ es un espacio métrico completo (por ser $M_n(\mathbb{R})$ finito-dimensional). Ahora,

sabemos que con la topología inducida por $d_n$ en $M_n(\mathbb{R})$ la aplicación $\Phi$ es continua, por lo tanto, dado que $F_n$ se puede ver como

$$F_n = \Phi^{-1}\left([0,255]^{n^2}\right).$$

Entonces $F_n$ es cerrado en $M_n(\mathbb{R})$, es más, es un espacio métrico completo con la métrica $d_n$ restringida a $F_n$ (todo subconjunto cerrado de un espacio métrico completo es completo).
Así pues, tenemos que $d_n$ es una métrica en $F_n$, pero esta no es la mejor métrica para nuestros propósitos, recordemos que queremos medir la similitud, existen imágenes que a simple vista son casi idénticas pero que respecto a esta métrica están alejadas, por ejemplo, dos imágenes que solo se diferencian la una de la otra por un pixel. Para solventar este inconveniente modificamos la métrica de tal manera que tenga en cuenta el entorno, dicha forma es re-definirla de la siguiente manera:

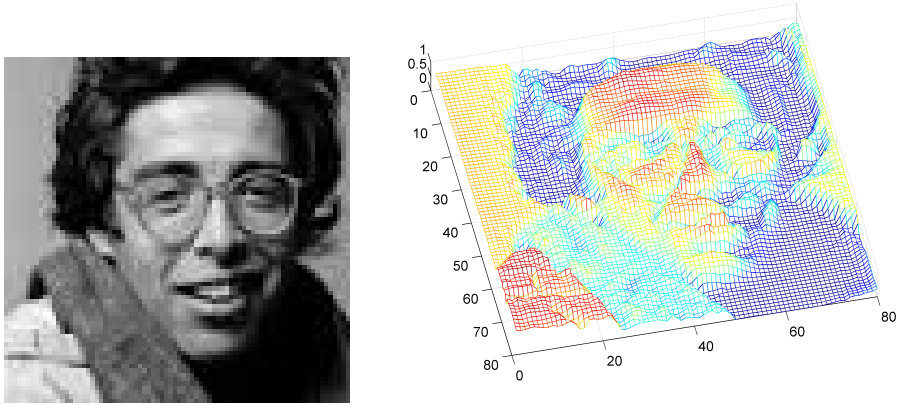
$$d_n(A,B) = \frac{1}{n^2} d(\Phi(A), \Phi(B)). \tag{2}$$

No es difícil ver que esta sigue siendo una métrica. La anterior es la primera de las métricas que prometimos, para definir la métrica que se ajusta completamente a nuestros propósitos debemos antes caracterizar a $F_n$ de otra forma que es lo que haremos a continuación.

Considere $I_n^2 := [0,n] \times [0,n] \subset \mathbb{R}^2$, entonces $F_n$ se puede caracterizar como

$$F_n = \left\{ A \in M_n(\mathbb{R})^n \;\middle|\; A(i,j) = f(i,j) \text{ para alguna función } f : I_n^2 \to [0,255] \right\}. \tag{3}$$

Una de las ventajas de esta caracterización es que permite ver a una imagen como una superficie, esto se ilustra en el ejemplo de la Figura 2.1.




(a) (b)

**Fig. 2.1.** (a) Imagen original. (a) Imagen original como superficie.

Consideremos una partición disyunta $\{\mathtt{R}_1, \mathtt{R}_2, \ldots, \mathtt{R}_N\}$ del conjunto $I_n^2$ tal que todos los $\mathtt{R}_k$ son cuadrados de igual tamaño, entonces, definimos

$$R_k := \{(i,j) \in \mathtt{R}_i \mid i,j \in \mathbb{Z}\}.$$

Sin perdida de generalidad podemos ver a $R_k$ como una matriz de la forma

$$R_k = \begin{pmatrix} r^k_{1,1} & r^k_{1,2} & \cdots & r^k_{1,m} \\ r^k_{2,1} & r^k_{2,2} & \cdots & r^k_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ r^k_{m,1} & r^k_{m,2} & \cdots & r^k_{m,m} \end{pmatrix} \tag{4}$$

donde cada $r^k_{i,j} \in \mathtt{R}_k \cap \mathbb{Z}^2$. Denotamos al conjunto de estas matrices como $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$ se sobreentiende que provienen de una partición disjunta de $I^2_n$. Una consecuencia es que todas estas matrices tendrán el mismo tamaño, denotamos ese tamaño común como $m \times m$, claramente $m \leq n$, otra consecuencia es que $mN = n$.
Ahora, dada $A \in F_n$ por (3) existe $f : I^2_n \to [0, 255]$ tal que $A(i, j) = f(i, j)$, entonces sin perdida de generalidad podemos asumir

$$f(R_k) = \begin{pmatrix} f\left(r^k_{1,1}\right) & f\left(r^k_{1,2}\right) & \cdots & f\left(r^k_{1,m}\right) \\ f\left(r^k_{2,1}\right) & f\left(r^k_{2,2}\right) & \cdots & f\left(r^k_{2,m}\right) \\ \vdots & \vdots & \ddots & \vdots \\ f\left(r^k_{m,1}\right) & f\left(r^k_{m,2}\right) & \cdots & f\left(r^k_{m,m}\right) \end{pmatrix}. \tag{5}$$

Esta caracterización permite ver $A$ de la siguiente forma

$$A = \begin{pmatrix} \boxed{f(R_1)} & \boxed{f(R_2)} & \cdots & \cdots \\ \cdots & \boxed{f(R_k)} & \boxed{f(R_{k+1})} & \cdots \\ \cdots & \cdots & \boxed{f(R_{N-1})} & \boxed{f(R_N)} \end{pmatrix}. \tag{6}$$

Es decir, estamos viendo $A$ a partir de determinados bloques. Otra consideración importante que se sigue de (5) es que podemos ver a $f(R_k)$ como una pequeña imagen, concretamente $f(R_k) \in F_m$, este conjunto esta dotado de la métrica $d_m$. Por lo tanto, definimos la métrica $d_{\mathcal{R}}$ en $F_n$ definida para $A, B \in F_n$ como

$$d_{\mathcal{R}}(A, B) := \sum_{k=1}^{N} d_m(f(R_k), g(R_k)) \tag{7}$$

donde $g : I^2_n \to [0, 255]$ es la función tal que $B(i, j) = g(i, j)$, es claro que esta métrica no depende de las funciones $f$ y $g$, pero si depende de la partición $\mathcal{R}$. Así pues, $(F_n, d_{\mathcal{R}})$ es un espacio métrico, más aún, es completo. Precisamente, $d_{\mathcal{R}}$ es la métrica con la que desarrollaremos los resultados en adelante.

## 3 Transformaciones afines, de contraste y de brillo

Para desarrollar el método de compresión es importante poder modificar algunas características de las imágenes, específicamente nuestro interés se centra en el contraste y el brillo de la imagen, otro aspecto también importante es poder rotar y trasladar partes de una imagen. Todas estas modificaciones se formalizan mediante transformaciones que realizan dichas acciones, en ese sentido tenemos las siguientes aplicaciones.

Una aplicación $v : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ tal que

$$v(x,y) = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ h \end{pmatrix}$$

donde $a, b, c, d, e, h$ son números reales, es llamada una *transformación afín.*

Una apropiada elección de los coeficientes $a, b, c, d, e$ y $h$ en la aplicación $S$ permiten rotar y mover la imagen.

La otra transformación con un rol importante en este articulo son las que permiten modificar el contraste y el brillo de una imagen, estas se caracterizan de la siguiente forma:

$$\begin{aligned} P_{s,\sigma} : F_n &\to F_n \\ A &\mapsto sA + \sigma \mathbb{1}_n. \end{aligned}$$

donde $s, \sigma \in \mathbb{R}$ con $0 \leq s$ y $\mathbb{1}_n$ la matriz de solo unos de tamaño $n \times n$. La aplicación $P_{s,\sigma}$ es llamada un *transformación de contraste y brillo (CBT)* donde $s$ modifica el contraste y $\sigma$ el brillo.

*Observacin 1.* Si consideramos $F_n$ con la métrica $d_n$ y $0 < s < 1$, entonces $P_{s,\sigma}$ es una aplicación contractiva, y ya que $(F_n, d_n)$ es un espacio métrico completo, entonces por el Teorema 1 $P_{s,\sigma}$ tiene un punto fijo en $F_n$.

## 4 El método de compresión

Como mencionamos al principio de este articulo, la mayoría de las imágenes cotidianas tienen partes de si misma que después de aplicarle adecuadas transformaciones afines (traslaciones y rotaciones), de contraste y de brillo son similares a otras partes de la misma imagen, usaremos esta situación desarrollar el método de compresión.

Sea $A \in F_n$ una imagen, entonces consideramos una partición disyunta $\{\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_N\}$ de $I_n^2$, como vimos en la Sección 2 esta partición determina el conjunto de matrices $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$ de tamaño $m \times m$ donde $m$ satisface $mN = n$. Ademas, por la caracterización (3) de $F_n$ existe una función $f : I_n^2 \to [0, 255]$ tal que $A(i,j) = f(i,j)$, entones $\mathcal{R}$ determina el conjunto de matrices $\{f(R_1), f(R_2), \ldots, f(R_N)\}$ donde cada una tiene la forma descrita en (5) y permiten expresar $A$ en la forma (6).

La idea del método de compresión consiste de varios pasos; **primero**, consideramos un numero entero $l$ tal que $m \leq l \leq n$ y consideramos el conjunto

$$\left\{ \mathtt{D} \subset I_n^2 \;\middle|\; \begin{array}{l} \mathtt{D} = \mathtt{U} \text{ or } \mathtt{D} = \mathbf{r}(\mathtt{U}) \text{ donde } \mathtt{U} \subset I_n^2 \text{ es un dominio cuadrado que contiene } m^2 \\ \text{pares enteros } (i,j) \text{ y donde } \mathbf{r} \text{ una rotación de 0, 90, 180 o 270 grados.} \end{array} \right\} \tag{8}$$

Les elementos en el conjunto anterior se pueden solapar. Por cada $\mathtt{D}$ en el conjunto anterior definimos

$$D := \{(i,j) \in \mathtt{D} \mid i, j \in \mathbb{Z}\}.$$

Sin perdida de generalidad $D$ se puede ver como una matriz

$$D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,l} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ d_{l,1} & d_{l,2} & \cdots & d_{l,l} \end{pmatrix} \tag{9}$$

donde cada $d_{i,j} \in \mathtt{D} \cap \mathbb{Z}^2$. Denotamos al conjunto de estas matrices como $\mathcal{D}$, se sobreentiende que provienen del conjunto de dominios definido en (8).
Antes de seguir describiendo el método es importante introducir las aplicaciones

$$\mathbf{T}_{l,m} : F_l \to F_m \qquad \text{y} \qquad \mathbf{T}_{m,l} : F_m \to F_l.$$

Estas aplicaciones redimensionan imágenes, es decir, $\mathbf{T}_{l,m}$ toma una imagen de tamaño $l \times l$ y la redimensiona uniformemente a el tamaño $m \times m$, de manera análoga, $\mathbf{T}_{m,l}$ toma una imagen de tamaño $m \times m$ y la redimensiona uniformemente a el tamaño $l \times l$. En `MatLab` esta acción se puede realizar con el comando `imresize`.

El **segundo** paso consiste en lo siguiente, por cada $R_k$ en $\mathcal{P}$ buscamos un $D_k$ en $\mathcal{D}$ y una transformación de contraste y brillo $P_{s_k,\sigma_k}$ con $0 < s_k < 1$ tal que la distancia

$$d_l \left( P_{s_k,\sigma_k}(f(D_k)), \mathbf{T}_{m,l}\left(f(R_k)\right)\right)$$

sea lo menor posible. Es importante recalcar la importancia de $\mathbf{T}_{m,l}$ en el calculo de la distancia anterior ya que de antemano $dP_{s_k,\sigma_k}(f(D_k))$ y $f(R_k)$ no tienen el mismo tamaño. Por otro lado, es fácil ver que

$$d_l \left( P_{s_k,\sigma_k}(f(D_k)), \mathbf{T}_{m,l}\left(f(R_k)\right)\right) \approx d_m \left( \mathbf{T}_{l,m}\left(P_{s_k,\sigma_k}(f(D_k))\right), f(R_k)\right) \tag{10}$$

Lo que nos dice (10) es que minimizar con $d_l$ es lo mismo que minimizar con $d_m$, recordemos que estas métricas fueron definidas en (2).
Note que

$$\mathbf{T}_{l,m}\left(P_{s_k,\sigma_k}(f(D_k))\right) = P_{s_k,\sigma_k}\left(\mathbf{T}_{l,m}\left(f(D_k)\right)\right). \tag{11}$$

Esta propiedad sera útil mas adelante.
Ahora, como $\frac{n}{m}$ es un entero definimos la aplicación

$$\begin{aligned} \mathbf{H} : I_n^2 &\longrightarrow I_{\frac{nl}{m}}^2 \\ (x,y) &\longmapsto \tfrac{l}{m}(x,y). \end{aligned} \tag{12}$$

Esta aplicación lineal (es más, es biyección) lo que hace es llevar los cuadrados de largo $m$ a cuadrados de largo $l$, en particular, como $R_k$ se puede ver como una matriz de $m \times m$ (ver (4)), entonces $\mathbf{H}(R_k)$ también se puede ver como una matriz de tamaño $l \times l$, las componentes de esta matriz son de la forma $\mathbf{H}(R_k)(i,j) = \mathbf{H}(r^k_{i,j})$.

El **tercer** y ultimo paso es encontrar una transformación afín $v_k$ tal que $v_k\left(\mathbf{H}(R_k)\right) = D_k$. Encontrar esta transformación afín no es difícil, en la programación del método no es necesario calcularla explicitamente, de modo que no haremos mucho enfaces en este paso.

Ahora, definimos la aplicación $w_k : F_m \to F_m$ definida para $B \in F_m$ por

$$w_k(B) := P_{s_k,\sigma_k}(B). \tag{13}$$

Claramente esta aplicación es una contracción en el espacio métrico $(F_m, d_m)$. Por lo tanto, definimos la aplicación $w : F_n \to F_n$ definida para cualquier $C \in F_n$ como

$$w(C) := \bigcup_{k=1}^{N} w_k\left(\mathbf{T}_{l,m}\left(g\left(v_k\left(\mathbf{H}(R_k)\right)\right)\right)\right) = \bigcup_{k=1}^{N} w_k\left(\mathbf{T}_{l,m}\left(g\left(D_k\right)\right)\right) \tag{14}$$

donde $g : I_n^2 \to [0, 255]$ es una función tal que $C(i, j) = g(i, j)$ (esta es dada por la caracterización (3)). Básicamente $w(C)$ visto como matriz por bloques es de la siguiente forma

$$w(C) = \begin{pmatrix} \boxed{w_1\left(\mathbf{T}_{l,m}\left(g\left(D_1\right)\right)\right)} & \boxed{w_2\left(\mathbf{T}_{l,m}\left(g\left(D_2\right)\right)\right)} & \cdots \\ \cdots & \boxed{w_k\left(\mathbf{T}_{l,m}\left(g\left(D_k\right)\right)\right)} & \cdots \\ \cdots & \cdots & \boxed{w_N\left(\mathbf{T}_{l,m}\left(g\left(D_N\right)\right)\right)} \end{pmatrix}. \tag{15}$$

Considerando $s = \max\{s_k \mid k = 1, 2, \ldots, N\}$, entonces $s < 1$, y $w$ es una contracción en el espacio métrico $(F_n, d_{\mathcal{R}})$ con constante de contracción $s$, es decir, para cualesquiera $C, B \in F_n$ después de algunos cálculos se concluye

$$d_{\mathcal{R}}(w(C), w(B)) \leq s d_{\mathcal{R}}(C, B). \tag{16}$$

Note que $w$ es descrita por los $s_k$, $\sigma_k$, las posiciones de los bloques $f(D_k)$ en la matriz $A$ ( que son dos datos por cada $k$) y la rotación del dominio $\mathtt{D}_k$ que es de donde proviene $D_k$. Entonces, como tenemos $N$ elementos en $\mathcal{R}$, entonces $w$ es descrita por un total de $5N$ datos que es mucho menor a $n^2$ que son la cantidad de datos que constituyen la imagen $A$ (pues $A$ es una imagen de $n \times n$). Por lo tanto, **la compresión de la imagen** $A$ **es** $w$. Mas adelante demostraremos el procedimiento para descomprimir, es decir, como llegar de $w$ a la imagen $A$.

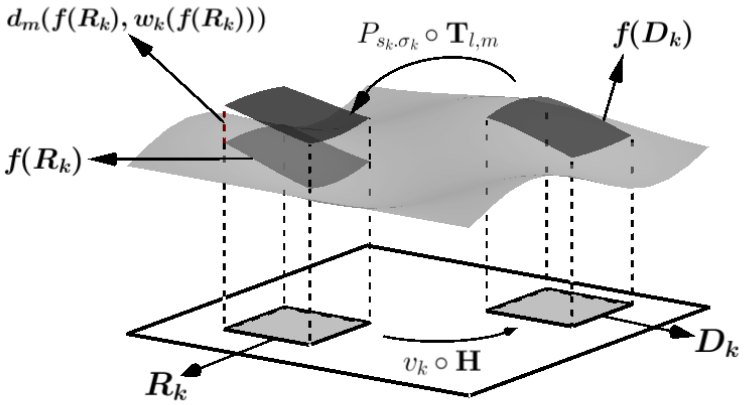



**Fig. 4.1.** Idea gráfica del método.

## 5 El problema de optimización en el método

El problema de optimización esta en el segundo paso del método de compresión. Concretamente, para cada $k = 1, 2, \ldots, N$ tenemos el siguiente problema

$$\begin{array}{rll} P: & \text{Minimizar} & d_l\left(P_{s_k,\sigma_k}(f(D_k)), \mathbf{T}_{m,l}\left(f(R_k)\right)\right) \\ & \text{Sujeto a} & 0 \leq s_k < 1 \end{array} \tag{17}$$

El cál debemos solucionar para $s_k$ y $\sigma_k$. Note que $P_{s_k,\sigma_k}(f(D_k))$ y $\mathbf{T}_{m,l}(f(R_k))$ pueden verse como matrices, especialmente, $P_{s_k,\sigma_k}(f(D_k))$ es caracterizada como la siguiente matriz:

$$\begin{pmatrix} s_k f\left(d^k_{1,1}\right)+\sigma_k & s_k f\left(d^k_{1,2}\right)+\sigma_k & \cdots & s_k f\left(d^k_{1,l}\right)+\sigma_k \\ s_k f\left(d^k_{2,1}\right)+\sigma_k & s_k f\left(d^k_{2,2}\right)+\sigma_k & \cdots & s_k f\left(d^k_{2,l}\right)+\sigma_k \\ \vdots & \vdots & \ddots & \vdots \\ s_k f\left(d^k_{l,1}\right)+\sigma_k & s_k f\left(d^k_{l,2}\right)+\sigma_k & \cdots & s_k f\left(d^k_{l,l}\right)+\sigma_k \end{pmatrix} \tag{18}$$

donde $d^k_{i,j} \in D_k$. Pero recordamos que $d_l$ se definió en (2) a partir del isomorfismo canónico $\Phi$ definido en (1), en ese sentido, $\Phi(P_{s_k,\sigma_k}(f(D_k)))$ es un vector que se puede ver de la siguiente forma:

$$\Phi(P_{s_k,\sigma_k}(f(D_k))) = \begin{pmatrix} s_k f\left(d^k_{1,1}\right)+\sigma_k \\ \vdots \\ s_k f\left(d^k_{l,1}\right)+\sigma_k \\ \vdots \\ s_k f\left(d^k_{1,l}\right)+\sigma_k \\ \vdots \\ s_k f\left(d^k_{l,l}\right)+\sigma_k \end{pmatrix} = \begin{pmatrix} f\left(d^k_{1,1}\right) & 1 \\ \vdots & \vdots \\ f\left(d^k_{l.1}\right) & 1 \\ \vdots & \vdots \\ f\left(d^k_{1,l}\right) & 1 \\ \vdots & \vdots \\ f\left(d^k_{l,l}\right) & 1 \end{pmatrix} \begin{pmatrix} S_k \\ \sigma_k \end{pmatrix}.$$

Y a $\Phi(\mathbf{T}_{m,l}\left(f(R_k)\right))$ lo asumimos como un vector de longitud $l^2$ dispuesto verticalmente. Con estas claridades y teniendo en cuenta la definición de la métrica $d_l$ la cual es a partir de la métrica euclidiana $d$ de $\mathbb{R}^2$, se obtiene que el problema de optimización (17) es equivalente al siguiente problema

$$\begin{aligned} P: \text{Minimizar} \quad & d\left(\Phi(P_{s_k,\sigma_k}(f(D_k))), \Phi(\mathbf{T}_{m,l}\left(f(R_k)\right))\right) \\ \text{Sujeto a} \quad & 0 \leq s_k < 1 \end{aligned} \tag{19}$$

Que a su vez es análogo a resolver el problema

$$\begin{aligned} P: \text{Minimizar} \quad & \left(\Phi(P_{s_k,\sigma_k}(f(D_k))) - \Phi(\mathbf{T}_{m,l}\left(f(R_k)\right))\right)^T \left(\Phi(P_{s_k,\sigma_k}(f(D_k))) - \Phi(\mathbf{T}_{m,l}\left(f(R_k)\right))\right) \\ \text{Sujeto a} \quad & 0 \leq s_k < 1 \end{aligned} \tag{20}$$

Pero el este ultimo problema que de antemano pareciera no tan amigable es tan solo un caso particular del siguiente problema de optimización

$$\begin{aligned} P: \text{Minimizar} \quad & (\mathbf{Ax}-\mathbf{B})^T(\mathbf{Ax}-\mathbf{B}) \\ \text{Sujeto a} \quad & \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}\mathbf{x} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned} \tag{21}$$

donde $\mathbf{A}$ es una metriz de $2 \times l^2$, $\mathbf{B}$ un vector vertical de $l^2$ componentes y $\mathbf{x} \in \mathbb{R}^2$ es la variable del problema. Existen varias formas de solucionar problemas de la forma expuesta en (21) pero en este articulo hemos decidido emplear un método numérico, específicamente el **Método de Barrera logaritmica** donde en cada iteración usaremos el **Método de Newton modificado con Backtracking**.

**Algoritmo del Método de Barrera logarítmica:**

**Haga** $t = t_0 = 10$, $\mu = 20$ y $\mathbf{x} = \mathbf{x}_0 = (0.3, 0)$.
**Repetir** hasta $\frac{2}{t} < \varepsilon_1$
$F(x) = \|\mathbf{Ax} - \mathbf{B}\|^2 - \frac{1}{t}\log(1 - \mathbf{x}(1)) - \frac{1}{t}\log(\mathbf{x}(1))$;
**Repetir** hasta $\|\nabla F(\mathbf{x})\| < \varepsilon_2$
$P = \nabla^2 F(\mathbf{x})\nabla F(\mathbf{x})$;
—-*Backtracking*—
$\alpha = 1$;
$\rho = 0.5$;
$\mathbf{y} = \mathbf{x} - \alpha P$;
**Repetir** hasta $0 < \mathbf{y}(1) < 1$
$\alpha = \rho\alpha$;
$\mathbf{y} = \mathbf{x} - \alpha P$;
**Fin** repetición
$\mathbf{x} = \mathbf{y}$;
**Fin** repetición
$t = \mu t$;
**Fin** repetición.

Método de Newton

Hemos tomado $\varepsilon_1$ y $\varepsilon_2$ heuristicamente, sus valores en el algoritmo son ambos 0.0000001. Frecuentemente el Método de Newton no incluye un backtracking pero en este caso es necesario ya que se debe evitar el factor $\mathbf{x} - P$ pueda salir del dominio de la función $F$, experimentalmente se comprobó que esa situación ocurría con demasiada frecuencia haciendo del método numérico inestable.

## 6 El método de descompresión

Inicialmente, sabemos que $w$ es una contracción en el espacio métrico $(F_n, d_{\mathcal{R}})$ (ver final de la Sección 4) con constante de contracción $s = \max s_1, \ldots, s_N < 1$, es decir, para cualesquiera $C, B \in F_n$ se tiene

$$d_{\mathcal{R}}(w(C), w(B)) \leq s d_{\mathcal{R}}(C, B). \tag{22}$$

Entonces por el Teorema 1 (punto fijo de Banach) existe una única imagen $Q \in F_n$ tal que $w(Q) = Q$.

Ahora, para cualquier imagen $B \in F_n$ denotamos por $w^r(B)$ la iteración $r$-veces de $B$ bajo $w$, es decir

$$w^r(B) = \underbrace{w\left(w\left(\cdots\left(w\left(B\right)\right)\right)\right)}_{r-\text{veces}}$$

Entonces se tiene que $\lim_{r\to\infty} w^r(B) = Q$, de modo que dado $\varepsilon > 0$ (pequeño) se sigue

$$d_{\mathcal{R}}\left(w^r\left(B\right), Q\right) < \varepsilon$$

para $r$ apropiadamente grande. Pero por otro lado $w$ tambien satisface el Corolario 1 (Collage theorem), por lo tanto, se tiene

$$d_{\mathcal{R}}\left(A, Q\right) \leq \frac{1}{1-s} d_{\mathcal{R}}\left(A, w(A)\right).$$

Así pues, para cualquier imagen $B \in F_n$ y $r$ suficientemente grande tenemos

$$\begin{aligned} d_{\mathcal{R}}\left(A, w^r\left(B\right)\right) &\leq d_{\mathcal{R}}\left(A, Q\right) + d_{\mathcal{R}}\left(Q, w^r\left(B\right)\right) \\ &< \frac{1}{1-s} d_{\mathcal{R}}\left(A, w\left(A\right)\right) + \varepsilon \end{aligned}$$

Por lo tanto, el la descompresión de la imagen $A$ es la imagen $W^r(B)$ para $r$ suficientemente grande, note que no recuperamos $A$ exactamente sino una aproximación, la calidad de esta aproximación depende únicamente de la cercania de $w(A)$ respecto a la imagen original $A$, es decir, depende de que tan pequeño es el termino $\frac{1}{1-s} d_{\mathcal{R}}(A, w(B))$ el cual ya es lo mas pequeño posible pues así se especifico en la construcción de $w$.

### 6.1 Ejemplo de descompresión

Dado el costo comunicacional del método expuesto en este articulo en la Figura 6.1 presentamos un ejemplo con una imagen de tamaño $80 \times 80$, para este caso hemos considerado a $\mathcal{R}$ con matrices de tamaño $5 \times 5$, eso quiere decir que $\mathcal{R}$ tiene 256 matrices, y a $\mathcal{D}$ con matrices de tamaño $10 \times 10$, esto significa que $\mathcal{D}$ tiene 19600 matrices, entonces la aplicación $w$ que actúa como la compresión de la imagen original esta definida por 1280 datos mientras que la imagen original esta definida por $80 \times 80 = 6400$ datos, tenemos una reducción del 80% (ver el final de la Sección 4).

La Figura 6.1 muestra que en este ejemplo la convergencia es rápida, en la cuarta iteración de $w$ sobre la imagen negra ya se ha llegado al punto fijo de $w$, aunque la calidad de dicha imagen no es la mejor es una buena estimación ya que se obtuvo una reducción 80% de los datos, pero si el deseo es la calidad por encima de la compresión esto se puede solventar en parte tomando $\mathcal{R}$ y $\mathcal{D}$ con matrices de menor tamaño, pero hacerlo implicara un costo comunicacional aun mayor, por ejemplo, para el caso de este ejemplo en un computador con especificaciones normales el proceso de compresión tomo aproximada mente 4 horas, note que por cada bloque en $\mathcal{R}$ se resolvieron 19600 problemas de optimización de la forma (21), esto de por si ya es dispendioso.

## 7 Comentarios sobre el algoritmo en `MatLab`

Adjunto al presente articulo se encontrara las rutinas de `MatLab` que permiten comprimir y descomprimir imagen con el método descrito en es este articulo, de modo que tomaremos estas lineas para explicar su uso.

Tenemos cuatro 4 archivos que son

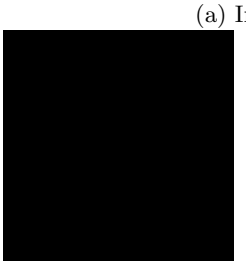
`CompresorImagen.m`
`DescompresorImagen.m`
`BarreraMethodImageMod.m`
`NewtonMethodBlock.m`

Los archivos `BarreraMethodImageMod.m` y `NewtonMethodBlock.m` son usados en `CompresorImagen.m` que es la rutina que descomprime las imágenes, por otro lado, `DescompresorImagen.m` es independiente de los demás archivos y es el que descomprime las imágenes.
Para ejecutar el proceso de compresión debe alojar estos archivos en una carpeta, luego en `MatLab` se debe diseccionar a la carpeta donde se albergan los archivos.
Supongamos que queremos comprimir $A$ que es una imagen en blanco y negro en formato .jpg (el proceso es análogo para otros formatos), entonces debemos llevar a $A$ en su forma matricial, para tal fin ejecutamos en `MatLab` el comando
>> `B = imread('A.jpg');`

A pesar de estas $A$ en formato .jpg y estar en blanco y negro esta $B$ tiene 3 capas, solo nos interesa la capa de grises, en ese sentido escribimos

>> `C = A(:, :, 1);`

(a) Imagen original.



(b) Iteración 0.

(c) Iteración 1.

(d) Iteración 2

(e) Iteración 3

(f) Iteración 4

(g) Iteración 100

**Fig. 6.1.** Ejemplo del método de descompresión, a partir de una imagen negra se intenta reconstruir la imagen original.

Pero esta aún no esta en la forma matricial deseada, dicha forma se obtiene con la siguiente linea
$>>$ `A` $=$ `mat2gray(C)`;

La rutina `CompresorImagen.m` trabaja con esta ultima matriz, esta toma $A$, el largo de las matrices cuadrada de $\mathcal{R}$ (esto es discrecional del usuario) que llamaremos `tamBloR` y el largo de las matrices cuadrada de $\mathcal{D}$ (esto es discrecional del usuario) que llamaremos `tamBloD`, y devuelve cinco matrices de tamaño `tamBloR` $\times$ `tamBloR` que son

`K`, `L`, `Rot`, `SS` y `Sigma`.

Teniendo en cuenta que tenemos tantos elementos en $\mathcal{R}$ como pares es $(i, j)$ en $\{1, 2, \ldots,$ `tamBloD`$\}^2$, de modo que podemos identificar los elementos en $\mathcal{R}$ con pares $(i, j)$, la forma de hacerlo es sugerida por la expresión (6). Asi pues, se tiene

`K:=` Matriz que en la posición $(i, j)$ esta la posición $i$ en la matriz $A$ de la primera componente de del bloque $D$ asociado a $R_{(i,j)}$ (sin rotar) en el sentido del segundo paso del método de compresión descrito en la Sección 4. (Estamos viendo a $D$ como un bloque de $A$)

`L:=` Matriz que en la posición $(i, j)$ esta la posición $j$ en la matriz $A$ de la primera componente de del bloque $D$ asociado a $R_{(i,j)}$ en el sentido descrito en el caso de `K`.

`Rot:=` Matriz que en la posición $(i, j)$ es 1, 2, 3 o 4 donde cada uno de estos números representa una rotación de $D$, donde $D$ es el bloque asociado a $R_{(i,j)}$ en el sentido descrito en el caso de `K`. En este caso 1 es *no rotar*, 2 es rotar 90 grados, 3 es rotar 180 grados y 4 es rotar 270 grados. Así pues, a posición $(i, j)$ es la mejor rotación de $D$.

`SS:=` Matriz que en la posición $(i, j)$ es el valor del factor de contracción $s_{i,j}$ asociados a $D$ después de aplicarle la mejor rotación, donde $D$ es el bloque asociado a $R_{(i,j)}$.

`Sigma:=` Matriz que en la posición $(i, j)$ es el valor del factor de brillo $\sigma_{(i,j)}$ asociados a $D$ después de aplicarle la mejor rotación, donde $D$ es el bloque asociado a $R_{(i,j)}$.

Estas matrices contiene todos los datos para construir la aplicación $w$. Asi pues, para obtenerlas ejecutamos la linea

```
>> [K, L, Rot, SS, Sigma] = CompresorImagen(tamBloR, tamBloD, A);
```

Para descomprimir la imagen ejecutamos la linea

```
>> DescompresorImagen(tamBloR, tamBloD, K, L, Rot, SS, Sigma, Imag0, iter);
```

donde `iter` es el numero de iteraciones que queremos que haga $w$ sobre `Imag0` que es una imagen cualquiera en escala de grises de tamaño igual al de $A$, en el ejemplo visto en la Figura 6.1 esa imagen fue una imagen negra.

## 8 Conclusiones

El método expuesto es costoso computacionalmente, no obstante, varios artículos recalcan que para una adecuada elección de $\mathcal{R}$ y $\mathcal{D}$ y con una computadora supremamente eficiente se obtienen mejores resultados que los demás métodos de compresión existentes, entendiendo mejor como el equilibrio entre disminución del peso de la imagen versus calidad de la descompresión.
Otro aspecto que debemos mencionar es que este método se puede programar usando computación paralela ya que la mayoría de las iteraciones son independientes la una de la otra.

## Referencias

1. Alexander, S. K. & Vrscay, E.R; Tsurumi, S., *A simple,general Model for theaffine self-similarity of Imges.* In: Image Analysis and Recognition, M.; Kamel, A; Campilho, Springer. Montreal. pp: 192-203 (2007)
2. Fisher, Y., *Fractal image compression: Theory an Applications.* Springer. New York. pp: 5-23 (1995)
3. S. Boyd & L. Vanderberghe, *Convex optimization*, Cambridge University Press, 160-174 (2004)