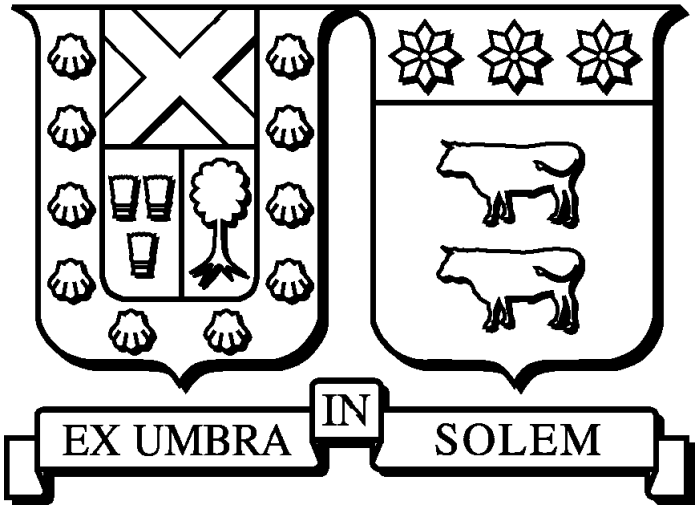
2017

COMPRESIÓN Y CORRECCIÓN DE IMAGENES MEDIANTE ALGORITMOS DE ONDÍCULAS Y OPTIMIZACIÓN

ROLDÁN CONTRERAS, FERNANDO MARCELO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
Departamento de Matemática
Valparaíso-Chile



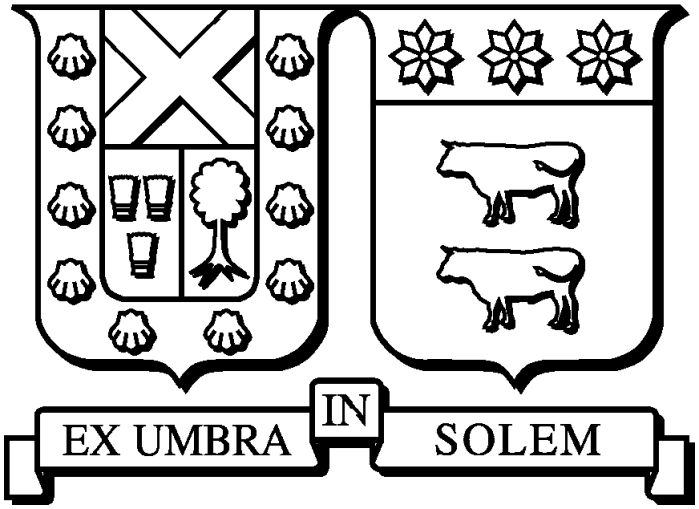
Compresión y corrección de imágenes mediante algoritmos de ondículas y optimización

Memoria presentada por:
Fernando Marcelo Roldán Contreras
Como requisito parcial para optar al título profesional Ingeniero Civil Matemático

Profesor Guía:
Juan Peypouquet

Noviembre 2017

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
Departamento de Matemática
Valparaíso-Chile



Compresión y corrección de imágenes mediante algoritmos de ondículas y optimización

Memoria presentada por:
Fernando Marcelo Roldán Contreras

Como requisito parcial para optar al título profesional Ingeniero Civil Matemático

Profesor Guía:
Juan Peypouquet

Examinadores:
Luis Briceño
Alberto Mercado

Noviembre, 2017

TÍTULO DE LA MEMORIA:
Compresión y corrección de imágenes mediante algoritmos de ondículas y optimización.

AUTOR: Fernando Marcelo Roldán Contreras.

TRABAJO DE MEMORIA, presentado como requisito parcial para optar al título profesional Ingeniero Civil Matemático de la Universidad Técnica Federico Santa María.

COMISIÓN EVALUADORA:

Integrantes	Firma
Juan Peypouquet Universidad Técnica Federico Santa María, Chile.	_____
Luis Briceño Universidad Técnica Federico Santa María, Chile.	_____
Alberto Mercado Universidad Técnica Federico Santa María, Chile.	_____

Valparaíso, Noviembre 2017.

Agradecimientos

En primer lugar quería agradecer a mí familia, durante todas mis etapas como estudiante siempre han estado apoyándome. A mí padre que en momentos muy difíciles estuvo dispuesto a dar todo por mí hermana y por mí, siempre fuimos su prioridad, me apoya en mis decisiones y constantemente está dispuesto a ayudarme, su amor hacia mí es gigante. A mis abuelos, desde pequeño me acogieron, como a un hijo me cuidaron y enseñaron, agradezco por sus atenciones, amor y preocupación, mucho de lo que soy hoy es gracias a ellos. A mí hermana, siempre ha sido enriquecedor ver como se desarrolla, sus logros y propósitos, gracias por su amor y disposición. A mí tía, que este ultimo tiempo ha sido de mucha ayuda en mí casa, cocinando sus ricas comidas, siempre dispuesta a aportar y preocupada por mí. A mí familia en general, primos y tíos, quienes siempre son un ejemplo, es reconfortante y valioso compartir con ellos.

A mis amigos, polola y compañeros en general. Todos han sido de mucha ayuda, siempre aportando algo. A mis primeros compañeros y amigos electricistas, he compartido muchas cosas con ustedes, gracias por considerarme y por su ayuda sobre todo en los primeros años. A mis amigos y compañeros de matemática, fueron muchos momentos que disfrutamos a lo largo de estos años, noches de estudio, risas, actividades extras, compartir experiencias en el extranjero, muchas gracias por todos los buenos momentos. A mí polola que este ultimo tiempo ha sido de mucha ayuda, preocupada, apoyándome en momentos difíciles, dándome animo y cariño. A mis amigos del colegio, tenis, fútbol y otros, gracias por los momentos que me entregaron, donde siempre fue bueno despejarse de los quehaceres, jugar, reír y conversar. A mis amigos de la iglesia, algunos también compañeros de universidad, muchas gracias por su compañía, atención y preocupación, por siempre estar ahí para mí, aconsejarme y pasar gratos momentos.

A los profesores, secretarias y personal de la universidad en general. Estoy muy agradecido de cada profesor, por sus enseñanzas y buena disposición, la mayoría siempre dispuesto a recibirme en su oficina para atender consultas, agradezco cada cosa que aprendí de ellos, lo cual utilizaré en el desarrollo de mi carrera. A mis profesores de matemáticas uno, Sergio y Salomón, les agradezco por su motivación y apoyo para ingresar a matemática, el que continuo durante todos estos años, compartiendo en ayudantías o simplemente conversando en los pasillos. A Alberto, quien fue jefe de carrera la mayoría de mí periodo, muchas gracias por su dedicación, por ayudarme con temas administrativos, trabajamos en muchas ayudantías y actividades extras, siempre dispuesto a recibirme y conversar. A Juan, mí profesor guía en esta memoria, con quien además cursé muchos cursos, estuvo de inmediato dispuesto a trabajar conmigo, como siempre lo estaba para atender mis consultas o inquietudes. A Cesare, quien me apoyó mucho en este trabajo, constantemente me solucionaba dudas y siempre fue agradable conversar con él. Quiero agradecer especialmente a las secretarias del departamento de matemáticas, Astrik y Andrea, quienes fueron de gran ayuda, siempre amables y serviciales, conversar con ellas me permitía distraerme además de pasar buenos momentos.

A mí Dios, el único Dios, a quien le debo todo, todo lo mencionado arriba es por Él, mis capacidades y virtudes, y aún muchas cosas más, las cuales no se comparan a nada.

Índice general

Introducción

Hoy en día las imágenes tienen un rol importante para la sociedad, cada día las personas tienen acceso más directo a fotografiar un instante. Las imágenes pueden guardar momentos con la familia y amigos, recuerdos de un viaje, toda cuenta en una red social debe tener una, incluso pueden representar una galaxia a millones de años luz de distancia. Por otro lado, también pueden ser utilizadas por detectives, por ejemplo, para identificar un delincuente mediante su huella digital, cual huella se guarda en una imagen.

El *procesamiento de imágenes* es un conjunto de técnicas que se aplican a imágenes digitales con diferentes objetivos, como resaltar o buscar cierta información, mejorar la calidad quitando ruido, extraer elementos de la imagen, detectar bordes, comprimir la imagen sin perder fidelidad, entre otros. A su vez, en las diferentes técnicas se utilizan herramientas matemáticas como estadística, control óptimo de EDP, optimización convexa, ondículas, además de sus respectivos modelos e implementaciones computacionales.

Retomando el ejemplo de las imágenes de galaxias, pongámonos en el caso que un Astrónomo en uno de sus días de estudio se encuentra con un planeta o estrella, que era hasta ese momento desconocida para la humanidad. Este científico, emocionado, quiere mostrar una imagen de su hallazgo a sus seres más cercanos, lamentablemente la imagen es de alta resolución y Gmail no permite enviar este archivo debido a su gran tamaño. Seria de utilidad para el astrónomo poder guardar la imagen de alguna manera más condensada y así enviarla inmediatamente, manteniendo cierta calidad de manera que su descubrimiento pueda ser notado.

Otra problemática recurrente es que ciertas veces el dispositivo que captura una imagen produce cierto error "natural", provocando que la imagen se vea borrosa, no se puedan diferenciar contornos o la figura se vea difusa. Esto es muy importante, por ejemplo, en imágenes médicas, poder dirimir de manera fehaciente si un paciente presenta alguna enfermedad o no, puede salvar su vida.

Considerando como motivación los problemas mencionados arriba, en el presente trabajo se presenta la teoría necesaria de ondículas (en ingles *wavelets*) como herramienta para solucionar estos problemas. Se muestran definiciones, resultados, demostraciones, ejemplos, todo en una mirada a través del álgebra lineal. Primeramente se muestran las ondículas unidimensionales para posteriormente introducir las ondículas bidimensionales utilizadas en imágenes. Esta teoría no es fácil de encontrar y normalmente no está bien detalla, por lo que será de ayuda para quienes quieran comenzar a estudiarla. A partir de esto, se presentarán métodos de corrección y compresión los cuales se desarrollan utilizando las buenas propiedades de las ondículas.

Se describen dos algoritmos llamados *EZW (embedded zerotree wavelet)* y *SPIHT (set partitioning in hierarchical trees)*, los cuales se utilizan para comprimir imágenes y son ideados

en base a las cualidades de las ondículas y por ende estas toman un rol importante para estos procesos. Se muestran ejemplos, ilustrativos y aplicados a imágenes, además de resultados computacionales, comparándolos en tiempo de ejecución, nivel de compresión y error.

Presentaremos dos modelos de corrección de imágenes, para los cuales se describirán tres algoritmos buscando enfrentar estos modelos. Estos métodos utilizan técnicas de optimización que involucran problemas con funciones *convexas, diferenciables y no diferenciables*, problemas *cuadráticos*, *separación de variables*, entre otras. Se entregan también algunas herramientas necesarias propias de la optimización y el análisis convexo que se emplean en estos métodos. Se propondrá un cuarto método, el cual busca acelerar la convergencia de otro, esta convergencia se probará experimentalmente.

Finalmente se muestran algunos experimentos que vienen a aclarar dudas canónicas que surgen a partir de los algoritmos propuestos, como por ejemplo cuál es más eficiente o efectivo, cómo afecta el nivel de la transformada ondícula en los algoritmos, como afecta la elección de los parámetros a los algoritmos, qué modelo de corrección es mejor, entre otras. Validaremos experimentalmente el algoritmo de corrección propuesto y se contrastará con los demás.

Capítulo 1

Ondículas en una dimensión

Antes de comenzar con las ondículas en dos dimensiones es bueno tener una noción de las ondículas en una dimensión, las cuales están orientadas a ondas unidimensionales como por ejemplo ondas de sonido. Los resultados presentados en esta sección están detallados con su respectiva demostración en [1], para mayores detalles se recomienda revisar este texto, donde además se encuentran las herramientas necesarias de álgebra lineal .

1. Preliminares y definiciones en $\ell^2(\mathbb{Z}_N)$

En lo que sigue, consideraremos

$$\mathbb{Z}_N = \{0, 1, \ldots, N-1\}.$$

Trabajaremos en el espacio

$$\ell^2(\mathbb{Z}_N) = \left\{ z = (z(0), z(1), \ldots, z(N-1)) : z(j) \in C, 0 \leq j \leq N-1 \right\},$$

con la suma usual y multiplicación usual de escalar componente a componente. Así, $\ell^2(\mathbb{Z}_N)$ es un espacio vectorial de dimensión N sobre $\mathbb{C}$. Una base de este espacio es la estándar $E = \{e_0, e_1, \ldots, e_{N-1}\}$, donde $e_j(n) = 1$ si $j = n$ y $e_j(n) = 0$ si $j \neq n$. $\ell^2(\mathbb{Z}_N)$ puede ser dotado del producto interno

$$\langle z, w \rangle = \sum_{k=0}^{N-1} z(k)\overline{w(k)},$$

con su norma asociada

$$\|z\| = \left(\sum_{k=0}^{N-1} |z(k)|^2 \right)^{1/2}.$$

Como convención, extenderemos z a todos los enteros de manera periódica, esto es:

$$z(j+N) = z(j), \text{ para todo } j \in \mathbb{Z}.$$

Definición 1.1. Definimos $E_0, E_1, \ldots, E_{N-1} \in \ell^2(\mathbb{Z}_n)$ por

$$E_m(n) = \frac{1}{\sqrt{N}} e^{2\pi i m n/N} \text{ para } 0 \leq m, n \leq N-1. \tag{1.1}$$

Lema 1.2. *El conjunto $\{E_0, E_1, \ldots, E_{N-1}\}$ forma una base ortonormal para $\ell^2(\mathbb{Z}_n)$.*

Definición 1.3. Sea $z = (z(0), \ldots z(N-1)) \in \ell^2(\mathbb{Z}_n)$. Para $m = 0, 1, \ldots, N-1$, definimos

$$\hat{z}(m) = \sum_{n=0}^{N-1} z(n) e^{2\pi i m n/N} \tag{1.2}$$

y así definimos $\hat{z}$ como

$$\hat{z} = (\hat{z}(0), \dots \hat{z}(N-1)),$$

según esto $\hat{z} \in \ell^2(\mathbb{Z}_n)$. La función $\hat{}: \ell^2(\mathbb{Z}_n) \to \ell^2(\mathbb{Z}_n)$, que toma z y lo lleva a $\hat{z}$ es llamada *transformada de Fourier discreta* y es abreviada como DFT.

OBSERVACIÓN 1.4. No es difícil notar, mediante la definición, que:

$$\hat{z}(m+N) = \hat{z}(m) = \sqrt{N}\langle z, E_m\rangle. \tag{1.3}$$

Esto nos lleva al siguiente resultado

TEOREMA 1.5. *Sea $z, w \in \ell^2(\mathbb{Z}_N)$. Entonces*

i. *(Fórmula de la transformada inversa de Fourier)*

$$z(n) = \frac{1}{N}\sum_{m=0}^{N-1} \hat{z}(m)e^{2\pi imi/N} \text{ para } n = 0, 1 \dots N-1. \tag{1.4}$$

ii. *(Identidad de Parseval)*

$$\langle z, w\rangle = \frac{1}{N}\sum_{m=0}^{N-1} \hat{z}(m)\overline{\hat{w}(m)} = \frac{1}{N}\langle \hat{z}, \hat{w}\rangle. \tag{1.5}$$

iii. *(Fórmula de Plancherel)*

$$\|z\|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |\hat{z}(k)|^2 = \frac{1}{N}\|\hat{z}\|^2 \tag{1.6}$$

DEFINICIÓN 1.6. Para $m = 0, 1, \dots, N-1$, definimos $F_m \in \ell^2(\mathbb{Z}_N)$ por

$$F_m(n) = \frac{1}{N}e^{2\pi imn/N}, \text{ para } n = 0, 1, \dots, N-1. \tag{1.7}$$

Sea

$$F = \{F_0, F_1, \dots F_{N-1}\}.$$

Llamaremos F a la base de Fourier de $\ell^2(\mathbb{Z}_N)$.

OBSERVACIÓN 1.7. Por la definición 1.1 podemos notar que $F_m = N^{1/2}E_m$, por esto, F es en efecto una base ortogonal. Además se tiene

$$z = \sum_{m=0}^{N-1} \hat{z}(m)F_m. \tag{1.8}$$

Esto nos dice que la representación en base de Fourier de un vector z es $\hat{z}$, es decir

$$\hat{z} = [z]_F.$$

La fórmula de inversión 1.4 sugiere que la transformación de Fourier es una función biyectiva y por ende invertible.

DEFINICIÓN 1.8. Para $w = (w(0), \dots, w(N-1)) \in \ell^2(\mathbb{Z}_n)$, definimos

$$\check{w}(n) = \frac{1}{N}\sum_{m=0}^{N-1} w(m)e^{2\pi imn/N}, \text{ para } n = 0, 1, \dots, N-1, \tag{1.9}$$

y así

$$\check{w} = (\check{w}(0), \check{w}(1), \dots, \check{w}(N-1)).$$

La función $\check{}: \ell^2(\mathbb{Z}_n) \to \ell^2(\mathbb{Z}_n)$ es la *transformada de Fourer discreta inversa*, abreviada IDFT.

DEFINICIÓN 1.9. Sea $z \in \ell^2(\mathbb{Z}_n)$ y $k \in \mathbb{Z}$. Definimos

$$(R_k z)(n) = z(n-k) \text{ para } n \in \mathbb{Z}.$$

Llamamos $R_k z$ la traslación de z en k unidades.

LEMA 1.10. *Sea $z \in \ell^2(\mathbb{Z}_n)$ y $k \in \mathbb{Z}$. Entonces para cualquiera $m \in \mathbb{Z}$,*

$$\widehat{(R_k z)}(m) = e^{-2\pi i m k/N} \hat{z}(m).$$

DEFINICIÓN 1.11. Para $z \in \ell^2(\mathbb{Z}_n)$, definimos $\overline{z}$ como

$$\overline{z} = (\overline{z(0)}, \ldots, \overline{z(N-1)}),$$

esto es, $\overline{z}(n) = \overline{z(n)}$.

LEMA 1.12. *Para $z \in \ell^2(\mathbb{Z}_n)$*

$$\widehat{(\overline{z})}(m) = \overline{\hat{z}(-m)} = \overline{\hat{z}(N-m)},$$

para todo $m \in \mathbb{Z}$.

DEFINICIÓN 1.13. Para $z, w \in \ell^2(\mathbb{Z}_n)$, la *convolución* $z * w \in \ell^2(\mathbb{Z}_n)$ es el vector con componentes

$$z * w(m) = \sum_{n=0}^{N-1} z(m-n) w(n),$$

para todo m.

LEMA 1.14. *Sean $z, w \in \ell^2(\mathbb{Z}_n)$. Entonces para cada m se tiene*

$$\widehat{(z * m)}(m) = \hat{z}(m)\hat{w}(m).$$

Diremos que un vector $z \in \ell^2(\mathbb{Z}_n)$ es localizado en el espacio cerca de n_0 si la mayoría de sus componentes $z(n)$ son cero o relativamente pequeñas, a excepción de algunos valores de n cercanos de n_0. Como veremos más adelante, esta idea de localización es una propiedad importante. Por ejemplo, la base de Fourier F_m no es localizada en el espacio ya que sus componentes tienen igual magnitud, $1/N$, para cada $n \in \mathbb{Z}_n$.

DEFINICIÓN 1.15. Para cada $w \in \ell^2(\mathbb{Z}_n)$, definimos $\tilde{w} \in \ell^2(\mathbb{Z}_n)$ por:

$$\tilde{w}(n) = \overline{w(-n)} = \overline{w(N-n)} \text{ para cada } n. \tag{1.10}$$

Llamaremos $\tilde{w}$ la *reflexión conjugada* de w.

No es difícil notar que para cada n se tiene

$$\widehat{(\tilde{w})}(n) = \overline{\tilde{w}(n)}. \tag{1.11}$$

LEMA 1.16. *Sean $z, w \in \ell^2(\mathbb{Z}_n)$. Para cualquier $k \in \mathbb{Z}$,*

$$z * \tilde{w}(k) = \langle z, R_k w \rangle \tag{1.12}$$

y

$$z * w(k) = \langle z, R_k \tilde{w} \rangle. \tag{1.13}$$

Este lema propone la siguiente idea. Sea $w \in \ell^2(\mathbb{Z}_n)$ tal que $B = \{R_k w\}_{k=0}^{N-1}$ es una base ortonormal de $\ell^2(\mathbb{Z}_n)$. Entonces los coeficientes de la expansión de un vector z en términos de B son los productos internos $\langle z, R_k w\rangle$ y por ende

$$[z]_B = z * \tilde{w}$$

debido a la primera ecuación del lema anterior, así $[z]_B$ puede ser obtenido de manera rápida mediante la transformada rápida de Fourier (revisar [1] página 151), debido a esto es de gran conveniencia pensar en las propiedad de una base localizada en frecuencias, es decir en el dominio de Fourier.

LEMA 1.17. *Sea $w \in \ell^2(\mathbb{Z}_n)$. Entonces $\{R_k w\}_{k=0}^{N-1}$ es una base ortonormal para $\ell^2(\mathbb{Z}_n)$ si y solo si $|\hat{w}(n)| = 1$ para todo $n \in \mathbb{Z}_n$.*

Esta condición parece ser simple y beneficiosa pero se aleja de nuestra intensión de obtener una base localizada. Por esta razón surge la siguiente definición, en busca de acercarnos a estas dos útiles propiedades.

DEFINICIÓN 1.18. Suponga N un entero par, digamos $N = 2M$ para algún $M \in \mathbb{N}$. Una base ortonormal de $\ell^2(\mathbb{Z}_n)$ de la forma

$$\{R_{2k} u\}_{k=0}^{M-1} \cup \{R_{2k} v\}_{k=0}^{M-1}$$

para $u, v \in \ell^2(\mathbb{Z}_n)$, es denominada *base ondícula* de primera etapa para $\ell^2(\mathbb{Z}_n)$. En lo que sigue se caracterizará el par u, v de manera que genere una base ondícula de primera etapa.

LEMA 1.19. *Sea $M \in \mathbb{N}$, $N = 2M$, y $z \in \ell^2(\mathbb{Z}_N)$. Si definimos $z^* \in \ell^2(\mathbb{Z}_n)$ por*

$$z^*(n) =)(-1)^n z(n) \text{ para todo } n. \tag{1.14}$$

Entonces

$$\widehat{(z^*)}(n) = \hat{z}(n + M) \text{ para todo } n. \tag{1.15}$$

LEMA 1.20. *Supongamos $M \in \mathbb{N}$, $N = 2M$, y $w \in \ell^2(\mathbb{Z}_N)$. Entonces $\{R_{2k} w\}_{k=0}^{M-1}$ es un conjunto ortonormal con M elementos si y solo si*

$$|\hat{w}(n)|^2 + |\hat{w}(n + M)|^2 = 2 \text{ para } n = 0, 1, \ldots, M - 1. \tag{1.16}$$

DEFINICIÓN 1.21. Sea $M \in \mathbb{N}$, $N = 2M$, y $u, v \in \ell^2(\mathbb{Z}_N)$. Para $n \in \mathbb{Z}$, definimos $A(n)$, el sistema de matrices asociado a u, v por

$$A(n) = \frac{1}{\sqrt{2}} \begin{bmatrix} \hat{u}(n) & \hat{v}(n) \\ \hat{u}(n + M) & \hat{v}(n + M) \end{bmatrix}. \tag{1.17}$$

TEOREMA 1.22. *Supongamos $M \in \mathbb{N}$, $N = 2M$. Sean $u, v \in \ell^2(\mathbb{Z}_N)$. Entonces*

$$B = \{R_{2k} u\}_{k=0}^{M-1} \cup \{R_{2k} v\}_{k=0}^{M-1}$$

es una base ortonormal para $\ell^2(\mathbb{Z}_N)$ si y solo si el sistema de matrices $A(n)$ de u, v es unitario para cada $n = 0, 1, \ldots, M-1$. Equivalentemente, B es una base ondícula de primera etapa para $\ell^2(\mathbb{Z}_N)$ si y solo si

$$|\hat{u}(n)|^2 + |\hat{u}(n + M)|^2 = 2, \tag{1.18}$$

$$|\hat{v}(n)|^2 + |\hat{v}(n + M)|^2 = 2, \tag{1.19}$$

y

$$\hat{u}(n)\overline{\hat{v}(n)} + \hat{u}(n + M)\overline{\hat{v}(n + M)} = 0, \tag{1.20}$$

para todo $n = 0, 1, \ldots, M - 1$.

EJEMPLO 1.23. Base de Shannon de primera etapa. Sea N divisible por 4. Definimos $\hat{u}, \hat{v} \in \ell^2(\mathbb{Z}_N)$ por

$$\hat{u}(n) = \begin{cases} \sqrt{2} & \text{si} \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \text{ ó } n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots N - 1, \\ 0 & \text{si} \quad n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{3N}{4} - 1, \end{cases} \tag{1.21}$$

y

$$\hat{v}(n) = \begin{cases} 0 & \text{si} \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \text{ ó } n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots N - 1, \\ \sqrt{2} & \text{si} \quad n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{3N}{4} - 1. \end{cases} \tag{1.22}$$

Usando el teorema anterior se puede concluir que $\{R_{2k}u\}_{k=0}^{M-1} \cup \{R_{2k}v\}_{k=0}^{M-1}$ es una base ortonormal.

LEMA 1.24. *Sea $M \in \mathbb{N}$, $N = 2M$, y $u \in \ell^2(\mathbb{Z}_N)$ tal que $\{R_{2k}u\}_{k=0}^{M-1}$ es un conjunto ortonormal con M elementos. Si definimos $v \in \ell^2(\mathbb{Z}_N)$ como*

$$v(k) = (-1)^{k-1}\overline{u(1-k)} \tag{1.23}$$

para cada k. Entonces $\{R_{2k}u\}_{k=0}^{M-1} \cup \{R_{2k}v\}_{k=0}^{M-1}$ es una base ondícula de primera etapa para $\ell^2(\mathbb{Z}_N)$.

Ahora que tenemos la base ondícula de primera etapa y podemos llevar un vector $z \in \ell^2(\mathbb{Z}_N)$ a coordenadas en esta base, de manera rápida por medio de la igualdad

$$[z]_B = \begin{bmatrix} z * \tilde{v}(0) \\ \vdots \\ z * \tilde{v}(N-2) \\ z * \tilde{u}(0) \\ \vdots \\ z * \tilde{u}(N-2) \end{bmatrix}. \tag{1.24}$$

Definimos a continuación dos operadores que se utilizaran en lo que sigue.

DEFINICIÓN 1.25. Sea $M \in \mathbb{N}$ y $N = 2M$. Definimos $D : \ell^2(\mathbb{Z}_N) \to \ell^2(\mathbb{Z}_M)$ para $z \in \ell^2(\mathbb{Z}_N)$ por

$$D(z)(n) = z(2n), \text{ para } n = 0, 1, \ldots, M - 1. \tag{1.25}$$

El operador D es denominado *downsampling*.

Este operador permite dejar solo las componentes que se utilizaran de los vectores $z * \tilde{v}, z * \tilde{u}$ para juntarlas y así formar $[z]_B$

DEFINICIÓN 1.26. Sea $M \in \mathbb{N}$ y $N = 2M$. Definimos $U : \ell^2(\mathbb{Z}_M) \to \ell^2(\mathbb{Z}_N)$ para $z \in \ell^2(\mathbb{Z}_M)$ por

$$U(z)(n) = \begin{cases} z(n/2) & \text{si} \quad n \text{ es par}, \\ 0 & \text{si} \quad n \text{ es impar}. \end{cases} \tag{1.26}$$

El operador U es denominado *upsampling*.

Como se ha mencionado anteriormente, buscamos aprovechar la propiedad de localización en el vector $[z]_B$ y así eliminar las componentes que son cercanas a cero, esto para mantener la mayor cantidad de información relevante y quitar la menos importante. Así es posible ahorrar espacio pensando en almacenar esta información en un dispositivo. Luego de esto, es necesario volver a nuestras coordenadas originales para así mostrar la información como se debe.

El siguiente esquema muestra estos procesos, donde se propone encontrar $\tilde{s}, \tilde{t}$ de manera que permitan recuperar el vector z de manera exacta. $\downarrow 2$ representa el operador downsampling y $\uparrow 2$ representa el operador upsampling.

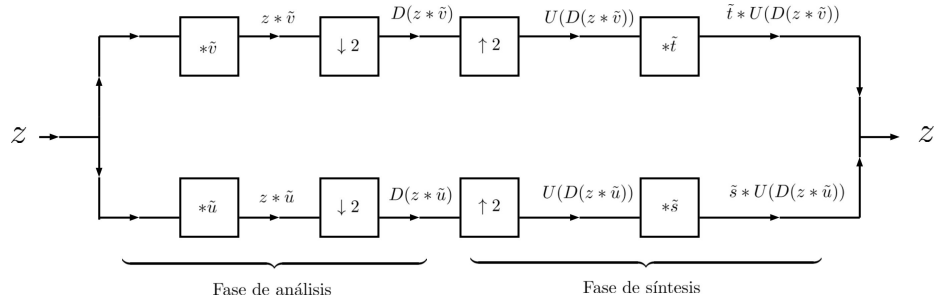


FIGURA 1. Proceso de transformación y reconstrucción de una señal mediante ondículas.

El siguiente resultado nos proporciona la caracterización de t y s para tener reconstrucción perfecta.

LEMA 1.27. *Sea $M \in \mathbb{N}$, $N = 2M$ y $u, v, s, t \in \ell^2(\mathbb{Z}_N)$. Para $n = 0, 1, \ldots, N-1$, sea $A(n)$ el sistema de matrices para u y v. Entonces se tiene reconstrucción perfecta mediante el proceso de la figura (a), esto es,*

$$\tilde{t} * U(D(z * \tilde{v})) + \tilde{s} * U(D(z * \tilde{u})) = z$$

para todo $z \in \ell^2(\mathbb{Z}_N)$, si y solo si

$$A(n) \begin{bmatrix} \hat{s}(n) \\ \hat{t}(n) \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} \tag{1.27}$$

para todo $n = 0, 1, \ldots, N-1$. En el caso que $A(n)$ es unitaria, se tiene que $\hat{t}(n) = \overline{\hat{v}(n)}$ y $\hat{s}(n) = \overline{\hat{u}(n)}$. Si $A(n)$ es unitaria para todo n entonces $t = \tilde{v}$ y $s = \tilde{u}$.

Capítulo 2

Ondículas en Imágenes

Una imagen en un computador cualquiera puede ser tratada como una matriz o arreglo de números donde el tamaño de esta dependerá de las dimensiones de la imagen. Por ejemplo, una imagen de 256×256 pixeles reportará una matriz del mismo tamaño. Es por esto que se necesita extender la noción de ondículas a dos dimensiones, para ello se trabaja en el espacio $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$.

1. Preliminares y definiciones en $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$.

Trabajaremos las ondículas en dos dimensiones en el espacio

$$\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}) = \Big\{ z = \big[z(i,j)\big] : z(i,j) \in \mathbb{C}, 0 \leq i \leq N_1, 0 \leq j \leq N_2 \Big\}.$$

El cual es un $\mathbb{C}$-espacio vectorial con la suma componente a componente y multiplicación por escalar usual. Se puede dotar del producto interno, para $z, w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$

$$\langle z, w \rangle = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} z(n_1, n_2)\overline{w(n_1, n_2)}. \tag{1.1}$$

Podemos considerar la base $\{e_{n_1,n_2}\}_{n_1 \in \mathbb{Z}_{N_1}}^{n_2 \in \mathbb{Z}_{N_2}}$ como

$$e_{n1,n2}(i,j) = \begin{cases} 1, & \text{si } i = n_1, j = n_2, \\ 0, & \text{en otro caso.} \end{cases} \tag{1.2}$$

de donde se desprende que $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ es $N_1 \cdot N_2$ dimensional.

Proposición 2.1. *Sea $\{B_0, \ldots, B_{N_1-1}\}$ y $\{C_0, \ldots, C_{N_1-1}\}$ bases ortogonales (ortonormales) de $\ell^2(\mathbb{Z}_{N_1})$ y $\ell^2(\mathbb{Z}_{N_2})$ respectivamente. Para $0 \leq m_1 \leq N_1 - 1$ y $0 \leq m_2 \leq N_2 - 1$ definimos $D_{m_1,m_2} \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ por*

$$D_{m_1,m_2}(n_1, n_2) = B_{m_1}(n_1) \cdot C_{m_2}(n_2). \tag{1.3}$$

De acuerdo a esta definición $D = \{D_{m_1,m_2}\}_{m_1 \in \mathbb{Z}_{N_1}}^{m_2 \in \mathbb{Z}_{N_2}}$ es una base ortogonal (ortonormal) de $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$.

Demostración. Sean $0 \leq m_1, k_1 \leq N_1 - 1$ y $0 \leq m_2, k_2 \leq N_2 - 1$, notemos que

$$\begin{aligned}
\langle D_{m_1,m_2}, D_{k_1,k_2} \rangle_{\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})} &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} D_{m_1,m_2}(n_1,n_2)\overline{D_{k_1,k_2}(n_1,n_2)} \\
&= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} B_{m_1}(n_1)C_{m_2}(n_2)\overline{B_{k_1}(n_2)C_{k_2}(n_2)} \\
&= \sum_{n_1=0}^{N_1-1} B_{m_1}(n_1)\overline{B_{k_1}(n_2)} \sum_{n_2=0}^{N_2-1} C_{m_2}(n_2)\overline{C_{k_2}(n_2)} \\
&= \langle B_{m_1}, B_{k_1} \rangle_{\ell^2(\mathbb{Z}_{N_2})} \cdot \langle C_{m_2}, C_{k_2} \rangle_{\ell^2(\mathbb{Z}_{N_2})} \\
&= \delta_{m_1,k_1} \cdot \delta_{m_2,k_2} \\
&= \begin{cases} 1, & \text{si } m_1 = k_1, m_2 = k_2, \\ 0, & \text{en otro caso.} \end{cases}
\end{aligned}$$

Esta última igualdad nos dices que tenemos $N1 \cdot N2$ elementos ortogonales y de norma 1 (en el caso que las bases sean ortonormales) por ende el conjunto D es una base ortonormal. ■

En este caso se especificó el producto interno correspondiente a cada espacio, en lo que sigue se omitirá esta especificación.

De acuerdo a este resultado, naturalmente se define:

DEFINICIÓN 2.2. Definimos la base $E_f = \{E_{m_1,m_2}\}_{m_1 \in \mathbb{Z}_{N_1}}^{m_2 \in \mathbb{Z}_{N_2}}$, de $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ como

$$E_{m_1,m_2}(n_1,n_2) = \frac{1}{\sqrt{N_1 N_2}} e^{2\pi i m_1 n_1/N_1} e^{2\pi i m_2 n_2/N_2} = E_{m_1}(n_1) \cdot E_{m_2}(n_2).$$

Análogamente la base de Fourier, $F = \{F_{m_1,m_2}\}_{m_1 \in \mathbb{Z}_{N_1}}^{m_2 \in \mathbb{Z}_{N_2}}$ como

$$F_{m_1,m_2}(n_1,n_2) = \frac{1}{N_1 N_2} e^{2\pi i m_1 n_1/N_1} e^{2\pi i m_2 n_2/N_2} = F_{m_1}(n_1) \cdot F_{m_2}(n_2).$$

Se sigue de la proposición anterior (más observación 1.4) que estos conjuntos son efectivamente una base ortonormal y ortogonal respectivamente. A continuación extendemos la transformada de Fourier discreta a dos dimensiones.

DEFINICIÓN 2.3. Para $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ definimos $\hat{z} \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ como

$$\hat{z}(m_1,m_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} z(n_1,n_2) e^{-2\pi i m_1 n_1/N_1} e^{-2\pi i m_2 n_2/N_2}. \tag{1.4}$$

La función$\hat{}: \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}) \to \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ que a z le asocia $\hat{z}$ es denominada la *transformada de Fourier discreta en dos dimensiones.*

PROPOSICIÓN 2.4. *La transformada de Fourier discreta en dos dimensiones en una función biyectiva.*

Demostración. Basta notar que

$$\hat{F}_{m_1,m_2}(n_1,n_2) = \begin{cases} 1/(N_1 N_2), & \text{si } m_1 = n_1, m_2 = n_2, \\ 0, & \text{en otro caso.} \end{cases}$$

Por lo tanto la transformada de Fourier discreta en dos dimensiones lleva base ortogonales en bases ortogonales. ■

Al igual que en una dimensión $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ se puede extender para todo $(n_1, n_2) \in \mathbb{Z} \times \mathbb{Z}$ con periodo N_1 y N_2 para la primera y segunda variable respectivamente, esto es,

$$z(n_1 + N_1, n_2 + N_2) = z(n_1, n_2). \tag{1.5}$$

Ya que la transformada de Fourier discreta en dos dimensiones en una función biyectiva es natural pensar en su transformada inversa.

DEFINICIÓN 2.5. Para $w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ definimos $\check{w} \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ como

$$\check{w}(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} w(m_1, m_2) e^{2\pi i m_1 n_1 / N_1} e^{2\pi i m_2 n_2 / N_2}. \tag{1.6}$$

La función $\check{}: \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}) \to \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ que a w le asocia $\check{w}$ es denominada la *transformada inversa de Fourier discreta en dos dimensiones.*

Veamos que efectivamente esta transformación hace honor a su nombre.

PROPOSICIÓN 2.6. *Para $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ tenemos $(\hat{z})\check{} = z$*

Demostración.

$$\begin{aligned}
&(\hat{z})\check{}(n_1, n_2) = \\
&= \frac{1}{N_1 N_2} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} (\hat{z}(m_1, m_2))\, e^{2\pi i m_1 n_1 / N_1} e^{2\pi i m_2 n_2 / N_2} \\
&= \frac{1}{N_1 N_2} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} \left(\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} z(k_1, k_2) e^{-2\pi i m_1 k_1 / N_1} e^{-2\pi i m_2 k_2 / N_2} \right) e^{2\pi i m_1 n_1 / N_1} e^{2\pi i m_2 n_2 / N_2} \\
&= \frac{1}{N_1 N_2} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} \left(\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} z(k_1, k_2) \delta_{k_1, m_1} \cdot \delta_{k_2, m_2} \right) \\
&= \frac{1}{N_1 N_2} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} z(n_1, n_2) \\
&= z(n_1, n_2).
\end{aligned}$$

■

Revisemos ahora una propiedad interesante.

PROPOSICIÓN 2.7. *Sean $w, z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ entonces*

$$\langle z, w \rangle = \frac{1}{N_1 \cdot N_2} \langle \hat{z}, \hat{w} \rangle \tag{1.7}$$

Demostración. Primero notemos que

$$\begin{aligned}
\langle z, E_{n_1, n_2} \rangle &= \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} z(m_1, m_2) E_{n_1, n_2}(m_1, m_2) \\
&= \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} z(m_1, m_2) \frac{1}{\sqrt{N_1 N_2}} e^{2\pi i m_1 n_1 / N_1} e^{2\pi i m_2 n_2 / N_2} \\
&= \check{z}(n_1, n_2) \cdot \sqrt{N_1 N_2}
\end{aligned}$$

Ahora, como E_f es una base ortonormal, la identidad de Parseval del álgebra lineal nos proporciona la primera igualdad, luego, utilizando lo mostrado arriba se sigue que

$$\begin{aligned}\langle z', w' \rangle &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \langle z', E_{n_1,n_2} \rangle \overline{\langle w', E_{n_1,n_2} \rangle} \\ &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} N_1 N_2 \hat{z'}(n_1, n_2) \overline{\hat{w'}(n_1, n_2)} \\ &= N_1 N_2 \langle \hat{z'}, \hat{w'} \rangle.\end{aligned}$$

Tomando $z' = \check{z}, w' = \check{w}$ tenemos

$$\langle \hat{z}, \hat{w} \rangle = N_1 N_2 \langle z, w \rangle$$

y el resultado se sigue. ■

DEFINICIÓN 2.8. *(Convolución en dos dimensiones.)* Sean $z, w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$, la convolución $z * w$ entre z y w es el vector con componentes

$$z * w(m_1, m_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-2} z(m_1 - n_1, m_2 - n_2) w(n_1, n_2), \tag{1.8}$$

para cada m_1, m_2.

Revisemos algunas propiedades que se cumplen para la convolución en dos dimensiones.

PROPOSICIÓN 2.9. *(Conmutatividad de la convolución.) Sean $z, w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ entonces se tiene*

$$z * w = w * z. \tag{1.9}$$

Demostración. Considerando el cambio $m_1 - n_1 = k_1$ y $m_2 - n_2 = k_2$ y la periodicidad de z

$$\begin{aligned}z * w(m_1, m_2) &= \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-2} z(m_1 - n_1, m_2 - n_2) w(n_1, n_2) \\ &= \sum_{k_1=m_1}^{m_1-N-1} \sum_{k_2=m_2}^{m_2-N-2} z(k_1, k_2) w(m_1 - k_1, m_2 - k_2) \\ &= \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} z(k_1, k_2) w(m_1 - k_1, m_2 - k_2) \\ &= \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} w(m_1 - k_1, m_2 - k_2) z(k_1, k_2) \\ &= w * z(m_1, m_2).\end{aligned}$$

■

Verifiquemos ahora que el buen comportamiento de la transformada de Fourier con la convolución se mantiene en dos dimensiones.

PROPOSICIÓN 2.10. *Para $z, w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ y cada m_1, m_2 enteros se cumple que*

$$\widehat{(z * w)}(m_1, m_2) = \hat{z}(m_1, m_2) \hat{w}(m_1, m_2). \tag{1.10}$$

Demostración. Se utilizara el cambio $n_1 - k_1 = \ell_1$ y $n_2 - k_2 = \ell_2$ y bajo la periodicidad de z (análogo a la demostración anterior) se cambian los indices en la suma correspondiente.

$$
\begin{aligned}
&(z * w)\hat{}(m_1, m_2) = \\
&\quad = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} (z * w)(n_1, n_2) e^{-2\pi i m_1 n_1/N_1} e^{-2\pi i m_2 n_2/N_2} \\
&\quad = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left(\sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} z(n_1 - k_1, n_2 - k_2) w(k_1, k_2) \right) e^{-2\pi i m_1 n_1/N_1} e^{-2\pi i m_2 n_2/N_2} \\
&\quad = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left(\sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} z(n_1 - k_1, n_2 - k_2) w(k_1, k_2) \right) \cdot \\
&\qquad\qquad e^{-2\pi i m_1 (n_1-k_1)/N_1} e^{-2\pi i m_2 (n_2-k_2)/N_2} e^{-2\pi i m_1 k_1/N_1} e^{-2\pi i m_2 k_2/N_2} \\
&\quad = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} w(k_1, k_2) e^{-2\pi i m_1 k_1/N_1} e^{-2\pi i m_2 k_2/N_2} \cdot \\
&\qquad\qquad \left(\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} z(n_1 - k_1, n_2 - k_2) e^{-2\pi i m_1 (n_1-k_1)/N_1} e^{-2\pi i m_2 (n_2-k_2)/N_2} \right) \\
&\quad = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-2} w(k_1, k_2) e^{-2\pi i m_1 k_1/N_1} e^{-2\pi i m_2 k_2/N_2} \cdot \\
&\qquad\qquad \left(\sum_{\ell_1=0}^{N_1-1} \sum_{\ell_2=0}^{N_2-1} z(\ell_1, \ell_2) e^{-2\pi i m_1 \ell_1/N_1} e^{-2\pi i m_2 \ell_2/N_2} \right) \\
&\quad = \hat{z}(m_1, m_2) \hat{w}(m_1, m_2).
\end{aligned}
$$

■

A continuación definimos el operador de traslación para caso bidimensional.

DEFINICIÓN 2.11. Denotaremos por $R_{k_1,k_2} z$ a la traslación de z en k_1 unidades en la primera componente y k_2 unidades en la segunda componente, esto es

$$R_{k_1,k_2} z(n_1, n_2) = z(n_1 - k_1, n_2 - k_2). \tag{1.11}$$

Otra definición importante y útil para nuestro propósito es la de reflexión conjugada.

DEFINICIÓN 2.12. Para $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ definimos $\tilde{w}(z) \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ por

$$\tilde{w}(n_1, n_2) = \overline{w(-n_1, -n_2)} = \overline{w(N_1 - n_1, N_2 - n_2)}, \text{ para cada } n_1, n_2 \text{ enteros.} \tag{1.12}$$

Denominamos a $\tilde{w}$ la *reflexión conjugada* de w.

Notemos que

$$\begin{aligned}
(\tilde{w})\hat{}(n_1,n_2) &= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} \tilde{w}(k_1,k_2)e^{-2\pi i k_1 n_1/N_1}e^{-2\pi i k_2 n_2/N_2} \\
&= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} \overline{w(N_1-k_1,N_2-k_2)}e^{-2\pi i k_1 n_1/N_1}e^{-2\pi i k_2 n_2/N_2} \\
&= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} \overline{w(N_1-k_1,N_2-k_2)}e^{-2\pi i k_1 n_1/N_1}e^{-2\pi i k_2 n_2/N_2}e^{2\pi i N_1 m_1/N_1}e^{2\pi i N_2 m_2/N_2} \\
&= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} \overline{w(N_1-k_1,N_2-k_2)}e^{2\pi i (N_1-k_1) n_1/N_1}e^{2\pi i (N_2-k_2) n_2/N_2} \\
&= \overline{\hat{w}(n_1,n_2)}
\end{aligned}$$

Por lo tanto hemos probado que

LEMA 2.13. *Para $w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ se tiene*

$$(\tilde{w})\hat{}(n_1,n_2) = \overline{\hat{w}(n_1,n_2)}. \tag{1.13}$$

Presentamos el siguiente Lema pensando en expresar un vector z de manera eficiente en ciertos tipos de bases.

LEMA 2.14. *Sean $z, w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$. Para cualquiera k_1, k_2 enteros se tiene*

$$z * \tilde{w}(k_1,k_2) = \langle z, R_{k_1,k_2} w\rangle \tag{1.14}$$

y

$$z * w(k_1,k_2) = \langle z, R_{k_1,k_2} \tilde{w}\rangle. \tag{1.15}$$

Demostración. Por definición

$$\begin{aligned}
\langle z, R_{k_1,k_2} w\rangle &= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-2} z(n_1,n_2)\overline{R_{k_1,k_2} w(n_1,n_2)} \\
&= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-2} z(n_1,n_2)\overline{w(n_1-k_1,n_2-k_2)} \\
&= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-2} z(n_1,n_2)\tilde{w}(k_1-n_1,k_2-n_2) \\
&= \tilde{w} * z(k_1,k_2) \\
&= z * \tilde{w}(k_1,k_2).
\end{aligned}$$

La segunda igualdad surge del hecho que $\tilde{\tilde{w}} = w$. ■

Similarmente al caso unidimensional, buscamos una base que sea de frecuencia localizada (para tener la información concentrada en algunos vectores de la base) y así aprovechar las propiedades de la convolución y la transformada de Fourier vistas hasta ahora. Si tenemos una base de la forma $\{R_{n_1,n_2} w\}_{n_1\in\mathbb{Z}_{N_1}}^{n_2\in\mathbb{Z}_{N_2}}$ podemos aprovechar que $[z]_B = z * \tilde{w}$, esto debido a la identidad de Parseval y el resultado anterior. Lamentablemente, de igual manera que en el caso unidimensional tenemos los siguientes resultados.

LEMA 2.15. *Sea $u \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$. Entonces $\{R_{n_1,n_2}u\}_{n_1\in\mathbb{Z}_{M_1}}^{n_2\in\mathbb{Z}_{M_2}}$ es un conjunto ortonormal de $M_1 \cdot M_2$ si, y solamente si,*

$$\langle u, R_{k_1,k_2}u\rangle = \begin{cases} 1, & \text{si } k_1 = k_2 = 0, \\ 0, & \text{en otro caso.} \end{cases} \tag{1.16}$$

Demostración. Notemos primero que si $\{R_{n_1,n_2}u\}_{n_1\in\mathbb{Z}_{M_1}}^{n_2\in\mathbb{Z}_{M_2}}$ es un conjunto ortonormal entonces

$$\langle R_{s_1,s_2}u, R_{k_1,k_2}u\rangle = \begin{cases} 1, & \text{si } k_1 = s_1 \text{ y } k_2 = s_2, \\ 0, & \text{en otro caso.} \end{cases}$$

para todo $s_1, k_1 \in \mathbb{Z}_{M_1}$ y $s_2, k_2 \in \mathbb{Z}_{M_2}$, en particular para $s_1 = s_2 = 0$.

Para la implicancia en el otro sentido consideremos $s_1 - k_1 = n_1$ y $s_2 - k_2 = n_2$

$$\begin{aligned}
\langle R_{s_1,s_2}u, R_{k_1,k_2}u\rangle &= \sum_{n_1'=0}^{M_1-1}\sum_{n_2'=0}^{M_2-1} R_{s_1,s_2}u(n_1',n_2')R_{k_1,k_2}u(n_1',n_2') \\
&= \sum_{n_1'=0}^{M_1-1}\sum_{n_2'=0}^{M_2-1} R_{n_1+k_1,n_2+k_2}u(n_1',n_2')R_{k_1,k_2}u(n_1',n_2') \\
&= \sum_{n_1'=0}^{M_1-1}\sum_{n_2'=0}^{M_2-1} R_{k_1,k_2}\left(R_{n_1,n_2}u(n_1',n_2')\cdot u(n_1',n_2')\right) \\
&= \sum_{n_1'=0}^{M_1-1}\sum_{n_2'=0}^{M_2-1} R_{n_1,n_2}u(n_1',n_2')\cdot u(n_1',n_2') \\
&= \begin{cases} 1, & \text{si } n_1 = n_2 = 0, \\ 0, & \text{en otro caso.} \end{cases}
\end{aligned}$$

Como observación, se sacó la traslación R_{k_1,k_2} de la expresión pues estamos sumando sobre todos los términos, por lo tanto, la traslación sólo produce un cambio en el orden de los sumandos. Si $n_1 = n_2 = 0$ tenemos que $s_1 = k_1$ y $s_2 = k_2$ por lo que

$$\langle R_{s_1,s_2}u, R_{k_1,k_2}u\rangle = \begin{cases} 1, & \text{si } k_1 = s_1 \text{ y } k_2 = s_2, \\ 0, & \text{en otro caso.} \end{cases}$$

y así el conjunto es ortonormal. ■

PROPOSICIÓN 2.16. *Sea $w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$. Entonces $\{R_{n_1,n_2}w\}_{n_1\in\mathbb{Z}_{N_1}}^{n_2\in\mathbb{Z}_{N_2}}$ es una base ortonormal de $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ si y solo si $|\hat{w}(n_1,n_2)| = 1$ para cada $(n_1,n_2) \in \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$.*

Demostración. Del Lema (2.15) tenemos que

$$\langle w, R_{n_1,n_2}w\rangle = \begin{cases} 1, & \text{si } n_1 = n_2 = 0, \\ 0, & \text{en otro caso.} \end{cases} \tag{1.17}$$

además sabemos que $\langle w, R_{n_1,n_2}w\rangle = w * \tilde{w}(n_1,n_2)$ del Lema 2.14 por lo tanto si aplicamos la transformada de Fourier tenemos $(w*\tilde{w})\hat{}(n_1,n_2) = 1$ para todo $(n_1,n_2) \in \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$ entonces, utilizando el resultado 2.13 tenemos

$$1 = (w*\tilde{w})\hat{}(n_1,n_2) = \hat{w}(n_1,n_2)(\tilde{w})\hat{}(n_1,n_2) = \hat{w}(n_1,n_2)\overline{\hat{w}(n_1,n_2)} = |\hat{w}(n_1,n_2)|^2.$$

■

El resultado anterior nos aleja de obtener una base con las propiedades de localización en frecuencia, por esto debemos seguir en busca de una base que se acerque a aquellas propiedades.

DEFINICIÓN 2.17. Sean $M_1, M_2 \in \mathbb{N}$ y $N_1 = 2M_1$, $N_2 = 2M_2$. Sea $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$, definimos z^*, z^{**} y z^{***} como

$$\begin{aligned} z^*(n_1, n_2) &= (-1)^{n_1} z(n_1, n_2), \\ z^{**}(n_1, n_2) &= (-1)^{n_2} z(n_1, n_2), \\ z^{***}(n_1, n_2) &= (-1)^{n_1+n_2} z(n_1, n_2). \end{aligned}$$

De acuerdo a esto no es difícil notar que

$$z(n_1, n_2) + z(n_1, n_2)^* + z(n_1, n_2)^{**} + z(n_1, n_2)^{***} = \begin{cases} 4z(n_1, n_2), & \text{si } n_1, n_2 \text{ son pares ,} \\ 0, & \text{en otro caso.} \end{cases} \tag{1.18}$$

Además se tiene que

$$(z^*)\hat{}(n_1, n_2) = \hat{z}(n_1 + M_1, n_2), \tag{1.19}$$

$$(z^{**})\hat{}(n_1, n_2) = \hat{z}(n_1, n_2 + M_2), \tag{1.20}$$

$$(z^{***})\hat{}(n_1, n_2) = \hat{z}(n_1 + M_1, n_2 + M_2). \tag{1.21}$$

Esto último pues

$$\begin{aligned} (z^*)\hat{}(n_1, n_2) &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} z * (k_1, k_2) e^{-2\pi i k_1 n_1/N_1} e^{-2\pi i k_2 n_2/N_2} \\ &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} (-1)^{k_1} z(k_1, k_2) e^{-2\pi i k_1 n_1/N_1} e^{-2\pi i k_2 n_2/N_2} \\ &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} e^{-i\pi k_1} z(k_1, k_2) e^{-2\pi i k_1 n_1/N_1} e^{-2\pi i k_2 n_2/N_2} \\ &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} z(k_1, k_2) e^{-2\pi i k_1 (n_1+M_1)/N_1} e^{-2\pi i k_2 n_2/N_2} \\ &= \hat{z}(n_1 + M_1, n_2). \end{aligned}$$

Análogo para los dos casos restantes.

LEMA 2.18. *Sean $M_1, M_2 \in \mathbb{N}$ y $N_1 = 2M_1$, $N_2 = 2M_2$. Si tomamos $w \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ el conjunto $R = \{R_{2k_1,2k_2} w\}_{n_1 \in \mathbb{Z}_{M_1}}^{n_2 \in \mathbb{Z}_{M_2}}$ es un conjunto ortonormal con $M_1 M_2$ elementos si y solo si*

$$|\hat{w}(n_1, n_2)|^2 + |\hat{w}(n_1 + M_1, n_2)|^2 + |\hat{w}(n_1, n_2 + M_2)|^2 + |\hat{w}(n_1 + M_1, n_2 + M_2)|^2 = 4 \tag{1.22}$$

para $n_1 \in \mathbb{Z}_{N_1}$ y $n_2 \in \mathbb{Z}_{N_2}$.

Demostración. Sabemos del Lema 2.15 que R es ortonormal si y solo si

$$\begin{aligned} w * \tilde{w}(2k_1, 2k_2) &= \langle w, R_{2k_1,2k_2} w \rangle \\ &= \begin{cases} 1, & \text{si } k_1, k_2 = 0, \\ 0, & \text{en otro caso.} \end{cases} \end{aligned}$$

Además sabemos de la ecuación 1.18 que

$$\begin{aligned} w * \tilde{w}(n_1, n_2) + (w * \tilde{w})^*(n_1, n_2) + (w * \tilde{w})^{**}(n_1, n_2) + (w * \tilde{w})^{***}(n_1, n_2) \\ = \begin{cases} 4w * \tilde{w}(n_1, n_2), & \text{si } n_1, n_2 \text{ son pares ,} \\ 0, & \text{en otro caso.} \end{cases} \end{aligned}$$

Juntando esto tenemos que la primera ecuación se cumple para valores pares si y solo si

$$(w*\tilde{w}+(w*\tilde{w})^{*}+(w*\tilde{w})^{**}+(w*\tilde{w})^{***})(2k_1,2k_2)=\begin{cases}4, & \text{si } k_1,k_2=0,\\ 0, & \text{en otro caso.}\end{cases}$$

Para valores impares de n_1, n_2, $(w*\tilde{w}+(w*\tilde{w})^{*}+(w*\tilde{w})^{**}+(w*\tilde{w})^{***})(n_1,n_2)$ es automáticamente cero. Entonces la primera ecuación se cumple si y solo si

$$(w*\tilde{w}+(w*\tilde{w})^{*}+(w*\tilde{w})^{**}+(w*\tilde{w})^{***})=4\delta.$$

Aplicando la transformada de Fourier obtenemos, para cada n_1, n_2,

$$\begin{aligned}
&(w*\tilde{w}+(w*\tilde{w})^{*}+(w*\tilde{w})^{**}+(w*\tilde{w})^{***})\hat{}\,(n_1,n_2)=(4\delta)\hat{}\,(n_1,n_2)\\
\iff\quad &((w*\tilde{w})\hat{}+((w*\tilde{w})^{*})\hat{}+((w*\tilde{w})^{**})\hat{}+((w*\tilde{w})^{***})\hat{})(n_1,n_2)=4\\
\iff\quad &\hat{w}(n_1,n_2)\overline{\tilde{w}(n_1,n_2)}+\hat{w}(n_1+M_1,n_2)\overline{\tilde{w}(n_1+M_1,n_2)}\\
&+\hat{w}(n_1,n_2+M_2)\overline{\tilde{w}(n_1,n_2+M_2)}+\hat{w}(n_1+M_1,n_2+M_2)\overline{\tilde{w}(n_1+M_1,n_2+M_2)}=4\\
\iff\quad &|\hat{w}(n_1,n_2)|^2+|\hat{w}(n_1+M_1,n_2)|^2+|\hat{w}(n_1,n_2+M_2)|^2+|\hat{w}(n_1+M_1,n_2+M_2)|^2=4.
\end{aligned}$$

Estas equivalencias debido a la periodicidad de w y a las ecuaciones 1.19 a 1.21. ■

TEOREMA 2.19. *Sean $u_0,u_1,u_2,u_3\in\ell^2(\mathbb{Z}_{N_1}\times\mathbb{Z}_{N_2})$. Sea $A(n_1,n_2)$ una matriz cuya m-esima fila es*

$$\frac{1}{2}\begin{bmatrix}\hat{u}_m(n_1,n_2)\\ \hat{u}_m(n_1+M_1,n_2)\\ \hat{u}_m(n_1,n_2+M_2)\\ \hat{u}_m(n_1+M_1,n_2+M_2)\end{bmatrix}\tag{1.23}$$

para $m=0,1,2,3$. Entonces el conjunto

$$U=\bigcup_{i=0}^{3}\{R_{2n_1,2n_2}u_i\}_{n_1\in\mathbb{Z}_{M_1}}^{n_2\in\mathbb{Z}_{M_2}}$$

es una base ortonormal de $\ell^2(\mathbb{Z}_{N_1}\times\mathbb{Z}_{N_2})$ si, y solamente si, $A(n_1,n_2)$ es unitaria para todo $n_1=0,1,\ldots,M_1-1$ y todo $n_2=0,1,\ldots,M_2-1$, con $2M_1=N_1$ y $2M_2=N_2$ enteros.

Demostración. Verificaremos que

$$\langle R_{2n_1,2n_2}u_i,R_{2k_1,2k_2}u_j\rangle=\begin{cases}1, & \text{si } n_1=k_1,n_2=k_2,i=j,\\ 0, & \text{en otro caso,}\end{cases}$$

para $j,i=0,1,2,3$, $n_1,k_1=0,\ldots,M_1$ y $n_2,k_2=0,\ldots,M_1$, es equivalente a que nuestras matrices $A(n_1,n_2)$ sean unitarias. De Acuerdo a los resultados de álgebra lineal sabemos que para que una matriz sea unitaria es necesario y suficiente que cada columna tenga norma uno y entre si sean ortogonales. Cada columna tiene norma uno si

$$\left(\frac{1}{2}\left(|\hat{u}_m(n_1,n_2)|^2+|\hat{u}_m(n_1+M_1,n_2)|^2+|\hat{u}_m(n_1,n_2+M_2)|^2+|\hat{u}_m(n_1+M_1,n_2+M_2)|^2\right)\right)^{1/2}=1$$

cual resultado es equivalente a nuestro Lema 2.18 por lo tanto resta verificar que

$$\langle R_{2n_1,2n_2}u_i,R_{2k_1,2k_2}u_j\rangle=\begin{cases}1, & \text{si } n_1=k_1,n_2=k_2\\ 0, & \text{en otro caso,}\end{cases}$$

para $j,i=0,1,2,3$, $n_1,k_1=0,\ldots,M_1$ y $n_2,k_2=0,\ldots,M_1$ con $j\neq i$. Análogamente a la demostración del Lema 2.14 esto es equivalente a

$$\langle u_i,R_{2k_1,2k_2}u_j\rangle=\begin{cases}1, & \text{si } k_1,k_2=0\\ 0, & \text{en otro caso,}\end{cases}$$

para $j, i = 0, 1, 2, 3$, $k_1 = 0, \ldots, M_1$ y $k_2 = 0, \ldots, M_1$ con $j \neq i$. Esta expresión a su vez es equivalente por el Lema 2.14 a $u_i * \tilde{u}_j(2k_1, 2k_2)$. Por lo tanto si $i \neq j$ se tiene (para los valores impares de n_1, n_2 la expresión es automáticamente cero por la ecuación 1.18)

$$\begin{aligned}
& (u_i * \tilde{u_j} + (u_i * \tilde{u_j})^* + (u_i * \tilde{u_j})^{**} + (u_i * \tilde{u_j})^{***})\hat{}\,(n_1, n_2) = 0 \\
\iff \quad & ((u_i * \tilde{u_j})\hat{} + ((u_i * \tilde{u_j})^*)\hat{} + ((u_i * \tilde{u_j})^{**})\hat{} + ((u_i * \tilde{u_j})^{***})\hat{})(n_1, n_2) = 0 \\
\iff \quad & \hat{u}_i(n_1, n_2)\overline{\tilde{u}_j(n_1, n_2)} + \hat{u}_i(n_1 + M_1, n_2)\overline{\tilde{u}_j(n_1 + M_1, n_2)} \\
& + \hat{u}_i(n_1, n_2 + M_2)\overline{\tilde{u}_j(n_1, n_2 + M_2)} + \hat{u}_i(n_1 + M_1, n_2 + M_2)\overline{\tilde{u}_j(n_1 + M_1, n_2 + M_2)} = 0.
\end{aligned}$$

Esta última igualdad nos dice que las columnas son ortogonales y así tenemos la equivalencia. ■

PROPOSICIÓN 2.20. *Sean $M_1, M_2 \in \mathbb{Z}, 2M_1 = N_1, 2M_2 = N_2$. Supongamos que $\{R_{2k}v_1\}_{k=0}^{N_1-1} \cup \{R_{2k}u_1\}_{k=0}^{N_1-1}$ es una base de ondículas de primera etapa para $\ell^2(\mathbb{Z}_{N_1})$ y $\{R_{2k}v_2\}_{k=0}^{N_2-1} \cup \{R_{2k}u_2\}_{k=0}^{N_1-1}$ es una base de ondículas de primera etapa para $\ell^2(\mathbb{Z}_{N_2})$. Si definimos*

$$\begin{aligned}
w_0(n_1, n_2) &= v_1(n_1) \cdot v_2(n_2), \\
w_1(n_1, n_2) &= u_1(n_1) \cdot v_2(n_2), \\
w_2(n_1, n_2) &= v_1(n_1) \cdot u_2(n_2), \\
w_3(n_1, n_2) &= u_1(n_1) \cdot u_2(n_2),
\end{aligned}$$

Entonces

$$\bigcup_{i=0}^{3} \{R_{2k_1, 2k_2} w_i\}_{n_1 \in \mathbb{Z}_{M_1}}^{n_2 \in \mathbb{Z}_{M_2}} \tag{1.24}$$

es una base ortonormal de $\ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$.

Demostración. Notemos que

$$\begin{aligned}
R_{2k_1, 2k_2} w_0(n_1, n_2) &= w_0(n_1 - 2k_1, n_2 - 2k_2) \\
&= v_1(n_1 - 2k_1) \cdot v_2(n_2 - 2k_2) \\
&= R_{2k_1} v_1(n_1) R_{2k_2} v_2(n_2).
\end{aligned}$$

Realizando lo mismo para el resto de los vectores, el resultado se sigue de la proposición 2.1. ■

Extendemos a continuación los operadores D y U a dos dimensiones.

DEFINICIÓN 2.21. Dados $M_1, M_2 \in \mathbb{Z}$, $2M_1 = N_1$, $2M_2 = N_2$. Definimos el operador $D : \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}) \to \ell^2(\mathbb{Z}_{M_1} \times \mathbb{Z}_{M_2})$ llamado *downsampling* para $z \in \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ por

$$D(z)(k_1, k_2) = z(2k_1, 2k_2) \tag{1.25}$$

DEFINICIÓN 2.22. Dados $M_1, M_2 \in \mathbb{Z}$, $2M_1 = N_1$, $2M_2 = N_2$. Definimos el operador $U : \ell^2(\mathbb{Z}_{M_1} \times \mathbb{Z}_{M_2}) \to \ell^2(\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2})$ llamado *upsampling* para $z \in \ell^2(\mathbb{Z}_{M_1} \times \mathbb{Z}_{M_2})$ por

$$U(z)(k_1, k_2) = \begin{cases} z(\frac{k_1}{2}, \frac{k_2}{2}), & \text{si } k_1, k_2 \text{ son pares} \\ 0, & \text{en otro caso.} \end{cases} \tag{1.26}$$

Observación 2.23. Bajo estas definiciones no es difícil comprobar que

$$U(D(z)) = \frac{1}{4}\left(z + z^{*} + z^{**} + z^{**}\right). \tag{1.27}$$

Siguiendo la misma idea que en una dimensión, buscamos aprovechar la propiedad de localización en el vector $[z]_B$ y así eliminar las componentes que son cercanas a cero conservando la información que es relevante. Luego de esto queremos tener nuestro vector z nuevamente en la base original.

El siguiente esquema muestra estos procesos, donde se propone encontrar $\tilde{s_0}, \tilde{s_1}, \tilde{s_2}$, y $\tilde{s_3}$ de manera que permitan recuperar el vector z de manera exacta, $\downarrow 2$ representa el operador downsampling y $\uparrow 2$ representa el operador upsampling.

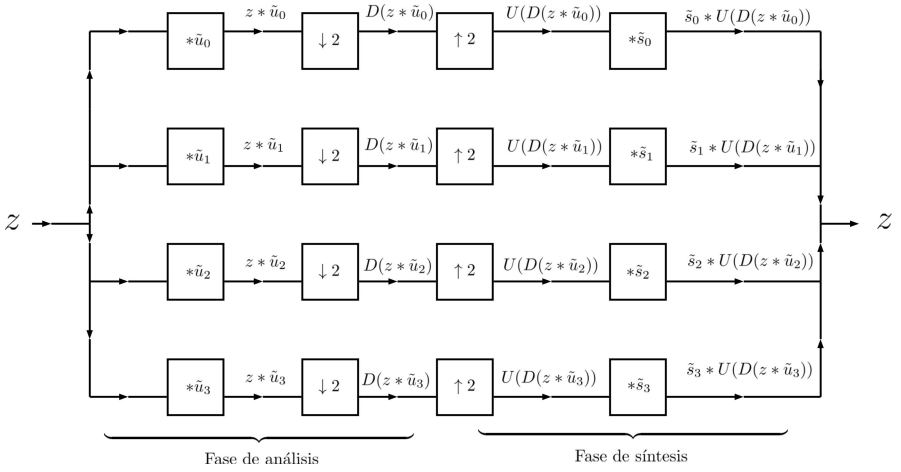


Figura 1. Proceso de transformación y reconstrucción de una imagen mediante ondículas.

Para este propósito se tiene el siguiente resultado.

Teorema 2.24. *Para las matrices $A(n_1, n_2)$ definidas en el teorema 2.19 tenemos reconstrucción perfecta mediante el esquema anterior, esto es*

$$\sum_{j=0}^{3} \tilde{s}_j * U(D(z * \tilde{u}_j)) = z, \tag{1.28}$$

si y solamente si

$$A(n_1, n_2) \cdot \begin{bmatrix} \hat{s}_0(n_1, n_2) \\ \hat{s}_1(n_1, n_2) \\ \hat{s}_2(n_1, n_2) \\ \hat{s}_3(n_1, n_2) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{1.29}$$

para cada n_1, n_2. Además, si $A(n_1, n_2)$ son unitarias tenemos $s_j = \tilde{u}_j$.

Demostración. Como notamos en la observación 2.23 tenemos que

$$U(D(z * \tilde{u}_j)) = \frac{1}{4}(z * \tilde{u}_j + (z * \tilde{u}_j)^{*} + (z * \tilde{u}_j)^{**} + (z * \tilde{u}_j)^{***})$$

utilizando la transformada de Fourier y las ecuaciones 1.18 a 1.21 esta ecuación es equivalente a

$$\begin{aligned} U(D(z * \tilde{u}_j))\hat{}(n_1, n_2) = \frac{1}{4} [&\hat{z}_i\,(n_1, n_2)\overline{\tilde{u}_j(n_1, n_2)} + \hat{z}_i(n_1 + M_1, n_2)\overline{\tilde{u}_j(n_1 + M_1, n_2)} \\ &+ \hat{z}_i(n_1, n_2 + M_2)\overline{\tilde{u}_j(n_1, n_2 + M_2)} \\ &+ \hat{z}_i(n_1 + M_1, n_2 + M_2)\overline{\tilde{u}_j(n_1 + M_1, n_2 + M_2)}\Big] \end{aligned}$$

para cada $n_1 \in \mathbb{Z}_{N_1}, n_2 \in \mathbb{Z}_{N_1}$

Notemos que por los resultados 2.10 y 2.13 tenemos la igualdad

$$\overline{\hat{s}_j} \cdot U(D(z * \tilde{u}_j))\hat{}(n_1, n_2) = (\tilde{s}_j * U(D(z * \tilde{u}_j)))\hat{}\,(n_1, n_2).$$

Por lo tanto, multiplicando la igualdad previa por $\overline{\hat{s}_j}(n_1, n_2)$, utilizando la igualdad anterior y sumando en j tenemos:

$$\begin{aligned} \sum_{j=0}^{3} (\tilde{s}_j * U(D(z * \tilde{u}_j)))\hat{}\,(n_1, n_2) = \frac{1}{4} \Bigg[&\sum_{j=0}^{3} \overline{\hat{s}_j}(n_1, n_2) \cdot \Big(\hat{z}_i(n_1, n_2)\overline{\tilde{u}_j(n_1, n_2)} \\ &+ \hat{z}_i(n_1 + M_1, n_2 + M_2)\overline{\tilde{u}_j(n_1 + M_1, n_2 + M_2)} \\ &+ \hat{z}_i(n_1, n_2 + M_2)\overline{\tilde{u}_j(n_1, n_2 + M_2)} + \hat{z}_i(n_1 + M_1, n_2)\overline{\tilde{u}_j(n_1 + M_1, n_2)} \Big) \Bigg] \\ = \frac{1}{4} \Bigg[&\hat{z}(n_1, n_2) \sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1, n_2)} \\ &+ \hat{z}(n_1 + M_1, n_2) \sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1 + M_1, n_2)} \\ &+ \hat{z}(n_1, n_2 + M_2) \sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1, n_2 + M_2)} \\ &+ \hat{z}(n_1, n_2 + M_2) \sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1 + M_1, n_2 + M_2)} \Bigg] \end{aligned}$$

donde esto último es igual a $\hat{z}(n_1, n_2)$ si y solamente si

$$\begin{aligned} &\sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1, n_2)} = 4 \\ &\sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1 + M_1, n_2)} = 0 \\ &\sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1, n_2 + M_2)} = 0 \\ &\sum_{j=0}^{3} \overline{\hat{s}_j(n_1, n_2)\hat{u_j}(n_1 + M_1, n_2 + M_2} = 0 \end{aligned}$$

Lo que matricialmente es

$$A(n_1,n_2)\cdot\begin{bmatrix}\hat{s}_0(n_1,n_2)\\ \hat{s}_1(n_1,n_2)\\ \hat{s}_2(n_1,n_2)\\ \hat{s}_3(n_1,n_2)\end{bmatrix}=\begin{bmatrix}2\\0\\0\\0\end{bmatrix},$$

que es lo que buscábamos. Ahora si $A(n_1,n_2)$ es unitaria, se tiene que $A^{-1}=\overline{A^T}$ entonces

$$\begin{bmatrix}\hat{s}_0(n_1,n_2)\\ \hat{s}_1(n_1,n_2)\\ \hat{s}_2(n_1,n_2)\\ \hat{s}_3(n_1,n_2)\end{bmatrix}=\overline{A^T}(n_1,n_2)\cdot\begin{bmatrix}2\\0\\0\\0\end{bmatrix}=\begin{bmatrix}\overline{\hat{u}_0}(n_1,n_2)\\ \overline{\hat{u}_1}(n_1,n_2)\\ \overline{\hat{u}_2}(n_1,n_2)\\ \overline{\hat{u}_3}(n_1,n_2)\end{bmatrix}$$

Aplicando la transformada inversa de Fourier en cada componente obtenemos

$$s_j(n_1,n_2)=\tilde{u}_j(n_1,n_2)$$

para cada $j=0,1,2,3$, concluyendo así la demostración. ■

2. Ejemplos

Considerando $u_{N_1},v_{N_1}\in\ell^2(\mathbb{Z}_{N_1}),N_1=2M_1\in\mathbb{Z}$ dados por

$$u_{N_1}=\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,\ldots,0\right),$$

$$v_{N_1}=\left(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},0,\ldots,0\right),$$

se puede comprobar que $\{R_{2k}u_{N_1}\}_{k=0}^{M-1}\cup\{R_{2k}v_{N_1}\}_{k=0}^{M-1}$ es una base ortonormal de $\ell^2(\mathbb{Z}_{N_1})$. Esta base es denominada base de *Haar*.

Utilizando el resultado de la proposición 2.20 y los vectores $u_{N_1},v_{N_1},u_{N_2},v_{N_2}$, con $N_2=2M_2\in\mathbb{Z}$, formamos una base para $\ell^2(\mathbb{Z}_{N_1}\times\mathbb{Z}_{N_2})$

$$\begin{aligned}u_0(n_1,n_2)&=v_{N_1}(n_1)\cdot v_{N_2}(n_2),\\ u_1(n_1,n_2)&=u_{N_1}(n_1)\cdot v_{N_2}(n_2),\\ u_2(n_1,n_2)&=v_{N_1}(n_1)\cdot u_{N_2}(n_2),\\ u_3(n_1,n_2)&=u_{N_1}(n_1)\cdot u_{N_2}(n_2).\end{aligned}$$

Está última base es ortonormal y por ello satisface todos los resultados vistos previamente.

La imagen de la Figura 2 nos acompañara a lo largo de nuestros experimentos, esta se puede encontrar en `http://links.uwaterloo.ca/Repository.html`.

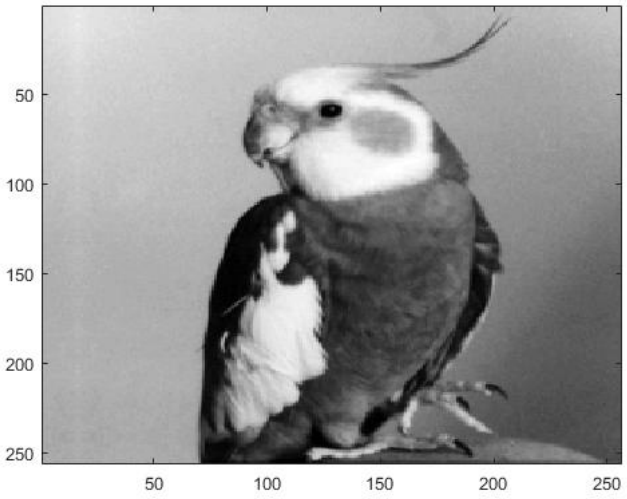
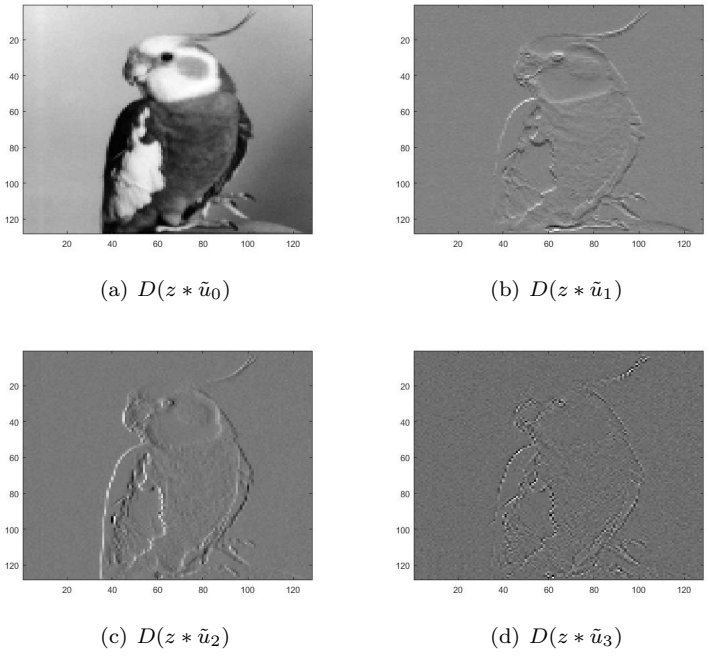


FIGURA 2. Imagen original.

Realizando el proceso de la Figura 1, hasta antes de comenzar nuestra reconstrucción, obtenemos las matrices $D(z * \tilde{u}_j)$ para $j = 0, 1, 2, 3$ de la Figura 3.



(a) $D(z * \tilde{u}_0)$ (b) $D(z * \tilde{u}_1)$

(c) $D(z * \tilde{u}_2)$ (d) $D(z * \tilde{u}_3)$

FIGURA 3. Imágenes Obtenidas.

En la figura 3 se puede notar que mayoritariamente la información se encuentra en nuestra imagen (a), es más, parece ser una replica más pequeña de nuestra imagen original. Aún así, en las imágenes (b), (c) y (d) se puede distinguir la silueta del pájaro, esto pues estas imágenes hacen referencia a las diferencias verticales, horizontales y diagonales de los pixeles de nuestra imagen, a su vez la imagen (a) es un promedio de los pixeles, esto explica la similitud a la imagen original. Estás 4 imágenes se denotan por LL, HL, LH y HH respectivamente, además suelen juntarse y representarse como una a cual imagen se le denomina la *transformada ondícula* de nuestra imagen original como se muestra en la Figura 4.

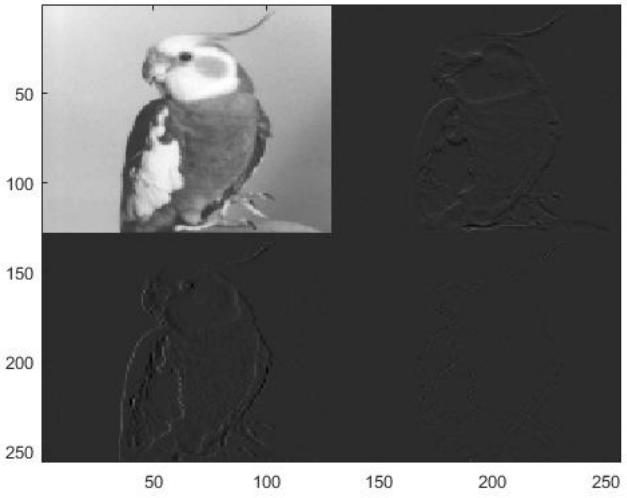


FIGURA 4. Transformada ondícula nivel 1.

La descomposición siempre sigue el esquema:

$$\begin{bmatrix} LL & HL \\ LH & HH \end{bmatrix}.$$

La Figura 4 muestra el primer nivel de la transformada ondícula, pues solo se ha realizado una vez el proceso. Para obtener el siguiente nivel se toma el bloque LL y se vuelve a aplicar la transformación hasta el nivel que se requiera. En la figura 5 se muestra el tercer nivel de la transformada ondícula para nuestra imagen.

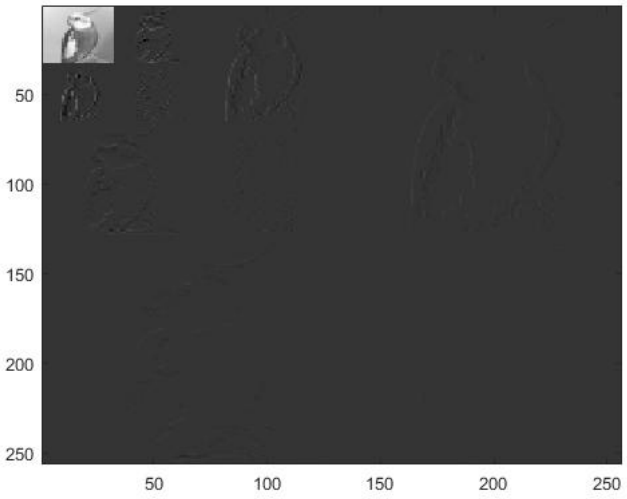


FIGURA 5. Transformada ondícula nivel 3.

De esta manera se tiene una representación equivalente a la original, pero que mediante ciertos algoritmos puede ser almacenada utilizando menos espacio. Debido a que la información principal está concentrada en cierta región de la imagen, es posible eliminar la información menos relevante y ahorrar espacio.

Capítulo 3

Algoritmos de Compresión de imágenes

1. Preliminares

Una de las principales aplicaciones de las ondículas, como ya se ha mencionado, es la compresión de imágenes. Existen dos tipos de compresión de imágenes, compresión con pérdidas y compresión sin pérdidas. La compresión sin pérdidas, como su nombre lo dice, recupera la imagen como era exactamente antes del proceso de compresión, desafortunadamente para ciertas imágenes, por ejemplo escenas naturales, es casi imposible obtener compresión libre de errores más allá de razón 2:1. Se puede obtener mayor razón de compresión si es que se permite algún error, esto es compresión con pérdidas.

Nos enfocaremos en la compresión con pérdidas y presentaremos dos algoritmos que trabajan esta compresión. Para ello, revisemos algunas definiciones que nos serán útiles más adelante. Los algoritmos y definiciones pueden ser revisados en [2].

En lo que sigue consideraremos nuestras imágenes como matrices en el espacio $\mathcal{M}_{N_1\times N_2}(\mathbb{R})$ de las matrices de orden N_1 por N_2 de coeficientes reales. El valor del pixel (i,j) será denominado por p_{ij}. Se asumirá que N_1, N_2 son potencias de dos dado que las dimensiones de las imágenes lo suelen ser.

Como queremos saber qué tan lejos o cerca está una imagen reconstruida de una original es necesario pensar en medidas para los errores, las cuales serán utilizadas en nuestros experimentos.

Definición 3.1. Dadas imagenes $I_1, I_2 \in \mathcal{M}_{N_1\times N_2}(\mathbb{R})$ se define el *error cuadrático medio*, *MSE* (*mean square error*), entre estas imágenes como

$$\mathrm{MSE}(I_1, I_2) = \frac{1}{N_1 \cdot N_2} \sum_{j\in\mathbb{Z}_{N_1}} \sum_{k\in\mathbb{Z}_{N_2}} \left(I_1(j,k) - I_2(j,k)\right)^2, \tag{1.1}$$

a partir del MSE definimos el PSNR dado por

$$\mathrm{PSNR}(I_1, I_2) = 10\log_{10}\left(\frac{(MAX_I)^2}{\mathrm{MSE}(I_1, I_2)}\right), \tag{1.2}$$

donde MAX_I es el máximo valor posible que puede tomar un pixel en la imagen. A medida que el PSNR aumenta, aumenta la calidad de la imagen reconstruida. En general si el PSNR es mayor o igual a 40 las imágenes comparadas son prácticamente idénticas para el ojo humano. Para más detalles revisar [2].

Algo muy importante antes de describir los algoritmos es definir el orden en el cual recorreremos los pixeles de nuestra imagen. La relevancia y utilidad quedará clara más adelante. El orden que utilizaremos recibe el nombre de orden Z el cual puede ser revisado en [17]. El orden se

sigue esquemáticamente en la Figura 1 mientras la Figura 2 muestra la enumeración en que se recorre una matriz de orden 8×8 mediante el orden Z.

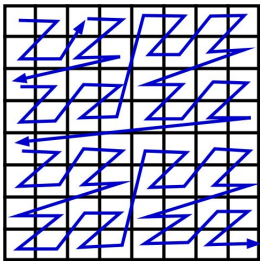


FIGURA 1. Orden Z.

1	2	5	6	17	18	21	22
3	4	7	8	19	20	23	24
9	10	13	14	25	26	29	30
11	12	15	16	27	28	31	32
33	34	37	38	49	50	53	54
35	36	39	40	51	52	55	56
41	42	45	46	57	58	61	62
43	44	47	48	59	60	63	64

FIGURA 2. Orden Z 8×8.

A continuación se muestra la matriz que nos acompañará como ejemplo a medida que explicamos los algoritmos.

58	-35	51	8	5	14	-10	5
-30	25	12	-14	3	1	5	-2
15	7	2	-9	5	-10	9	12
-10	-6	-11	5	7	-2	4	9
-2	11	-1	47	6	6	-2	3
0	3	-2	0	1	-4	3	1
1	-4	8	-2	4	5	3	12
5	11	3	3	-2	5	-4	1

(1.3)

2. Algoritmo EZW

El algoritmo EZW, debido a su nombre en ingles *embedded zerotree wavelet*, fue uno de los primeros algoritmos en mostrar el poderío de trabajar en base de ondículas para comprimir imágenes. Fue introducido por Shapiro en [3].

Este algoritmo consiste en separar las magnitudes de nuestros pixeles en dos listas, una para los valores significativos y otra para los insignificantes, esto en relación a un umbral T.

DEFINICIÓN 3.2. Dado $T \in \mathbb{R}$ un umbral y $p_{i,j}$ un pixel de nuestra imagen $I \in \mathcal{M}_{N_1 \times N_2}(\mathbb{R})$. Diremos que $p_{i,j}$ es *significativo* en relación a T si $|p_{i,j}| \geq T$. Si $|p_{i,j}| < T$ diremos entonces que es *insignificante*.

A continuación definimos una de las nociones y aportes más importantes del algoritmo EZW la cual es responsable de la compresión de información.

DEFINICIÓN 3.3. Dada una imagen $I \in \mathcal{M}_{N_1 \times N_2}(\mathbb{R})$ y dado un pixel $p_{i,j} \in I$ llamaremos *hijos* o *descendencia directa* al conjunto $\{p_{2i-1,2j-1}; p_{2i,2j-1}; p_{2i-1,2j}; p_{2i,2j}\}$ siempre que todos estos valores estén en el rango de nuestra imagen. Llamaremos *descendencia completa* o simplemente *descendencia* del pixel $p_{i,j}$ al conjunto de todos los pixeles que están en la cadena de hijos de $p_{i,j}$. Cuando hablamos del pixel $p_{i,j}$ en relación a sus hijos o descendencia lo llamaremos *padre*.

EJEMPLO 3.4. En nuestra matriz de enumeración de la figura 2 los hijos del pixel $(2,2)$, número 4 en orden Z, son los pixeles $(3,3);(3,4);(4,3);(4,4))$ los cuales tienen orden $13, 14, 15$ y 16 respectivamente. A su vez los hijos del pixel 13 son los pixeles $49, 50, 51$ y 52 de orden Z. Se sigue que la descendencia del pixel 4 en orden Z es $\{13, 14, 15, 16, 49, 50, 51, \ldots, 62, 63, 64\}$, como se muestra en la figura 3.

OBSERVACIÓN 3.5. Los hijos del pixel $p_{1,1}$ son $p_{1,2}, p_{2,1}, p_{2,2}$ por convención.

1	2	5	6	17	18	21	22
3	4	7	8	19	20	23	24
9	10	13	14	25	26	29	30
11	12	15	16	27	28	31	32
33	34	37	38	49	50	53	54
35	36	39	40	51	52	55	56
41	42	45	46	57	58	61	62
43	44	47	48	59	60	63	64

FIGURA 3. Ejemplo hijos y descendientes.

La importancia del orden de Morton radica en que siempre revisaremos los pixeles padres antes de sus hijos. Así, si un pixel es insignificante al igual que toda su descendencia, puede ser representada por el padre y no es necesario recordar que cada pixel es insignificante pues el padre habla por ellos. De acuerdo esto surge la siguiente definición.

DEFINICIÓN 3.6. Diremos que un pixel $p_{i,j}$ es una *raíz nula* si él y toda su descendencia son menores que cierto umbral T. Si $p_{i,j}$ es menor al umbral pero existe al menos un pixel en la descendencia de $p_{i,j}$ que es mayor al umbral entonces llamaremos al pixel padre *raíz aislada*.

EJEMPLO 3.7. Considerando un umbral $T = 32$ y nuestra matriz (1.3). El pixel $p_{2,2} = 25$ es un raíz nula pero el pixel $p_{2,1} = -30$ es una raíz aislada pues el pixel $p_{5,4} = 47$ está en su descendencia y es mayor al umbral.

En el algoritmo EZW se exportan listas binarias y listas de letras dependiendo de la etapa. Las listas de letras están compuestas por las letras T, Z, N y P. Su significado se comentará a continuación. A su vez, para alcanzar una mayor compresión las letras son codificadas por el

algoritmo de Huffman (revisar [10]), un ejemplo de codificación puede ser el siguiente código:

$$\begin{aligned} \mathrm{T} &= 0, \\ \mathrm{Z} &= 10, \\ \mathrm{N} &= 110, \\ \mathrm{P} &= 111. \end{aligned}$$

Estudios del algoritmo EZW y la codificación de Huffman pueden ser revisadas en [18]

Presentamos a continuación el algoritmo EZW.

Algoritmo EZW codificación

(1) **Iniciación:** *Incluir todos los coeficientes (pixeles) de nuestra imagen en la lista de valores insignificantes mientras que la lista de valores significativos inicia vacía. Comenzar con un umbral* $T_0 = 2^{\lfloor \log_2(C_{\text{máx}}) \rfloor}$ *donde* C_{max} *es el máximo valor absoluto de los coeficientes de la matriz. Tomar un número de iteraciones,* N*, menor a* $\lfloor \log_2(C_{\text{máx}}) \rfloor + 1$ *. Iniciar con* $n = 0$*.*

(2) **Etapa de pixeles significativos:** *Recorrer los valores* $p_{i,j}$ *en la lista de valores insignificantes mediante el orden Z y comparar con el umbral* T_n *:*

a) *Si* $|p_{i,j}|$ *es mayor que* T_n *y* $p_{i,j}$ *es positivo exportar la letra P y adicionar* $|p_{i,j}|$ *a la lista de valores significativos. Cambiar* $p_{i,j}$ *en la lista de insignificancia por* 0*.*
b) *Si* $|p_{i,j}|$ *es mayor que* T_n *y* $p_{i,j}$ *es negativo exportar la letra N y adicionar* $|p_{i,j}|$ *a la lista de valores significativos. Cambiar* $p_{i,j}$ *en la lista de insignificancia por* 0*.*
c) *Si* $|p_{i,j}|$ *es menor que* T_n *y* $p_{i,j}$ *es una raíz aislada exportar la letra Z.*
d) *Si* $|p_{i,j}|$ *es menor que* T_n *y* $p_{i,j}$ *es una raíz nula exportar la letra T.*

Codificar la cadena de letras mediante el algoritmo de Huffman.

(3) **Etapa de refinamiento:** *Recorrer los valores* C *en la lista de valores significativos. Para cada* C *calculamos* d_C *como:*

- *Comenzar con* $d_{C_0} = C$*. Iteramos para* $j = 0$ *hasta* $j = n$*. Si,* $d_{C_j} - T_j \geq 0$ *entonces* $d_{C_{j+1}} = d_{C_j} - T_j$*, si no,* $d_{C_{j+1}} = d_{C_j}$*. Finalmente* $d_C = d_{C_n}$*.*

Si $d_C \geq T_n/2$ *exportar 1, si no, exportar 0.*

(4) **Iterar:** *Si* $n = N$ *el algoritmo acaba. Si no, continuar con* $n \to n + 1$*,* $T_{n+1} = T_n/2$ *e ir al paso (2).* □

Ahora que ya sabemos llevar nuestra imagen a un código binario mediante el algoritmo EZW es importante poder decodificar esta información para así volver a nuestra imagen original. Se presenta a continuación el algoritmo de decodificación.

Algoritmo EZW decodificación

(1) **Iniciación:** *Iniciar todos los valores de nuestra matriz de reconstrucción como ceros. Comenzar con el mismo umbral* T_0 *que se comenzó la codificación. Tomar un número*

de iteraciones, N, menor a o igual al número de iteraciones utilizadas en la codificación. Iniciar $n = 0$.

(2) **Etapa de pixeles significativos:** *Lo primero es decodificar la lista de letras que fue llevada a binario. Luego, recorrer los valores L_j en la lista de letras de la iteración n donde j va desde 1 hasta el número de letras en la lista:*

   (a) *Si L_j =P adicionar el valor T_n a nuestra matriz de reconstrucción en la posición correspondiente según el orden Z.*
   (b) *Si L_j =N adicionar el valor $-T_n$ a nuestra matriz de reconstrucción en la posición posición correspondiente según el orden Z.*
   (c) *Si L_j =T o L_j =Z, hacer nada. Es importante tener en cuenta que la letra T nos dice que los descendiente son insignificantes por lo que estas entradas deben ser consideradas como llenas y ser saltadas en los casos (a) y (b).*

(3) **Etapa de refinamiento:** *El número de valores que fueron actualizados en la matriz de reconstrucción en la etapa anterior, debidos a las letras P y N, coincide con el número de ceros y unos en nuestra cadena de números de la iteración n. Por esto, se recorren estos valores actualizados y se les asigna el correspondiste valor binario en nuestra lista, si este valor es cero, hacer nada. Si este valor es uno y el valor de la matriz de reconstrucción es positivo se le debe adicionar $T_n/2$, si el valor es negativo se le debe restar esta cantidad.*

(4) **Iterar:** *Si $n = N$ el algoritmo acaba. Si no, continuar con $n \to n + 1$, $T_{n+1} = T_n/2$ e ir al paso (2).* □

Como es de esperar, mayores niveles de compresión se alcanzan realizando menos iteraciones de nuestro algoritmo pues guardamos menos cadenas de letras y de números. Esto también produce que se tenga menos información para reconstruir la matriz. Ejemplos de codificación y decodificación para el algoritmo EZW pueden ser revisados en el Anexo.

Presentamos a continuación otro algoritmo de compresión, el cual se basa en ideas del algoritmo EZW.

3. Algoritmo SPIHT

El algoritmo SPIHT, siglas debidas a su nombre en inglés *set partitioning in hierarchical trees,* toma muchas propiedades del algoritmo EZW y busca implementar otras de manera de obtener un algoritmo más rápido. El principal propósito es incorporar una mejor regla de clasificación de los pixeles, para más detalles se puede revisar [4].

Uno de los problemas detectados en el algoritmo EZW es que si un coeficiente era definido significativo el algoritmo debía continuar por sus hijos para determinar si ellos eran significativos o no. Por esto, uno de los objetivos es crear nuevas particiones en las cuales los subconjuntos de coeficientes insignificantes contengan una gran cantidad de elementos mientras que los subconjuntos de coeficientes significativos contengan solo un elemento. Para esto utilizaremos la siguiente definición.

Definición 3.8. Dado $\mathcal{T}$ un conjunto de coordenadas de nuestra imagen $I \in \mathcal{M}_{N_1 \times N_2}(\mathbb{R})$ definimos la función de *significatividad* como

$$S_T(\mathcal{T}) = \begin{cases} 1, & \text{si } \max_{(i,j\in\mathcal{T})} \{|c_{i,j}|\} \leq T \\ 0, & \text{en otro caso,} \end{cases} \tag{3.1}$$

para un umbral $T \in \mathbb{R}$. Si $\mathcal{T} = \{(i,j)\}$ simplemente utilizaremos la notación $S_T(i,j)$.

Definimos a continuación los siguientes conjuntos de coordenadas.

$\mathcal{O}(i,j)$: Conjunto de coordenadas de todos los hijos del pixel (i,j),

$\mathcal{D}(i,j)$: Conjunto de coordenadas de todos los descendientes del pixel (i,j),

$\mathcal{H}$: Conjunto de coordenadas de todas las raíces del árbol de orientación espacial, esto es $\{(1,1),(1,2),(2,1),(2,2)\}$,

$\mathcal{L}(i,j) := \mathcal{D}(i,j) \setminus \mathcal{O}(i,j)$.

A diferencia del algoritmo EZW, este algoritmo trabaja con listas de posiciones, cuales listas son para trabajar con durante algoritmo, no para guardarlas como información. A continuación se definen las listas de nuestro algoritmo.

LIS: Lista de conjuntos de pixeles insignificantes (*list of insignificant sets*),

LIP: Lista de pixeles insignificantes (*list of insignificant pixels*),

LSP: Lista de pixeles significativos (*list of significant pixels*).

Las listas LIP y LIS contienen los pixeles insignificantes y significativos respectivamente, representados por su posición. La lista LIS contiene conjuntos de pixeles insignificantes, representados por una posición y una letra A o una letra B, la utilidad de estas letras quedará clara cuando se presente el algoritmo.

Guardaremos dos listas para la decodificación. Una lista que contiene ceros, unos y las letras P y N, esta lista se completará en la fase de clasificación, denominaremos esta lista como *lista de letras*. La segunda lista solo contiene ceros y unos y se completará en la fase de refinamiento, denominaremos esta lista como *lista de números*. La primera lista nombrada se codificara mediante el algoritmo de Huffman.

Presentamos a continuación el algoritmo SPIHT.

Algoritmo SPIHT codificación

(1) **Iniciación:** *Comenzar con un umbral* $T_0 = 2^{\lfloor \log_2(C_{\max}) \rfloor}$ *donde* C_{max} *es el máximo valor absoluto de los coeficientes de la matriz. Tomar un número de iteraciones,* N*, menor a* $\lfloor \log_2(C_{\max}) \rfloor + 1$ *. Iniciar con* $n = 0$*. Fijar inicialmente la lista LSP vacía, rellenar lista LIP con todas las coordenadas del conjunto* $\mathcal{H}$*, incluir en la lista LIS las coordenadas de* $\mathcal{H}$ *que tienen hijos no incluidos (esto excluye al (1,1)) y se declaran tipo A.*

(2) **Etapa de clasificación:** *Recorrer los valores $p_{i,j}$ en la lista de valores insignificantes mediante el orden Z y comparar con el umbral T_n :*

(2.1) *Por cada entrada $(i,j) \in LIP$*
(2.1.1) *Exportar el valor de $S_{T_n}(i,j)$ si es igual a cero.*
(2.1.2) *Si $S_{T_n}(i,j) = 1$ mover (i,j) a la lista LSP y exportar el signo de $C_{i,j}$ (si es positivo la letra P, si es negativo la letra N).*

(2.2) *Por cada entrada $(i,j) \in LIS$*
(2.2.1) *Si la entrada es tipo A*
- *exportar $S_{T_n}(\mathcal{D}(i,j))$,*
- *si $S_{T_n}(\mathcal{D}(i,j)) = 1$ entonces:*
  * *Por cada $(k,l) \in \mathcal{O}(i,j)$:*
    · *importar $S_{T_n}(k,l)$,*
    · *si $S_{T_n}(k,l) = 1$ entonces agregar (k,l) a la lista LSP y exportar el signo de $p_{k,l}$,*
    · *si $S_{T_n}(k,l) = 0$ entonces agregar (k,l) al final de la lista LIP.*
  * *Si $\mathcal{L}(i,j) \neq \emptyset$ entonces mover (i,j) al final de la lista LIS como entrada tipo B, si no, remover la entrada (i,j) de la lista LIS.*

(2.2.2) *Si la entrada es tipo B entonces*
- *Exportar $S_{T_n}(\mathcal{L}(i,j))$,*
- *Si $S_{T_n}(\mathcal{L}(i,j)) = 1$ entonces*
  * *agregar cada $(k,l) \in \mathcal{O}(i,j)$ al final de la lista LIS como entrada tipo A,*
  * *Remover (i,j) de la lista LIS.*

Codificar la cadena de letras mediante el algoritmo de Huffman.

(3) **Etapa de refinamiento:** *Recorrer las posiciones $(i,j) \in LSP$ y asignar $C = p_{i,j}$. Para cada (i,j) calculamos d_C como:*

- Comenzar con $d_{C_0} = C$. Iteramos para $j = 0$ hasta $j = n$. Si, $d_{C_j} - T_j \geq 0$ entonces $d_{C_{j+1}} = d_{C_j} - T_j$, si no, $d_{C_{j+1}} = d_{C_j}$. Finalmente $d_C = d_{C_n}$.

Si $d_C \geq T_n/2$ exportar 1, si no, exportar 0.

(4) **Iterar:** *Si $n = N$ el algoritmo acaba. Si no, continuar con $n \to n+1$, $T_{n+1} = T_n/2$ e ir al paso (2).* □

En el Anexos es posible encontrar un ejemplo de la codificación SPHIT para nuestro matriz ejemplo.

Ahora necesitamos decodificar esta información para así volver a nuestra imagen original. La bibliografía suele decir que para obtener el algoritmo de decodificación basta cambiar los "exportar" por "importar" en la etapa de clasificación. Para finalizar este capitulo se presenta un algoritmo de decodificación para el algoritmo SPIHT que sigue básicamente esa idea.

Algoritmo SPIHT codificación

(1) **Iniciación:** *Iniciar todos los valores de nuestra matriz de reconstrucción como ceros. Comenzar con el mismo umbral T_0 que se comenzó la codificación. Tomar un número de*

iteraciones, N, menor a o igual al número de iteraciones utilizadas en la codificación. Iniciar las listas LIP,LIS y LSP tal cual como se comenzó la codificación. $n = 0$.

(2) **Etapa de clasificación:** *Recorrer los valores ℓ en la lista de letras (ya decodificada mediante Huffman) y al mismo tiempo las posiciones de nuestra matriz mediante el orden Z :*

(2.1) *Por cada entrada $(i, j) \in LIP$*

(2.1.1) *Importar el valor de ℓ.*

(2.1.2) *Si $\ell = P$ o $\ell = N$ mover (i, j) a la lista LSP. Si $\ell = P$ sumar T_n a la entrada (i, j) de nuestra matriz de reconstrucción, si $\ell = N$ restar T_n a la entrada (i, j) de nuestra matriz de reconstrucción.*

(2.2) *Por cada entrada $(i, j) \in LIS$*

(2.2.1) *Si la entrada es tipo A*

- *importar ℓ,*
- *si $\ell = 1$ entonces:*
  - *Por cada $(k, l) \in \mathcal{O}(i, j)$:*
    - *importar ℓ,*
    - *si $\ell = P$ o $\ell = N$ entonces agregar (k, l) a la lista LSP. Si $\ell = P$ sumar T_n a la entrada (k, l) de nuestra matriz de reconstrucción, si $\ell = N$ restar T_n a la entrada (k, l) de nuestra matriz de reconstrucción,*
    - *si $\ell = 0$ entonces agregar (k, l) al final de la lista LIP.*
  - *Si $\mathcal{L}(i, j) \neq \emptyset$ entonces mover (i, j) al final de la lista LIS como entrada tipo B, si no, remover la entrada (i, j) de la lista LIS.*

(2.2.2) *Si la entrada es tipo B entonces*

- *Importar ℓ,*
- *Si $\ell = 1$ entonces*
  - *agregar cada $(k, l) \in \mathcal{O}(i, j)$ al final de la lista LIS como entrada tipo A,*
  - *Remover (i, j) de la lista LIS.*

(3) **Etapa de refinamiento:** *Recorrer las posiciones $(i, j) \in LSP$ las cuales fueron actualizadas en nuestra matriz de reconstrucción. el número de valores que fueron actualizados en la matriz de reconstrucción en la etapa anterior, debidos a las letras P y N, coincide con el número de ceros y unos en nuestra cadena de números de la iteración n. Por esto, se recorren estos valores actualizados y se les asigna el correspondiste valor binario en nuestra lista, si este valor es cero, hacer nada. Si este valor es uno y el valor de la matriz de reconstrucción es positivo se le debe adicionar $T_n/2$, si el valor es negativo se le debe restar esta cantidad.*

(4) **Iterar:** Si $n = N$ el algoritmo acaba. Si no, continuar con $n \to n + 1$, $T_{n+1} = T_n/2$ e ir al paso (2). □

Capítulo 4

Algoritmos de Corrección de imágenes

1. Preliminares

Muchas veces al capturar una imagen mediante un dispositivo, la imagen presenta ciertos errores o se ve borrosa. Es por esto que nos gustaría poder trabajar esta imagen de modo que represente más fehacientemente la realidad. Se han realizado muchos estudios de este problema mediante optimización y hay un gran listado de resultados y algoritmos propuestos para esta problemática. Estos algoritmos suelen trabajar la imagen en base ondícula para hacer referencia a comprimir la imagen, veremos más adelante como se encargan de eso. A continuación presentamos algunos resultados y definiciones propias del análisis convexo y optimización, para posteriormente enunciar algunos modelos y algoritmos de corrección que se analizarán en los experimentos.

Definición 4.1. Dada $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. Diremos que esta función es *convexa* siempre que

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{1.1}$$

para cada $x, y \in \text{dom}(f)$ y $\lambda \in (0,1)$. Si la desigualdad 1.1 es estricta se dice que f es *estrictamente convexa*.

Definición 4.2. Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ y $x_n \to x \in \mathbb{R}^n$. Diremos que f es *inferior semicontinua* si

$$f(x) \leq \liminf_{n\to\infty} f(x_n).$$

Entregaremos a continuación algunas propiedades de las funciones convexas que son continuas o diferenciables,

Proposición 4.3. *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función convexa. Entonces f es continua en* $\text{int}(\text{dom}(f))$.

Proposición 4.4. *Sea $A \subset \mathbb{R}^n$ un conjunto abierto y convexo. Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función Gâteaux diferenciable. Entonces son equivalentes:*

(i) *f es convexa.*
(ii) *$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle$ para todo $x, y \in A$.*
(iii) *$\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq 0$ para todo $x, y \in A$.*

La proposición 4.4 nos dice que para $f : \mathbb{R}^n \to \mathbb{R}$ convexa y Gâteaux diferenciable en un punto $x \in \mathbb{R}^n$ se tiene

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle$$

para cada $y \in \mathbb{R}^n$. Esto dice que el hiperplano

$$v = \{(y, z) \in \mathbb{R}^n \times \mathbb{R} : f(x) + \langle f(x), y - x\rangle = z\}$$

se encuentra contenido en el conjunto epi(f) y lo toca en el punto $(x, f(x))$.

Se busca generalizar esta idea para funciones no diferenciables.

DEFINICIÓN 4.5. Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función propia y convexa. Definimos el *subdiferencial* de f en un punto $x \in \mathbb{R}^n$ como el conjunto $\partial f(x)$ dado por.

$$\partial f(x) = \{x^* \in \mathbb{R}^n \ : \ f(y) \geq f(x) + \langle x^*, y - x\rangle\} \tag{1.2}$$

Si $\partial f(x) \neq \emptyset$ diremos que f es *subdiferenciable* en x.

la siguiente proposición nos dice que efectivamente el subdiferencial es una extensión de la noción de derivada.

PROPOSICIÓN 4.6. *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ convexa. Si f es Gâteaux diferenciable en x, entonces $\partial f(x) = \{\nabla f(x)\}$.*

Veamos que el conjunto $\partial f(x)$ tiene muy buenas propiedades.

PROPOSICIÓN 4.7. *Para cada $x \in \mathbb{R}^n$ el conjunto $\partial f(x)$ es cerrado y convexo.*

Presentamos a continuación un resultado que sigue extendiendo las propiedades de la derivada al subdiferencial.

TEOREMA 4.8. Regla de Fermat *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función propia y convexa. Entonces $\hat{x}$ es un minimzador global de f si, y solo si, $0 \in \partial f(\hat{x})$.*

PROPOSICIÓN 4.9. *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una convexa. Si f es continua en x entonces $\partial f(x)$ es no vacío y acotado.*

Revisemos algunas condiciones de optimalidad para problemas con restricciones. Consideremos $C \subset \mathbb{R}^n$ un conjunto cerrado y convexo. La siguiente proposición caracteriza la optimalidad del problema

$$\text{mín}\{f(x) \ : \ x \in C\}. \tag{1.3}$$

PROPOSICIÓN 4.10. *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función propia, semicontinua inferior y convexa, sea $C \subset \mathbb{R}^n$ cerrado y convexo. Supongamos que f es continua en algún punto del conjunto C o existe algún punto en el interior de C donde f es acotada. Entonces $\hat{x}$ minimiza f en C si, y solamente si, existe $p \in \partial f(\hat{x})$ tal que $-p \in \partial \delta_C(\hat{x})$. Con*

$$\delta_C(x) = \begin{cases} 0, & \text{si } x \in C, \\ +\infty, & \text{en otro caso.} \end{cases}$$

Consideremos ahora el conjunto C dado por

$$C = \{x \in \mathbb{R}^n \ : \ Ax = b\} \tag{1.4}$$

para $A \in \mathcal{L}(\mathbb{R}^m, \mathbb{R}^n)$ y $b \in \mathbb{R}^n$.

TEOREMA 4.11. *Sea $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ una función propia, semicontinua inferior y convexa, sea $C \subset \mathbb{R}^n$ como en (1.4). Supongamos que f es continua en algún punto del conjunto C. Entonces $\hat{x}$ minimiza f en C si, y solamente si, $A\hat{x} = b$ y existe $\hat{y} \in \mathbb{R}^m$ tal que $-A^T\hat{y} \in \partial f(\hat{x})$*

2. Algoritmo GPSR

Damos a continuación las nociones básicas del algoritmo propuesto en [7] en el cual se presenta un problema de optimización cuadrático para corregir imágenes.

Comenzamos con el siguiente problema de optimización sin restricciones

$$\min_x \frac{1}{2}\|y - Ax\|_2^2 + r\|x\|_1 \tag{2.1}$$

donde $x \in \mathbb{R}^n$, $y \in \mathbb{R}^k$ y A es una matriz de orden $k \times n$, r es un parámetro no negativo, $\|v\|_2 = \left(\sum_j |v_j|^2\right)^{1/2}$ la norma euclidiana y $\|v\|_1 = \sum_j |v_j|$ la norma de ℓ_1.

Este modelo propone recuperar el vector x a partir de una observación y que se obtuvo mediante el proceso A el cual produce ruido y borrosidad. Por esto el término $\frac{1}{2}\|y - Ax\|_2^2$ en nuestra función objetivo, de modo de establecer similitudes entre y y Ax. La presencia del termino $r\|x\|_1$ busca incitar que las pequeñas componentes de x sean exactamente cero. En función de esto ultimo se puede considerar $x = Wh$ donde h es nuestra imagen original y W es la transformada ondícula, y así $A = RW^{-1}$ donde R es el operador de ruido, esto para aprovechar la propiedad de concentrar la información que ya hemos revisado en los capítulos anteriores.

Para llevar el problema 2.1 a una forma cuadrática, se separa la variable x en dos variables no negativas u, v.

$$x = u - v, \quad u \geq 0,\ v \geq 0.$$

Este cambio es factible por ejemplo para $u_i = (x_i)_+$, $v_i = (-x_i)_+$ donde $(a)_+ = \max\{0, a\}$. De acuerdo a esto tenemos que $\|x\|_1 = 1_n^T u + 1_n^t v$. Así, el problema original puede ser llevado a un problema cuadrático con restricciones:

$$\begin{aligned} \min_{u,v} \quad & \frac{1}{2}\|y - A(u - v)\|_2^2 + r1_n^T u + r1_n^T v \\ \text{sujeto a} \quad & u \geq 0 \\ & v \geq 0. \end{aligned}$$

Notemos que trasladamos u y v por un vector s, $u \leftarrow u + s$ y $v \leftarrow v + s$ con $s \geq 0$ el valor de la norma ℓ_2 en la función objetivo no cambia sin embargo este cambio aumenta el valor de la función objetivo en $2r1_n^T s \geq 0$ por lo tanto las soluciones del problema deben ser de la forma $u_i = 0$ o $v_i = 0$ para $i = 1, \ldots, n$ esto dice que la solución debe ser de la forma mencionada arriba $u_i = (x_i)_+$, $v_i = (-x_i)_+$.

Considerando $z = \begin{bmatrix} u \\ v \end{bmatrix}$, $b = A^T y$, $c = r1_{2n} + \begin{bmatrix} -b \\ b \end{bmatrix}$ y

$$B = \begin{bmatrix} A^T A & -A^t A \\ -A^T A & A^T A \end{bmatrix}$$

el problema cuadrático puede ser llevado a una forma más estándar:

$$\begin{aligned} \min_z \quad & c^T z + \frac{1}{2} z^T B z \\ \text{sujeto a} \quad & z \geq 0. \end{aligned}$$

Se propone resolver este problema con algoritmos tipo gradiente proyectado con reglas Armijo (revisar [11], página 266). Las iteraciones del algoritmo llevaran $z^{(k)}$ en $z^{(k+1)}$, consideraremos

la función objetivo $F : \mathbb{R}^{2n} \to \mathbb{R}$ dada por

$$F(z) = c^T z + \frac{1}{2} z^T B z. \tag{2.2}$$

Para cada k eligiéremos un $\alpha^{(k)} > 0$ y definimos la cantidad

$$w^{(k)} = \left(z^{(k)} - \alpha^{(k)} \nabla F(z^{(k)}) \right)_+ . \tag{2.3}$$

Elegimos un escalar $\lambda^{(k)} \in [0, 1]$ y establecemos

$$z^{(k+1)} = z^{(k)} + \lambda(k) \left(w^{(k)} - z^{(k)} \right). \tag{2.4}$$

$\alpha^{(k)}$ será elegido mediante la regla de Armijo. Para elegir $\lambda^{(k)}$ definimos el vector $g^{(k)}$ dado por

$$g_i^{(k)} = \begin{cases} \left(\nabla F(z^{(k)}) \right)_i, & \text{si } z_i^{(k)} < 0 \text{ o } \left(\nabla F(z^{(k)}) \right)_i < 0, \\ 0, & \text{en otro caso.} \end{cases}$$

Entonces se define

$$\alpha_0 = \arg\min_{\alpha} F \left(z^{(k)} - \alpha g^{(k)} \right)$$

cual cantidad puede ser calcula explicitamente como

$$\alpha_0 = \frac{(g^{(k)})^T g^{(k)}}{(g^{(k)})^T B g^{(k)}}. \tag{2.5}$$

Para evitar que los valores α_0 sean o muy grandes o muy pequeños consideraremos un intervalo $[\alpha_{\text{mín}}, \alpha_{\text{máx}}]$ con $0 < \alpha_{\text{mín}} < \alpha_{\text{máx}}$. Definimos el operador $\text{mid}(a, b, c)$ para $a, b, c \in \mathbb{R}$ el cual entrega el valor que esta acotado por arriba y abajo por los otros dos (el valor de en medio). Presentamos a continuación el algoritmo propuesto en [7] denominado algoritmo GPSR.

Algoritmo GPSR

(1) Iniciación: *Comenzar con $z^{(0)}$ arbitrario, elegir parametros $\beta \in (0, 1)$ y $\mu \in (0, 1/2)$. Establecer k=0.*

(2) *Calcular α_0 mediante la relación 2.5 y actulizarlo mediante*

$$\alpha_0 \leftarrow \text{mid}(\alpha_{min}, \alpha_0, \alpha_{\text{máx}}).$$

(3) *Elegir $\alpha^{(k)}$ el primer numero de la sucesión $\alpha_0, \beta\alpha_0, \beta^2\alpha_0, \ldots$ tal que*

$$F\left(\left(z^{(k)} - \alpha^{(k)} \nabla F\left(z^{(k)} \right) \right)_+ \right) \leq F\left(z^{(k)} \right) - \mu \nabla F\left(z^{(k)} \right)^T \left(z^{(k)} - \left(z^{(k)} - \alpha^{(k)} \nabla F\left(z^{(k)} \right) \right)_+ \right),$$

y actualizar

$$z^{(k+1)} = \left(z^{(k)} - \alpha^{(k)} \nabla F(z^{(k)}) \right)_+ .$$

(4) *Verificar si $z^{(k+1)}$ satisface el criterio de convergencia preestablecido, si es así terminar. En otro caso actualizar $k \leftarrow k + 1$ e ir al paso (2).*

□

Un criterio de convergencia utilizado es

$$\| z - (z - \overline{\alpha} \nabla F(z))_+ \| \leq \text{tolP}$$

para $\overline{\alpha}$ y tolP parámetros arbitrarios. Más criterios son descritos en la publicación original.

3. Algoritmo FISTA

Describimos a continuación lo que se propone en [8] donde se busca dar una solución más rápida y eficiente a problemas del tipo

$$\text{mín}\{F(x) \equiv f(x) + g(x) \ : \ x \in \mathbb{R}^n\} \tag{3.1}$$

donde $g : \mathbb{R}^n \to \mathbb{R}$ es una función continua y convexa pero no necesariamente diferenciable, $f : \mathbb{R}^n \to \mathbb{R}$ es una función convexa y diferenciable con su gradiente Lipschitz continuo de constante $L(f)$.

Entra en este marco funciones como $f(x) = \|Ax - b\|_2^2$ y $g(x) = \|x\|_1$ las cuales, como vimos anteriormente, son de interés para nuestros propósitos.

Para $L > 0$ se considera la aproximación cuadrática de $F(x) = f(x) + g(x)$ en un punto y dada por

$$Q_L(x, y) := f(y) + \langle x - y, \nabla f(y)\rangle + \frac{L}{2}\|x - y\|^2 + g(x), \tag{3.2}$$

la cual admite un único minimizador dado por

$$p_L(y) := \text{argmin}\{Q_L(x, y) \ : \ x \in \mathbb{R}^n\}. \tag{3.3}$$

Ignorando los términos en y que son constantes esta cantidad puede ser llevada a

$$p_L(y) = \text{argmin}_x \left\{ g(x) + \frac{L}{2} \left\| x - \left(y - \frac{1}{L}\nabla f(y) \right) \right\|^2 \right\}.$$

Bajo estas definiciones se tiene el algoritmo ISTA con paso constante.

Algoritmo ISTA con paso constante

(1) Iniciación: *Comenzar con $x_0 \in \mathbb{R}^n$ arbitrario, definir $L := L(f)$ la constate de Lipschitz de ∇f. Iniciar $k = 0$.*

(2) *Actualizar x_{k-1} a x_k mediante*

$$x_k = p_L(x_{k-1}).$$

(3) Si se cumple algún criterio de convergencia terminar, si no ir al paso (2) con $k \leftarrow k+1$.

□

En [8] proponen el algoritmo FISTA, una versión mejorada del algoritmo ISTA en cuanto a la velocidad de convergencia.

Algoritmo FISTA con paso constante

(1) Iniciación: *Comenzar con $y_1 = x_0 \in \mathbb{R}^n$ arbitrario, definir $L := L(f)$ la constate de Lipschitz de ∇f. Iniciar $k = 0$ y $t_1 = 0$.*

(2) *Actualizar x_k, y_{k+1} y t_{k+1} mediante*

$$x_k = p_L(y_k),$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2},$$

$$y_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}} \right)(x_k - x_{k-1}).$$

(3) Si se cumple algún criterio de convergencia terminar, si no ir al paso (2) con $k \leftarrow k+1$.

□

La gran diferencia entre FISTA e ISTA es que p_L no es evaluado en el paso anterior x_{k-1}, si no en y_k una combinación lineal de x_{k-1} y x_{k-2} la cual proporciona mejores tiempo de convergencia. Se obtienen el sigueinte resultado de covergencia, los detalles y la demostración de este hecho puede ser revisado en [8].

TEOREMA 4.12. *Sean* $\{x_k\}, \{y_k\}$ *sucesiones generadas por FISTA. Entonces para cualquier* $k \geq 1$ *se tiene*

$$F(x_k) - F(x^*) \leq \frac{2L(f)\|x_0 - x^*\|^2}{(k+1)^2} \quad \forall x^* \textit{minimizador de } F. \tag{3.4}$$

Retomando nuestro caso de interés, podemos tomar $f(x) = \|Ax - b\|_2^2$ y $g(x) = r\|x\|_1$ con $r > 0$. Debido a que en este caso g es separable permite calcular $p_L(x_k)$ para el algoritmo ISTA mediante:

$$p_L(x_k) = \mathcal{T}_{rt}\left(x_{k-1} - 2tA^T(Ax_{k-1} - b)\right). \tag{3.5}$$

con $t = \frac{1}{L(f)}$.

Mientras que para el algoritmo FISTA calculamos $p_L(x_k)$ mediante:

$$p_L(x_k) = \mathcal{T}_{rt_k}\left(y_k - 2t_kA^T(Ay_k - b)\right). \tag{3.6}$$

Donde $\mathcal{T}_\alpha(x)$ es el vector dado por:

$$\mathcal{T}_\alpha(x)_i = \text{signo}(x_i)(|x_i| - \alpha)_+. \tag{3.7}$$

Al igual que en el algoritmo anterior es posible considerar $x = Wu$ y $A = RW^{-1}$ y así trabajar el termino $r\|x\|_1$ en base ondícula para que la norma uno lleve términos a cero.

4. Algoritmo AFBS

Presentamos otra formulación que puede ser utilizada para reconstruir una imagen, pero el problema a cual viene a dar solución esta formulación es mucho más general. La demostración de los resultados y más detalles pueden ser revisados en [16]. Las siglas AFBS provienen de *Alternating Forward-Backward Splitting* nombre del algoritmo propuesto por los autores.

Buscamos soluciones numéricas para el problema:

$$\text{Encontrar } (x^*, y^*) \in \text{argmin}\{f(x) + g(y) \ : \ (x, y) \in C\}, \tag{4.1}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$ es una función convexa y diferenciable, $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ es una función propia, inferior semicontinua y convexa. El conjunto $C \subset \mathbb{R}^n \times \mathbb{R}^m$ está dado por:

$$C = \{(x, y) \in X \times Y \ : \ Ax + By = c\}, \tag{4.2}$$

para operadores lineales continuos $A : \mathbb{R}^n \to \mathbb{R}^d$ y $B : \mathbb{R}^m \to \mathbb{R}^d$, $c \in \mathbb{R}^d$. El método propuesto es un algoritmo de tipo gradiente-proximal con dos pasos basados en alternar la minimización de la función

$$\mathcal{L}_\lambda(x, y) = f(x) + g(y) + \frac{\gamma}{2\lambda}\|Ax + By - c\|^2 \tag{4.3}$$

con respecto a las variables x e y en cada iteración.

Algoritmo AFBS

(1) Iniciación: *Tomar* $(\lambda_n)_{n\in\mathbb{N}}, \gamma$ *adecuados, elegir* (x_0, y_0) *convenientes,*

(2) *Actualizar* x_n, y_n *mediante*

$$x_{n+1} = x_n - \lambda_n \nabla f(x_n) - \gamma A^*(Ax_n + By_n - c),$$

$$y_{n+1} = \operatorname{argmin}_{y \in Y} \left\{ g(y) + \frac{1}{2\lambda_n}\|y - y_n\|^2 + \frac{\gamma}{2\lambda_n}\|Ax_{n+1} + By - c\|^2 \right\},$$

(3) Si se cumple algún criterio de convergencia o se llega al máximo de iteraciones terminar, si no, ir al paso (2) con $n \leftarrow n + 1$. □

La elección de λ_n y γ se desprende de las condiciones que siguen a continuación, las cuales son las hipótesis para el resultado principal de [16].

Hipótesis (H)

(H_1) La función $f : \mathbb{R}^n \to \mathbb{R}$ es convexa, diferenciable y el gradiente ∇f es Lipschitz-continuo con constante L. La función $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ es propia, inferior semicontinua, y convexa. Los operadores $A : \mathbb{R}^n \to \mathbb{R}^d$ y $B : \mathbb{R}^m \to \mathbb{R}^d$ son lineales y continuos.

(H_2) El conjunto solución $\mathcal{S}$ es no vacío y $(x^*, y^*) \in \mathcal{S}$ si, y solamente si, existe z^* in $\mathbb{R}^d$ tal que (x^*, y^*, z^*) satisface

$$\begin{cases} -A^*z^* &= \nabla f(x^*), \\ -B^*z^* &\in \partial g(y^*), \\ c &= A^*x + b^*. \end{cases}$$

(H_3) El tamaño del paso λ_n forma una sucesión no creciente en ℓ^2 tal que $\sup_{n\in\mathbb{N}} \left(\frac{1}{\lambda_{n+1}} - \frac{1}{\lambda_n} \right) < +\infty$.

(H_4) El parámetro γ satisface $\gamma \in \left(0, \frac{2}{\|A\|^2}\right)$.

En [16] se menciona que es posible considerar $\lambda_n = 1/n^q$ con $1/2 < q \leq 1$. En el capítulo siguiente, veremos que la elección de λ_n puede aportar a la rapidez de convergencia del algoritmo, por ejemplo tomando sucesiones constantes a trozos.

Bajo estas hipótesis se presenta el siguiente resultado de convergencia.

TEOREMA 4.13. *Supongamos que las hipótesis (H) se satisfacen. Entonces toda sucesión* (x_n, y_n) *dada por el algoritmo AFBS converge a un punto en* $\mathcal{S}$.

Volviendo a nuestro propósito original, podemos considerar el problema

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}\|Kx - h\|_2^2 + r\|Wx\|_1 \right\}, \tag{4.4}$$

donde r es un parámetro positivo, h es un vector en $\mathbb{R}^{M_1}$ y K, W son matrices en $\mathcal{M}_{M_1 \times N}(\mathbb{R})$ y $\mathcal{M}_{M_2 \times N}(\mathbb{R})$ respectivamente. Aquí h representa una observación de una imagen original x_e mediante K, operador de ruido, además de un error aditivo de media cero, ε, esto es

$$Kx_e + \varepsilon = h.$$

El operador W representa la transformada ondícula, y como en los casos anteriores viene a aprovechar la localización de la base ondículas para que así la norma uno lleve términos a cero

para comprimir información. Este problema puede ser llevado a nuestra formulación presentada anteriormente mediante la restricción $Wx - y = 0$ y así mirar el problema en la forma

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\|Kx-h\|_2^2 + r\|y\|_1 \ : \ Wx - y = 0\right\}. \tag{4.5}$$

Otra formulación que puede ser abordada por este algoritmo es considerar

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\|Kx-h\|_2^2 + r\|Dx\|_1\right\}, \tag{4.6}$$

donde D es una matriz en $\mathcal{M}_{M_2\times N}(\mathbb{R})$ que representa el operador de gradiente discreto. Este modelo es conocido como modelo ROF debido a sus autores Rudin, Osher y Fatemi en [19]. Esta formulación propone recuperar imágenes que sean constantes a trozos pero localizadas en el espacio. Por ello se minimiza la norma uno del gradiente, de modo que la mayoría de componentes de este sean ceros salvo algunas que representan los cambios de tonalidad. Como en el caso anterior, incluyendo la restricción $Dx - y = 0$, podemos resolver el problema

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\|Kx-h\|_2^2 + r\|y\|_1 \ : \ Dx - y = 0\right\}, \tag{4.7}$$

El algoritmo para estos problemas puede ser escrito de manera explicita como

Algoritmo AFBS para imágenes

(1) Iniciación: *Tomar* $(\lambda_n)_{n\in\mathbb{N}}, \gamma$ *adecuados, elegir* (x_0, y_0) *convenientes,*

(2) *Actualizar* x_n, y_n *mediante*

$$x_{n+1} = x_n - \lambda_n K^*(Kx_n - h) - \gamma A^*(Ax_n - y_n),$$

$$(y_{n+1})_i = \begin{cases} (\tilde{y}_n)_i + \Lambda_n & \text{si} \quad (\tilde{y}_n)_i < -\Lambda_n, \\ 0 & \text{si} \quad -\Lambda_n \le (\tilde{y}_n)_i \le -\Lambda_n, \\ (\tilde{y}_n)_i - \Lambda_n & \text{si} \quad (\tilde{y}_n)_i > \Lambda_n, \end{cases}$$

donde

$$\Lambda_n = \frac{r\lambda_n}{1+\gamma} \qquad \text{y} \qquad \tilde{y}_n = \frac{y_n + \gamma A x_{n+1}}{1+\gamma}.$$

(3) Si se cumple algún criterio de convergencia o se llega al máximo de iteraciones terminar, si no, ir al paso (2) con $n \leftarrow n+1$. □

Donde el operador A corresponde a W o D dependiendo del caso.

Una observación importante es que los algoritmos previos, GPSR, ISTA y FISTA, no pueden ser aplicados directamente a la formulación (4.7). En [7], página 587, los autores de GPSR mencionan esto, mientras que en [20], los autores de FISTA, proponen resolver el modelo ROF utilizando FISTA en la formulación dual de este modelo, aumentando así las dimensiones del problema.

5. Algoritmo FAFBS

Pensando en el paso del algoritmo ISTA al algoritmo FISTA se propone una aceleración similar para pasar del algoritmo AFBS al algoritmo FAFBS, para resolver problemas del tipo (4.1). Las condiciones suficientes para la convergencia y el análisis de esta se proponen para un trabajo futuro. Nos limitaremos a proponer el algoritmo y analizar de manera experimental la convergencia.

Algoritmo FAFBS

(1) **Iniciación:** *Tomar* $(\lambda_n)_{n\in\mathbb{N}}, \gamma$ *adecuados,* $t_1 = 0$*, elegir* (x_0, y_0) *convenientes,* $w_0 = x_0$, $z_0 = y_0$,

(2) *Actualizar* x_n*,* y_n *mediante*

$$x_{n+1} = w_n - \lambda_n \nabla f(w_n) - \gamma A^*(Aw_n + Bz_n - c),$$

$$y_{n+1} = \operatorname{argmin}_{y\in Y} \left\{ g(y) + \frac{1}{2\lambda_n}\|y - z_n\|^2 + \frac{\gamma}{2\lambda_n}\|Ax_{n+1} + By - c\|^2 \right\},$$

$$t_{n+1} = \frac{1 + \sqrt{1 + 4t_n^2}}{2},$$

$$w_{n+1} = x_n + \left(\frac{t_n - 1}{t_{n+1}}\right)(x_n - x_{n-1}),$$

$$z_{n+1} = y_{n+1} + \left(\frac{t_n - 1}{t_{n+1}}\right)(y_{n+1} - y_n).$$

(3) Si se cumple algún criterio de convergencia o se llega al máximo de iteraciones terminar, si no, ir al paso (2) con $n \leftarrow n + 1$. □

De igual manera, aplicando este algoritmo para una formulación de corrección de imágenes, como en (4.5) o en (4.7) obtenemos

Algoritmo FAFBS para imágenes

(1) **Iniciación:** *Tomar* $(\lambda_n)_{n\in\mathbb{N}}, \gamma$ *adecuados,* $t_1 = 0$*, elegir* (x_0, y_0) *convenientes,* $w_0 = x_0$, $z_0 = y_0$,

(2) *Actualizar* x_n*,* y_n *mediante*

$$x_{n+1} = w_n - \lambda_n K^*(Kw_n - h) - \gamma A^*(Aw_n - z_n),$$

$$(y_{n+1})_i = \begin{cases} (\tilde{y}_n)_i + \Lambda_n & \text{si} \quad (\tilde{y}_n)_i < -\Lambda_n, \\ 0 & \text{si} \quad -\Lambda_n \leq (\tilde{y}_n)_i \leq -\Lambda_n, \\ (\tilde{y}_n)_i - \Lambda_n & \text{si} \quad (\tilde{y}_n)_i > \Lambda_n, \end{cases}$$

$$t_{n+1} = \frac{1 + \sqrt{1 + 4t_n^2}}{2},$$

$$w_{n+1} = x_n + \left(\frac{t_n - 1}{t_{n+1}}\right)(x_n - x_{n-1}),$$

$$z_{n+1} = y_{n+1} + \left(\frac{t_n - 1}{t_{n+1}}\right)(y_{n+1} - y_n).$$

Donde

$$\Lambda_n = \frac{r\lambda_n}{1 + \gamma} \qquad \text{y} \qquad \tilde{y}_n = \frac{z_n + \gamma A x_{n+1}}{1 + \gamma},$$

(3) Si se cumple algún criterio de convergencia o se llega al máximo de iteraciones terminar, si no, ir al paso (2) con $n \leftarrow n + 1$. □

El operador A corresponde a W o D dependiendo del caso.

Capítulo 5

Experimentos

1. Algoritmos de Compresión

1.1. SPIHT versus EZW. Como vimos en el capitulo dos, el algoritmo SPIHT surge a partir del algoritmo EZW buscando mejorar algunos de sus aspectos como la forma de guardar la información, trabajar con la posición de los pixeles, tener grandes listas insignificantes y pequeñas listas significativas. Aún así, en cada etapa los pixeles significativos son los mismos en cada algoritmo, esto nos dice que la información que se guarda es la misma pero de diferente manera.

Presentamos los Cuadros 1 y 2 para los algoritmos EZW y SPHIT respectivamente, con diferentes números de iteraciones, sus respectivos tiempos de computación, MSE, PSNR, tamaño en disco y el porcentaje del tamaño de la imagen comprimida en relación a la original. En la Figura 3 se muestra la imagen original la cual tienen un tamaño de 65.666 [B].

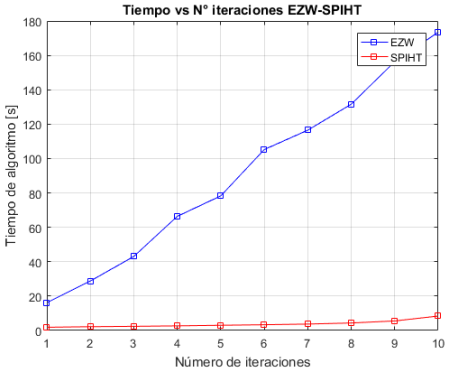
Para contrastar con mayor facilidad estos resultados se muestran los gráficos de la Figura 1. En (a) se muestra el tiempo de computación de cada algoritmo, se puede apreciar que el algoritmo SPHIT es mucho más rápido que el EZW, esto debido a que es mucho más dinámico en la manera que busca los valores significativos. En la Figura 1 (b) se enseña el tamaño en disco ocupado por la imagen comprimida, podemos notar que la información se guarda de manera más eficiente mediante el algoritmo SPIHT. De la Figura 1 (c) podemos ver que el PSNR es igual para ambos algoritmos (una gráfica encima de la otra), esto es ya que dado un número de iteraciones la información que se guarda en ambos algoritmos es la misma, pero el SPIHT lo hace de manera más rápida y eficiente.

Iteraciones	Tiempo [S]	MSE	PSNR [dB]	Tamaño [B]	Compresión %
1	1,87	3027,13	13,32	375	0,57 %
2	2,21	391,33	22,21	671	1,02 %
3	2,42	72,48	29,53	896	1,36 %
4	2,69	21,63	34,78	1.251	1,91 %
5	3,03	7,23	39,54	1.929	2,94 %
6	3,35	2,17	44,77	3.331	5,07 %
7	3,74	0,75	49,36	5.708	8,69 %
8	4,36	0,28	53,73	9.465	14,41 %
9	5,56	0,09	58,65	16.321	24,85 %
10	8,39	0,02	64,82	29.582	45,05 %

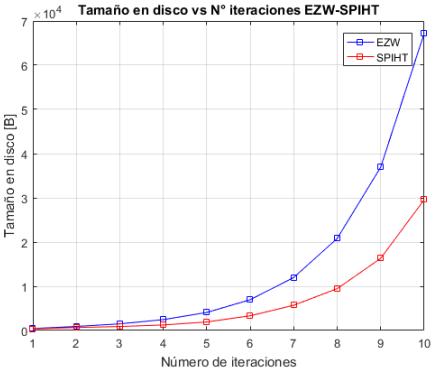
Cuadro 1. Resultados algoritmo EZW

Iteraciones	Tiempo [s]	MSE	PSNR [dB]	Tamaño [B]	Compresión %
1	16,18	3027,13	13,32	441	0,67 %
2	28,77	391,33	22,21	918	1,40 %
3	43,06	72,48	29,53	1.499	2,28 %
4	66,39	21,63	34,78	2.458	3,74 %
5	78,28	7,23	39,54	4.079	6,21 %
6	105,22	2,17	44,77	6.966	10,61 %
7	116,47	0,75	49,36	11.991	18,26 %
8	131,61	0,28	53,73	20.816	31,70 %
9	156,32	0,09	58,65	36.942	56,26 %
10	173,47	0,02	64,82	67.213	102,36 %

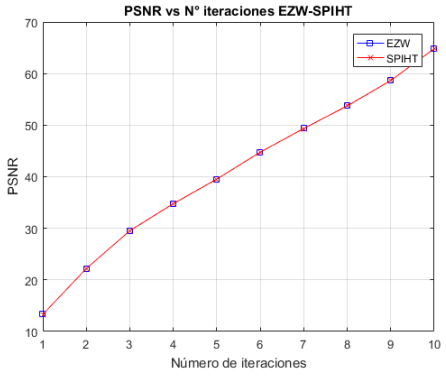
CUADRO 2. Resultados algoritmo SPIHT



(a)



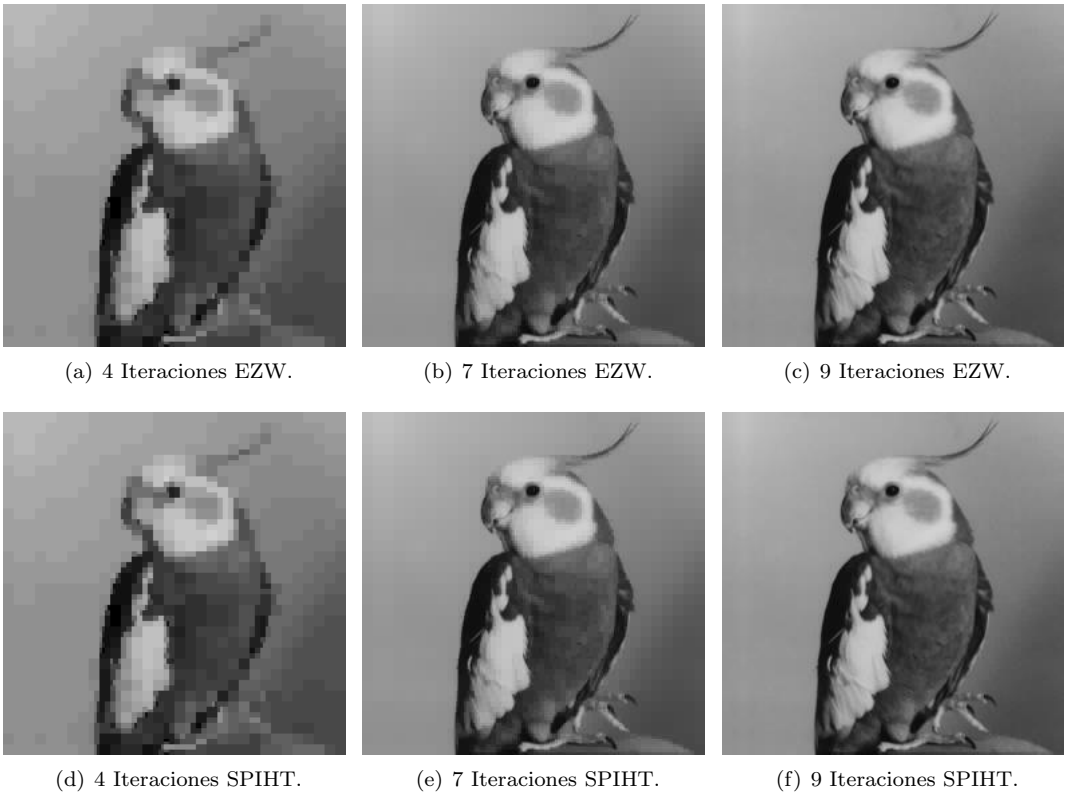
(b)



(c)

FIGURA 1. Algoritmos EZW y SPIHT

Se muestra en la Figura 2 las imágenes reconstruidas para 4, 7 y 9 iteraciones del algoritmo EZW y SPIHT en las cuales se puede apreciar que efectivamente no hay diferencias visuales entre los algoritmos.



(a) 4 Iteraciones EZW. (b) 7 Iteraciones EZW. (c) 9 Iteraciones EZW.

(d) 4 Iteraciones SPIHT. (e) 7 Iteraciones SPIHT. (f) 9 Iteraciones SPIHT.

FIGURA 2. Compresión, SPIHT versus EZW.

FIGURA 3. Imagen original.

1.2. Base Canónica versus Base de Ondículas. Como mencionamos en los capítulos de introducción a la teoría de ondículas y algoritmos de compresión, la razón por la que se trabaja con ondículas en lugar de la base canónica es que esta transformación se encarga de concentrar la información en ciertas regiones de la imagen.

En el Cuadro 3 están algunos resultados obtenidos para el algoritmo SPHIT empleado en la base canónica y base ondícula a nuestra imagen ejemplo. Podemos notar que utilizando ondículas el algoritmo es mucho más rápido, esto se puede explicar por la menor cantidad de pixeles que deben ser evaluados (hay más pixeles significativos). Por otro lado el comportamiento del PSNR tiende a ser mejor en la base canónica esto pues se trabaja con la imagen directamente y no pasa por el proceso de transformación e inversión de ondículas, aún así, las diferencias son pequeñas. Presentamos gráficamente estos resultados en la Figura 4.

En la Figura 5, mostramos las imágenes reconstruidas para 4 y 7 iteraciones. Se puede ver que la calidad de la imagen a partir de la base canónica es mejor que para la base de ondículas. A pesar de esto en el cuadro 3 se puede ver que con 9 iteraciones en base ondículas el algoritmo tarda menos, comprime más y su PSNR es mayor que para 4 y 7 iteraciones en base canónica.

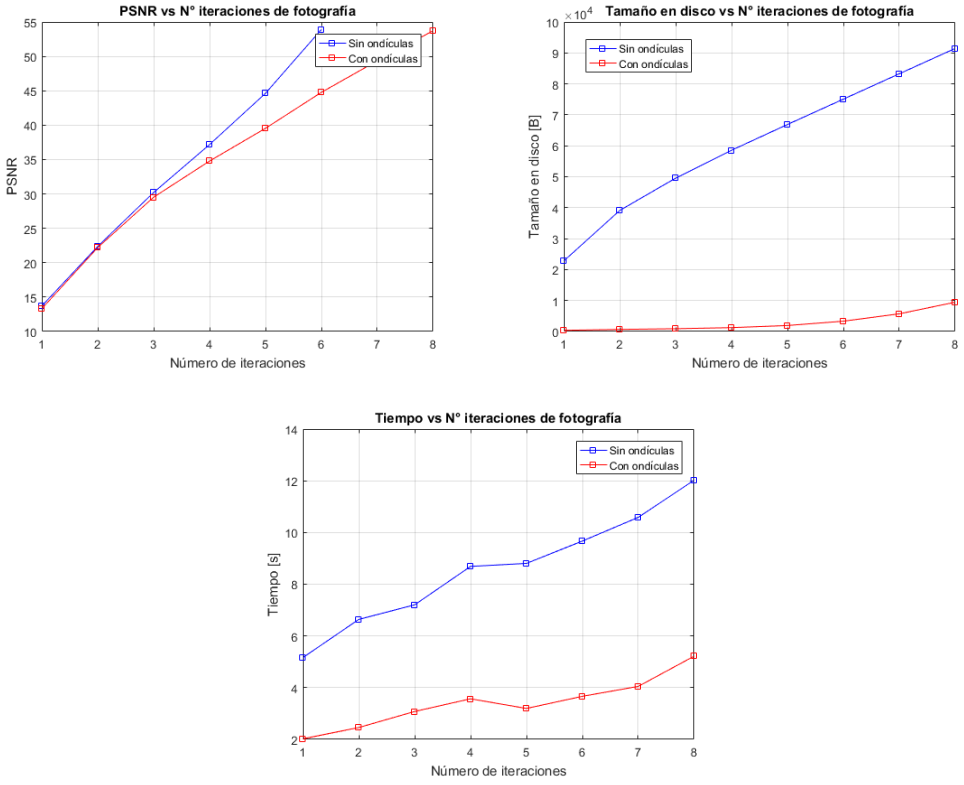


FIGURA 4. Base ondículas versus Base canónica de fotografía ejemplo

	Tiempo [s]		PSNR [dB]		Tamaño [B]	
Iteraciones	Canónica	Ondículas	Canónica	Ondículas	Canónica	Ondículas
1	5,16	2,02	13,71	13,32	22.797	375
2	6,64	2,46	22,38	22,21	39.104	671
3	7,20	3,08	30,21	29,53	49.552	896
4	8,70	3,57	37,18	34,78	58.561	1.251
5	8,81	3,20	44,63	39,54	66.880	1.929
6	9,67	3,67	53,91	44,77	75.072	3.331

CUADRO 3. Base ondículas versus Base canónica de fotografía ejemplo.

(a) 4 Iteraciones base ondículas.

(b) 7 Iteraciones base ondículas.

(c) 4 Iteraciones base canónica.

(d) 7 Iteraciones base canónica.

FIGURA 5. Contraste Base ondículas-Base canónica

1.3. Nivel de la transformada ondícula. En lo anterior hemos trabajado con la transformada ondícula de nivel tres. Veremos a continuación como afecta disminuir o aumentar el nivel de la transformación en nuestro algoritmo SPHIT. Para ello mantendremos el número de iteraciones de nuestro algoritmo SPHIT en cuatro y variaremos el nivel de la transformación ondícula.

Nivel ondícula	Tiempo [s]	MSE [dB]	PSNR [dB]	Tamaño [B]
1	4,42	17,52	35,70	16437
2	3,26	18,32	35,50	4387
3	2,68	21,63	34,78	1251
4	2,19	33,92	32,83	357
5	1,78	65,56	29,96	107
6	1,33	181,43	25,54	29
7	0,78	377,44	22,36	7

CUADRO 4. Nivel de transformada ondículas en algoritmo SPHIT.

Del cuadro 4 y la Figura 6 se puede ver que a medida que aumenta el nivel de la transformada ondícula disminuye el tiempo de computación esto pues hay menos pixeles significativos. Al aumentar el nivel también disminuye el MSE ya que hay menos información que se transmite, por esto igualmente disminuye el tamaño en disco de la compresión.

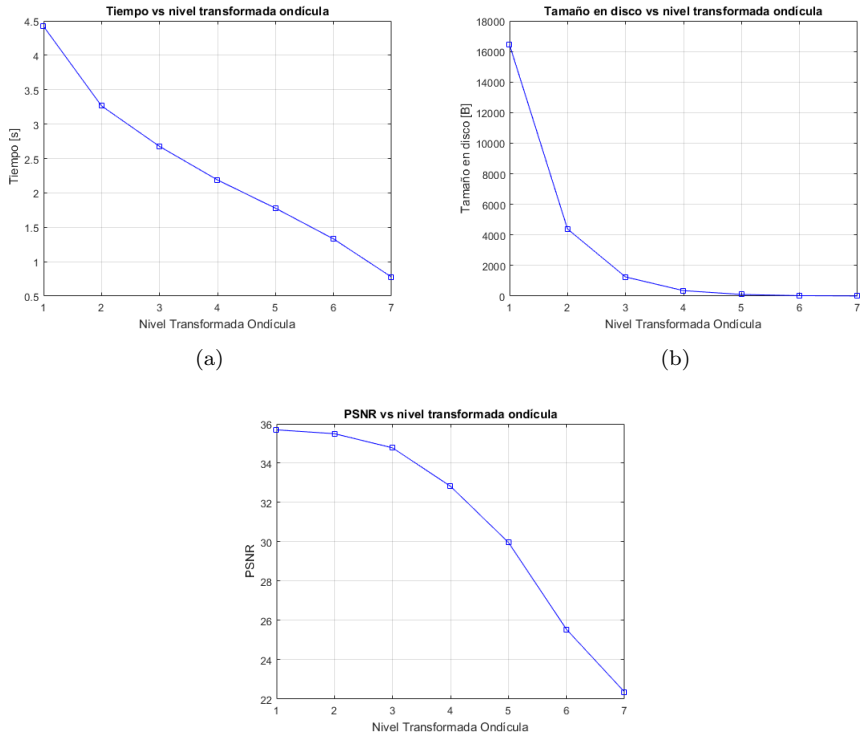


FIGURA 6. Nivel de transformada ondículas en algoritmo SPHIT.

2. Algoritmos de Corrección

2.1. Elección λ_n algoritmo AFBS y FAFBS. Antes de comparar los algoritmos de corrección propuestos, estudiaremos el comportamiento del algoritmo AFBS y FAFBS para diferentes sucesiones $(\lambda_n)_{n\in\mathbb{N}}$. Por ejemplo, es posible considerar

(a) $\lambda_n = 1/n$,
(b) $\lambda_n = 1{,}7/n^{0{,}505}$,
(c) λ_n constante a trozos, para ellos tomamos $m \in \mathbb{N}$ y definimos por ejemplo:

$$\lambda_n = \begin{cases} \frac{17}{10}\left(\frac{m}{n}\right)^{0{,}505} & \text{si} \quad n \in \{m \cdot k \mid k \in \mathbb{Z}\}, \\ \lambda_{n-1} & \text{en otro caso.} \end{cases} \tag{2.1}$$

Para el algoritmo AFBS evaluaremos los casos (a), (b) y (c) para $m = 50$ y $m = 100$. Para FAFBS los casos (b) y (c) con $m = 100$. En la Figura 7 se muestra el PSNR de la imagen obtenida en cada iteración en relación a la imagen original, Figura 8 (a), se trabajó en la imagen dañada de la Figura 8 (b). Se puede apreciar las variaciones en la convergencia dependiendo de la elección de λ_n. En negro se muestra la recta de 40[dB] del PSNR, valor donde la imagen ya presenta similitudes a la original. Podemos notar que cada caso está sobre esta recta por lo que la corrección es buena, aún así, mejores resultados se encuentran para la sucesión constante a trozos para $m = 100$ tanto como para AFBS como para FAFBS. Los demás parámetros de estos algoritmos se fijaron en $r = 10^{-4}$, $\gamma = 1$ para AFBS y $r = 10^{-4}$, $\gamma = 0{,}99$ para FAFBS.

De la figura 7 también podemos notar la convergencia del algoritmo FAFBS, la cual es más rápida que la del AFBS, esto valida experimentalmente nuestra propuesta.

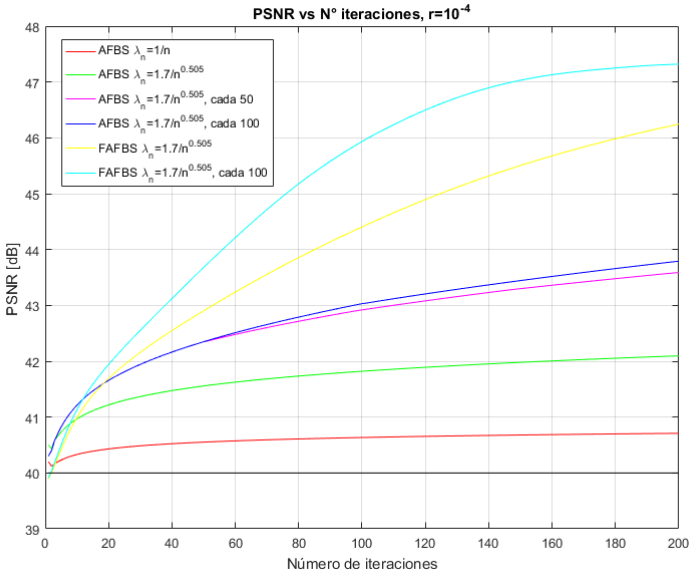


FIGURA 7. AFBS y FAFBS para diferentes λ_n

(a) Imagen original (b) Imagen dañada

FIGURA 8. Imagen original y dañada.

La imagen dañanda, Figura 8 (b), se obtiene a partir de la imagen original mediante las funciones *fspecial* y *imfilter* de MATLAB, aplicando un factor de borrosidad Gaussiano de tamaño 9×9 y de desviación estándar 4, seguido de un ruido Gaussiano blanco de media 0 y desviación estándar de 10^{-3}. Para obtener el operador asociado al factor de borrosidad A y su respectivo operador adjunto, A^*, es recomendable revisar [12].

2.2. Comparación algoritmos Corrección. En esta sección compararemos los cuatro algoritmos de corrección presentados GPSR, FISTA, AFBS, FAFBS, incluyendo también el algoritmo ISTA. Como vimos, estos algoritmos pueden resolver problemas que involucren la función objetivo

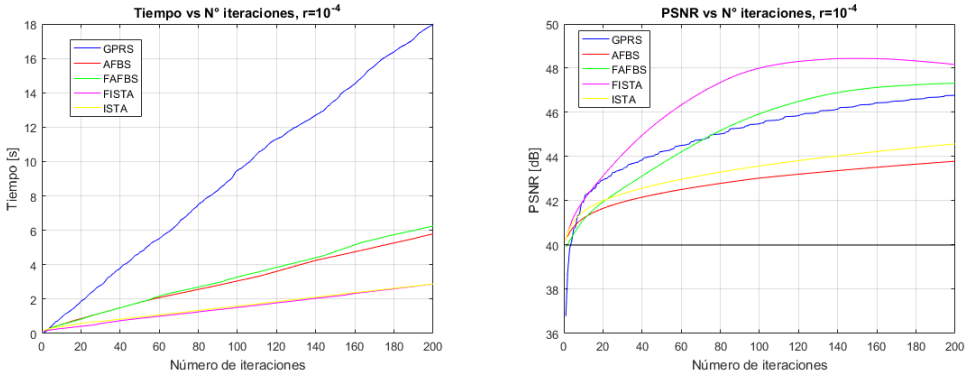
$$\frac{1}{2}\|Ax - h\|_2^2 + r\|x\|_1.$$

Contrastaremos estos algoritmos evaluando el tiempo de computo y la similitud de la imagen reconstruida a la original. Para los algoritmos GPSR, ISTA y FISTA se considera $x = Wu$ donde W es la transformada ondícula de nivel 3 dada por la base de Haar. Para el algoritmo GPSR se utilizaran parámetros $\beta = 1/2, \gamma = 1/10$. En el algoritmo AFBS utilizaremos $\gamma = 1$, λ_n constante a trozos, $m = 100$, en el algoritmo FAFBS $\gamma = 0{,}99$ y λ_n constante a trozos, $m = 100$, como en 2.1. En todos utilizaremos $r = 10^{-4}$.

De la Figura 9 (a) notamos que el algoritmo FISTA el más rápido, seguido de cerca por AFBS y FAFBS. Para la fidelidad de la reconstrucción. De la Figura 9 (b) observamos que todos los algoritmos corrigen de buena manera al estar sobre los 40[dB] del PSNR, el mejor resultado lo entrega el algoritmo FISTA seguido de FAFBS.

Se muestra en la Figura 10 los resultados para 200 iteraciones de los diferentes algoritmos buscando reconstruir la imagen dañada de la Figura 8 (b).

Finalmente concluimos que bajo las condiciones propuestas, el algoritmo que corrige de mejor manera es el algoritmo FISTA, superando en tiempo de cálculo y fidelidad de reconstrucción a los otros. Esto se debe principalmente a que ocupa técnicas más sofisticadas de optimización en relación al algoritmo GPSR y resuelve de manera más directa el problema propuesto que el algoritmo AFBS. Este último abarca formulaciones más generales, como veremos adelante permite resolver otros modelos que llegan a corregir mejor.



(a) Tiempo Corrección

(b) PSNR diferentes algoritmos

FIGURA 9. Análisis algoritmos de corrección.

(a) 200 iteraciones GPSR, $r = 10^{-4}$

(b) 200 iteraciones FISTA, $r = 10^{-4}$

(c) 200 iteraciones ISTA, $r = 10^{-4}$

(d) 200 iteraciones AFBS, $r = 10^{-4}$

(e) 200 iteraciones FAFBS, $r = 10^{-4}$

FIGURA 10. Resultados reconstrucción diferentes algoritmos.

2.3. Comparaciones modelos de Corrección. En el capítulo 4 sección 4 presentamos dos modelos de corrección:

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\|Kx-h\|_2^2+r\|Wx\|_1\right\}, \tag{2.2}$$

y

$$\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\|Kx-h\|_2^2+r\|Dx\|_1\right\}. \tag{2.3}$$

Estos modelos son similares, difiriendo sólo en los operadores W y D. Por un lado el operador W representa la transformada ondícula de modo que la norma uno lleve las componentes de Wx a cero para así encontrar una representación localizada. Por su parte el operador de gradiente discreto, D, busca mediante la norma uno disminuir los cambios de tonalidades en la imagen. El primer modelo se revisó anteriormente para los algoritmos GPRS, FISTA y AFBS. Como comentamos en el Capítulo 3 los dos primeros no pueden ser aplicados directamente al segundo modelo (ROF). Mediante la restricción $Dx-y=0$ podemos trabajar este modelo con los algoritmos AFBS y FAFBS.

Comparamos a continuación los dos modelos aplicados en los algoritmos AFBS y FAFBS. Para el modelo en base ondículas en AFBS utilizamos $\gamma=1$ mientras que en FAFBS $\gamma=0{,}99$. En el modelo ROF utilizamos $\gamma=0{,}05$ tanto como para AFBS como para FAFBS. En el modelo ROF para FAFBS tomamos $\lambda_n=1{,}7/n^{0{,}505}$. En el resto de casos tomamos λ_n constante a trozos para $m=100$, como en la igualdad (2.1).

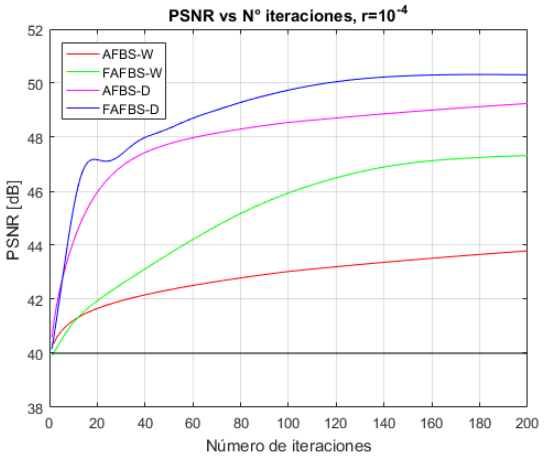


FIGURA 11. Comparación modelos de corrección.

De la Figura 11 podemos notar que el modelo tipo ROF, entrega mayor fidelidad de corrección en relación al modelo en base ondícula. Se puede notar además que el algoritmo FAFBS, aplicando al modelo ROF (FAFBS-D), converge más rápido que el algoritmo AFBS en el mismo modelo, esto reafirma la propuesta de aceleración. En la Figura 12 se muestran las imágenes corregidas a partir de nuestra imagen dañada de la Figura 8.

De la figura 13 podemos notar que AFBS y FAFBS mediante el modelo ROF corrigen incluso de mejor manera que FISTA en el modelo de ondículas. Esto demuestra la utilidad y versatilidad de AFBS, la cual hereda FAFBS.

(a) 200 iteraciones AFBS, Operador W (b) 200 iteraciones FAFBS, Operador W

(c) 200 iteraciones AFBS, Operador D (d) 200 iteraciones FAFBS, Operador D

FIGURA 12. Resultados reconstrucción diferentes modelos.

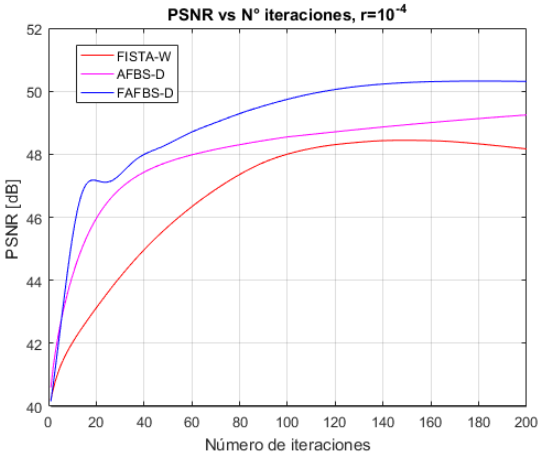


FIGURA 13. Modelos de corrección

Conclusión y trabajos futuros

A lo largo de los capítulos vimos que las ondículas son utilizadas en el proceso tanto de compresión como de corrección. Poder formalizar la teoría y resultados relacionados con las ondículas es de ayuda al momento de utilizarlas, para así hacerlo de manera correcta.

Las ondículas, nos permiten concentrar la información relevante en las regiones superiores de una imagen, esto ayuda a los algoritmos de compresión. Los algoritmos EZW y SPIHT se basan en esta idea de canalizar la información para así guardar la que es más relevante y obviar la que no, permitiendo de esta manera condensar la información. Vimos que el algoritmo SPIHT es mucho más eficiente y efectivo que el algoritmo EZW, esto debido a que el primero surge como una mejora del segundo basándose en la idea de pixeles significativos e insignificantes, además del concepto de descendencia.

Se presentaron tres algoritmos de corrección de imágenes, GPSR, FISTA, AFBS y se propuso un cuarto, FAFBS, los cuales pueden resolver el modelo de corrección en base ondículas. En el orden que fueron presentados van aumentando en el nivel de generalización del problema a tratar, de manera de estar enfocado directamente a resolver el problema de corrección a enfrentarlo como un caso particular. Pudimos notar que la elección de los parámetros para el algoritmo AFBS influye en la rapidez de convergencia, por lo que hacer una buena elección es de importancia. De los experimentos de comparación se obtuvo que el algoritmo FISTA destaca por sobre los otros tres en la formulación ondícula, esto se puede explicar por lo mencionado arriba, no es una formulación ni muy simple ni muy lejana del problema original de corrección de imágenes. Por otro lado, también se presentó el modelo ROF, modelo que entre los algoritmos presentados sólo puede ser resuelto por AFBS y FAFBS. Al momento de comparar las formulaciones en base ondículas y de gradiente discreto, se obtuvieron mejores resultados de corrección para la formulación de gradiente discreto, incluso mejorando a FISTA con el modelo de ondículas.

Experimentalmente se notó que el algoritmo propuesto, FAFBS, acelera de gran manera su primera versión AFBS, al ser este un algoritmo para problemas más generales que los que resuelve por ejemplo FISTA, sería de gran interés y aporte poder demostrar teóricamente la convergencia de este método, ya que puede ser aplicado en problemas de otras áreas como se menciona en [16].

La elección de la base de ondículas fue algo que no se tomó en cuenta. Existen publicaciones donde se realizan estudios y se proponen métodos para encontrar la mejor base dependiendo del propósito, [13], [14]. Adicionar estas variantes a los algoritmos puede ser beneficioso. Otro problema interesante de analizar es identificar el operador de ruido aplicado a la imagen que se quiere corregir. Los errores que se persiven al capturar una imagen rara vez pueden ser modelados directamente como se ha realizado en estos experimentos. En [15] se propone encontrar estos operadores mediante un problema de optimización.

Anexos

Presentamos a continuación ejemplos de los algoritmos EZW y SPIHT, los cuales pueden ser de utilidad para enfrentar los algoritmos por primera vez. Las iteraciones siguen a partir de la matriz (2.4) y del orden Z de la Figura 14.

58	-35	51	8	5	14	-10	5
-30	25	12	-14	3	1	5	-2
15	7	2	-9	5	-10	9	12
-10	-6	-11	5	7	-2	4	9
-2	11	-1	47	6	6	-2	3
0	3	-2	0	1	-4	3	1
1	-4	8	-2	4	5	3	12
5	11	3	3	-2	5	-4	1

(2.4)

1	2	5	6	17	18	21	22
3	4	7	8	19	20	23	24
9	10	13	14	25	26	29	30
11	12	15	16	27	28	31	32
33	34	37	38	49	50	53	54
35	36	39	40	51	52	55	56
41	42	45	46	57	58	61	62
43	44	47	48	59	60	63	64

FIGURA 14. Orden Z matriz 8×8.

Ejemplo algoritmo EZW

(1) **Iniciación:** Tenemos que $C_{\text{máx}} = 58$ entonces $T_0 = 2^5 = 32$. Se fija $N \leq 32$ y $n = 0$. La lista significativa contiene todos los valores de la matriz.

(2.1) **Etapa de pixeles significativos:**

i) Comenzamos con $p_{1,1} = 58$, como $|58| > T_0 = 32$ y $58 > 0$ exportamos la letra P, fijamos $p_{1,1} = 0$ y agregamos 58 a nuestra lista significativa.

ii) Seguimos con $p_{1,2} = -35$, como $|-35| > T_0 = 32$ y $-32 > 0$ exportamos la letra N, fijamos $p_{1,2} = 0$ y agregamos 35 a nuestra lista de significativa.

iii) Para $p_{2,1} = -30$, tenemos $|-30| < T_0 = 32$ pero uno de sus descendientes, $|p_{5,4}| = |47| > 32$, por lo tanto se exporta la letra Z.

iv) Para $p_{2,2} = 25$, tenemos $25| < T_0 = 32$ y todos sus de descendientes menores que nuestro umbral T_0 por lo que se exporta la letra T.

v) Para $p_{1,3} = 51$, como $|51| > T_0 = 32$ y $51 > 0$ exportamos la letra P y fijamos $p_{1,3} = 0$.

Se continúa hasta evaluar toda la lista significativa y se obtiene la cadena de letras

P-N-Z-T-P-T-T-T-T-Z-T-T-T-T-T-T-T-P-T-T.

Nuestra lista significativa queda como

[58 35 51 47],

a su vez nuestra la lista de insignificancia expresada como matriz en el orden Z

0	0	0	8	5	14	-10	5
-30	25	12	-14	3	1	5	-2
15	7	2	-9	5	-10	9	12
-10	-6	-11	5	7	-2	4	9
-2	11	-1	0	6	6	-2	3
0	3	-2	0	1	-4	3	1
1	-4	8	-2	4	5	3	12
5	11	3	3	-2	5	-4	1

.

(3.1) **Etapa de refinamiento:** Tenemos nuestra lista significativa [58 35 51 47].

i) Para $C = 58$. Fijamos $D_{C_0} = 58$, tenemos que $58 - 32 > 0$ entonces fijamos $D_{C_1} = 26$. Como $26 > 32/2 = 16$ exportamos el valor 1.

ii) Para $C = 35$. Fijamos $D_{C_0} = 58$, tenemos que $35 - 32 > 0$ entonces fijamos $D_{C_1} = 3$. Como $3 < 16$ exportamos el valor 0.

iii) Para $C = 51$. Fijamos $D_{C_0} = 58$, tenemos que $51 - 32 > 0$ entonces fijamos $D_{C_1} = 19$. Como $19 > 16$ exportamos el valor 1.

iv) Para $C = 47$. Fijamos $D_{C_0} = 58$, tenemos que $47 - 32 > 0$ entonces fijamos $D_{C_1} = 15$. Como $15 < 16$ exportamos el valor 0.

Así exportamos la cadena de números

1-0-1-0.

(4.1) Actualizamos $n \to n + 1$ y continuamos con $T_1 = 32/2 = 16$ en el paso (2).

(2.2) **Etapa de pixeles significativos:**

(i) Comenzamos con $p_{1,1} = 0$, como $|0| < T_1 = 16$ pero uno de sus descendientes, $|p_{2,1}| = |29| > 16$. Se exporta la letra Z.

(ii) Comenzamos con $p_{1,1} = 0$, como $|0| < T_1 = 16$ y todos sus descendientes son menores que nuestro umbral T_1 por lo que se exporta la letra T.

(iii) Para $p_{2,1} = -30$, tenemos $| - 30| > T_1 = 16$ y $-30 < 0$ por lo que se exporta la letra N. Fijamos $p_{2,1} = 0$ y agregamos 30 a nuestra lista significativa.

(iv) Para $p_{2,2} = 25$, tenemos $|25| > T_1 = 16$ y $25 > 0$ por lo que se exporta la letra P. Fijamos $p_{2,2} = 0$ y agregamos 30 a nuestra lista significativa. Se continúa hasta evaluar toda la lista de significativa y se obtiene la cadena de letras

Z-T-N-P-T-T-T-T-T-T-T-T.

Nuestra lista significativa queda como

[58 35 51 47 30 25],

a su vez nuestra la lista de insignificancia expresada como matriz en el orden Z

0	0	0	8	5	14	-10	5
0	0	12	-14	3	1	5	-2
15	7	2	-9	5	-10	9	12
-10	-6	-11	5	7	-2	4	9
-2	11	-1	0	6	6	-2	3
0	3	-2	0	1	-4	3	1
1	-4	8	-2	4	5	3	12
5	11	3	3	-2	5	-4	1

.

(3.2) **Etapa de refinamiento:** Ahora tenemos nuestra lista significativa [58 35 51 47 30 25].

(i) Para $C = 58$. Fijamos $C_{58_0} = 58$. Para $j = 0$ tenemos que $58 - T_0 = 58 - 32 > 0$ entonces fijamos $C_{58_1} = 26$. Para $j = 1$ tenemos que $C_{58_0} - T_1 = 26 - 16 = 10 > 0$ entonces $C_{58_2} = 10$. Fijamos $d_{58} = 10$, como $10 > 16/2 = 8$ exportamos el valor 1.

(ii) Para $C = 35$. Fijamos $C_{35_0} = 35$. Para $j = 0$ tenemos que $35 - T_0 = 35 - 32 > 0$ entonces fijamos $C_{35_1} = 3$. Para $j = 1$ tenemos que $C_{35_1} - T_1 = 3 - 16 = -13 < 0$ entonces $C_{35_2} = 3$. Fijamos $d_{35} = 3$, como $10 > 16/2 = 8$ exportamos el valor 0.

(iii) Para $C = 51$. Fijamos $C_{51_0} = 51$. Para $j = 0$ tenemos que $51-T_0 = 51-32 = 19 > 0$ entonces fijamos $C_{35_1} = 19$. Para $j = 1$ tenemos que $C_{51_1} - T_1 = 19 - 16 = 3 > 0$ entonces $C_{35_2} = 3$. Fijamos $d_{35} = 3$, como $3 < 8$ exportamos el valor 0.

(iv) Para $C = 47$. Fijamos $C_{47_0} = 47$. Para $j = 0$ tenemos que $47-T_0 = 47-32 = 15 > 0$ entonces fijamos $C_{47_1} = 15$. Para $j = 1$ tenemos que $C_{47_1} - T_1 = 15 - 16 = -1 < 0$ entonces $C_{47_2} = 15$. Fijamos $d_{47} = 15$, como $15 > 8$ exportamos el valor 1.

(v) Para $C = 30$. Fijamos $C_{30_0} = 30$. Para $j = 0$ tenemos que $30-T_0 = 30-32 = -2 < 0$ entonces fijamos $C_{47_1} = 30$. Para $j = 1$ tenemos que $C_{47_1} - T_1 = 30-16 = 14 > 0$ entonces $C_{30_2} = 14$. Fijamos $d_{30} = 14$, como $14 > 8$ exportamos el valor 1.

(vi) Para $C = 25$. Fijamos $C_{25_0} = 25$. Para $j = 0$ tenemos que $25-T_0 = 25-32 = -7 < 0$ entonces fijamos $C_{25_1} = 25$. Para $j = 1$ tenemos que $C_{25_1} - T_1 = 25-16 = 9 < 0$ entonces $C_{25_2} = 9$. Fijamos $d_{25} = 9$, como $9 > 8$ exportamos el valor 1.

Así exportamos la cadena de números

1-0-0-1-1-1.

(4.2) Actualizamos $n \to n + 1$ y continuamos con $T_2 = 16/2 = 8$ en el paso (2).

(2.3) Continuar iterando ... □

Mostramos a continuación algunas iteraciones de nuestro algoritmo de decodificación para los cadenas obtenidas en el ejemplo de codificación.

(1) **Iniciación:** Iniciamos todos los valores de nuestra matriz de reconstrucción como ceros. Comenzamos con el umbral $T_0 = 32$ que se comenzó la codificación. Iniciamos con $n = 0$.

(2.1) **Etapa de pixeles significativos:** Nuestra lista de letras es P-N-Z-T-P-T-T-T-T-Z-T-T-T-T-T-T-T-P-T-T.

(i) Tenemos L_1 =P adicionar el valor $T_0 = 32$ a nuestra matriz en la posición 1.

(ii) L_2 =N adicionar el valor $-T_0 = -32$ a nuestra matriz en la posición 2.

(iii) L_3 =Z, hacemos nada.

(iv) L_4 =T, esto nos dice que la descendencia de la entrada 4 son nulos. Por ende bloqueamos las entradas $\{13, 14, 15, 16, 49, 50, 51, \ldots, 62, 63, 64\}$.

(v) L_5 =P adicionar el valor $T_0 = 32$ a nuestra matriz en la posición 5 que es la posición que sigue a ser llenada y no ha sido bloqueada.

(vi) Iterar hasta el final de la lista de letras ...

Al finalizar esta etapa se obtiene la siguiente matriz de reconstrucción:

32	-32	32	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	32	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(3.1) **Etapa de refinamiento:** Recorremos nuestra cadena de números 1-0-1-0. El primer uno nos dice que debemos sumar $T_0/2 = 16$ a nuestra primera entrada, el siguiente valor cero nos dices que no debemos hacer nada con la segunda entrada, el siguiente uno nos dice que debemos sumar 16 a nuestra entrada número 5, finalmente el último cero nos dice que no debemos hacer nada con la entrada número 38. Esto reporta la siguiente matriz de reconstrucción:

48	-32	48	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	32	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(4.1) **Iterar:** Continuar con $n \to n + 1 = 1$ actualizar $T_1 = T_0/2 = 16$ e ir al paso (2).

(2.2) **Etapa de pixeles significativos:** Esta etapa entrega la matriz

48	-32	48	0	0	0	0	0
-16	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	32	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(3.2) **Etapa de refinamiento:** Al finalizar esta etapa se reporta la matriz

56	-32	48	0	0	0	0	0
-24	24	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	40	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

.

(4.2) **Iterar:** Continuar con $n \to n+1 = 2$ actualizar $T_2 = T_1/2 = 8$ e ir al paso (2).

(2.3) Continuar iterando ... □

Ejemplo algoritmo SPIHT

Presentamos a continuación una iteración del algoritmo codificación SPIHT para nuestra matriz de Ejemplo (2.4).

(1) **Iniciación:** Comenzamos con el umbral $T_0 = 32$, iniciamos con $n = 0$, LSP= { }, LIP= $\{(0,0),(0,1),(1,0),(1,1)\}$, LIS= $\{(0,1)A,(1,0)A,(1,1)A\}$.

(2) **Etapa de clasificación:**

(2.1) Comenzamos con $(i,j) = (1,1)$, $p_{1,1} = 58$ entonces $S_{32}(1,1) = 1$ y $58 > 0$ por lo tanto exportamos la letra P y movemos el $(0,0)$ a nuestra lista LSP.

Seguimos con $(i,j) = (1,2)$, $p_{1,2} = -35$ entonces $S_{32}(1,2) = 1$ y $-35 < 0$ por lo tanto exportamos la letra N y movemos el $(0,1)$ a nuestra lista LSP.

Continuamos con $(i,j) = (2,2)$, $p_{2,1} = -30$ entonces $S_{32}(2,1) = 0$ por lo tanto exportamos el valor 0.

Finalmente para $(i,j) = (2,2)$, $p_{2,2} = 25$ entonces $S_{32}(2,2) = 0$ por lo tanto exportamos el valor 0.

Pasamos a la siguiente etapa con las listas LIS= $\{(1,2)A,(2,1)A,(2,2)A\}$, LIP= $\{(2,1),(2,2)\}$, LSP= $\{(1,1),(1,2)\}$. Por ahora nuestra lista de letras es P-N-0-0.

(2.2) Comenzamos con $(i,j) = (1,2)$ cual entrada es tipo A. Tenemos que $\mathcal{D}(1,2) = \{(1,3),(1,4),(2,3),(2,4)\}$ y para estas coordenadas los valores en la matriz son $51, 8, 12$ y -14 respectivamente, por lo tanto $S_{32}(\mathcal{D}(1,2)) = 1$ y entonces debemos revisar cada uno de estos elementos. Para $(k,l) = (1,3)$ tenemos $p_{1,3} = 51$ por lo tanto $S_{32}(1,3) = 1$ entonces agregamos (k,l) a la lista LSP y exportamos P ya que $51 > 0$. Para $(k,l) = (1,4)$ tenemos $p_{1,4} = 8$ por lo tanto $S_{32}(1,4) = 0$ así agregamos (k,l) al final de la lista LIP, esto mismo ocurre para $(k,l) = (2,3)$ y $(k,l) = (2,4)$ pues $p_{2,3} = 12$ y $p_{2,4} = -14$ entonces $S_{32}(2,3) = S_{32}(2,4) = 0$. Como $\mathcal{L}(1,2) \neq \emptyset$ debemos mover $(1,2)$ al final de la lista LIS como entrada tipo B.

Ahora, debemos hacer el mismo proceso anterior para $(i,j) = (2,1)$ ya que es tipo A, luego se continúa con $(i,j) = (2,2)$ tipo A, de igual manera. Se debe seguir

con este proceso hasta terminar con esta lista, es importante notar que esta se va actualizando y también se debe pasar por estos valores nuevos.

Al finalizar esta etapa se obtiene
- LIS=$\{(2,2)A, (1,2)B, (3,1)A, (4,1)A, (4,2)A\}$,
- LIP=$\{(2,1), (2,2), (1,4), (2,3), (2,4), (3,1), (3,2), (4,1), (4,2), (5,1), (6,3), (6,4)\}$,
- LSP=$\{(1,1), (1,2), (1,3), (5,3)\}$.

La lista de letras a exportar es

P-N-0-0-1-P-0-0-0-1-0-0-0-0-0-0-1-0-1-0-P-0-0-0-0.

(3) **Etapa de refinamiento:** Esta etapa es análoga a la etapa de refinamiento del algoritmo EZW pero con los valores correspondientes a las coordenadas de la lista LSP (que son los mismos del ejemplo del algoritmo EZW). Como primera iteración se reporta la lista de números

1-0-1-0.

(4) **Iterar:** Pasamos a $n = 2$, entonces $T_2 = T_n/2 = 16$ e ir al paso (2).

(2) Continuar iterando ... □

Bibliografía

[1] Frazier Michael, An introduction to wavelets through linear algebra. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1999. xvi+501 pp.

[2] K.R. Rao, P.C. Yip, The Transform and Data Compression Handbook. Electrical Engineering & Applied Signal Processing Series. CRC Press LLC, Boca Raton, 2001. 408 pp.

[3] J. M. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficients. IEEE Transactions on Signal Processing, Vol 41, no 12, 1993, pp. 3445-3462.

[4] A. Said, W. A. Pearlman, A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. IEEE Transactions on Circuits and Systems for Video Technology, Vol 6, no 3, 1996, pp. 243-250.

[5] M. Misiti, Y. Misiti, G. Oppenheim, J. M. Poggi, Wavelets and their Applications. Digital Signal and image processing series. Wiley-ISTE, 2007, 330pp.

[6] Peypouquet, Juan Convex optimization in normed spaces. Theory, methods and examples. With a foreword by Hedy Attouch. SpringerBriefs in Optimization. Springer, Cham, 2015. xiv+124 pp.

[7] M. A. T. Figueiredo, R. D. Nowak, S, J. Wright, Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems. IEEE Journal of Selected Topics in Signal Processing, Vol 1, no 4, 2007, pp. 586-597.

[8] T. Beck, A. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. 2 (2009), no. 1, pp. 183–202.

[9] Walker, James S. A primer on wavelets and their scientific applications. Second edition. Studies in Advanced Mathematics. Chapman & Hall/CRC, Boca Raton, FL, 2008. xvi+300 pp.

[10] David A. Huffman, A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE, vol. 40, no 9, 1952, pp. 1098-1101.

[11] D. P. Bertsekas. Bertsekas, Dimitri P. Nonlinear programming. Second edition. Athena Scientific Optimization and Computation Series. Athena Scientific, Belmont, MA, 1999. xiv+777 pp.

[12] Hansen, Per Christian; Nagy, James G.; O'Leary, Dianne P. Deblurring images. Matrices, spectra, and filtering. Fundamentals of Algorithms, 3. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006. xiv+130 pp.

[13] K. Ramchandran, M. Vetterli, Best wavelet packet bases in a rate-distortion sense. IEEE Transactions on Image Processing, vol. 2, no 2, 1993, pp. 160-175.

[14] R. Coifman, V. Wickerhauser, Entropy-based algorithms for best basis selection, IEEE Trans. on Information Theory, vol. 38, no 2, 1992, pp. 713-718.

[15] De los Reyes, Juan Carlos; Schönlieb, Carola-Bibiane Image denoising: learning the noise model via nonsmooth PDE-constrained optimization. Inverse Probl. Imaging 7 (2013), no. 4, pp. 1183–1214.

[16] C. Molinari, J. Peypouquet, F. Roldán, Alternating Forward-Backward Splitting for Linearly Constrained Optimization Problems. *Under review.*

[17] G. M. Morton, A computer Oriented Geodetic Data Base; and a New Technique in File Sequencin. Technical Report, Ottawa, Canada: IBM Ltd, 1999, 19 pp.

[18] R. Janaki, A. Tamilarasi, Still Image Compression by Combining EZW Encoding with Huffman Encoder. International Journal of Computer Applications, vol. 13 , no 7, 2011, pp. 1-7.

[19] L. I. Rudin, S. Osher, E. Fatemi, Nonlinear Total Variation Based Noise Removal Algorithms. Physica D, vol. 60, 1992, pp. 259-268.

[20] A. Beck, M Teboulle, Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems. IEEE Transactions on image processing, vol.18, no. 11, 2009, pp.2419-2434.