UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

PROCESAMIENTO DE IMÁGENES CON WAVELETS

Autora: Lucia Bordajandi Moleón

Director: Dr. Joaquim Ortega Cerdà
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 14 de enero de 2025

Abstract

The theory of wavelets is part of the field of harmonic analysis, with various applications in signal processing. In this project, we will review wavelets and apply the fast wavelet transform algorithm to some problems in image processing, such as denoising and compression.

Resumen

La teoría de wavelets forma parte del campo de análisis armónico, con diversas aplicaciones en procesamiento de señales. En este proyecto, haremos un repaso de las wavelets y aplicaremos el algoritmo de la transformada rápida de wavelets a algunos problemas de procesamiento de imágenes, como limpieza de ruido y compresión. [1]

[1] 2020 Mathematics Subject Classification. 42C40, 65T60, 94A12

Agradecimientos

En primer lugar, me gustaría agradecer todo su esfuerzo y dedicación a mi tutor Joaquim Ortega. Este proyecto no habría sido posible sin su ayuda.

También quiero dar las gracias a los amigos que me han acompañado durante la carrera. Gracias a Nel, Alberto, Clara y Mireia por convertir estos años en una época muy especial. Quiero agradecer a Paloma y Manu por animarme siempre y apoyarme para terminar mis estudios.

Finalmente, quiero agradecer a toda mi familia por su apoyo incondicional a lo largo de los años. Quisiera agradecer a mis hermanos por estar siempre ahí. Gracias a mi padre por animarme a tener curiosidad desde pequeña, y a mi madre por servirme de inspiración.

Índice

1. Introducción

Una gran variedad de datos puede ser representada como señales de distintas dimensiones, desde audio hasta imágenes y vídeo, además de toda la información expresada en forma de series temporales. En este trabajo aplicaremos herramientas de teoría de wavelets [PW12] para procesamiento de imágenes. En concreto, presentamos aplicaciones de compresión de imágenes para optimización de espacio de memoria y eliminación de ruido conservando la información más relevante.

La teoría de wavelets fue desarrollada desde principios del siglo pasado [Haa10], ofreciendo una alternativa natural a las aproximaciones a través de series de Fourier [PW12]. Cuando se aproximan funciones mediante una base de wavelets, se aproxima con funciones localizadas en el tiempo y subespacios de aproximaciones y detalles provistos de una estructura interna. Esto último tiene aplicaciones relevantes, como el desarrollo del formato de compresión de imágenes JPEG 2000 [MGBB00].

El primer ejemplo de wavelet y el más sencillo, es el de la función de Haar [Haa10], que trataremos en detalle para ilustrar conceptos más complejos. Haremos una revisión de teoría de wavelets [PW12], comenzando por el análisis de Haar y comprobando que las wavelets de Haar forman un sistema completo ortonormal. Formalizaremos la definición de función wavelet e introduciremos el Análisis de Multiresolución (MRA) junto a las funciones de escalado asociadas a este último. Veremos que, por el teorema de Mallat [Mal99], cada MRA define únicamente un sistema de wavelets, y demostraremos este enunciado. Describiremos la *Fast Wavelet Transform*, el algoritmo computacionalmente eficiente para hallar los coeficientes en la base de wavelets mediante sus filtros. Finalmente, aplicaremos este proceso a distintas imágenes para eliminar los menores coeficientes en este espacio y deshacernos del ruido además de ahorrar espacio de memoria.

Estructura de la Memoria

Empezaremos el trabajo con la Sección 2, presentando una serie de resultados preliminares que utilizaremos posteriormente. En la Sección 3 haremos una introducción a la teoría de wavelets a través del sistema de Haar. La sección 4 introduce el concepto de MRA Ortogonal, que será central para plantear y demostrar el Teorema de Mallat en la Sección 5. Desarrollaremos el algoritmo de la transformada de wavelets durante la Sección 6, que aplicaremos finalmente en la Sección 7 para el procesamiento de imágenes mediante wavelets.

2. Conceptos preliminares

El espacio de funciones $L^2(\mathbb{R})$ es el espacio vectorial formado por el conjunto de funciones de cuadrado integrable en el sentido de Lebesgue junto la suma y el producto por escalares. La norma L^2 viene inducida por el producto escalar

$$\langle f, g\rangle \coloneqq \int_{\mathbb{R}} f(x)\overline{g(x)}dx,$$

tal que $||f||_2 = \langle f, f\rangle^{1/2}$. En concreto, este espacio es un *espacio de Hilbert*, es decir un espacio vectorial completo respecto la norma inducida de un producto escalar. Recordamos que dos funciones $f, g \in L^2(\mathbb{R})$ son *ortogonales* si su producto escalar es 0, es decir, $\langle f, g\rangle = 0$. A su vez, decimos que una familia de funciones $\{f_i\}_{i\in I}$ es una *familia ortogonal* si $\langle f_i, f_j\rangle = 0$ para todo $i, j \in I, i \neq j$.

Definición 2.1. Sea $\{f_i\}_{i\in I}$ una familia ortogonal en un espacio de Hilbert H, decimos que forma *base ortogonal* de este espacio si cualquier elemento de H puede expresarse como combinación lineal finita o numerable de $\{f_i\}_{i\in I}$. Diremos que esta base es *ortonormal* si además $\langle f_i, f_i\rangle = 1$ para todo $i \in I$.

El concepto de *energía* en procesamiento de señales se define como $E_s = \langle x(t), x(t)\rangle$, donde $x(t)$ es una señal para tiempo continuo o $E_s = \sum_{-\infty}^{\infty} |x(n)|^2$ para $x(n)$ señal en tiempo discreto. Observamos que si E_s es finita, la señal no es periódica ya que los coeficientes deben decaer a partir de cierto valor [Pro01].

Recordamos que un subespacio $W \in L^2(\mathbb{R})$ es cerrado si toda sucesión convergente de elementos de W en $L^2(\mathbb{R})$ converge a un punto en W.

Teorema 2.2 (Proyección ortogonal)**.** *Dado un subespacio cerrado W del espacio $L^2(\mathbb{R})$ y dada una función f en $L^2(\mathbb{R})$, existe una única función $P_W f$ en W que minimiza la distancia de f a W. Esto es,*

$$||f - g||_2 \geq ||f - P_W f||_2 \quad \textit{para todo } g \in W$$

Además, el vector $f - P_W f$ queda caracterizado por ser ortogonal a cualquier vector de W.

Llamaremos proyección ortogonal a esta función $P_W f$. Podemos encontrar una prueba de este teorema en [Rud87].

El teorema 2.2 implica que si x es un vector de $L^2(\mathbb{R})$, $x \notin W$, proyectando x de forma ortogonal al subespacio W y calculando la norma del vector diferencia $||x - P_W x||_2$ obtenemos la distancia mínima de x al subespacio W en $L^2(\mathbb{R})$, de manera análoga a cómo lo haríamos en espacios euclídeos.

A lo largo de este trabajo haremos uso de la transformada de Fourier de funciones $f \in L^2(\mathbb{R})$ y de sus propiedades básicas.

Definición 2.3. Sea $f \in L^2(\mathbb{R})$. Definimos la *transformada de Fourier* de f como

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(x)e^{-2\pi i\xi x}dx.$$

Recordamos que la transformada de Fourier es una transformación lineal. Es decir, si $f, g \in L^2(\mathbb{R})$ y $a, b \in \mathbb{C}$, entonces

$$\widehat{(af + bg)} = a\hat{f} + b\hat{g}. \tag{2.1}$$

También utilizaremos que la transformada de Fourier de una traslación es una modulación y viceversa. Es decir,

$$\widehat{\tau_h f(\xi)} = \widehat{f(\xi - h)} = e^{-2\pi i h\xi}\hat{f}(\xi) \tag{2.2}$$

$$\widehat{e^{2\pi i h\xi} f}(\xi) = \hat{f}(\xi - h) = \tau_h \hat{f}(\xi). \tag{2.3}$$

3. Introducción a la teoría de Wavelets

En esta sección presentamos el concepto de wavelet mediante la función de Haar. Esta función fue la primera wavelet en ser descubierta y es la más simple. Comprobaremos que las familias de estas funciones dan lugar a bases ortonormales de $L^2(\mathbb{R})$ y definiremos el concepto de wavelet de manera formal. Los resultados de esta sección se derivan de [PW12].

3.1. Primeros pasos: Análisis de Haar

Presentamos uno de los primeros ejemplos de wavelets, introducido en 1910 [Haa10]. Este sencillo ejemplo nos permite mostrar qué propiedades básicas buscamos para las wavelets. Primero necesitamos introducir la geometría de los intervalos diádicos, que caracteriza el análisis de Haar.

Por comodidad, utilizaremos la siguiente notación. Dado un intervalo $I = [a, b) \in \mathbb{R}$, denotamos la mitad izquierda como $I_l = [a, (a+b)/2)$ y la derecha como $I_r = [(a+b)/2, b)$.

Definición 3.1 (Intervalos diádicos)**.** Llamaremos *intervalos diádicos* a los intervalos de la forma

$$I_j^k = [k2^{-j}, (k+1)2^{-j}) \quad \text{para enteros } j, k.$$

Sea D el conjunto de todos los intervalos diádicos en $\mathbb{R}$ y D_j el conjunto de intervalos $I_j^k \in D$ de longitud fija 2^{-j}. Está claro que $D = \cup_{j\in\mathbb{Z}} D_j$. Notamos que cada D_j establece una partición de la recta real. En consecuencia, cada intervalo diádico I_j^k pertenece a una única partición o generación D_j. Además, I_j^k se puede dividir en dos intervalos diádicos (una mitad izquierda y una derecha) de D_{j+1} y está contenido en un intervalo en la generación anterior D_{j-1}. Esto se puede resumir en que dos intervalos diádicos o bien están anidados o bien son disjuntos.

Lema 3.2. *Si $I, J \in D$, entonces $I \cap J = \emptyset$ o $I \subseteq J$ o $J \subseteq I$.*

Demostración. Supongamos que $I \cap J \neq \emptyset$. Si I, J están en la misma generación, está claro que son iguales. Supongamos, sin pérdida de generalidad, que I es de mayor generación que J, es decir, $i > j$. Sean $I = [k2^{-i}, (k+1)2^{-i})$, $J = [n2^{-j}, (n+1)2^{-j})$ con $i, j, n, k \in \mathbb{Z}$. Definimos $l = i - j$. Tenemos que

$$J = [n2^{l-i}, (n+1)2^{l-i}) = \left[\frac{n2^l}{2^i}, \frac{(n+1)2^l}{2^i}\right).$$

Si $k < n2^l$ entonces por ser ambas partes de la desigualdad enteros tenemos que $k+1 \leq n2^l$. Este caso es imposible puesto que $I \cap J \neq \emptyset$. Por lo tanto, $k \geq n2^l$. Por otro lado, si $k+1 > (n+1)2^l$ entonces $k \geq (n+1)2^l$ por ser enteros, pero esto es imposible de nuevo porque I y J no son disjuntos. Así pues $I \subseteq J$. □

Definición 3.3 (Función de Haar)**.** La función de Haar asociada al intervalo I es la función h_I

$$h_I(x) \coloneqq (1/\sqrt{|I|})(\chi_{I_r}(x) - \chi_{I_l}(x)).$$

Lema 3.4. *La familia de funciones de Haar $\{h_I\}_{I \in D}$ indexadas por los intervalos diádicos es una familia ortonormal.*

Demostración. Sean $I, J \in D$ con $I \neq J$, por 3.2 o bien son intervalos disjuntos o bien uno está contenido en el otro. Si $I \cap J = \emptyset$ entonces $\langle h_I, h_J \rangle = 0$, ya que los soportes son disjuntos. Si I está contenido estrictamente en J, entonces h_J es constante en I, y de hecho

$$\langle h_I, h_J \rangle = \int_I h_I(x) h_J(x) dx = \frac{\pm 1}{|J|^{1/2}} \int_I h_I(x) dx = 0$$

por construcción de h_I. Análogamente tenemos el caso en el que $J \subsetneq I$, por lo que hemos comprobado que es una familia ortogonal.

Para probar la normalidad notamos que, si $I = J$,

$$\langle h_I, h_I \rangle = ||h_I||_2 = \int_I |h_I(x)|^2 dx = \frac{1}{|I|} \int_I dx = 1$$

por las definiciones de la norma L^2 y h_I. □

3.2. Completitud del sistema de Haar

A continuación, mostraremos que la familia de funciones de Haar forma un sistema completo. Este hecho será relevante, ya que en última instancia buscamos expresar las funciones originales $f \in L^2(\mathbb{R})$ en bases más convenientes de $L^2(\mathbb{R})$.

Teorema 3.5. *La familia de funciones de Haar $\{h_I\}_{I \in D}$ forma una base ortonormal de $L^2(\mathbb{R})$.*

Para demostrar la completitud del sistema utilizaremos varios resultados auxiliares. Además, necesitaremos definir los operadores promedio y diferencia. Haremos uso de la siguiente notación:

Definición 3.6. Dada f localmente integrable, definimos el valor medio de f en el intervalo I como

$$m_I f \coloneqq \frac{1}{|I|} \int_I f(x) dx.$$

El operador promedio toma los valores medios de f en cada intervalo diádico de generación j, siendo constante en cada I_j.

Definición 3.7. Definimos el *operador promedio* como la aplicación $P_j : L^2(\mathbb{R}) \to L^2(\mathbb{R})$, $j \in \mathbb{Z}$, tal que

$$P_j f(x) \coloneqq \frac{1}{|I_j|} \int_{I_j} f(t) dt$$

Donde $I_j = I_j(x)$ es el intervalo diádico de la generación j que contiene x.

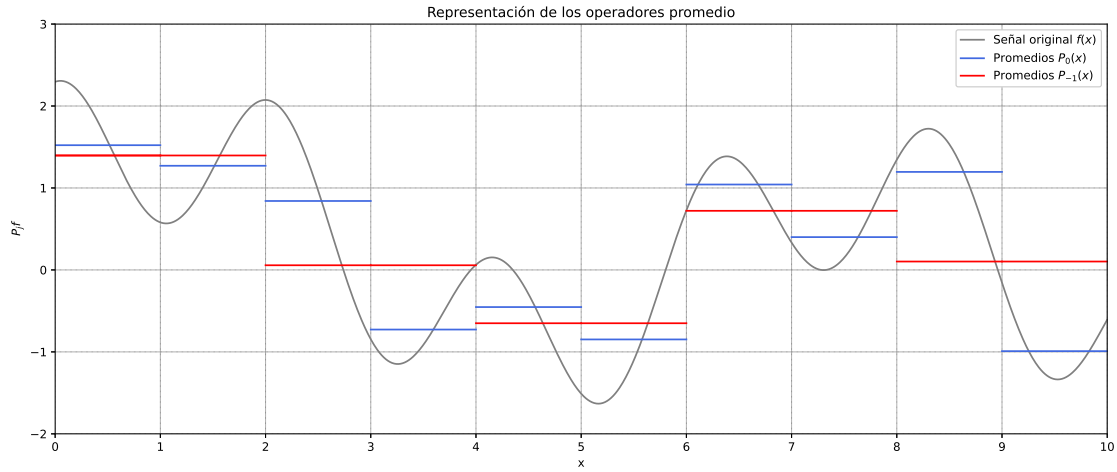


Figura 1: Ejemplo de valores promedios para escalas 0 y -1.

Notamos que $P_j f(x) = m_{I_j} f$ para todo $x \in I_j$ por definición de valor medio (3.6).

Los operadores diferencia obtienen la información necesaria para pasar a mejores aproximaciones, es decir, de $P_j f$ a $P_{j+1} f$.

Definición 3.8 (Operador diferencia)**.** Definimos los operadores diferencia $Q_j : L^2(\mathbb{R}) \to L^2(\mathbb{R})$, $j \in \mathbb{Z}$, como

$$Q_j f(x) \coloneqq P_{j+1} f(x) - P_j f(x)$$

A continuación, comprobaremos que $Q_j f$ es combinación lineal de las funciones de Haar a escala j.

Lema 3.9. *Para $f \in L^2(\mathbb{R})$, $Q_j f(x) = \sum_{I \in D_j} \langle f, h_I \rangle h_I(x)$.*

Demostración. Por definición de h_I y puesto que $|I| = 2|I_r| = 2|I_l|$,

$$\langle f, h_I \rangle h_I(x) = \frac{\sqrt{|I|}}{2} \left(\frac{1}{|I_r|} \int_{I_r} f - \frac{1}{|I_l|} \int_{I_l} f \right) h_I(x) = \frac{\sqrt{|I|}}{2} (m_{I_r} f - m_{I_l} f) h_I(x)$$

Debido a que $h_I(x) = 1/\sqrt{|I|}$ si $x \in I_r$ y $h_I(x) = -1/\sqrt{|I|}$ si $x \in I_l$

$$\langle f, h_I \rangle h_I(x) = \begin{cases} (m_{I_r} f - m_{I_l} f)/2, & \text{si } x \in I_r \\ -(m_{I_r} f - m_{I_l} f)/2, & \text{si } x \in I_l \end{cases}$$

Por ser I un intervalo diádico en D_j, para todo $x \in I$, $P_j f(x) = m_I f$. Observamos que si $x \in I_r$, entonces $P_{j+1} f(x) = m_{I_r} f$. Análogamente, si $x \in I_l$ tenemos $P_{j+1} f(x) = m_{I_l} f$. Por tanto,

$$Q_j f(x) = \begin{cases} m_{I_r} f - m_I f, & \text{si } x \in I_r \\ m_{I_l} f - m_I f, & \text{si } x \in I_l \end{cases}$$

Puesto que $m_I f = (m_{I_l} f + m_{I_r} f)/2$,

$$m_{I_r} f - m_I f = (m_{I_r} f - m_{I_l} f)/2 = m_I f - m_{I_l} f$$

Por lo que $Q_j f(x) = \sum_{I \in D_j} \langle f, h_I \rangle h_I(x)$ para $x \in I$. □

Para probar la completitud del sistema de Haar, mostraremos que para cualquier $f \in L^2(\mathbb{R})$, podemos expresar f como

$$f(x) = \sum_{I \in D} \langle f, h_I \rangle h_I(x)$$

Por el lema 3.9, esto es equivalente a

$$f(x) = \lim_{M,N \to \infty} \sum_{-M \leq j < N} Q_j f(x)$$

Observamos que

$$\sum_{M \leq j < N} Q_j f(x) = \sum_{M \leq j < N} (P_{j+1} f(x) - P_j f(x)) = P_N f(x) - P_M f(x)$$

por un argumento de sumas telescópicas. Por tanto, si logramos comprobar que

$$f(x) = \lim_{N \to \infty} P_N f(x) - \lim_{M \to -\infty} P_M f(x)$$

lograremos nuestro objetivo. Es suficiente con demostrar el siguiente teorema.

Teorema 3.10. *Para $f \in L^2(\mathbb{R})$, tenemos*

$$\lim_{M \to -\infty} ||P_M f||_2 = 0 \tag{3.1}$$

$$\lim_{N \to \infty} ||P_N f - f||_2 = 0 \tag{3.2}$$

Haremos uso del siguiente lema de análisis real:

Lema 3.11. *Una función $f : \mathbb{R} \to \mathbb{C}$ pertenece a $L^p(\mathbb{R})$ si y solo si las funciones*

$$f_k \coloneqq f\chi_{[k2^{-j},(k+1)2^{-j}]} \in L^p(\mathbb{R}) \quad y \quad \sum_{k \in \mathbb{Z}} ||f_k||_p^p < \infty$$

para cualquier escala $j \in \mathbb{Z}$. Además, $\sum_{k \in \mathbb{Z}} ||f_k||_p^p = ||f||_p^p$.

Demostración. Una función $f \in L^p(\mathbb{R})$ si $\int_{\mathbb{R}} |f(x)|^p dx < +\infty$. Fijamos j y seccionamos la recta real en intervalos $[k2^{-j}, (k+1)2^{-j}]$ para $k \in \mathbb{Z}$. Obtenemos

$$||f||_p^p = \int_{\mathbb{R}} |f(x)|^p dx = \sum_{k=-\infty}^{\infty} \int_{k2^{-j}}^{(k+1)2^{-j}} |f(x)|^p dx = \sum_{k \in \mathbb{Z}} ||f_k||_p^p.$$

Podemos deducir el enunciado de esta igualdad, ya que $f \in L^p(\mathbb{R})$ si y solo si todos los términos son finitos. □

Veremos que 3.10 es consecuencia de los siguientes lemas.

Lema 3.12. *Los operadores P_j están acotados uniformemente en $L^2(\mathbb{R})$. En particular, para cada $f \in L^2(\mathbb{R})$ y cada entero j,*

$$||P_j f||_2 \leq ||f||_2$$

Demostración. En primer lugar, acotamos para $x \in I \in D_j$

$$|P_j f(x)|^2 = \left| \frac{1}{|I|} \int_I f(x)dx \right|^2 \leq \frac{1}{|I|^2} \int_I 1^2 dt \int_I |f(t)|^2 dt = \frac{1}{|I|} \int_I |f(t)|^2 dt.$$

Donde hemos utilizado la desigualdad de Cauchy-Schwarz. Integrando sobre I,

$$\int_I |P_j f(x)|^2 dx \leq \int_I |f(t)|^2 dt.$$

Si sumamos sobre todos los intervalos diádicos D_j y aplicamos el lema 3.11,

$$\int_{\mathbb{R}} |P_j f(x)|^2 dx = \sum_{I \in D_j} \int_I |P_j f(x)|^2 dx \leq \sum_{I \in D_j} \int_I |f(t)|^2 dt = \int_{\mathbb{R}} |f(t)|^2 dt$$

ya que los intervalos diádicos son una partición de la recta real. □

Lema 3.13. *Sea $g \in L^2(\mathbb{R})$, g continua, y con soporte compacto en el intervalo $[-K, K]$, entonces se satisfacen las igualdades*

$$\lim_{M \to -\infty} ||P_M g||_2 = 0 \tag{3.3}$$

$$\lim_{N \to \infty} ||P_N g - g||_2 = 0 \tag{3.4}$$

Demostración. Veamos (3.3). Sea g continua y con soporte compacto en $[-K, K]$, escogemos j tal que $K < 2^j$. Si $x \in [0, 2^j) \subset D_j$, entonces $|P_{-j} g(x)| = \frac{1}{2^j} \int_0^K |g(t)|dt$. Aplicando la desigualdad de Cauchy-Schwartz,

$$|P_{-j} g(x)| = \frac{1}{2^j} \int_0^K |g(t)|dt \leq \frac{1}{2^j} \left(\int_0^K 1^2 dt \right)^{1/2} \left(\int_0^K |g(t)|^2 dt \right)^{1/2} \leq \frac{\sqrt{K}}{2^j} ||g||_2$$

Análogamente, tenemos la misma desigualdad para $x < 0$. Si $|x| \geq 2^j$, entonces $P_{-j} g(x) = 0$, ya que el intervalo en D_j que contiene x es disjunto con el soporte de g. Observamos que

$$||P_{-j} g||_2^2 = \int_{-2^j}^{2^j} |P_{-j} g(x)|^2 dx \leq \frac{1}{2^{2j}} K ||g||_2^2 \int_{-2^j}^{2^j} 1 dx = 2^{-j+1} K ||g||_2^2$$

nos da una cota de la norma L^2 de $P_{-j} g$. Si escogemos un N suficientemente grande, tenemos que $2^{-N+1} K ||g||_2^2 < \varepsilon$. Es decir, dado $\varepsilon > 0$, existe un $N > 0$ tal que

$$||P_{-j} g||_2 < \varepsilon$$

para todo $j > N$. Esto prueba la expresión (3.3).
Para ver (3.4), observamos que bajo la hipótesis del enunciado, g es continua con soporte compacto en $[-K, K] \subset [-2^{M_0}, 2^{M_0}]$. Esto implica que g es uniformemente continua, y por definición, dado $\varepsilon > 0$, existe $\delta > 0$ tal que

$$|g(y) - g(x)| < \varepsilon / \sqrt{2^{M_0+1}} \quad \text{si } |y - x| < \delta.$$

Escogemos ahora $N > M_0$ que satisfaga $2^{-j} < \delta$ para todo $j > N$. Puesto que D_j es una partición de $\mathbb{R}$, cada x pertenece a un único intervalo $I \in D_j$, con $|I| = 2^{-j} < \delta$. Por tanto, $|y - x| \leq \delta$ para todo $y \in I$, y

$$|P_j g(x) - g(x)| \leq \frac{1}{|I|} \int_I |g(y) - g(x)|dy \leq \varepsilon / \sqrt{2^{M_0+1}}$$

Elevando al cuadrado e integrando sobre $\mathbb{R}$,

$$\int_{\mathbb{R}} |P_j g(x) - g(x)|^2 dx = \int_{|x|\leq 2^{M_0}} |P_j g(x) - g(x)|^2 dx < \frac{\varepsilon^2}{2^{M_0+1}} \int_{|x|\leq 2^{M_0}} 1 dx = \varepsilon^2.$$

Observa que si $|x| > 2_0^M$, entonces para $j > N \geq M_0$, $P_j g(x)$ es la media sobre un intervalo $I \in D_j$ que esté completamente fuera del soporte de g. Para cada x y j, $P_j g(x) = 0$, y por tanto la integral es nula a partir de $|x| > 2^{M_0}$. Hemos probado que, dado $\varepsilon > 0$, existe un $N > 0$ tal que para todo $n > N$,

$$||P_j g - g||_2 \leq \varepsilon.$$

Esto demuestra la ecuación (3.4) para funciones continuas con soporte compacto. □

Haremos uso del siguiente resultado de análisis funcional:

Teorema 3.14. *Las funciones continuas en un intervalo cerrado acotado I son densas en $L^p(I)$, para $1 \leq p \leq \infty$.*

Podemos encontrar una demostración de este teorema en [Rud87].

Lema 3.15. *Las funciones continuas con soporte compacto son densas en $L^2(\mathbb{R})$. En otras palabras, dado $f \in L^2(\mathbb{R})$, para cualquier $\varepsilon > 0$ existen funciones g y h tales que $f = g + h$, donde g es una función continua con soporte compacto en un intervalo $[-K, K]$ y $h \in L^2(\mathbb{R})$ es una función tal que $||h||_2 < \varepsilon$.*

Demostración. Escogemos K suficientemente grande tal que

$$||f\chi_{\{x\in\mathbb{R}:|x|>K\}}||_2 \leq \varepsilon/3$$

es decir, la cola de f tiene norma L^2 menor que $\varepsilon/3$. Haciendo uso del teorema 3.14, las funciones continuas son densas en $L^2([-K, K])$. Podemos escoger g_1 continua en $[-K, K]$ tal que

$$||(f - g_1)\chi_{[-K,K]}||_2 \leq \varepsilon/3.$$

Notamos que puede ocurrir que g_1 sea continua en $[-K, K]$, pero cuando la extendamos a la recta real con $g_1(x) = 0$ si $x \notin [-K, K]$ no sea continua. Podemos definir g tal que

$$g(x) \coloneqq \begin{cases} g_1(x), & \text{si } x \in [-K + \delta, K - \delta] \\ 0, & \text{si } x \notin [-K, K] \\ \frac{1}{\delta}(Kg_1(-K + \delta) + g_1(-K + \delta)x), & \text{si } x \in [-K, -K + \delta) \\ \frac{1}{\delta}(Kg_1(K - \delta) - g_1(K - \delta)x), & \text{si } x \in (K - \delta, K] \end{cases}$$

De esta manera, g es continua en $\mathbb{R}$. Escogemos δ tal que $||g_1 - g||_2 \leq \varepsilon/3$. Consideramos

$$h = f - g = f\chi_{\{x\in\mathbb{R}:|x|>K\}} + (f - g_1)\chi_{[-K,K]} + (g_1 - g)\chi_{[-K,K]}.$$

Por la desigualdad triangular,

$$||h||_2 \leq ||f\chi_{\{x\in\mathbb{R}:|x|>K\}}||_2 + ||(f - g_1)\chi_{[-K,K]}||_2 + ||(g_1 - g)\chi_{[-K,K]}||_2 \leq \varepsilon$$

□

Haremos uso del hecho de que los operadores P_j son lineales.

Proposición 3.16. *Los operadores P_j son lineales.*

Demostración. Es inmediato a partir de la linealidad de la integración. □

Completitud del sistema de Haar. Por el lema 3.15, dado $\varepsilon > 0$, podemos expresar f como $f = g + h$, donde g es continua con soporte compacto en $[-K, K]$ y $h \in L^2(\mathbb{R})$ con $||h||_2 < \varepsilon/4$. Por el lema 3.13, podemos escoger $N \in \mathbb{N}$ suficientemente grande tal que, para cualquier $j > N$,

$$||P_{-j}g||_2 \leq \varepsilon/2$$

Como los operadores promedio P_j son operadores lineales, $P_j(g+h) = P_jg + P_jh$. Por la desigualdad triangular y por el lema 3.12,

$$||P_{-j}f||_2 = ||P_{-j}g + P_{-j}h||_2 \leq ||P_{-j}g||_2 + ||P_{-j}h||_2 \leq \varepsilon/2 + ||h||_2 \leq \varepsilon$$

Esto prueba la primera ecuación del teorema 3.10.
Haciendo uso del mismo argumento, por el lema 3.13, escogiendo $N \in \mathbb{N}$ suficientemente grande tal que

$$||P_jg - g||_2 \leq \varepsilon/2$$

Haciendo uso de la desigualdad triangular y el lema 3.12,

$$||P_jf - f||_2 = ||P_jg + P_jh - g - h||_2 \leq ||P_jg - g||_2 + ||P_jh - h||_2 \leq \frac{\varepsilon}{2} + 2||h||_2 \leq \varepsilon$$

Esta última igualdad prueba la segunda ecuación del teorema 3.10. □

3.3. Definición de Wavelet

En el apartado anterior hemos estudiado el sistema de Haar y comprobado que las funciones de Haar forman una base ortonormal de $L^2(\mathbb{R})$. Sin embargo, estas funciones son discontinuas y no diferenciables, por lo que no son suficientemente buenas en la aproximación de señales mínimamente regulares. En concreto, para nuestra aplicación en imágenes necesitaremos funciones más suaves. Podemos generalizar el concepto tras estas funciones y definir las wavelets.

De manera informal, serán funciones localizadas en el tiempo cuyas dilataciones y traslaciones nos permitan expresar señales de una manera más conveniente. Nos daremos cuenta rápidamente de que hallar este tipo de funciones no es en absoluto inmediato.

Definición 3.17. Una función $\psi \in L^2(\mathbb{R})$ es una *wavelet* si la familia de funciones

$$\psi_j^k(x) = 2^{j/2}\psi(2^jx - k) \quad \text{para } j, k \in \mathbb{Z}$$

forma una base ortonormal en $L^2(\mathbb{R})$. Es decir, $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ es una familia ortonormal que satisface

$$\Big|\Big|f - \sum_{j,k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \psi_{j,k}\Big|\Big|_2 \to 0 \quad \text{para todo } f \in L^2(\mathbb{R}).$$

Hemos visto un ejemplo en el que obtenemos una función que satisface la definición:

Definición 3.18 (Wavelet de Haar)**.** La función de Haar (3.3) con intervalo $I = [0, 1]$ es una wavelet. La llamaremos *wavelet de Haar*. Explícitamente,

$$h(x) \coloneqq -\chi_{[0,1/2]}(x) + \chi_{[1/2,1]}(x).$$

Como hemos visto en el apartado anterior, la familia $h_{j,k}(x) := \{2^{j/2}h(2^j x - k)\}_{j,k}$ es una base ortonormal de $L^2(\mathbb{R})$.

4. MRA ortogonal

Como hemos comentado, hallar wavelets mediante la definición (3.17) no es un problema sencillo. A finales del siglo pasado, Meyer y Mallat desarrollaron un esquema general para hallar wavelets. Este método utiliza como base el Análisis de Multiresolución Ortogonal (MRA ortogonal) [Mal99].

Gracias a este desarrollo hoy en día disponemos de una librería extensa de wavelets, que utilizaremos en la parte práctica de este trabajo. Pese a que en este trabajo no desarrollaremos métodos para hallar wavelets, el concepto de MRA ortogonal sigue siendo central ya que lo utilizaremos para justificar el algoritmo de la *Fast Wavelet Transform*.

Formalizamos la definición de análisis de multiresolución ortogonal del espacio $L^2(\mathbb{R})$.

Definición 4.1 (MRA ortogonal)**.** Un *MRA ortogonal* con función de escalado φ es una colección de subespacios cerrados $\{V_j\}_{j\in\mathbb{Z}}$ de $L^2(\mathbb{R})$ tal que

1. $V_j \subset V_{j+1}$ para todo $j \in \mathbb{Z}$
2. $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$
3. $\bigcup_{j\in\mathbb{Z}} V_j$ es denso en $L^2(\mathbb{R})$
4. $f(x) \in V_j \iff f(2x) \in V_{j+1}$
5. $\varphi \in V_0$ y $\{\varphi(x-k)\}_{k\in\mathbb{Z}}$, forman una base ortonormal de V_0.

Comentamos esta definición en detalle. En primer lugar, los espacios V_j son los denominados *espacios de aproximación*. Estos subespacios albergan las funciones que utilizaremos para aproximar las señales en cada escala $j \in \mathbb{Z}$. Observamos que, por la primera propiedad, están anidados y que cuanto mayor sea j más "fina" es esta aproximación.

La segunda propiedad nos indica que la intersección de los espacios de aproximación para todas las escalas es trivial. La tercera propiedad impone que la unión de los espacios de aproximación para todas las escalas sea densa en $L^2(\mathbb{R})$, es decir, que cualquier función de $L^2(\mathbb{R})$ se pueda aproximar por funciones en el espacio de aproximación. La cuarta propiedad nos proporciona un mecanismo para movernos entre escalas.

Finalmente, se relaciona la *función de escalado* φ con los subespacios de aproximación. Esta función pertenece al espacio de aproximación central V_0 y sus traslaciones forman una base ortonormal de este espacio. En la práctica, las funciones φ codificarán la información de las aproximaciones de la señal para cada escala. Observamos además que una función de escalado φ determina completamente los subespacios anidados, ya que el espacio V_0 se define en la propiedad (5) y la propiedad (4) nos permite movernos a través de escalas. Cuando hallamos esta función de escalado, sus traslaciones y dilataciones forman una base ortonormal de V_j.

4.1. MRA ortogonal de Haar

A continuación, ilustramos el concepto de MRA ortogonal con el ejemplo de Haar. Comprobamos que satisface todas las propiedades de la definición.

Proposición 4.2. *La función de escalado $\varphi(x) = \chi_{[0,1]}(x)$ junto con los espacios de aproximación V_j constituidos por funciones "step" asociadas a intervalos diádicos $I_j^k = [k2^{-j}, (k+1)2^{-j})$ dan lugar a un MRA ortogonal.*

Antes de probar la proposición, damos la siguiente definición:

Definición 4.3 (Función step)**.** Diremos que una función h definida en I es una *función step* si existe una partición P de I y coeficientes reales $\{a_J\}_{J\in P}$ tales que

$$h(x) = \sum_{J\in P} a_J \chi_J(x),$$

donde $\chi_J(x)$ es la función característica en el intervalo J.

En concreto, haremos referencia a las funciones step asociadas a intervalos diádicos, donde $P = D_j$ para cierta $j \in \mathbb{Z}$.

Demostración. Comprobamos las propiedades de la definición 4.1.

1. Sea $h \in V_j$, podemos expresar h como

$$\begin{aligned} h(x) &= \sum_{k\in\mathbb{Z}} a_{I_j^k} \chi_{I_j^k}(x) = \sum_{k\in\mathbb{Z}} a_{I_j^k} \left(\chi_{I_{j+1}^{2k}}(x) + \chi_{I_{j+1}^{2k+1}}(x) \right) \\ &= \sum_{k\in\mathbb{Z}} a_{I_j^k} \chi_{I_{j+1}^{2k}}(x) + \sum_{k\in\mathbb{Z}} a_{I_j^k} \chi_{I_{j+1}^{2k+1}}(x) \end{aligned}$$

   Donde hemos hecho uso de la propiedad de los intervalos diádicos $I_j^k = I_{j+1}^{2k} \cup I_{j+1}^{2k+1} = I_l \cup I_r$.

2. Si $f \in \bigcap_{j\in\mathbb{Z}} V_j$, f es constante en todos los intervalos diádicos y de cuadrado integrable. Está claro que la única función constante que satisface esto es $f \equiv 0$.

3. Hemos visto en el teorema 3.5 que las funciones de Haar $\{h_I\}_{I\in D}$ forman una base ortonormal de $L^2(\mathbb{R})$. En concreto, estas funciones son densas en $L^2(\mathbb{R})$. Suponemos que tenemos $f \in L^2(\mathbb{R})$, entonces podemos expresar f como $f(x) = \sum_{I\in D} \langle f, h_I \rangle h_I(x)$ donde I es un intervalo diádico. Si I es de la generación j, entonces $h_I \in V_{j+1}$, así pues f se puede expresar como suma de funciones de $\bigcup_{j\in\mathbb{Z}} V_j$.

4. Observamos que $\chi_{I_j^k}(x) = \chi_{I_{j+1}^k}(2x)$. Con la definición de step function asociada a intervalos diádicos obtenemos el resultado.

5. Claramente $\varphi(x) = \chi_{[0,1]}(x) \in V_0$. Notamos que

$$\varphi(x-k) = \chi_{[0,1]}(x-k) = \chi_{[k,k+1]}(x).$$

   El conjunto $\{\chi_{[k,k+1]}(x)\}_{k\in\mathbb{Z}}$ genera el espacio V_0 debido a que V_0 es el espacio de funciones step escalonadas en los intervalos $[k, k+1)$. Los elementos de este conjunto son ortogonales dos a dos: está claro que, dado $x \in \mathbb{R}$, $\langle \chi_{[k_1,k_1+1)}(x), \chi_{[k_2,k_2+1)}(x) \rangle = 0$, ya que al menos uno de los elementos es nulo. Puesto que la norma de la función característica en $\mathbb{R}$ es uno, la base es ortonormal.

□

5. Teorema de Mallat

En esta sección veremos que una vez tenemos una función de escalado φ definida en un MRA ortogonal, podemos obtener una familia de wavelets asociada a ese MRA. Este resultado es conocido como el teorema de Mallat. Antes de eso necesitaremos definir los subespacios de detalles W_j. Los resultados que encontramos en esta sección se derivan de [PW12].

5.1. Subespacios de detalles

Nos falta una pieza clave muy relacionada con el concepto de MRA; los subespacios de detalles W_j. Estos subespacios codificarán la información necesaria para pasar de unas aproximaciones a otras y más adelante veremos que están generados por bases de wavelets.

Definición 5.1 (Subespacio de detalles)**.** Dado un MRA ortogonal con subespacios de aproximación V_j con $j \in \mathbb{Z}$, diremos que W_j es el *subespacio de detalles a escala* 2^{-j} si es el complemento ortogonal de V_j en V_{j+1}. De manera explícita,

$$W_j \coloneqq \{h \in V_{j+1} : \langle h, g\rangle = 0, \text{ para todo } g \in V_j\}.$$

Observamos que, por definición, se satisface que $V_j \perp W_j$ para todo j entero. En particular, los espacios detalle son los subespacios diferencia entre V_j y V_{j+1}, i.e.,

$$V_{j+1} = V_j \oplus W_j$$

donde $\oplus$ denota la suma directa entre subespacios ortogonales. Es decir, cualquier vector $v_{j+1} \in V_{j+1}$ puede expresarse como $v_{j+1} = v_j + w_j$ con $v_j \in V_j$, $w_j \in W_j$ y $v_j \perp w_j$.

Observamos que estos subespacios son ortogonales entre sí.

Proposición 5.2. *Si $j \neq k$, entonces $W_j \perp W_k$.*

Demostración. Sean W_j, W_k, $j \neq k$,

$$W_j \coloneqq \{h \in V_{j+1} : \langle h, g\rangle = 0, \text{ para todo } g \in V_j\}$$

$$W_k \coloneqq \{l \in V_{k+1} : \langle l, g\rangle = 0, \text{ para todo } g \in V_k\}$$

Supongamos que $j > k$. En ese caso, $V_k \subsetneq V_j$ y $V_{k+1} \subseteq V_j$. Por tanto, si $h \in W_j$ y $l \in W_k$ entonces $\langle h, l\rangle = 0$ ya que $l \in V_{k+1} \subseteq V_j$. □

A continuación, comprobamos que la suma directa de los subespacios de detalle W_j para todo j da lugar al espacio $L^2(\mathbb{R})$. Daremos una base de este espacio en función de una wavelet ψ.

Proposición 5.3. *Para cada subespacio V_j tenemos una descomposición ortogonal en términos del espacio de aproximación menos detallado V_n y los subespacios de detalle W_k de resoluciones intermedias, donde $n \leq k < j$:*

$$V_j = V_n \oplus W_n \oplus W_{n-1} \oplus \cdots \oplus W_{j-2} \oplus W_{j-1}$$

Demostración. Sabemos que $V_{j+1} = V_j \oplus W_j$. Aplicando inducción obtenemos el resultado. □

Los subespacios anidados de aproximación $\{V_j\}$ definen el MRA ortogonal y por la propiedad (3) de MRA son densos en $L^2(\mathbb{R})$. Por la proposición 5.3, los subespacios de detalle W_j nos dan la siguiente descomposición ortogonal del espacio $L^2(\mathbb{R})$:

$$L^2(\mathbb{R}) = \overline{\bigoplus_{j\in\mathbb{Z}} W_j}.$$

Mostraremos que la función de escalado φ determina completamente una wavelet ψ tal que $\{\psi(x-k)\}_{k\in\mathbb{Z}}$ es una base ortonormal del espacio de aproximación W_0. Una vez que tengamos esta wavelet ψ, tenemos que el subespacio de detalles W_j es una dilatación de W_0, y la función

$$\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$$

pertenece a W_j. Por tanto, la familia de wavelets $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ formará una base ortonormal de W_j.

Comprobamos este último resultado. Vemos en primer lugar que es familia ortonormal. Si $\psi_{j,k}, \psi_{j,k'} \in \{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$,

$$\begin{aligned}\langle \psi_{j,k}, \psi_{j,k'}\rangle &= \int_{\mathbb{R}} \psi_{j,k}(x)\psi_{j,k'}(x)dx = \int_{\mathbb{R}} 2^{j/2}\psi(2^j x - k)2^{j/2}\psi(2^j x - k')dx \\ &= \int_{\mathbb{R}} 2^j \psi(y-k)\psi(y-k')2^{-j}dy = \int_{\mathbb{R}} \psi(y-k)\psi(y-k')dy \\ &= \begin{cases} 0, & \text{si } k \neq k' \\ 1, & \text{si } k = k' \end{cases}\end{aligned}$$

donde la última integral toma estos valores por ser $\{\psi(x-k)\}_{k\in\mathbb{Z}}$ una base ortonormal.

Comprobamos a continuación que esta familia de funciones es base de W_j. Para cualquier $f \in W_0$,

$$f(x) = \sum_{k\in\mathbb{Z}} \langle f, \psi_{0,k}\rangle \psi(x-k).$$

Mostraremos que las funciones de W_j pueden expresarse en función de $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$. Sea g una función de W_j,

$$g(x) = f(2^j x) = \sum_{k\in\mathbb{Z}} \langle f, \psi_{0,k}\rangle \psi(2^j x - k),$$

ya que $f(x) \in W_0$ implica que $f(2^j x) \in W_j$ utilizando la propiedad (4) de la definición 4.1 reiteradamente. Multiplicando y dividiendo por $2^{j/2}$ la última igualdad,

$$\begin{aligned}&\sum_{k\in\mathbb{Z}} \langle f, \psi_{0,k}\rangle \frac{1}{2^{j/2}} 2^{j/2}\psi(2^j x - k) = \sum_{k\in\mathbb{Z}} 2^{-j/2}\langle f, \psi_{0,k}\rangle \psi_{j,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \left(\int_{\mathbb{R}} 2^{-j/2} f(x)\psi(x-k)dx\right)\psi_{j,k}(x) = \sum_{k\in\mathbb{Z}} \left(\int_{\mathbb{R}} 2^{-j/2} f(2^j y)\psi(2^j y - k)2^j dy\right)\psi_{j,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \left(\int_{\mathbb{R}} f(2^j y)2^{j/2}\psi(2^j y - k)dy\right)\psi_{j,k}(x) = \sum_{k\in\mathbb{Z}} \left(\int_{\mathbb{R}} g(y)\psi_{j,k}(y)dy\right)\psi_{j,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \langle g, \psi_{j,k}\rangle \psi_{j,k}(x),\end{aligned}$$

donde hemos hecho el cambio de variable $x = 2^j y$.

Es decir, dada una función de escalado tendremos bases ortonormales de wavelets para todas las escalas $j \in \mathbb{Z}$. Si tenemos en cuenta todos los índices enteros de todas las posibles escalas, la familia de funciones $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ es una base ortonormal del espacio $L^2(\mathbb{R})$.

5.2. Demostración del teorema de Mallat

Teorema 5.4 (Teorema de Mallat)**.** *Dado un MRA ortogonal con función de escalado φ, existe una wavelet $\psi \in L^2(\mathbb{R})$ tal que para cada $j \in \mathbb{Z}$, la familia $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ es una base ortonormal para W_j. Esto implica que $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ es una base ortonormal de $L^2(\mathbb{R})$.*

Serán necesarios varios resultados previos a la demostración de este teorema. Haremos especial énfasis en conceptos relevantes en las secciones posteriores, como la ecuación de dilatación y los coeficientes de filtro.

Consideramos la función de escalado de un MRA ortogonal. Por la propiedad (1) de la definición 4.1, $\varphi \in V_0 \subset V_1$. Además, $\varphi(2x) \in V_1$ por la propiedad (4), y $\{\varphi(x-k)\}_{k\in\mathbb{Z}}$ forma una base de V_0 por (5). Observamos que $\{\varphi(2x-k)\}_{k\in\mathbb{Z}}$ es una base del espacio V_1. Esto se debe a que si tenemos una función $f(2x) \in V_1$, entonces $f(x)$ se puede expresar como combinación lineal de $\{\varphi(x-k)\}_{k\in\mathbb{Z}}$, y $f(2x)$ a su vez puede expresarse como combinación lineal de $\{\varphi(2x-k)\}_{k\in\mathbb{Z}}$. Añadiendo $\sqrt{2}$ a esta base, obtenemos la base ortonormal $\{\sqrt{2}\varphi(2x-k)\}_{k\in\mathbb{Z}}$.

Proposición 5.5. *Sea φ la función de escalado de un MRA ortogonal. Existen coeficientes $\{h_k\}_{k\in\mathbb{Z}}$ tales que $\sum_{k\in\mathbb{Z}} |h_k|^2 < \infty$ para los que se satisface*

$$\varphi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \varphi(2x-k). \tag{5.1}$$

Demostración. Puesto que las funciones $\varphi_{1,k}(x) = \{\sqrt{2}\varphi(2x-k)\}_{k\in\mathbb{Z}}$ forman base ortonormal de V_1, φ puede expresarse como

$$\varphi(x) = \sum_{k\in\mathbb{Z}} \langle \varphi, \varphi_{1,k}(x)\rangle \varphi_{1,k}(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \varphi(2x-k)$$

donde $h_k \coloneqq \langle \varphi, \varphi_{1,k}\rangle$. Además, $\sum_{k\in\mathbb{Z}} |h_k|^2 = ||\varphi||_2^2 < \infty$. □

A raíz de la proposición anterior obtenemos las siguientes definiciones:

Definición 5.6. Definimos los *coeficientes de filtro* o *filtro paso bajo* h_k de una wavelet como los coeficientes de la función de escalado φ en la base V_1, es decir $h_k \coloneqq \langle \varphi, \varphi_{1,k}\rangle$.

Definición 5.7. Definimos los *filtro paso alto* g_k de una wavelet como los coeficientes de ψ en la base V_1, es decir $g_k \coloneqq \langle \psi, \varphi_{1,k}\rangle$.

Definición 5.8. Llamaremos *ecuación de dilatación* o *escalado* a la expresión (5.1).

Observamos que esta ecuación expresa la relación entre la función de escalado y sus dilataciones en otras escalas. Los coeficientes de filtro serán clave en la aplicación práctica, ya que determinan completamente la wavelet y podemos calcular utilizando solamente estos filtros. Para demostrar el teorema de Mallat haremos uso de herramientas de análisis de Fourier.

Definición 5.9. Si pasamos la ecuación de dilatación (5.1) al lado de Fourier, gracias a las propiedades de los preliminares (2.2) y (2.3), obtenemos

$$\hat{\varphi}(\xi) = H(\xi/2)\hat{\varphi}(\xi/2). \tag{5.2}$$

Donde $H(\xi) = (1/\sqrt{2})\sum_{k\in\mathbb{Z}} h_k e^{-2\pi ik\xi}$ son los coeficientes de filtro en Fourier, llamados *filtros paso bajo*. Observamos que $H(\xi)$ es una función de periodo 1 de $L^2(\mathbb{R})$.

Son de especial interés los MRAs cuyos coeficientes h_k son nulos a excepción de ciertos $k \in \{0, \dots, L-1\}$. En estos casos, diremos que el filtro H tiene longitud L. Estos filtros se conocen comúnmente en ingeniería como FIRs, *Finite Impulse Response*. Por conveniencia, solamente estudiaremos wavelets asociadas a FIRs, que corresponden a funciones de escalado con soporte compacto y se pueden implementar computacionalmente. Obtenemos así el polinomio trigonométrico

$$H(\xi) = (1/\sqrt{2})\sum_{k=0}^{L-1} h_k e^{-2\pi ik\xi}.$$

Por el mismo razonamiento que en la definición 5.9, para cualquier $f \in V_1$, debe existir una función de periodo 1, $m_f(\xi) \in L^2([0,1])$ tal que

$$\hat{f}(\xi) = m_f(\xi/2)\hat{\varphi}(\xi/2). \tag{5.3}$$

En el caso de $\hat{f} \equiv \hat{\varphi}$, la función de periodo 1 m_f corresponde al filtro paso bajo H.

Lema 5.10. *Sea $f \in L^2(\mathbb{R})$. La familia de traslaciones por enteros de f es ortonormal si y solo si*

$$\sum_{n\in\mathbb{Z}} |\hat{f}(\xi+n)|^2 = 1 \ \textit{a.e.}\ \xi \in \mathbb{R}$$

Demostración. Consideramos la familia de traslaciones de f $\{f_{0,k} = \tau_k f\}_{k\in\mathbb{Z}}$, donde $\tau_k f(x) \coloneqq f(x-k)$. Observamos que $\langle \tau_k f(x), \tau_m f(x)\rangle = \langle \tau_{k-m} f(x), f(x)\rangle$ haciendo un cambio de variable. Queremos mostrar que

$$\langle \tau_k f, f\rangle = \delta_k \text{ para todo } k \in \mathbb{Z}.$$

Pasando al lado de Fourier,

$$\delta_k = \langle \widehat{\tau_k f}, \widehat{f}\rangle = \int_{\mathbb{R}} \widehat{\tau_k f}(\xi)\overline{\hat{f}(\xi)}d\xi = \int_{\mathbb{R}} e^{-2\pi ik\xi}\hat{f}(\xi)\overline{\hat{f}(\xi)}d\xi = \int_{\mathbb{R}} e^{-2\pi ik\xi}|\hat{f}(\xi)|^2 d\xi$$

donde la primera igualdad es consecuencia de que la transformada de Fourier conserva el producto escalar, y en la tercera igualdad hemos utilizado que la transformada de Fourier de $\tau_k f$ es una modulación de $\hat{f}$. Por la aditividad de la integral y haciendo el cambio de variable $\eta = \xi - n$:

$$\delta_k = \sum_{n\in\mathbb{N}} \int_n^{n+1} e^{-2\pi ik\xi}|\hat{f}(\xi)|^2 d\xi = \int_0^1 e^{-2\pi ik\xi} \sum_{n\in\mathbb{N}} |\hat{f}(\eta+n)|^2 d\eta.$$

Con esta última igualdad tenemos que la función periódica de periodo uno dada por $F(\eta) = \sum_{n\in\mathbb{Z}} |\hat{f}(\eta+n)|^2$ tiene el k-ésimo coeficiente de Fourier igual a la delta de Kronecker δ_k. Por tanto debe de ser 1 casi en todas partes. □

Lema 5.11. *Sea φ la función de escalado de un MRA ortogonal. Entonces,*

$$\sum_{n\in\mathbb{Z}} |\hat{\varphi}(\xi+n)|^2 = 1 \tag{5.4}$$

Para casi todo $\xi \in \mathbb{R}$.

Demostración. El enunciado se satisface por el lema 5.10, ya que la familia de traslaciones por enteros de φ forma una familia ortonormal. □

A continuación, introducimos una condición necesaria sobre los filtros paso bajo conocida como la *quadrature mirror filter* (QMF). Esta propiedad nos permite la reconstrucción exacta de un par de filtros y la necesitaremos para la prueba de Mallat.

Lema 5.12. *Dado un MRA Ortogonal con función de escalado φ y filtros paso bajo H, que asumimos que es un polinomio trigonométrico. Entonces, para todo $\xi \in \mathbb{R}$,*

$$|H(\xi)|^2 + |H(\xi+1/2)|^2 = 1 \tag{5.5}$$

Demostración. Insertando la ecuación (5.2) en la ecuación (5.4),

$$1 = \sum_{n\in\mathbb{Z}} |\hat{\varphi}(\xi+n)|^2 = \sum_{n\in\mathbb{N}} |H((\xi+n)/2)|^2 |\hat{\varphi}((\xi+n)/2)|^2.$$

Si separamos la suma en pares e impares y utilizamos el hecho de que H es de periodo uno, obtenemos

$$|H(\xi/2)|^2 \sum_{k\in\mathbb{Z}} |\hat{\varphi}(\xi/2+k)|^2 + |H(\xi/2+1/2)|^2 \sum_{k\in\mathbb{Z}} |\hat{\varphi}((\xi+1)/2+k)|^2$$

Aplicando (5.4) a ambas sumas, la última expresión es equivalente a

$$|H(\xi/2)|^2 + |H(\xi/2+1/2)|^2 = 1$$

para casi todo $\xi \in \mathbb{R}$. Puesto que H es un polinomio trigonométrico, H es continuo y se satisface para todo $\xi \in \mathbb{R}$. □

Lema 5.13. *Si H y G son funciones periódicas de periodo uno, entonces la función*

$$F(\xi) = G(\xi/2)\overline{H(\xi/2)} + G(\xi/2+1/2)\overline{H(\xi/2+1/2)} \tag{5.6}$$

también es de periodo uno.

Demostración. Vemos que $F(\xi+1) = F(\xi)$,

$$\begin{aligned} F(\xi+1) &= G((\xi+1)/2)\overline{H((\xi+1)/2)} + G((\xi+1)/2+1/2)\overline{H((\xi+1)/2+1/2)} \\ &= G(\xi/2+1/2)\overline{H(\xi/2+1/2)} + G(\xi/2+1)\overline{H(\xi/2+1)} \\ &= G(\xi/2+1/2)\overline{H(\xi/2+1/2)} + G(\xi/2)\overline{H(\xi/2)} = F(\xi). \end{aligned}$$

Donde hemos utilizado $G(\xi) = G(\xi+1)$ y $\overline{H(\xi)} = \overline{H(\xi+1)}$. □

Lema 5.14. *Una función $\lambda(\xi)$ es de periodo uno y satisface $\lambda(\xi+1/2) = -\lambda(\xi)$ si y solamente si $\lambda(\xi) = e^{2\pi i\xi}\sigma(\xi)$, donde $\sigma(\xi)$ es de periodo $1/2$.*

Demostración. Vemos la primera implicación. Expresamos λ en serie de Fourier

$$\lambda(\xi) = \sum_{n\in\mathbb{Z}} c_n e^{2\pi i n\xi}.$$

Desarrollando $\lambda(\xi + 1/2)$ obtenemos

$$\begin{aligned}\lambda(\xi+1/2) &= \sum_{n\in\mathbb{Z}} c_n e^{2\pi i n(\xi+1/2)} = \sum_{n\in\mathbb{Z}} c_n e^{2\pi i n\xi} e^{\pi i n} \\ &= \sum_{k\in\mathbb{Z}} c_{2k} e^{2\pi i k} e^{2\pi i (2k)\xi} + \sum_{k\in\mathbb{Z}} c_{2k+1} e^{\pi i (2k+1)} e^{2\pi i (2k+1)\xi} \\ &= \sum_{k\in\mathbb{Z}} c_{2k} e^{2\pi i (2k)\xi} + \sum_{k\in\mathbb{Z}} -c_{2k+1} e^{2\pi i (2k+1)\xi}.\end{aligned}$$

Por otro lado, el desarrollo de $-\lambda(\xi)$ es

$$-\lambda(\xi) = -\sum_{n\in\mathbb{Z}} c_n e^{2\pi i n\xi} = \sum_{k\in\mathbb{Z}} -c_2 k e^{2\pi i 2k\xi} + \sum_{k\in\mathbb{Z}} -c_{2k+1} e^{2\pi i (2k+1)\xi}$$

La condición $\lambda(\xi + 1/2) = -\lambda(\xi)$ implica que los coeficientes pares c_{2n} son nulos.

A continuación, vemos la otra implicación. Sea $\sigma(\xi)$ tal que $\sigma(\xi) = \sigma(\xi + 1/2)$,

$$\lambda(\xi+1) = e^{2\pi i(\xi+1)}\sigma(\xi+1) = e^{2\pi i\xi} e^{2\pi i}\sigma(\xi+1) = e^{2\pi i\xi}\sigma(\xi) = \lambda(\xi)$$

ya que $\sigma(\xi + 1) = \sigma(\xi + 1/2) = \sigma(\xi)$. Comprobamos que $\lambda(\xi + 1/2) = -\lambda(\xi)$,

$$\lambda(\xi+1/2) = e^{2\pi i(\xi+1/2)}\sigma(\xi+1/2) = e^{2\pi i\xi} e^{\pi i}\sigma(\xi) = -e^{2\pi i\xi}\sigma(\xi) = -\lambda(\xi).$$

□

Lema 5.15. *Una función $f \in W_0$ si y solo si existe una función $v(\xi)$ de periodo uno tal que*

$$\hat{f}(\xi) = e^{\pi i\xi} v(\xi)\overline{H(\xi/2+1/2)}\hat{\varphi}(\xi/2)$$

Demostración. Suponemos que $f \in W_0$, y por tanto $f \in V_1$ y $f \perp V_0$. Observamos que, por la expresión (5.3), si comprobamos que

$$m_f(\xi) = e^{2\pi i\xi}\sigma(\xi)\overline{H(\xi+1/2)},$$

donde $\sigma(\xi)$ es una función de periodo $1/2$, habremos visto el enunciado. En este caso, $v(\xi) = \sigma(\xi/2)$ será una función de periodo uno.

Puesto que $f \perp V_0$, $\langle f, \varphi_{0,k}\rangle = 0$ para todo k entero obtenemos

$$0 = \langle \hat{f}, \widehat{\varphi_{0,k}}\rangle = \int_{\mathbb{R}} \hat{f}(\xi)\overline{\widehat{\varphi_{0,k}}(\xi)}d\xi = \int_{\mathbb{R}} \hat{f}(\xi)\overline{\widehat{\varphi}(\xi-k)}d\xi = \int_{\mathbb{R}} e^{2\pi ik\xi}\hat{f}(\xi)\overline{\widehat{\varphi}(\xi)}d\xi.$$

Haciendo uso de las expresiones (5.3) y (5.2) respectivamente,

$$\begin{aligned}0 &= \int_{\mathbb{R}} e^{2\pi ik\xi} m_f(\xi/2)\hat{\varphi}(\xi/2)\overline{\widehat{\varphi}(\xi)}d\xi = \int_{\mathbb{R}} e^{2\pi ik\xi} m_f(\xi/2)\hat{\varphi}(\xi/2)\overline{H(\xi/2)\widehat{\varphi}(\xi/2)}d\xi \\ &= \int_{\mathbb{R}} e^{2\pi ik\xi} m_f(\xi/2)\overline{H(\xi/2)}|\widehat{\varphi}(\xi/2)|^2 d\xi.\end{aligned}$$

A continuación, integramos la expresión anterior sobre $\mathbb{R}$ como suma de los intervalos $[n, n+1)$, hacemos un cambio de variable para obtener una correspondencia con los intervalos unidad, y obtenemos

$$\begin{aligned} 0 &= \sum_{n\in\mathbb{N}} \int_n^{n+1} e^{2\pi i k\xi} m_f(\xi/2)\overline{H(\xi/2)}|\widehat{\varphi}(\xi/2)|^2 d\xi \\ &= \int_0^1 e^{2\pi i k\zeta} \sum_{n\in\mathbb{N}} m_f((\zeta+n)/2)\overline{H((\zeta+n)/2)}|\widehat{\varphi}((\zeta+n)/2)|^2 d\zeta. \end{aligned}$$

Donde hemos utilizado la periodicidad de la exponencial al aplicar el cambio de variable $\zeta = \xi + n$. Notamos que las funciones m_f y H son de periodo 1. Queremos utilizar su periodicidad, pero estamos añadiendo $1/2$. Separando las sumas en pares e impares,

$$\begin{aligned} 0 &= \int_0^1 e^{2\pi i k\zeta} [m_f(\zeta/2)\overline{H(\zeta/2)} \sum_{m\in\mathbb{Z}} |\widehat{\varphi}(\zeta/2+m)|^2 \\ &\quad + m_f(\zeta/2+1/2)\overline{H(\zeta/2+1/2)} \sum_{m\in\mathbb{Z}} |\widehat{\varphi}((\zeta+1)/2+m)|^2] d\zeta \\ &= \int_0^1 e^{2\pi i k\zeta} [m_f(\zeta/2)\overline{H(\zeta/2)} + m_f(\zeta/2+1/2)\overline{H(\zeta/2+1/2)}] d\zeta, \end{aligned} \tag{5.7}$$

donde hemos hecho uso de la igualdad (5.4) dos veces en el último paso. Consideramos la función

$$F(\zeta) = m_f(\zeta/2)\overline{H(\zeta/2)} + m_f(\zeta/2+1/2)\overline{H(\zeta/2+1/2)}.$$

Por el lema 5.13, F es de periodo uno. Puesto que la integral (5.7) es igual a 0, por definición, los coeficientes de Fourier de F son todos nulos. Por tanto, $F(\zeta)$ es la función cero casi para todo $\zeta \in \mathbb{R}$, y se satisface

$$m_f(\zeta/2)\overline{H(\zeta/2)} + m_f(\zeta/2+1/2)\overline{H(\zeta/2+1/2)} = 0 \quad \text{a.e.} \tag{5.8}$$

Esto implica que, casi para todo ζ, el vector $\vec{v} \in \mathbb{C}^2$

$$\vec{v} = (m_f(\zeta), m_f(\zeta+1/2))$$

es ortogonal al vector $\vec{w} \in \mathbb{C}^2$

$$\vec{w} = (H(\zeta), H(\zeta+1/2)).$$

Por la condición *QMF* (lema 5.12), sabemos que $\vec{w} \neq 0$. Queremos expresar el espacio generado por $\vec{v}$ en función de este vector. Sabemos que $\mathbb{C}^2$ es un espacio dos-dimensional sobre $\mathbb{C}$, y por tanto el complemento ortogonal del subespacio generado por $\vec{w}$ es de una dimensión. El vector

$$\vec{u} = (-\overline{H(\zeta+1/2)}, \overline{H(\zeta)})$$

es no nulo y ortogonal a $\vec{w}$, por lo que caracteriza el espacio de vectores ortogonales a $\vec{w}$. En particular, $\vec{v} = \lambda\vec{u} = \lambda(-\overline{H(\zeta+1/2)}, \overline{H(\zeta)})$, para $\lambda \in \mathbb{C}$. Por tanto, la función m_f de periodo uno debe cumplir

$$m_f(\zeta) = -\lambda(\zeta)\overline{H(\zeta+1/2)}, \quad m_f(\zeta+1/2) = -\lambda(\zeta)\overline{H(\zeta)}.$$

En consecuencia, $\lambda(\zeta)$ es una función de periodo uno tal que $-\lambda(\zeta+1/2) = \lambda(\zeta)$. Por el lema (5.14), $\lambda(\zeta) = e^{2\pi i\zeta}\sigma(\zeta)$, donde $\sigma(\zeta)$ es una función de periodo $1/2$. Es decir, tenemos que $f \in W_0$ si y solo si

$$m_f(\zeta) = e^{2\pi i\zeta}\sigma(\zeta)\overline{H(\zeta+1/2)}.$$

□

Ahora podemos demostrar el resultado principal 5.4 y presentar la wavelet ψ asociada al MRA ortogonal con función de escalado φ del enunciado.

Teorema de Mallat. Buscamos una función wavelet $\psi \in W_0$. Por el lema 5.15, esta ψ debe satisfacer

$$\widehat{\psi}(\xi) = m_\psi(\xi/2)\widehat{\psi}(\xi/2),$$

donde

$$m_\psi(\xi) = e^{2\pi i\xi}\sigma(\xi)\overline{H(\xi+1/2)} \tag{5.9}$$

y σ es una función de periodo $1/2$.

Queremos ver que las traslaciones por enteros de ψ son una familia ortonormal de $L^2(\mathbb{R})$. Aplicando el lema 5.10, obtenemos que para casi todo $\xi \in \mathbb{R}$,

$$|m_\psi(\xi)|^2 + |m_\psi(\xi+1/2)|^2 = 1$$

Observamos que esta expresión es análoga a la propiedad *QMF* para la función $m_\psi(\xi)$. Haciendo uso de la expresión (5.9) obtenemos

$$|\sigma(\xi)|^2|H(\xi+1/2)|^2 + |\sigma(\xi+1/2)|^2|H(\xi)|^2 = 1$$

para casi todo $\xi \in \mathbb{R}$, donde $\sigma(\xi)$ es de periodo $1/2$ y H satisface la propiedad *QMF* (5.12). Extrayendo el factor común $|\sigma(\xi)|^2 = |\sigma(\xi+1/2)|^2$ deducimos que $|\sigma(\xi)| = 1$ casi para todo $\xi \in \mathbb{R}$.

Escogemos $\sigma(\xi) \equiv 1$, de periodo $1/2$ y con $|\sigma(\xi)| = 1$ para todo ξ. Definimos la wavelet ψ en el lado de Fourier como

$$\widehat{\psi}(\xi) \coloneqq G(\xi/2)\widehat{\varphi}(\xi/2),$$

donde

$$G(\xi) \coloneqq e^{2\pi i\xi}\overline{H(\xi+1/2)}, \quad \text{tal que} \quad G(\xi) = G(\xi+1).$$

Observamos que $\psi \in W_0$ por el lema 5.15. Puesto que en el lado de Fourier las traslaciones son modulaciones,

$$\widehat{\psi_{0,k}}(\xi) \coloneqq e^{-2\pi ik\xi}G(\xi/2)\widehat{\varphi}(\xi/2),$$

por lo que $\{\psi_{0,k}\}_{k\in\mathbb{Z}} \in W_0$. Hacemos uso del lema 5.15 de nuevo, tomando $v(\xi) = e^{-2\pi ik\xi}$. Puesto que G satisface la propiedad *QMF* (5.12), concluimos que la familia $\{\psi_{0,k}\}_{k\in\mathbb{Z}}$ es una familia ortonormal en W_0.

Vemos que la familia $\{\psi_{0,k}\}_{k\in\mathbb{Z}}$ genera el espacio de detalles W_0. Sea $f \in W_0$, por el lema 5.15 existe $v \in L^2(\mathbb{R})$ de periodo uno tal que

$$\hat{f}(\xi) = v(\xi)e^{\pi i\xi}\overline{H(\xi/2+1/2)}\widehat{\varphi}(\xi/2) = v(\xi)\widehat{\psi}(\xi). \tag{5.10}$$

Podemos descomponer $v(\xi)$ como serie trigonométrica, $v(\xi) = \sum_{k\in\mathbb{Z}} a_k e^{-2\pi ik\xi}$, con $\sum_{k\in\mathbb{Z}} |a_k|^2 < \infty$. Sustituyendo en la expresión (5.10) y haciendo uso de que las modulaciones en el lado de Fourier son traslaciones,

$$\hat{f}(\xi) = \sum_{k\in\mathbb{Z}} a_k e^{-2\pi ik\xi}\widehat{\psi}(\xi) = \sum_{k\in\mathbb{Z}} a_k \widehat{\psi_{0,k}}(\xi).$$

Aplicando la transformada inversa de Fourier obtenemos

$$f(x) = \sum_{k\in\mathbb{Z}} a_k \psi_{0,k}(x).$$

Es decir, f puede expresarse como combinación lineal de traslaciones de ψ, y por tanto estas forman base de W_0. Como vimos al principio de este apartado, por la propiedad (4) de la definición de MRA ortogonal 4.1, esto implica que la familia de funciones $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ forma base de W_j. En conclusión, $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ es una base ortonormal de $L^2(\mathbb{R})$.

□

Notamos que, como consecuencia del teorema de Mallat, un MRA ortogonal queda definido unívocamente por la función de escalado φ o por sus filtros.

5.3. Condiciones necesarias para φ

Como consecuencia del teorema de Mallat, podemos reducir el problema de hallar bases de wavelets a hallar una función de escalado para un MRA Ortogonal. Este problema se puede resolver con el *cascade algorithm* [PW12]. Mediante este método podemos definir los filtros de un MRA, siempre que cumplan ciertas condiciones, y aproximar iterativamente la función de escalado φ asociada a estos filtros. No lo desarrollaremos aquí ya que se aleja de la motivación de este trabajo. Sin embargo, sí comentaremos condiciones necesarias para que φ sea función de escalado de un MRA Ortogonal.

En primer lugar, sabemos por el apartado anterior que φ tiene que satisfacer la ecuación de dilatación.

En el capítulo anterior hicimos uso de la condición QMF 5.12 para demostrar el teorema de Mallat. Esta condición es una condición necesaria que podemos utilizar para comprobar si un polinomio trigonométrico H puede ser el filtros paso bajo de un MRA ortogonal.

5.4. Ejemplos de wavelets a partir de MRAs

Como hemos visto, siempre que tengamos un MRA ortogonal con función de escalado φ existe una wavelet ψ asociada a este. A continuación, comentaremos varias wavelets obtenidas a partir de MRAs mediante este resultado y algoritmos de aproximación.

Existen distintas propiedades que se buscan en este tipo de funciones, algunas excluyentes entre ellas. Entre estas propiedades tenemos la ortogonalidad, el soporte compacto, la regularidad y los momentos de orden k nulos. La ortogonalidad nos permite hacer cálculos directos con los coeficientes. El soporte compacto es importante para los cálculos numéricos. La regularidad suaviza el error de aproximación. El soporte compacto y la regularidad están muy relacionadas; como regla general cuánto más suave es una wavelet mayor será el soporte, y no existe una wavelet C^∞ con soporte compacto. Finalmente, consideramos los *momentos de orden k nulos*:

Definición 5.16. Diremos que ψ tiene *n momentos de orden k igual a 0* si

$$\int_{\mathbb{R}} x^k \psi(x) dx = 0, \quad k = 0, 1, \ldots, n.$$

Es decir, por definición ψ es ortogonal a x^k y a sus combinaciones lineales, que son los polinomios de grado n. Esto implica que si la función original f es k veces diferenciable podemos aproximarla mediante un polinomio de Taylor p de grado k, y entonces

$$\begin{aligned}\int_{\mathbb{R}} \psi(x)f(x)dx &= \int_{\mathbb{R}} \psi(x)(f(x)-p(x)+p(x))dx \\ &= \int_{\mathbb{R}} \psi(x)(f(x)-p(x))dx + \int_{\mathbb{R}} \psi(x)p(x)dx = \int_{\mathbb{R}} \psi(x)R_n(x)dx + 0 = \varepsilon\end{aligned}$$

donde $R_n(x)$ es el resto del polinomio de Taylor p y ε es un valor pequeño. A efectos prácticos, los *momentos de orden k nulos* hacen 'pequeños' los coeficientes de las funciones bien aproximadas por polinomios. También detectan singularidades, ya que en los puntos que no son suficientemente regulares esta ε no es pequeña. Visualmente, haciendo la transformada wavelet de la función $f(x) = |x|$ con $db3$,

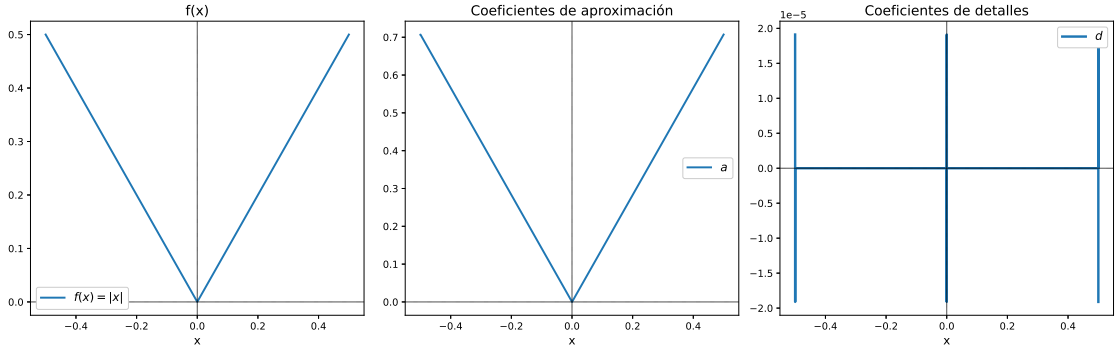
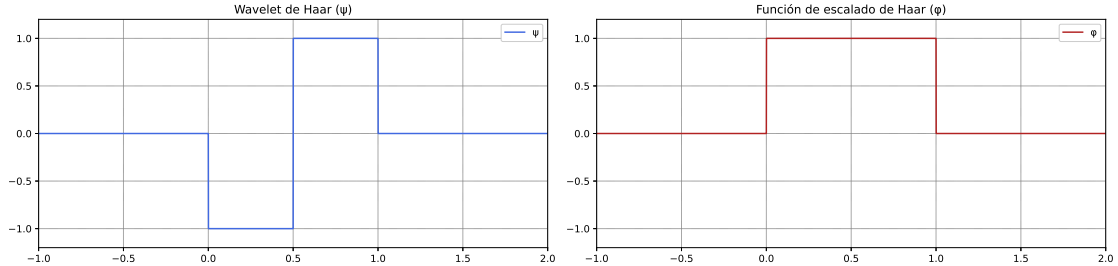


Figura 2: Ejemplo con los coeficientes wavelet calculados mediante la librería pywavelets [LWVK24].

Hemos profundizado en el ejemplo de Haar, que utilizamos para poder ilustrar la mayoría de conceptos de este trabajo por el hecho de que es el más simple. Recordamos que las funciones de escalado y la wavelet de Haar vienen dadas por:

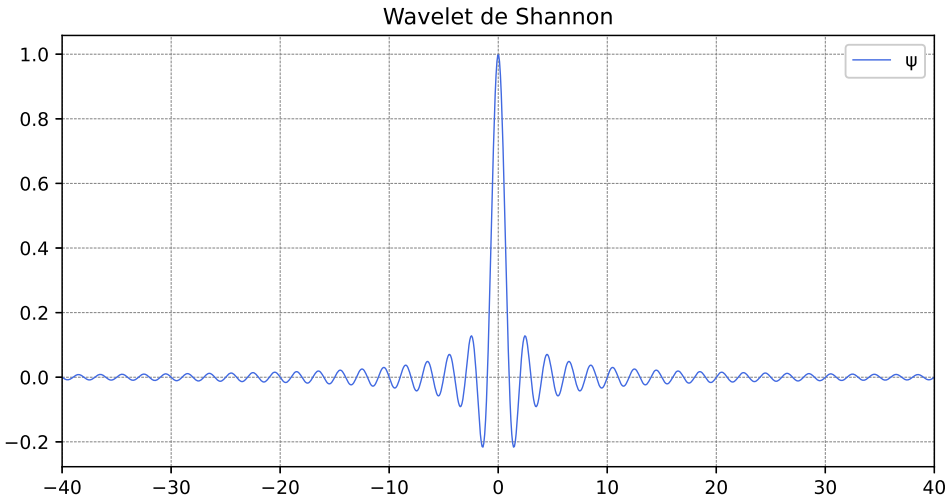
$$\varphi := \chi_{[0,1]}(x), \quad \psi := -\chi_{[0,1/2]}(x) + \chi_{[1/2,1]}(x).$$

Tenemos que la wavelet de Haar está completamente localizada en el tiempo, y de hecho tiene el soporte más corto posible dentro de las wavelets. Notamos que está poco localizada en frecuencia. Comentamos en secciones anteriores que es discontinua, hecho que hace que sea difícil utilizarla en la aproximación de funciones mínimamente regulares.



Otro ejemplo de wavelet es la wavelet de Shannon, en concreto es el otro extremo en cuanto a propiedades comparando con las funciones de Haar. Esta wavelet no tiene

soporte compacto y no está localizada en el tiempo, pero sí está completamente localizada en frecuencia. Al contrario que Haar, es infinitamente diferenciable. El hecho de no estar localizada en el tiempo es una desventaja en el análisis de imágenes y en general no es buena para las implementaciones. Necesitamos una wavelet con propiedades "intermedias" entre las dos anteriores.

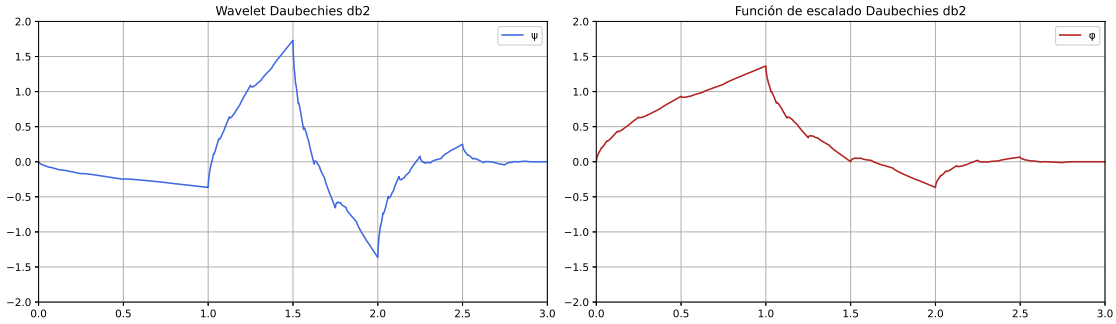


Presentamos a continuación las wavelets más utilizadas y estandard en procesamiento de imágenes; las wavelets de Daubechies. Estas funciones se pueden hallar mediante el *cascade algorithm*, y dibujar sus gráficas mediante este algoritmo, pero no podemos escribirlas explícitamente ya que no son de forma cerrada.

Para cada $N \geq 1$ existe un MRA ortogonal que genera una wavelet de este tipo. Las wavelets de Daubechies son de soporte compacto y se puede ver que es el mínimo posible por sus momentos de orden k nulos. En particular, dbN tiene N momentos de orden k igual a 0, longitud de soporte $2N$, y sus filtros tienen $2N$ coeficientes. Para $N = 2$, la wavelet de Daubechies $db1$ se corresponde con la de Haar. Caracterizamos esta familia de wavelets por sus filtros, por ejemplo para $db2$ obtenemos los filtros de paso bajo

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

Cuánto mayor es la longitud de estos filtros, más suave es la función.



6. Algoritmo de la transformada de wavelets

Supongamos que tenemos una señal representada numéricamente como un vector de muestras. Queremos utilizar la transformada de wavelets para representar estos datos como vectores de coeficientes de detalles y de aproximación en la base wavelet escogida. Existe un algoritmo optimizado para este propósito conocido como *Fast Wavelet Algorithm* (FWT). Una vez procesados los datos, necesitaremos un algoritmo de reconstrucción para ser capaces de volver al espacio inicial. Los resultados principales de esta sección pueden consultarse en [PW12] [Chu16].

Formalizamos primero la definición de transformada de wavelets.

Definición 6.1. La *transformada ortogonal de wavelets* es la aplicación $W : L^2(\mathbb{R}) \to \ell^2(\mathbb{Z}^2)$ que asigna a cada función de $L^2(\mathbb{R})$ la sucesión de sus coeficientes de wavelets

$$Wf(j,k) \coloneqq \langle f, \psi_{j,k} \rangle = \int_{\mathbb{R}} f(x)\overline{\psi_{j,k}(x)}dx.$$

Mostraremos cómo implementar la transformada de manera eficiente, haciendo uso de *filter banks*. Comprobaremos que la optimización de las operaciones proviene del hecho de que los coeficientes de filtro tienen longitud finita y fija, no muy elevada.

6.1. Filter Banks

De manera general, un *filter bank* es un conjunto de filtros (*bandpass filters*) que se utilizan para descomponer la señal en distintas componentes, cada una con una subbanda de la señal original. La idea es separar la señal en bandas de frecuencias diferentes. Estas partes pueden ser codificadas eficientemente, cada una por su cuenta, y más tarde utilizarse para reconstruir la señal.

Para optimizar la transformada de wavelets, realizamos los cálculos en función de los *filtros paso bajo* h y *filtros paso bajo* g. Veremos que el filtro h nos proporcionará las "aproximaciones" en la representación de multiresolución, mientras que utilizaremos el filtro g para obtener los "detalles" en cada escala. Uno de estos filtros por sí solo no es suficiente, ya que no son invertibles y está claro que no contienen toda la información. Es por esto que necesitaremos un *filter bank* formado por ambos.

Si un *filter bank* recupera la señal exactamente al final del proceso, diremos que es un *perfect reconstruction filter bank*. Veremos que al final del siguiente apartado conseguiremos uno de estos *filter banks*.

6.2. Fast Wavelet Transform

En esta sección presentamos el conocido algoritmo de la Fast Wavelet Transform. En primer lugar, necesitaremos definir los coeficientes de aproximación y detalles. Son estos coeficientes los que queremos calcular. Los definimos como las componentes de una señal f en los espacios de aproximación y de detalles respectivamente:

Definición 6.2. Fijada una $f \in L^2(\mathbb{R})$, definimos el *coeficiente de aproximación* $a_{j,k}$ como el producto interno de f con la función de escalado $\varphi_{j,k}$,

$$a_{j,k} \coloneqq \langle f, \varphi_{j,k} \rangle$$

Definición 6.3. Fijada una $f \in L^2(\mathbb{R})$, definimos el *coeficiente de detalles* $d_{j,k}$ como el producto interno de f con la wavelet $\psi_{j,k}$,

$$d_{j,k} \coloneqq \langle f, \psi_{j,k} \rangle$$

A continuación definimos la convolución entre dos sucesiones. Notamos que para que esta operación esté bien definida una de las sucesiones debe tener un número finito de coeficientes no nulos.

Definición 6.4. Dadas dos sucesiones $a = \{a_m\}_{m\in\mathbb{Z}}$, $b = \{b_m\}_{m\in\mathbb{Z}}$ indexadas por enteros. Su *convolución* $a * b$ es la sucesión cuyo ℓ-ésimo término viene definido mediante

$$a * b(\ell) = \sum_{m\in\mathbb{Z}} a_{\ell-m} b_m$$

Presentamos ahora el resultado que nos proporciona un método para obtener los coeficientes de aproximación y de detalles, garantizando complejidad computacional lineal. Sea $N = 2^J$ el número de muestras o datos y L la longitud del filtro:

Teorema 6.5. *Supongamos que tenemos los coeficientes de aproximación para escala J-ésima $\{a_{J,k}\}_{k=0}^{N-1}$, para una función definida en $[0,1]$, donde $2^J = N$. Entonces la aproximación más gruesa $a_{j,k}$ y los coeficientes de detalles $d_{j,k}$ para escalas $0 \le j < J$, donde $k = 0, 1, \ldots, 2^j - 1$, pueden ser calculados con un número de operaciones de orden LN.*

Demostración. Denotamos por a_j la sucesión $\{a_{j,k}\}_{k=0}^{2^j-1}$ y por dj la sucesión $\{d_{j,k}\}_{k=0}^{2^j-1}$. Observamos que queremos calcular un total de $2(1 + 2 + 2^2 + \cdots + 2^{J-1}) \sim 2^{J+1} = 2N$ coeficientes. Comprobaremos que calcular cada coeficiente $a_{j,\ell}$ y $d_{j,\ell}$ requiere un máximo de L multiplicaciones cuando tenemos los coeficientes de aproximación más finos $\{a_{j+1,k}\}_{k=0}^{2^{J+1}-1}$. Como consecuencia, la complejidad computacional del algoritmo será del orden de $\mathcal{O}(LN)$ operaciones. Consideramos primero el primer paso, calcular a_0 y d_0 dado a_1. La ecuación de escalado relaciona $\varphi_{0,\ell}(x)$ con $\{\varphi_{1,m}\}_{m\in\mathbb{Z}}$ de la siguiente manera:

$$\begin{aligned}
\varphi_{0,\ell}(x) = \varphi(x-\ell) &= \sum_{k\in\mathbb{Z}} h_k \varphi_{1,k}(x-\ell) \\
&= \sum_{k\in\mathbb{Z}} h_k \sqrt{2}\varphi(2x - (2\ell + k)) \\
&= \sum_{m\in\mathbb{Z}} h_{m-2\ell}\varphi_{1,m}(x) = \sum_{m\in\mathbb{Z}} \overline{\tilde{h}_{2\ell-m}}\varphi_{1,m}(x), \qquad (6.1)
\end{aligned}$$

donde $\tilde{h}_k \coloneqq \overline{h_{-k}}$ con h_k filtro de baso bajo y hemos hecho el cambio de índices $m = 2\ell + k$. Ahora podemos calcular a_0 en función de a_1,

$$\begin{aligned}
a_{0,\ell} = \langle f, \varphi_{0,\ell}\rangle &= \sum_{m\in\mathbb{Z}} \tilde{h}_{2\ell-m}\langle f, \varphi_{1,m}\rangle \\
&= \sum_{m\in\mathbb{Z}} \tilde{h}_{2\ell-m} a_{1,m} = \tilde{h} * a_1(2\ell). \qquad (6.2)
\end{aligned}$$

Notamos que la sucesión $\tilde{h}$ tiene la misma longitud L que la sucesión h. Por tanto, esta convolución solamente requiere L multiplicaciones.

Vemos ahora los coeficientes de detalle $d_{0,\ell}$. De manera similar, la ecuación de escalado relaciona $\psi_{0,\ell}$ con $\{\varphi_{1,m}\}_{m\in\mathbb{Z}}$. La diferencia es que ahora trabajaremos con los filtros de paso alto $\{g_k\}_{k\in\mathbb{Z}}$. Calculando de manera análoga a (6.1) obtenemos

$$\psi_{0,\ell}(x) = \sum_{m\in\mathbb{Z}} \overline{\tilde{g}_{2\ell-m}}\varphi_{1,m}.$$

Con el mismo argumento que hemos utilizado en (6.2), obtenemos que $d_{0,\ell} = \tilde{g} * a_1(2\ell)$, donde $\tilde{g}_k \coloneqq \overline{g_{-k}}$.

Vemos a continuación el caso general, para todo $j \in \mathbb{Z}$, de manera análoga a (6.1). Para cualquier $j \in \mathbb{Z}$,

$$\begin{aligned}\varphi_{j,\ell}(x) = \varphi_j(x-\ell) &= \sum_{k\in\mathbb{Z}} h_k\varphi_{j+1,k}(x-\ell) = \sum_{k\in\mathbb{Z}} h_k\sqrt{2}\varphi_j(2x-(2\ell+k)) \\ &= \sum_{m\in\mathbb{Z}} h_{m-2\ell}\sqrt{2}\varphi_j(2x-m) = \sum_{m\in\mathbb{Z}} h_{m-2\ell}\varphi_{j+1,m}(x) \end{aligned} \tag{6.3}$$

Comprobamos a continuación que se satisface la ecuación diferencia

$$\psi_{j,\ell} = \sum_{m\in\mathbb{Z}} g_{m-2\ell}\varphi_{j+1,m}. \tag{6.4}$$

Por la definición 5.7,

$$\begin{aligned}\psi_{j,\ell}(x) = \psi_j(x-\ell) &= \sum_{k\in\mathbb{Z}} g_k\varphi_{j+1,k}(x-\ell) = \sum_{k\in\mathbb{Z}} g_k\sqrt{2}\varphi_j(2x-(2\ell+k)) \\ &= \sum_{m\in\mathbb{Z}} g_{m-2\ell}\sqrt{2}\varphi_j(2x-m) = \sum_{m\in\mathbb{Z}} g_{m-2\ell}\varphi_{j+1,m}(x).\end{aligned}$$

Como consecuencia de (6.3) y (6.4), para todo $j \in \mathbb{Z}$ obtenemos

$$a_{j,\ell} = \sum_{n\in\mathbb{Z}} \tilde{h}_{2\ell-n}a_{j+1,n} = \tilde{h} * a_{j+1}(2\ell) \tag{6.5}$$

$$d_{j,\ell} = \sum_{n\in\mathbb{Z}} \tilde{g}_{2\ell-n}a_{j+1,n} = \tilde{g} * a_{j+1}(2\ell) \tag{6.6}$$

Hemos expresado todos los coeficientes de detalles y aproximaciones para la generación j como convolución de los coeficientes de aproximación de generación $j+1$ y $\tilde{h}$, $\tilde{g}$. Puesto que los filtros son de longitud L, calcular estos coeficientes requiere L multiplicaciones. □

Definimos a continuación los operadores de *downsampling* y *upsampling* siguiendo las notaciones de [Chu16].

Definición 6.6. El operador de *downsampling*, denotado por $(\downarrow 2)$, toma un vector de N componentes y los mapea a un vector con $N/2$ componentes descartando las entradas impares,

$$Ds(n) \coloneqq s(2n)$$

En concreto, el operador acorta a la mitad un vector. Aplicar el operador a un vector v equivale a multiplicar por la siguiente *matriz de downsampling*:

$$(\downarrow 2) \coloneqq \begin{bmatrix} 1 & & & & & \\ 0 & 0 & 1 & & & \\ & & 0 & 0 & 1 & \\ & & & & . & . \end{bmatrix} \tag{6.7}$$

Definición 6.7. El operador de *upsampling* $(\uparrow 2)$ toma un vector de N componentes y los mapea a un vector con $2N$ componentes intercalando ceros en las posiciones impares,

$$Us(n) \coloneqq \begin{cases} s(n/2), & \text{si } n \text{ par} \\ 0, & \text{si } n \text{ impar} \end{cases}$$

Cuando aplicamos upsampling, normalmente devolvemos los vectores a la longitud original, añadiendo ceros. Aplicar el operador a un vector es equivalente a hacer el producto por la *matriz de upsampling* $(\uparrow 2) \coloneqq (\downarrow 2)^T$.

Podemos obtener las reconstrucciones de las señales originales con un algoritmo de complejidad lineal. El siguiente resultado nos proporciona el método para las reconstrucciones de las aproximaciones y los detalles, con complejidad computacional LN.

Teorema 6.8. *La reconstrucción de los datos o muestras a nivel $j+1$ a través de los coeficientes de aproximación y de detalles del nivel j es también un algoritmo con número de operaciones de orden L. Si decidimos reconstruir, a partir de aproximaciones más gruesas a nivel M y los detalles de niveles $M \leq j < M+J$, todas las aproximaciones a escalas más finas $M < j \leq M+J$, el número de operaciones del algoritmo será de orden LN.*

Demostración. Vemos cómo pasar de los coeficientes de detalle y aproximación más gruesos a_j, d_j a los coeficientes de aproximación más finos de escala $j+1$, a_{j+1}. Recordamos que $P_{j+1}f = P_jf + Q_jf$ (definición 3.8), y en consecuencia

$$\sum_{m\in\mathbb{Z}} a_{j+1,m}\varphi_{j+1,m} = \sum_{\ell\in\mathbb{Z}} a_{j,\ell}\varphi_{j,\ell} + \sum_{\ell\in\mathbb{Z}} d_{j,\ell}\psi_{j,\ell}. \tag{6.8}$$

Podemos expresar $\varphi_{j,\ell}$ y $\psi_{j,\ell}$ en términos de $\{\varphi_{j+1,\ell}\}_{m\in\mathbb{Z}}$ a partir de (6.3) y (6.4). Aplicando estas fórmulas en el lado derecho de (6.8) obtenemos

$$\begin{aligned} \sum_{\ell\in\mathbb{Z}} a_{j,\ell}\varphi_{j,\ell} + \sum_{\ell\in\mathbb{Z}} d_{j,\ell}\psi_{j,\ell} &= \sum_{\ell\in\mathbb{Z}} a_{j,\ell} \sum_{m\in\mathbb{Z}} h_{m-2\ell}\varphi_{j+1,m} + \sum_{\ell\in\mathbb{Z}} d_{j,\ell} \sum_{m\in\mathbb{Z}} g_{m-2\ell}\varphi_{j+1,m} \\ &= \sum_{m\in\mathbb{Z}} \left[\sum_{\ell\in\mathbb{Z}} h_{m-2\ell}a_{j,\ell} + g_{m-2\ell}d_{j,\ell} \right] \varphi_{j+1,m}, \end{aligned}$$

donde hemos reagrupado los términos múltiples de $\varphi_{j+1,m}$ en la última igualdad. Por la ecuación (6.8), esto implica que

$$a_{j+1,m} = \sum_{\ell\in\mathbb{Z}} h_{m-2\ell}a_{j,\ell} + g_{m-2\ell}d_{j,\ell} \tag{6.9}$$

reescribiendo mediante la definición de convolución y upsamplings, obtenemos

$$h * Ua_j(m) + g * Ud_j(m) = \sum_{k\in\mathbb{Z}} h_{m-k}Ua_{j,k} + \sum_{k\in\mathbb{Z}} g_{m-k}Ud_{j,k} = \sum_{\ell\in\mathbb{Z}} h_{m-2\ell}a_{j,\ell} + \sum_{\ell\in\mathbb{Z}} g_{m-2\ell}d_{j,\ell} \tag{6.10}$$

donde primero restauramos las dimensiones originales mediante el upsampling y después aplicamos la convolución con los filtros de paso alto y bajo respectivamente.

Por tanto, $a_{j+1,m} = h * Ua_j(m) + g * Ud_j(m)$ y podemos calcular estos coeficientes con L operaciones. □

Observamos que, aplicando el teorema 6.5, podemos seguir el siguiente proceso para el cálculo de los coeficientes,

$$\begin{array}{ccccccc} a_j & \longrightarrow & a_{j-1} & \longrightarrow & a_{j-2} & & \cdots \\ & \searrow & & \searrow & & & \\ & & d_{j-1} & & d_{j-2} & & \cdots \end{array} \tag{6.11}$$

donde podemos representar cada paso esquemáticamente como

$$\begin{array}{ccccc} & & *\tilde{h} & \longrightarrow \downarrow 2 \longrightarrow & a_j \\ a_{j+1} & \longrightarrow & & & \\ & & *\tilde{g} & \longrightarrow \downarrow 2 \longrightarrow & d_j \end{array} \tag{6.12}$$

aplicando las expresiones (6.5) y (6.6). En ingeniería eléctrica, este proceso suele denominarse la *fase de análisis* de un *esquema de subbanda*. Aplicando el teorema 6.8 de reconstrucción de los coeficientes en cada paso, obtenemos el proceso general de reconstrucción

$$\begin{array}{ccccccc} a_j & \longrightarrow & a_{j+1} & \longrightarrow & a_{j+2} & & \cdots \\ & \nearrow & & \nearrow & & & \\ d_j & & d_{j+1} & & & & \cdots \end{array} \tag{6.13}$$

cuya *fase de síntesis* o reconstrucción en cada paso, puede ser representada como

$$\begin{array}{ccccccc} a_j & \longrightarrow \uparrow 2 \longrightarrow & *h & & & & \\ & & & \searrow & & & \\ & & & & \oplus & \longrightarrow & a_{j+1} \\ & & & \nearrow & & & \\ d_j & \longrightarrow \uparrow 2 \longrightarrow & *g & & & & \end{array} \tag{6.14}$$

aplicando la ecuación (6.10). Observamos que en las aplicaciones prácticas podemos calcular solamente con los coeficientes h y g, además de nuestros datos, dejando la wavelet y la función de escalado implícitas en los cálculos.

6.3. El algoritmo en el sistema de Haar

Ilustramos el algoritmo con un ejemplo numérico simple utilizando los filtros de Haar. Utilizamos la notación $a_j = [x_0, x_1, \ldots, x_n]$ para representar la sucesión con coeficientes nulos a excepción de $k \in \{0, \ldots, n\}$, en cuyo caso $a_{j,0} = x_0, \ldots, a_{j,n} = x_n$.

Suponemos que tenemos los coeficientes de aproximación para escala uno $a_1 = [1, 2, 3, 4]$. Los filtros de paso bajo de Haar vienen dados por $h = [\sqrt{2}/2, \sqrt{2}/2]$, y los de paso alto

$g = [-\sqrt{2}/2, \sqrt{2}/2]$. Calculamos mediante la expresión (6.5),

$$a_{0,\ell} = \sum_{n\in\mathbb{Z}} h_{n-2\ell} a_{1,n} = h_0 a_{1,2\ell} + h_1 a_{1,2\ell+1}.$$

Si sustituimos por los valores, obtenemos

$$a_{0,0} = h_0 a_{1,0} + h_1 a_{1,1} = \frac{\sqrt{2}}{2} \times 1 + \frac{\sqrt{2}}{2} \times 2 = \frac{3\sqrt{2}}{2}$$
$$a_{0,1} = h_0 a_{1,2} + h_1 a_{1,3} = \frac{\sqrt{2}}{2} \times 3 + \frac{\sqrt{2}}{2} \times 4 = \frac{7\sqrt{2}}{2}.$$

Por tanto, $a_0 = [\frac{3\sqrt{2}}{2}, \frac{7\sqrt{2}}{2}]$. Operamos de manera análoga para los coeficientes de detalle. Aplicando (6.6),

$$d_{0,\ell} = \sum_{n\in\mathbb{Z}} g_{n-2\ell} a_{1,n} = g_0 a_{1,2\ell} + g_1 a_{1,2\ell+1}.$$

Por tanto,

$$d_{0,0} = g_0 a_{1,0} + g_1 a_{1,1} = -\frac{\sqrt{2}}{2} \times 1 + \frac{\sqrt{2}}{2} \times 2 = \frac{\sqrt{2}}{2}$$
$$d_{0,1} = g_0 a_{1,2} + g_1 a_{1,3} = -\frac{\sqrt{2}}{2} \times 3 + \frac{\sqrt{2}}{2} \times 4 = \frac{\sqrt{2}}{2}$$

y $d_0 = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$.

Calculamos ahora los coeficientes de reconstrucción. Recordamos que, por (6.9),

$$a_{1,m} = \sum_{\ell\in\mathbb{Z}} h_{m-2\ell} a_{0,\ell} + g_{m-2\ell} d_{0,\ell} = h_m a_{0,0} + h_{m-2} a_{0,1} + g_m d_{0,0} + g_{m-2} d_{0,1}.$$

Sustituyendo por los valores obtenidos anteriormente,

$$a_{1,0} = h_0 * a_{0,0} + g_0 * d_{0,0} = \frac{\sqrt{2}}{2} \times \frac{3\sqrt{2}}{2} + \frac{-\sqrt{2}}{2} \times \frac{\sqrt{2}}{2} = 1$$
$$a_{1,1} = h_1 * a_{0,0} + g_1 * d_{0,0} = \frac{\sqrt{2}}{2} \times \frac{3\sqrt{2}}{2} + \frac{\sqrt{2}}{2} \times \frac{\sqrt{2}}{2} = 2$$
$$a_{1,2} = h_0 * a_{0,1} + g_0 * d_{0,1} = \frac{\sqrt{2}}{2} \times \frac{7\sqrt{2}}{2} + \frac{-\sqrt{2}}{2} \times \frac{\sqrt{2}}{2} = 3$$
$$a_{1,3} = h_1 * a_{0,1} + g_1 * d_{0,1} = \frac{\sqrt{2}}{2} \times \frac{7\sqrt{2}}{2} + \frac{\sqrt{2}}{2} \times \frac{\sqrt{2}}{2} = 4.$$

6.4. Implementación del algoritmo

Presentamos a continuación la implementación del algoritmo mediante pseudocódigo, siguiendo las notaciones de los apartados anteriores.

Para calcular los coeficientes de aproximación de escala j, a partir de los coeficientes de aproximación de escala $j+1$, utilizamos la expresión (6.5),

Algoritmo 1 Cálculo de coeficientes de aproximación a_j

1: **procedure** CalculaCoeficientesAproximacion(a_{j+1}, h, L, N)
2: $a_j \leftarrow$ vector de tamaño $N/2$ inicializado a 0
3: **for** $l \leftarrow 0$ to $N/2$ **do**
4: **for** $n \leftarrow 2l$ to $L + 2l$ **do**
5: $a_j \leftarrow a_j + h[n - 2 \cdot l] \cdot a_{j+1}[n]$
6: **return** a_j

donde a_{j+1} y a_j son los coeficientes en V_{j+1} y V_j respectivamente, h son los filtros de paso alto de la wavelet escogida, L la longitud de los filtros y N el número de coeficientes en el nivel $j + 1$-ésimo.

Para la obtención de los detalles d_j mediante a_{j+1} tenemos, de manera similar, por (6.6),

Algoritmo 2 Cálculo de coeficientes de detalles d_j

1: **procedure** CalculaCoeficientesDetalle(a_{j+1}, g, L, N)
2: $d_j \leftarrow$ vector de tamaño $N/2$ inicializado a 0
3: **for** $l \leftarrow 0$ to $N/2$ **do**
4: **for** $n \leftarrow 2l$ to $L + 2l$ **do**
5: $d_j \leftarrow d_j + g[n - 2 \cdot l] \cdot a_{j+1}[n]$
6: **return** d_j

donde d_j son los coeficientes de detalle en W_j y g son los filtros de paso alto de la wavelet.

Para la reconstrucción de los coeficientes de aproximación a_{j+1} a escala $j+1$ mediante a_j y d_j,

Algoritmo 3 Cálculo de los coeficientes de reconstrucción a_{j+1}

1: **procedure** CalculaCoeficientesReconstruccion(a_j, d_j, h, g, L, N)
2: $a_{j+1} \leftarrow$ vector de tamaño N inicializado a 0
3: **for** $m \leftarrow 0$ to M **do**
4: **for** $l \leftarrow 0$ to $m/2$ **do**
5: **if** $m - 2l < L$ **then**
6: $a_{j+1}[m] \leftarrow a_{j+1}[m] + h[m - 2l] \cdot a_j[l] + g[m - 2l] \cdot d_j[l]$
7: **return** a_{j+1}

Podemos observar que en los procedimientos (1) y (2) devolvemos vectores de longitud $N/2$ por el downsampling, mientras que en (3) obtenemos un vector de longitud N tras el upsampling.

Notamos que los algoritmos de cálculo de coeficientes (1) y (2) ilustran los cálculos representados en (6.12), mientras que (3) hace referencia al esquema (6.14).

Vemos a continuación el algoritmo general de cálculo de coeficientes para escalas j e i (6.11) con $j > i$, y su respectiva reconstrucción (6.13).

Algoritmo 4 Cálculo de $a_j, \ldots, a_i$ y $d_j, \ldots, d_i$ a partir de a_{j+1}

procedure ComputeFWT($a_1, h, g, L, N, scalesDiff$)
    $coefficients.a \leftarrow$ vector inicializado a 0
    $coefficients.d \leftarrow$ vector inicializado a 0
    for $j \leftarrow 0$ to $scalesDiff - 1$ **do**
        Append ComputeCoefficients(a_1, h, L, N) to $coefficients.a$
        Append ComputeCoefficients(a_1, g, L, N) to $coefficients.d$
        $N \leftarrow N/2$
        $a_1 \leftarrow coefficients.a[j]$
    return $coefficients$

Para la reconstrucción de los coeficientes de aproximación de escala $j+1$ a partir de los de escala i y todos los coeficientes intermedios, $i, i+1, \ldots, j$, podemos calcular

Algoritmo 5 Reconstrucción de a_{j+1} a partir de $a_j, \ldots, a_i$ y $d_j, \ldots, d_i$

procedure ReconstructIFWT($a, d, h, g, L, N, scalesDiff$)
    $coeff \leftarrow$ vector vacío
    $N \leftarrow N/2^{scalesDiff-1}$
    for $j \leftarrow scalesDiff - 1$ to 0 by step -1 **do**
        Append ComputeReconstructions($a[j], d[j], N$) to $coeff$
        $N \leftarrow N \cdot 2$
    return $coeff$

donde a y d son vectores cuyos elementos son los distintos coeficientes $a_j, \ldots, a_i$ y $d_j, \ldots, d_i$ respectivamente.

Observamos que en la implementación los cálculos se hacen en función de los filtros de paso alto y de paso bajo, ya que toda la información de las wavelets, funciones de escalado y el MRA se codifica a través en estos valores. Presentamos a continuación tablas con valores para los filtros de distintas wavelets de uso común [Was22] [LWVK24].

Wavelet	**Filtros** h
Haar	$1/\sqrt{2}$, $1/\sqrt{2}$
db2	$\frac{1+\sqrt{3}}{4\sqrt{2}}$, $\frac{3+\sqrt{3}}{4\sqrt{2}}$, $\frac{3-\sqrt{3}}{4\sqrt{2}}$, $\frac{1-\sqrt{3}}{4\sqrt{2}}$
db3	0.035226291, -0.085441273, -0.135011020, 0.459877502, 0.806891509, 0.332670552

Cuadro 1: Filtros paso bajo.

Wavelet	**Filtros** g
Haar	$-1/\sqrt{2}$, $1/\sqrt{2}$
db2	$\frac{1-\sqrt{3}}{4\sqrt{2}}$, $-\frac{3-\sqrt{3}}{4\sqrt{2}}$, $\frac{3+\sqrt{3}}{4\sqrt{2}}$, $-\frac{1+\sqrt{3}}{4\sqrt{2}}$
db3	-0.332670553, 0.806891509, -0.459877502, -0.13501102, 0.085441274, 0.035226292

Cuadro 2: Filtros paso alto.

7. Wavelets en dos dimensiones

Hasta ahora todos los conceptos con los que hemos trabajado eran sobre una dimensión. Para nuestra aplicación, implementaremos la FWT sobre señales de dos dimensiones, en concreto sobre imágenes. A continuación, veremos cómo podemos generalizar las bases en las que hemos trabajado a más dimensiones, y cómo podemos aplicar el algoritmo a estas señales [PW12].

7.1. Base de wavelets en 2D

Dada una base wavelet unidimensional $\{\varphi_{j,k}\}_{j,k\in\mathbb{Z}} \in L^2(\mathbb{R})$, queremos generar una base en $L^2(\mathbb{R}^2)$ a partir de esta. Existe un método general para construir bases de dos dimensiones mediante bases unidimensionales; el producto tensorial.

En primer lugar, recordemos las nociones básicas del producto tensorial. Sean V, W espacios vectoriales sobre un cuerpo $\mathbb{R}$, con bases B_V, B_W respectivamente, consideramos las siguientes definiciones.

Definición 7.1. Definimos el *producto tensorial* $V \otimes W$ como el conjunto de funciones de $B_V \times B_W$ a $\mathbb{R}$ con número finito de valores no nulos. Denotamos $v \otimes w$ a la función que envía (v, w) a 1 y el resto de elementos de $B_V \times B_W$ a 0. El conjunto $\{v \otimes w | v \in B_V, w \in B_W\}$ forma base de $V \otimes W$.

Consideramos la familia de productos tensoriales

$$\psi_{j,k;i,n}(x, y) = \psi_{j,k}(x)\psi_{i,n}(y).$$

Observamos que es una base ortonormal de $L^2(\mathbb{R}^2)$. Pero esta base no nos sirve, ya que con esta construcción hemos perdido la estructura de multiresolución al mezclar las escalas j, i. Podemos utilizar la misma idea con la función de escalado, "fijando" la escala, para poder mantener la estructura de multiresolución. Más tarde podremos pasar a las wavelets mediante la función de escalado. Consideramos los productos tensoriales

$$\varphi_{j,k,n}(x, y) = \varphi_{j,k}(x)\varphi_{j,n}(y)$$

Donde φ es la función de escalado de un MRA ortogonal y para cada j, $\{\varphi_{j,k}\}_{k\in\mathbb{Z}}$ y $\{\varphi_{j,n}\}_{n\in\mathbb{Z}}$ forman bases ortonormales de V_j.

Ahora consideramos el espacio generado por las combinaciones de k, n, fijada j, de estos productos tensoriales. Tomaremos el espacio $\mathcal{V}_j$ como la clausura de este espacio en $L^2(\mathbb{R}^2)$.

Definición 7.2. Definimos $\mathcal{V}_j$ como

$$\mathcal{V}_j = V_j \otimes V_j := \left\{ f(x, y) = \sum_{n,k\in\mathbb{Z}} a_{j,k,n}\varphi_{j,k,n}(x, y) : \sum_{n,k\in\mathbb{Z}} |a_{j,k,n}|^2 < \infty \right\}.$$

Notamos que hemos considerado la clausura algebraica en L^2 de la noción de producto tensorial de la definición 7.1, donde también consideramos series de ℓ^2 además de sumas finitas. Vemos que estos son los espacios de aproximación que buscamos, y que forman un MRA ortogonal.

Proposición 7.3. *Los espacios $\mathcal{V}_j$ forman un MRA ortogonal de $L^2(\mathbb{R}^2)$, con función de escalado $\varphi(x,y) = \varphi(x)\varphi(y)$.*

Demostración. Comprobamos las propiedades de la definición 4.1, teniendo en cuenta las propiedades del producto tensorial,

1. Puesto que $V_j \subset V_{j+1}$ para todo $j \in \mathbb{Z}$, tenemos que
$$\mathcal{V}_j = V_j \otimes V_j \subset V_{j+1} \otimes V_{j+1} = \mathcal{V}_{j+1}.$$

2. Puesto que la intersección de los espacios V_j con $j \in \mathbb{Z}$ es nula,
$$\bigcap_{j\in\mathbb{Z}} \mathcal{V}_j = \bigcap_{j\in\mathbb{Z}} (V_j \otimes V_j) = \bigcap_{j\in\mathbb{Z}} V_j \otimes \bigcap_{j\in\mathbb{Z}} V_j = \{0\} \otimes \{0\} = \{0\}$$

3. Vemos que $\bigcup_{j\in\mathbb{Z}} \mathcal{V}_j$ es densa. Sabemos que la familia $\psi_{j,k;i,n}(x,y)$ forma base ortonormal de $L^2(\mathbb{R}^2)$. Por tanto, es suficiente con comprobar que $\bigcup_{j\in\mathbb{Z}} \mathcal{V}_j$ contiene los elementos de esta base. En concreto, tenemos que
$$\psi_j(x) = \sum_r c_r \varphi_{j+1,r}(x), \quad \sum_r |c_r|^2 = 1$$
$$\psi_i(x) = \sum_k d_k \varphi_{i+1,k}(y), \quad \sum_k |d_k|^2 = 1$$
Puesto que $V_{j+1} = V_j \oplus W_j$, sabemos que $\psi_j \in W_j \subset V_{j+1}$ y $\psi_i \in W_i \subset V_{i+1}$. Si $i \leq j$ (análogamente para $j < i$), $V_{i+1} \subset V_{j+1}$ y obtenemos
$$\psi_j(x)\psi_i(x) = \sum_{r,k} c_r d'_k \varphi_{j,r}(x)\varphi_{j,k}(x)$$
con $\sum_r \sum_k |c_r d'_k|^2 = 1$, donde d'_k son los coeficientes de ψ_i en escala j. Como podemos expresar los elementos de $\psi_{j,k;i,n}(x,y)$ como combinación lineal infinita convergente de elementos de $\bigcup_{j\in\mathbb{Z}} \mathcal{V}_j$, hemos comprobado el enunciado.

4. Observamos que
$$\begin{aligned} f(2x,2y) &= \sum_{n,k\in\mathbb{Z}} a_{j,n,k}\varphi_{j,k,n}(2x,2y) = \sum_{n,k\in\mathbb{Z}} a_{j,n,k}\varphi_{j,k}(2x)\varphi_{j,n}(2y) \\ &= 2\sum_{n,k\in\mathbb{Z}} a_{j,n,k}\varphi_{j+1,k}(x)\varphi_{j+1,n}(y) = 2\sum_{n,k\in\mathbb{Z}} a_{j,n,k}\varphi_{j+1,k,n}(x,y) = 2f(x,y), \end{aligned}$$
por tanto, $f(z) \in \mathcal{V}_j$ si y solo si $f(2z) \in \mathcal{V}_{j+1}$.

5. Puesto que $\varphi \in V_0$, de manera trivial $\varphi_{0,k,n}(x,y) = \varphi_{0,k}(x)\varphi_{0,n}(y) \subset V_0 \otimes V_0$. Como $\{\varphi(x-k)\}_{k\in\mathbb{Z}}$ forma base ortonormal de V_0, $\varphi_{j,k,n}(x,y)$ es base ortonormal de $V_0 \otimes V_0$.

□

Hemos hallado los subespacios de aproximaciones, pero necesitamos también los de detalles. Puesto que estamos en un MRA ortogonal, el subespacio de detalles $\mathcal{W}_j$ es el

complemento ortonormal de $\mathcal{V}_j$ en $\mathcal{V}_{j+1}$. Por las propiedades del producto tensorial y la suma directa,

$$\begin{aligned}\mathcal{V}_{j+1} &= V_{j+1} \otimes V_{j+1} = (V_j \oplus W_j) \otimes (V_j \oplus W_j) \\ &= (V_j \otimes V_j) \oplus [(V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j)]\end{aligned}$$

Por tanto, podemos expresar

$$\mathcal{W}_j = (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j)$$

con $\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j$. Fijándonos en esta última expresión observamos que necesitaremos tres wavelets para poder generar cada subespacio de detalles $\mathcal{W}_j$. Las denotaremos como wavelet diagonal ψ^d, vertical ψ^v y horizontal ψ^h por los detalles que tienden a destacar. De esta manera,

$$\psi^d(x,y) = \psi(x)\psi(y), \quad \psi^v(x,y) = \psi(x)\varphi(y), \quad \psi^h(x,y) = \varphi(x)\psi(y)$$

7.2. Aplicación en imágenes

Observamos que la definición que hemos dado en el apartado anterior nos proporciona bases separables. Esto es muy conveniente en la práctica, ya que implementar la FWT para señales de dos dimensiones se traduce en aplicar sucesivamente el algoritmo sobre cada eje.

Para hacer las descomposiciones en espacios de aproximación y de detalle mediante la FWT podemos hacerlo de la siguiente manera. En primer lugar, suponemos que nuestra imagen original con la resolución más "fina" se halla en el espacio de aproximación $\mathcal{V}_j$. Hacemos uso de la misma notación que en apartados anteriores y llamamos a_2 a los coeficientes de aproximación para escala 2. A continuación descomponemos a_2 en una aproximación a_1 de menor resolución y los detalles d_1 como hemos descrito en el apartado anterior. Tendremos que a_1 contiene $1/4$ de los datos y cada uno de los detalles d_1^h, d_1^d y d_1^v también contiene $1/4$ del total, con lo que $a_2 = a_1 + d_1$. Podemos repetir este proceso las veces que sea necesario hasta la escala que queramos alcanzar o esté disponible en nuestros datos. Visualmente, obtenemos:

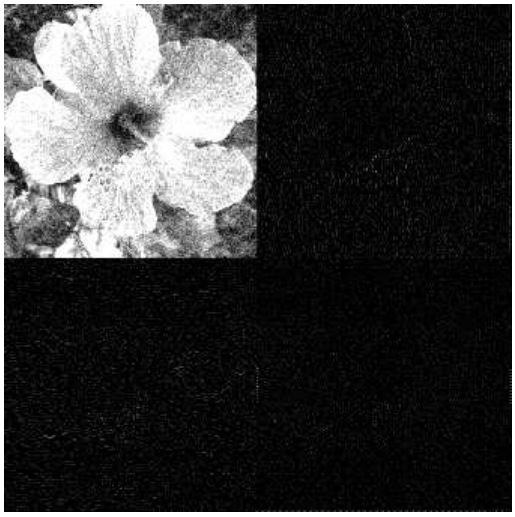


Figura 3: Imagen procesada mediante $db2$ para una escala.

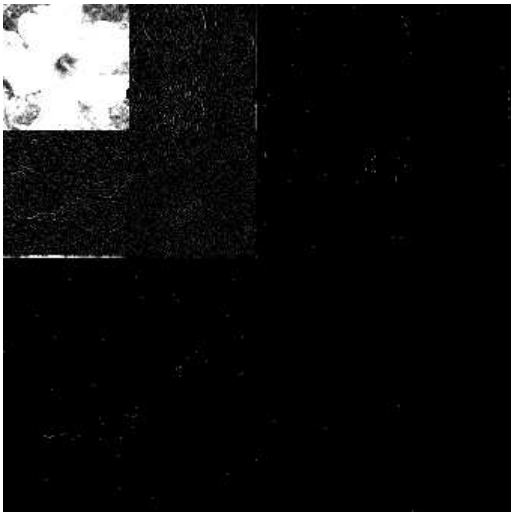


Figura 4: Imagen procesada mediante $db2$ para dos escalas.

Para las reconstrucciones podemos seguir el proceso inverso, siguiendo el esquema 6.13 para dos dimensiones.

7.3. Compresión y limpieza de ruido

Una tarea común en el campo de visión artificial y procesamiento de imágenes es la limpieza de ruido. El ruido en imágenes puede ser causado por diversos factores como la iluminación o limitaciones de la cámara, o la trasmisión de los datos de las imágenes. En este apartado truncamos los coeficientes de wavelets más pequeños en valor absoluto, asumiendo que algunos de ellos son los causantes del ruido en la imagen [FZFZ19]. Además, al truncar coeficientes wavelets y como consecuencia píxeles de la imagen, ahorramos espacio de memoria y estamos comprimiendo la imagen.

Podemos asumir que las imágenes naturales tienen cierta regularidad. Como consecuencia, muchos de los coeficientes wavelets son pequeños y si los eliminamos no alteramos la imagen en gran medida. Sin embargo, sí logramos ahorrar espacio de memoria. Esta técnica se conoce como *hard threshold* en el contexto de procesamiento de imágenes. Al mismo tiempo, sabemos que el ruido se distribuye de manera diferente al de las señales naturales [Gon09]. Por tanto, podemos asumir que la mayoría de coeficientes wavelet de valores pequeños son causados por el ruido. Al aplicar el *threshold* estamos eliminando parte de este ruido.

Observamos en las siguientes imágenes la eliminación de algunos de los coeficientes wavelet:

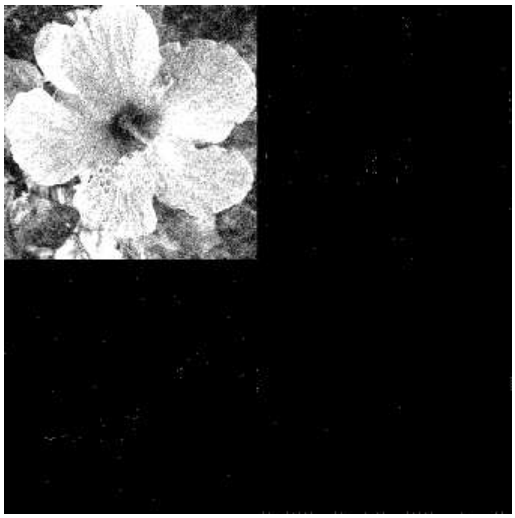


Figura 5: Eliminación de coeficientes wavelets en la figura 3.

Figura 6: Eliminación de coeficientes wavelets en la figura 4.

Tras hacer la reconstrucción de los coeficientes obtenemos los siguientes resultados:

Figura 7: Imagen original con todos los datos.

Figura 8: Imagen reconstruida tras el proceso de limpieza de ruido en una escala.

Notamos que la imagen 8 es un poco más suave que la original (7). En el código podemos comprobar que tras la limpieza de ruido hemos truncado a 0 alrededor de un 43 % de los datos iniciales, obteniendo una imagen más ligera. En concreto, hemos utilizado el límite inferior de 0,07 para truncar los píxeles.

Referencias

[Chu16] C. K. Chui. *An introduction to wavelets*. Elsevier, 2016.

[FZFZ19] L. Fan, F. Zhang, H. Fan, and C. Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, 2019.

[Gon09] R. C. Gonzalez. *Digital image processing*. Pearson education india, 2009.

[Haa10] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 71(1):38–53, 1910.

[LWVK24] G. R. Lee, F. Wasilewski, M. Vetterli, and M. Kutyłowski. Pywavelets: A python library for wavelet transform, 2006–2024.

[Mal99] S. Mallat. A wavelet tour of signal processing, 1999.

[MGBB00] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of jpeg-2000. In *Proceedings DCC 2000. Data Compression Conference*, pages 523–541, 2000.

[Pro01] J. G. Proakis. *Digital signal processing: principles algorithms and applications*. Pearson Education India, 2001.

[PW12] M. C. Pereyra and L. A. Ward. *Harmonic analysis: from Fourier to wavelets*, volume 63. American Mathematical Soc., 2012.

[Rud87] W. Rudin. *Real and complex analysis, 3rd ed.* McGraw-Hill, Inc., USA, 1987.

[Str96] G. Strang. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[Was22] F. Wasilewski. Wavelet properties browser, 2008–2022.

A. Código fuente en C++

En este anexo podemos encontrar el código desarrollado para la aplicación de procesamiento de imágenes. También podemos acceder a través del repositorio público de github `https://github.com/lbordajandi/wavelet-processing`.

A.1. Archivo main.cpp

```cpp
#include <math.h>
#include <stdint.h>

#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <vector>

#include "headers/image.h"
#include "headers/imageWaveletProcessor.h"
#include "headers/utils.h"
#include "headers/waveletTransform.h"

using std::vector;

int main(void) {
    // Wavelet filters
    vector<double> h = {0.035226292, -0.085441274, -0.13501102,
    0.459877502, 0.806891509, 0.332670553};
    vector<double> g = {-0.332670553, 0.806891509, -0.459877502,
    -0.13501102, 0.085441274, 0.035226292};

    WaveletTransform wavelet(h, g);
    int scalesDifference = 2;
    double threshold = 0.07;

    std::string fileName = "snow_monkey";
    std::string inputPath = "data/inputs/" + fileName + ".png";
    std::string outputPath = "data/outputs/" + fileName;

    Image image(inputPath);
    std::cout << std::endl
              << "Start processing..." << std::endl
              << "Number of rows and columns:" << std::endl
              << image.rows << " " << image.cols << std::endl;

    ImageWaveletProcessor image2DWavelet(wavelet, image, scalesDifference,
     outputPath);

    image2DWavelet.deconstructImage(threshold);
    image2DWavelet.reconstructImage();

    std::cout << "Finished processing." << std::endl;
    return 0;
}
```

A.2. Archivo waveletTransform.cpp

```cpp
#include "headers/waveletTransform.h"

#include <stdint.h>

WaveletTransform::WaveletTransform(const std::vector<double>& lowPass, const std::vector<double>& highPass) {
    if (lowPass.size() != highPass.size())
        throw std::invalid_argument("Low-pass and high-pass filters must have the same size.");
    h = lowPass;
    g = highPass;
    L = lowPass.size();
}

std::vector<double> WaveletTransform::computeCoefficients(std::vector<double>& a_1, std::vector<double>& f, int N) {
    std::vector<double> c_0(N / 2);
    for (int l = 0; l < N / 2; l++) {
        for (int n = 2 * l; n < L + 2 * l; n++) {
            if (n < N) {
                c_0[l] += f[n - 2 * l] * a_1[n];
            }
        }
    }
    return c_0;
}

std::vector<double> WaveletTransform::computeReconstructions(std::vector<double>& a_0, std::vector<double>& d_0, int N) {
    std::vector<double> a_1(N);
    for (int m = 0; m < N; m++) {
        for (int l = 0; l <= m / 2; l++) {
            if (m - 2 * l < L) {
                a_1[m] += h[m - 2 * l] * a_0[l] + g[m - 2 * l] * d_0[l];
            }
        }
    }
    return a_1;
}

WaveletCoeff WaveletTransform::computeFWT(std::vector<double> a_1, int N, int scalesDiff) {
    WaveletCoeff coefficients;
    for (int j = 0; j < scalesDiff; j++) {
        coefficients.a.push_back(computeCoefficients(a_1, h, N));
        coefficients.d.push_back(computeCoefficients(a_1, g, N));
        N /= 2;
        a_1 = coefficients.a[j];
    }
    return coefficients;
}

ApproxCoeff WaveletTransform::inverseFWT(ApproxCoeff& a, DetailsCoeff& d, int N, int scalesDiff) {
    ApproxCoeff coeff;
    N = N / pow(2, scalesDiff - 1);
    for (int j = scalesDiff - 1; j >= 0; j--) {
        coeff.push_back(computeReconstructions(a[j], d[j], N));
        N *= 2;
```

```
    }
    return coeff;
}
```

A.3. Archivo image.cpp

```
#include "headers/image.h"

using namespace cv;

Image::Image(const std::string& filepath) {
    image = cv::imread(filepath, IMREAD_GRAYSCALE);
    image.convertTo(image, CV_64F, 1.0 / 255.0);
    if (image.empty()) {
        throw std::runtime_error("Could not open or find the image: " +
    filepath);
    }
    int squareSize = std::min(image.rows, image.cols);
    cropTopLeftImage(squareSize);
    rows = squareSize;
    cols = squareSize;
}

void Image::cropTopLeftImage(int squareSize) {
    cv::Rect roi(0, 0, squareSize, squareSize);
    image = image(roi);
}

std::vector<double> Image::getRow(int row) const {
    if (row < 0 || row >= image.rows) {
        throw std::out_of_range("Row index out of range");
    }
    return std::vector<double>(image.ptr<double>(row), image.ptr<double>(
    row) + image.cols);
}

void Image::setRow(int row, const std::vector<double>& data) {
    if (row < 0 || row >= image.rows) {
        throw std::invalid_argument("Invalid row or row size does not
    match image dimensions");
    }
    double* rowPtr = image.ptr<double>(row);
    for (size_t i = 0; i < data.size(); i++) {
        rowPtr[i] = data[i];
    }
}

std::vector<double> Image::getColumn(int col) const {
    if (col < 0 || col >= image.cols) {
        throw std::out_of_range("Column index out of range");
    }
    std::vector<double> column(image.rows);
    for (int i = 0; i < image.rows; ++i) {
        column[i] = image.at<double>(i, col);
    }
    return column;
}

void Image::setColumn(int col, const std::vector<double>& data) {
    int size = data.size();
```

```
    if (col < 0 || col >= image.cols) {
        throw std::invalid_argument("Invalid column or column size does not match image dimensions");
    }
    for (int i = 0; i < size; i++) {
        image.at<double>(i, col) = data[i];
    }
}

void Image::save(const std::string& filepath) const {
    cv::Mat savedImage;
    image.convertTo(savedImage, CV_8U, 255.0);
    cv::imwrite(filepath, savedImage);
}
```

A.4. Archivo imageWaveletProcessor.cpp

```
#include "headers/imageWaveletProcessor.h"

ImageWaveletProcessor::ImageWaveletProcessor(WaveletTransform wavelet, Image image, int scalesDifference, std::string path)
    : wavelet(wavelet), image(image), scalesDifference(scalesDifference), path(path) {
}

void ImageWaveletProcessor::deconstructImage(double threshold) {
    int scale = 0, rowsToProcess = image.rows, colsToProcess = image.cols;
    while (scale < scalesDifference) {
        deconstructOneScale(rowsToProcess, colsToProcess);
        std::string outputPath = path + "_" + std::to_string(scale) + ".png";
        image.save(outputPath);
        if (threshold > 0) {
            denoiseProcessedImage(threshold, rowsToProcess, colsToProcess);
            outputPath = path + "_denoised_" + std::to_string(scale) + ".png";
            image.save(outputPath);
        }
        rowsToProcess = rowsToProcess / 2;
        colsToProcess = colsToProcess / 2;
        scale++;
    }
}

void ImageWaveletProcessor::deconstructOneScale(int rowsToProcess, int colsToProcess) {
    deconstructAxis(
        rowsToProcess,
        [this](int index) { return image.getRow(index); },
        [this](int index, const std::vector<double>& row) {
            image.setRow(index, row);
        });
    deconstructAxis(
        colsToProcess,
        [this](int index) { return image.getColumn(index); },
        [this](int index, const std::vector<double>& col) {
            image.setColumn(index, col);
        });
}
```

```
38
39 void ImageWaveletProcessor::deconstructAxis(
40     int axisToProcess,
41     std::function<std::vector<double>(int)> getAxis,
42     std::function<void(int, const std::vector<double>&)> setAxis) {
43     WaveletCoeff imageCoeff;
44     std::vector<double> axis, newAxis;
45
46     for (int index = 0; index < axisToProcess; index++) {
47         axis = getAxis(index);
48         imageCoeff = wavelet.computeFWT(axis, axisToProcess, 1);
49         newAxis = joinVectors(imageCoeff.a[0], imageCoeff.d[0]);
50         setAxis(index, newAxis);
51     }
52 }
53
54 double ImageWaveletProcessor::denoiseProcessedImage(double threshold, int
    rowsToProcess, int colsToProcess) {
55     vector<double> row, col;
56     int deletedPixels = 0;
57     denoiseAxis(rowsToProcess, threshold, deletedPixels, [this](int index)
     { return image.getRow(index); }, [this](int index, const std::vector<
    double>& row) { image.setRow(index, row); });
58     denoiseAxis(colsToProcess, threshold, deletedPixels, [this](int index)
     { return image.getColumn(index); }, [this](int index, const std::
    vector<double>& col) { image.setColumn(index, col); });
59     return computePercentageDeleted(deletedPixels);
60 }
61
62 void ImageWaveletProcessor::denoiseAxis(
63     int axisToProcess,
64     double threshold,
65     int& deletedPixels,
66     std::function<std::vector<double>(int)> getAxis,
67     std::function<void(int, const std::vector<double>&)> setAxis) {
68     for (int index = axisToProcess / 2; index < axisToProcess; index++) {
69         std::vector<double> axis = getAxis(index);
70         int axisSize = axis.size();
71         for (int i = 0; i < axisSize; i++) {
72             if (fabs(axis[i]) < threshold) {
73                 if (fabs(axis[i]) != 0.0) {
74                     deletedPixels++;
75                 }
76                 axis[i] = 0.0;
77             }
78         }
79         setAxis(index, axis);
80     }
81 }
82
83 double ImageWaveletProcessor::computePercentageDeleted(int deletedPixels)
    {
84     double percentageDeleted = (double(deletedPixels) / (image.rows *
    image.cols)) * 100.0;
85     std::cout << "Percentage of pixels deleted: " << percentageDeleted <<
    std::endl;
86     return percentageDeleted;
87 }
88
89 void ImageWaveletProcessor::reconstructImage() {
90     std::cout << "Starting reconstructions..." << std::endl;
```

```
    vector<double> row, col;
    int scale = 0;
    int rowsToProcess = image.rows / pow(2, scalesDifference - 1);
    int colsToProcess = image.cols / pow(2, scalesDifference - 1);

    while (scale < scalesDifference) {
        reconstructAxis(
            rowsToProcess,
            [this](int index) { return image.getRow(index); },
            [this](int index, const std::vector<double>& row) {
                image.setRow(index, row);
            });
        reconstructAxis(
            colsToProcess,
            [this](int index) { return image.getColumn(index); },
            [this](int index, const std::vector<double>& col) {
                image.setColumn(index, col);
            });
        scale++;
        rowsToProcess *= 2;
        colsToProcess *= 2;

        std::string outputPath = path + "_reconstructed_" + std::to_string
    (scale) + ".png";
        image.save(outputPath);
    }
}

void ImageWaveletProcessor::reconstructAxis(
    int axisToProcess,
    std::function<std::vector<double>(int)> getAxis,
    std::function<void(int, const std::vector<double>&)> setAxis) {
    std::vector<double> axis;
    int halfSize = axisToProcess / 2;
    for (int index = 0; index < axisToProcess; index++) {
        axis = getAxis(index);
        std::vector<double> approxCoeff(axis.begin(), axis.begin() +
    halfSize);
        std::vector<double> detailCoeff(axis.begin() + halfSize, axis.end
    ());
        std::vector<double> reconstructedAxis = wavelet.
    computeReconstructions(approxCoeff, detailCoeff, axisToProcess);
        setAxis(index, reconstructedAxis);
    }
}
```

A.5. Archivo utils.cpp

```
#include "headers/utils.h"

#include <tuple>

void printVector(const std::string& label, const std::vector<double>& vec)
     {
    std::cout << label << " [ ";
    for (const auto& val : vec) {
        std::cout << val << " ";
    }
    std::cout << "]" << std::endl;
}
```

```
12
13 void printVector(const std::string& label, const std::vector<std::vector<
       double>>& vec) {
14     std::cout << label << " [\n";
15     for (const auto& row : vec) {
16         std::cout << "  [ ";
17         for (const auto& val : row) {
18             std::cout << val << " ";
19         }
20         std::cout << "]\n";
21     }
22     std::cout << "]" << std::endl;
23 }
24
25 std::vector<double> joinVectors(std::vector<double> v1, std::vector<double
       > v2) {
26     v1.insert(v1.end(), v2.begin(), v2.end());
27     return v1;
28 }
29
30 std::tuple<std::vector<double>, std::vector<double>> breakVector(std::
       vector<double> v) {
31     int halfSize = v.size() / 2;
32     std::vector<double> v1(v.begin(), v.begin() + halfSize);
33     std::vector<double> v2(v.begin() + halfSize + 1, v.end());
34     return {v1, v2};
35 }
```

A.6. Headers

Listing 1: waveletTransform.h

```
#ifndef WAVELET_TRANSFORM_H
#define WAVELET_TRANSFORM_H

#include <vector>

#include "utils.h"

using ApproxCoeff = std::vector<std::vector<double>>;
using DetailsCoeff = std::vector<std::vector<double>>;

struct WaveletCoeff {
    ApproxCoeff a;
    DetailsCoeff d;
};

class WaveletTransform {
   private:
    std::vector<double> h;
    std::vector<double> g;

   public:
    int L;
    WaveletTransform(const std::vector<double>& lowPass, const std::vector
    <double>& highPass);
    std::vector<double> computeCoefficients(std::vector<double>& a_1, std
    ::vector<double>& f, int N);
    std::vector<double> computeReconstructions(std::vector<double>& a_0,
    std::vector<double>& d_0, int N);
    WaveletCoeff computeFWT(std::vector<double> a_1, int N, int scalesDiff
    );
    ApproxCoeff inverseFWT(ApproxCoeff& a, DetailsCoeff& d, int N, int
    scalesDiff);
};

#endif  // WAVELET_TRANSFORM_H
```

Listing 2: image.h

```
#ifndef IMAGE_DATA_H
#define IMAGE_DATA_H

#include <iostream>
#include <opencv2/opencv.hpp>
#include <stdexcept>

class Image {
   private:
    cv::Mat image;

   public:
    int rows, cols;
    Image(const std::string& filepath);

    std::vector<double> getRow(int row) const;
    std::vector<double> getColumn(int col) const;
    void setRow(int row, const std::vector<double>& data);
    void setColumn(int col, const std::vector<double>& data);
```

```
20
21      void cropTopLeftImage(int squareSize);
22      void save(const std::string& filepath) const;
23  };
24
25  #endif  // IMAGE_DATA_H
```

Listing 3: utils.h

```
1  #ifndef UTILS_H
2  #define UTILS_H
3  #include <stdint.h>
4
5  #include <iostream>
6  #include <vector>
7
8  void printVector(const std::string& label, const std::vector<double>& vec)
       ;
9  void printVector(const std::string& label, const std::vector<std::vector<
       double>>& vec);
10 std::tuple<std::vector<double>, std::vector<double>> breakVector(std::
       vector<double> v);
11 std::vector<double> joinVectors(std::vector<double> v1, std::vector<double
       > v2);
12
13 #endif  // UTILS_H
```

Listing 4: imageWaveletProcessor.h

```
1  #ifndef IMAGE_WAVELET_PROCESSOR_H
2  #define IMAGE_WAVELET_PROCESSOR_H
3
4  #include <stdint.h>
5
6  #include <vector>
7
8  #include "image.h"
9  #include "utils.h"
10 #include "waveletTransform.h"
11
12 using std::vector;
13
14 using ApproxCoeff = std::vector<std::vector<double>>;
15 using DetailsCoeff = std::vector<std::vector<double>>;
16
17 class ImageWaveletProcessor {
18    private:
19     void deconstructAxis(
20         int axisToProcess,
21         std::function<std::vector<double>(int)> getAxis,
22         std::function<void(int, const std::vector<double>&)> setAxis);
23     void deconstructOneScale(int rowsToProcess, int colsToProcess);
24
25     void denoiseAxis(
26         int axisToProcess,
27         double threshold,
28         int& deletedPixels,
29         std::function<std::vector<double>(int)> getAxis,
30         std::function<void(int, const std::vector<double>&)> setAxis);
31     double computePercentageDeleted(int deletedPixels);
32
```

```
33      void reconstructAxis(
34          int axisToProcess,
35          std::function<std::vector<double>(int)> getAxis,
36          std::function<void(int, const std::vector<double>&)> setAxis);
37
38    public:
39      WaveletTransform wavelet;
40      Image image;
41      int scalesDifference;
42      std::string path;
43      ImageWaveletProcessor(WaveletTransform wavelet, Image image, int
        scalesDifference, std::string path);
44
45      void deconstructImage(double threshold);
46      double denoiseProcessedImage(double threshold, int rowsToProcess, int
        colsToProcess);
47      void reconstructImage();
48  };
49
50  #endif  // IMAGE_WAVELET_PROCESSOR_H
```