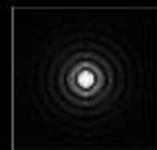
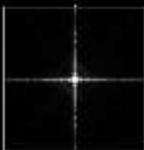


Procesamiento de Datos Discretos en 1D y 2D: Fourier, Coseno y Wavelet

Aplicaciones 1.0



Sea $X[n] = [1, 3, -4, 0, 1, 7, -9, -15, 6, \text{adelante}, \text{atrás}, -1/3]$;

Facultad de Ingenierías y Arquitectura
Universidad de Pamplona
2020

*Programas Ingeniería en **Telecomunicaciones,**
Electrónica e Industrial
Facultad de Ingenierías y Arquitectura
Universidad de Pamplona*

Procesamiento de Datos Discretos en 1D y 2D: Fourier, Coseno y Wavelet

Aplicaciones

V 1.0

**Editorial: Universidad de Pamplona
Colombia**

Procesamiento de Datos Discretos en 1D y 2D, Usando Fourier, Coseno y Wavelet

Aplicaciones

V 1.0

**Editorial: Universidad de Pamplona
Colombia**

**Procesamiento de datos discretos en 1D y 2D:
Fourier, Coseno y Wavelet.**

Derechos reservados

Primera edición

Autor Principal

Luis Enrique Mendoza

e-mail: luis.mendoza@unipamplona.edu.co

2020

Impreso en Colombia

ISBN: 978-958-53020-3-7

2020

**Universidad de Pamplona
Facultad de Ingenierías y Arquitectura
Ingeniería en Telecomunicaciones, Ingeniería
Electrónica e Ingeniería Industrial
Colombia
2020**

Agradezco a dios y la virgen por permiti  dedicar tiempo a este libro. A mis hijos Isabel Sof a, Jer nimo y mi esposa que son el motor de mi vida.

A la Universidad de Pamplona, por el apoyo recibido y darme la oportunidad de mi desarrollo acad mico, investigativo y administrativo.

A Leonor, Ronald y Alberto por su dedicaci n y apoyo en la escritura de los cap tulos.

Contenido

PREFACIO

GLOSARIO

CAPÍTULO I 16

TRANSFORMADA DISCRETA DE FOURIER (TDF)

16

| | |
|--|----|
| 1.1 Introducción TDF en el procesamiento de datos | 17 |
| 1.2 Estructura matemática de la TDF- 1D | 21 |
| 1.3 Ejemplos matemáticos de la TDF-1D y desarrollo gráfico. | 26 |
| 1.4 Estructura matemática de la TDF TC- 1D | 40 |
| 1.5 Estructura de matemática de la TDF- 2D. y TDFI-2D. | 44 |
| 1.6 Ejemplos matemáticos de la TDF-2D y desarrollo de ejercicios en datos reales. | 46 |
| 1.7 Aplicaciones en el procesamiento de datos | 61 |
| 1.7.1 Extracción de patrones | 62 |
| 1.7.2 Eliminación de información | 69 |
| 1.8 Ejercicios propuestos | 76 |
| Referencias | 79 |

CAPÍTULO II 88

TRANSFORMADA DISCRETA DEL COSENO

(TDC) 88

| | |
|--|-----|
| 2.1 Introducción de la TDC en el procesamiento de datos | 89 |
| 2.2 Estructura matemática de la TDC-1D | 90 |
| 2.3 Ejemplos matemáticos TDC | 95 |
| 2.4 Ejemplos TDC en señales de longitudes altas | 100 |
| 2.5 Estructura de matemática de la TDC-2D .. | 106 |
| 2.6 Ejemplos matemáticos de la TDC-2D | 107 |
| 2.7 Ejemplos TDC-2D en imágenes reales | 115 |
| 2.8 Aplicaciones en el procesamiento de datos | 119 |
| 2.8.1 Compresión | 119 |
| 2.8.2 Detección de contornos | 123 |
| 2.8.3 Extracción de patrones | 127 |
| 2.8.4 Eliminación de Información | 132 |
| 2.9 Ejercicios propuestos TDC 1D y 2D | 137 |
| Referencias | 141 |

CAPÍTULO III 150

TRANSFORMADA WAVELET DISCRETA (TWD)

150

| | |
|---|-----|
| 3.1 Introducción análisis Wavelet | 151 |
| 3.2 Transformada wavelet discreta inversa 1D | 159 |
| 3.3 Ejemplos transformada wavelet 1D | 163 |
| 3.3 1 Descomposición | 163 |
| 3.3 2 Eliminación de ruido | 169 |
| 3.3.3 Extracción de patrones | 171 |
| 3.3.4 Compresión de señales | 175 |
| 3.4 Transformada wavelet en 2D | 176 |

| | |
|---|-----|
| 3.4.1 Descomposición en 2D | 179 |
| 3.4.2 Compresión 2D | 185 |
| 3.4.3 Extracción de patrones | 190 |
| Referencias | 194 |

CAPÍTULO IV 203

DESARROLLO EN PYTHON 203

| | |
|---|-----|
| 4.1 ¿Qué es Python? | 204 |
| 4.2 ¿Qué es PyCharm? | 205 |
| 4.3 Instalación de Python | 205 |
| 4.4 Instalación y configuraciones iniciales de PyCharm | 210 |
| 4.5 Crear un nuevo proyecto en Python | 221 |
| 4.6 Algunas librerías importantes y su instalación | 227 |
| 4.6.1 Numpy | 227 |
| 4.6.2 Matplotlib | 228 |
| 4.6.3 OpenCV | 228 |
| 4.7 Ejemplos usando Python | 232 |
| 4.7.1 Filtrado de señales en 1D | 232 |
| 4.7.2 Extracción de patrones en señales 1D | 239 |
| 4.7.3 Extracción de patrones usando la transformada discreta del coseno TDC | 243 |
| 4.7.4 Compresión de información | 244 |
| 4.7.5 Análisis tiempo-frecuencia en 1D usando la transformada Wavelet Estacionaria | 249 |
| 4.7.6 Compresión y extracción de patrones utilizando TDC en 2D | 253 |
| Referencias | 260 |

***El conocimiento se adquiere y se fortalece,
si existe
Constancia, Paz, Dedicación y Amor
por lo que se hace.***

PREFACIO

El procesamiento de datos, en la actualidad, está siendo una de las áreas que más importancia y más crecimiento ha tenido en los últimos años, esto es debido a sus múltiples aplicaciones. Adicionalmente, es importante resaltar que el procesamiento de datos, está siendo usado en áreas como: Telecomunicaciones, Electrónica, Ambiental, Economía, Industrial, Mecatrónica, Mecánica, Sistema, Civil y Eléctrica entre otras. Es así, como se puede decir que el procesamiento de datos es una herramienta fundamental y transversal para las áreas que actualmente tienen auge. El procesamiento de datos se puede ver, de manera sencilla, como una forma o método de obtener, conseguir o visualizar información importante o relevante, que en el dominio del tiempo es poco probable observarla, como, por ejemplo, la frecuencia de una señal.

El procesamiento de datos se puede definir como la aplicación de herramientas matemáticas, a un conjunto de datos con el fin de realizar una aplicación. Estas aplicaciones pueden definirse como: compresión, extracción de patrones, cifrado, y eliminación de ruido entre otras.

En este libro se presenta, los métodos matemáticos en 1D y 2D, de herramientas matemáticas como: La transformada de Fourier, la transformada discreta del coseno y la transformada wavelet discreta, así mismo este libro presenta resultados de aplicaciones como: eliminación de ruido, compresión y cifrado. Es así como este libro está dividido en IV capítulos, en el cual el primero trata de todo lo relacionado con análisis de Fourier, teoría y práctica, el segundo capítulo trata todo lo relacionado tanto en el aspecto matemático como en el aspecto práctico de la transformada discreta del coseno, el tercer capítulo describe la transformada wavelet con sus respectivas demostraciones matemáticas y sus aplicaciones. Y el cuarto capítulo trata de la comparación de resultados y ejercicios resueltos en Python, así como ejercicios propuestos.

Así mismo para darle mayor cobertura las aplicaciones reales que desde el análisis frecuencial se pueden realizar, se ha incluido los términos de

visión artificial (VA), con el fin de mostrar la eficiencia o la potencialidad de las técnicas en diferentes áreas del conocimiento. La VA permite realizar aplicaciones en espacios de 2D, buscando una mejor visión y ampliar al margen de este libro.

GLOSARIO

Los conceptos que se presentan a continuación ayudan a entender de manera más eficiente la documentación expuesta y la terminología utilizada en capítulos siguientes.

A

Análisis de Fourier: estructura matemática que representa en función de senos y cosenos una señal y que en resultado se obtiene las componentes de frecuencia de una señal., 13

Análisis frecuencial: el análisis frecuencia se define como el mapeo que se realiza para lograr tener un espacio en función de la frecuencia que tiene la señal a procesar. Por ejemplo, en el tiempo no es claro que frecuencia tiene una señal no estacionara, haciendo uso del análisis frecuencial se obtiene las frecuencias correspondientes de dicha señal.

El análisis frecuencial es sumamente importante en temas como Telecomunicaciones, y procesamiento de la información, ya que en ocasiones es fundamental mapear al espacio de la frecuencia y obtener resultados satisfactorios del proceso.

Técnicas matemáticas: las técnicas matemáticas que se plantean estudiar en este libro son 3:

transformada discreta de Fourier, transformada discreta del coseno y transformada discreta wavelet. Es importante mencionar que este libro está enfocado en el estudio de datos en una dimensión (1D) y en datos de dos dimensiones (2D). Esto con el propósito de mostrar aplicaciones en imagenología y en tiempo real usando visión artificial.

....En la introducción se plantea una breve evolución por años, es importante mencionar que la TDF es muy utilizada y es la más conocida, ha conseguido realizar aplicaciones en un contexto para extracción de patrones, y que a su vez permitió realizar comparaciones con la TDC, y que la TDW logró vencer algunas de las debilidades de las técnicas comúnmente utilizadas., 72.

E

ECG: Electroencefalografía, señal bioeléctrica proveniente del corazón., 17.

EEG: Electroencefalografía, señal bioeléctrica que estudia el comportamiento neuroquímico del cerebro., 17.

Eliminación de ruido: se conoce como eliminación de ruido la eliminación de elementos en los datos que no son relevantes para los análisis a realizar o procesamiento siguiente., 17.

EOG: Electrooculograma, señal bioeléctrica que permite medir el movimiento de los ojos, consiste en ubicar electrodos alrededor de los ojos y medir la polarización de los musculos provocada por el movimiento., 17.

Extracción de patrones: se conoce como la búsqueda de una característica particular entre grupos o características que identifique o diferencien grupos para realizar un proceso, estas características deben ser de menor tamaño o longitud que el vector original., 20.

F

Frecuencia de la señal: La definición común es el número de veces que se repite la señal en el tiempo de registro. Sin embargo, en este libro se define como la velocidad de cambio o razón de cambio de la señal en un determinado tiempo, esto ya que las aplicaciones y las señales ejemplo son aleatorias., 18.

S

Señal: medición física de un sistema, en la cual está contenida información importante del procesos o sistema medido. Estas señales pueden ser 1D, es decir una variable independiente, ejemplos; la señal de voz, señal de temperatura, señal de los músculos, señal de velocidad, señal de corriente y 2D, que tiene dos variables independientes, un

ejemplo típico son las matrices y las imágenes a nivel de gris o binarias., 74.

Señal estacionaria: es una señal cuya frecuencia no varía en el tiempo, es decir la frecuencia o las frecuencias se mantienen durante todo el tiempo de registro., 16.

Señal no periódica: es una señal que no se repite en el tiempo, es decir que tiene diferentes formas de onda en todo su registro en el tiempo., 13.

Señal Periódica: Una señal periódica es aquí cuya frecuencia se repite en el tiempo o su forma de onda hace que tenga porciones de igual forma en diferentes tiempos., 14.

Señal Sparse: son señales poco densas, que tiene la característica de tener información que permite comprimir de mejor manera la información, una señal sparse son considera una señal con pocos puntos con valores altos en amplitud y gran cantidad de información con valores de amplitud pequeños., 71.

T

TDC: Transformada Discreta del Coseno., 71.

TDCI: Transformada discreta del coseno Inversa, 76.

TDF: Trnasformada Discreta de Fourier, 15

TDF-2D: Trnasformada Discreta de Fourier para 2 dimensiones, 17.

TDFI: Transformada Discreta de Fourier Inversa, 19.

TDF-TC: Transformada Discreta de Fourier en Tiempo Corto, 18.

Tiempo de muestreo: es el inverso del número de datos registrados en un segundo, se mide en el sistema internacional., 13.

Transformada discreta del coseno: estructura matemática que es real y que representa en función de cosenos una señal, para obtener una representación en frecuencia con una característica adicional de compactación de energía., 72.

TWD: Transformada Wavelet discreta., 119.

CAPÍTULO I

TRANSFORMADA DISCRETA DE FOURIER (TDF)

Luis Enrique Mendoza,
Ingeniería en Telecomunicaciones
Universidad de Pamplona

La TDF es una estructura matemática que viene de los años 70, que permite de manera fácil conocer la morfología de la señal en orden de las componentes de frecuencia, esto permite concluir que se realiza un mapeo entre el tiempo y la frecuencia. En TDF no existe resolución tiempo-frecuencia, solo existe máxima resolución en tiempo y máxima resolución en frecuencia.

1.1 Introducción TDF en el procesamiento de datos

La transformada de Fourier discreta (**TDF**) se define como una herramienta matemática, en la cual se hallan las componentes de frecuencia de una señal, en otras palabras, se pasa del espacio del tiempo al espacio de la frecuencia. Es importante resaltar que, por su estructura matemática de la TDF, se dice que solo se puede aplicar a datos o señales estacionarias, ya que su desventaja es que no crea un plano tiempo-frecuencia. Por otro lado, la TDF puede ser aplicada a cualquier tipo de datos, estacionarios o no estacionarios siempre y cuando no se necesite un espacio tiempo-frecuencia, lo cual quiere decir que solo se necesite para encontrar las componentes de frecuencia de una señal. De necesitarse un espacio tiempo –frecuencia se debe utilizar la TDF fraccionada, que no es más que dividir los datos en segmento de tiempo y aplicar la transformada.

La aplicación de la TDF, en el procesamiento de datos ha sido fundamental, ya que, desde sus inicios, se ha utilizado para aplicaciones de modulación, extracción de patrones y eliminación de ruido. Actualmente, la TDF, se utiliza para procesar señales de voz, señales metabólicas, ECG, EEG, EOG, imágenes de radar, tomografía y angiografía entre otras.

El procesamiento de datos utilizando la TDF, aunque cada día aparecen nuevas herramientas matemáticas, que vencen las debilidades de la TDF, sigue siendo una herramienta fundamental, ya que diferentes aplicaciones reales, son basadas en esta herramienta, lo cual hace que se habrá un campo de investigación con el objetivo de reemplazar las aplicaciones de la TDF [1,2].

Es este libro se presenta la TDF 1D y la TDF-2D, que son las estructuras matemáticas base para cualquier aplicación y representación de datos, esto quiere

decir que esta herramienta se puede aplicar a datos números y texto [1-7].

Tabla 1.1. Aplicación de la transformada de Fourier TDF- en su estado original
Longitud máxima de la señal

La Tabla 1.1 muestra un grafica del funcionamiento de la TDF, aquí se desea mostrar que la transformada de Fourier se aplica a toda la señal en su estado original, esto no indica que la señal no pueda tomarse en porciones y aplicar dicha transformada. Aquí la reflexión se da ya que la transformada tiene una gran debilidad y es que no realiza análisis tiempo-frecuencia, lo que permite concluir que tiene una máxima resolución en frecuencia y una baja resolución en tiempo. Así mismo se puede segmentar la señal para encontrar una resolución tiempo-frecuencia, esto se indica en la Tabla 1.2. Dicha división de la señal y la aplicación a cada división de la transformada de Fourier se conoce como transformada de Fourier en tiempo corto (TDF-

TC). La cual obtiene una resolución en tiempo-frecuencia de la señal a analizar, debido al control del tamaño de la ventana seleccionada [8-12].

Tabla 1.2. Aplicación de la transformada de Fourier en tiempo corto (TDF-TC)

| | | | | | | |
|------|----------|------|----------|------|----------|----------------|
| TDF1 | TDF2 | TDF3 | TDF4 | TDF5 | TDF6 | TDF27 |
| | τ_1 | | τ_2 | | τ_3 | $\tau_4 \dots$ |
| L | | | | | | |

La Tabla 1.2 muestra el desarrollo gráfico y funcional de la TDF-TC, nótese que la señal original es dividida en 7 secciones y cada sección es aplicada la TDF. Según el resultado encontrado en la porción número 1, se puede decir, que dichas frecuencias se encuentran en un tiempo entre 0 - τ_1 , lo cual da una relación tiempo-frecuencia, sin embargo entre más pequeña es la duración de las ventanas la calidad de la respuesta en funciones de la frecuencia disminuye, esto indica: a menor resolución en tiempo (duración de la ventana más alta) mayor resolución en frecuencia y a menor resolución en frecuencia

(duración de la ventana más baja) mayor resolución en tiempo.

A continuación, se muestran las presentaciones matemáticas de la TDF [1-12].

1.2 Estructura matemática de la TDF- 1D.

Matemáticamente la TDF-1D esta expresada como $c(k)$:

$$c(k) = \sum_{n=0}^{N-1} x(n) * e^{-2*pi*n*k/N} \quad (1. 1)$$

Aquí, $x(n)$, es el vector original, N = longitud del vector y n la posición.

Por otro lado, la estructura matemática de la transformada de Fourier inversa (**TDFI - 1D**), se describe como $x(n)$:

$$x(n) = \frac{1}{N} * \sum_{k=0}^{N-1} x(k) * e^{2*pi*k*n/N} \quad (1. 2)$$

La expresión matemática ecuación (1. 1), es fundamental, ya que al aplicarla como se ha venido hablando se obtiene las componentes de frecuencia de una señal. En el ejemplo 1, se muestra un

resultado de aplicar la TDF de una señal senoidal con frecuencia 125Hz y frecuencia de muestreo 1200Hz. Por otro lado, se debe entender muy bien su funcionamiento, ya que desde el espacio $x(k)$, es donde se puede realizar una serie de aplicaciones como: extracción de patrones, o filtrado.

Es importante recordar que la TDF, es una representación de seno y cosenos complejos de la señal $x(n)$, por lo tanto, $x(k)$, es un vector complejo, en el cual se tiene parte real y parte imaginaria. Esto significa que, para graficar la respuesta en un ejemplo real, se debe obtener la magnitud de cada dato, para saber su amplitud. El espacio de la gráfica en su eje x, significa la frecuencia y en su eje y la amplitud (ver

Figura 1.1). Adicionalmente a esto, se sabe que la TDF, cumple la propiedad de simetría par con respecto al punto medio de la máxima longitud del vector, esto quiere decir que el eje de simetría está relacionado con la frecuencia máxima, que se puede obtener según la frecuencia la que se a muestreado.

Por ejemplo: si una señal de voz es muestreada a 8Khz, su frecuencia máxima que puede encontrarse con la TDF es 4Khz, esto significa que el eje de simetría está en el 4Khz. Si a este ejemplo le incluimos, que la señal fue adquirida por 3 segundos, se puede decir que el eje de simetría está a 4Khz o en los 12.000 puntos, que son la mitad del tamaño real del vector [13-14].

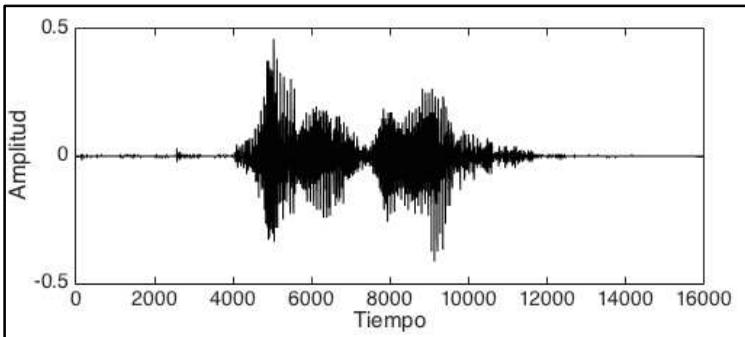


Figura 1.1.
Señal de voz en el tiempo.

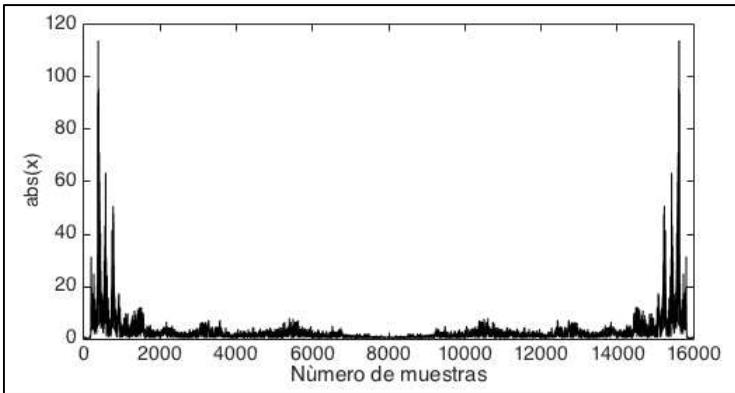


Figura 1.2.
Señal de voz en frecuencia.

La

Figura 1.1 y

Figura 1.2, se muestra una señal de voz real y su representación de la TDF. Aquí se observa los ejes de simetría, la longitud del vector. Nótese como la señal en frecuencia, tiene una longitud de 16000 datos, pero en realidad por la propiedad de simetría, solo se recomienda analizar para efectos de frecuencia del punto 1 al punto 8000. Se debe resaltar que los números del eje x, no indican valores en frecuencia para conocer los valores reales de

frecuencia se debe tener en cuenta: frecuencia de muestreo, tiempo de adquisición, ya que en el análisis de Fourier la longitud máxima de puntos es decir 16000 puntos, equivale a la frecuencia de muestreo, para este ejemplo se tomó 8Khz. En este ejemplo se puede mencionar que las componentes más representativas en amplitud están alrededor de los 50 a los 1000 puntos, pero no indica que son las frecuencias, para indicar la frecuencia real se concluye que 16000 puntos, son 8Khz, entonces 1000 equivalen a 500hz.

Por otro lado, todo término transformado, trae consigo la definición de su inversa en el caso de la TDF, se denomina transformada de Fourier discreta inversa (TDFI), la cual transforma los datos del dominio de la frecuencia al dominio del tiempo. Esto es relevante ya que en el espacio $x(k)$, la resolución de la información original se pierde, es decir que la forma de onda de los datos originales no se mantiene.

La estructura de TDFI ver ecuación (1. 2) , consigue obtener los datos originales si y solo si ningún coeficiente de $x(k)$, modifica sus características, el hecho de que al menos uno solo sea modificado, el resultado final sería $x'(n)$, donde la ' , quiere decir que el resultado es una aproximación de $x(n)$ original. Dependiendo el **número** de datos que se modifiquen y **cuales** datos sean modificados, se obtendrá una representación muy aproximada o muy diferentes de $x(n)$.

A continuación, se describen una serie de ejemplos en donde se explica de mejor manera la parte práctica de esta herramienta matemática [15-20].

1.3 Ejemplos matemáticos de la TDF-1D y desarrollo gráfico.

Ejemplo 1.1.

Hallar la TDF de $x(n) = [6, 3, 1]$ si $N = 3$

Si $X(n) = [6 \ 3 \ 1]$, hallar $c(k)$ $n = 3$

$$c(k) = \sum_{n=0}^2 x(n) e^{\frac{-2\pi jkn}{3}}$$

$$c(0) = \sum_{n=0}^2 x(n) e^0$$

$$c(0) = x(0) + x(1) + x(2)$$

$$c(0) = 10$$

$$c(1) = \sum_{n=0}^2 x(n) e^{\frac{-2\pi jn}{3}}$$

$$c(1) = x(0) e^0 + x(1) e^{\frac{-2\pi j}{3}} + x(2) e^{\frac{-4\pi j}{3}}$$

$$c(1) = x(0) + x(1) \left[\cos\left(\frac{2\pi}{3}\right) - j \operatorname{sen}\left(\frac{2\pi}{3}\right) \right] \\ + x(2) \left[\cos\left(\frac{4\pi}{3}\right) - j \operatorname{sen}\left(\frac{4\pi}{3}\right) \right]$$

$$c(1) = 6 + (3) \cos\left(\frac{2\pi}{3}\right) \\ + (1) \cos\left(\frac{4\pi}{3}\right) - j [(3) \operatorname{sen}\left(\frac{2\pi}{3}\right) \\ + (1) \operatorname{sen}\left(\frac{4\pi}{3}\right)]$$

$$x(1) = 4.0000 - 1.7321i$$

$$c(2) = \sum_{n=0}^2 x(n) e^{-\frac{4\pi j n}{3}}$$

$$c(2) = x(0) + x(1) e^{-\frac{4\pi j}{3}} + x(2) e^{-\frac{8\pi j}{3}}$$

$$c(2) = 6 + 3 \left(\cos\left(\frac{4\pi}{3}\right) - j \operatorname{sen}\left(\frac{4\pi}{3}\right) \right) + 1 \left[\cos\left(\frac{8\pi}{3}\right) - j \operatorname{sen}\left(\frac{8\pi}{3}\right) \right]$$

$$c(2) = 4.0000 + 1.7321i$$

$$c(k) = [10.0000, 4.0000 - 1.7321i, 4.0000 + 1.7321i]$$

Ejemplo 1.2.

Hallar la TDF de $x(n) = [-3, 5, 4, 6]$

$N = 4$, de la ecuación 1.1

$$c(k) = \sum_{n=0}^3 x(n) e^{\frac{-2\pi jkn}{4}}$$

$$c(0) = \sum_{n=0}^3 x(n) e^0$$

$$c(0) = x(0) + x(1) + x(2) + x(3)$$

$$c(0) = 12$$

$$c(1) = \sum_{n=0}^3 x(n) e^{\frac{-\pi jn}{2}}$$

$$c(1) = x(0) e^0 + x(1) e^{-\pi j/2} + x(2) e^{-\pi j} + x(3) e^{\frac{-3\pi j}{4}}$$

$$\begin{aligned} c(1) = x(0) + x(1) \left[\cos\left(\frac{\pi}{2}\right) - j \operatorname{sen}\left(\frac{\pi}{2}\right) \right] \\ + x(2) [\cos(\pi) - j \operatorname{sen}(\pi)] \\ + x(3) \left[\cos\left(\frac{3\pi}{4}\right) - j \operatorname{sen}\left(\frac{3\pi}{4}\right) \right] \end{aligned}$$

$$\begin{aligned} c(1) = -3 + 5 \left[\cos\left(\frac{\pi}{2}\right) - j \operatorname{sen}\left(\frac{\pi}{2}\right) \right] \\ + 4 [\cos(\pi) - j \operatorname{sen}(\pi)] \\ + 6 \left[\cos\left(\frac{3\pi}{4}\right) - j \operatorname{sen}\left(\frac{4\pi}{4}\right) \right] \end{aligned}$$

$$x(1) = -7.0 + 1.0i$$

$$c(2) = \sum_{n=0}^3 x(n) e^{-\pi j n}$$

$$c(2) = x(0) + x(1) e^{-\pi j} + x(2) e^{-2\pi j} + x(3) e^{-3\pi j}$$

$$c(2) = -3 + 5[\cos(\pi) - j \operatorname{sen}(\pi)] \\ + 4[\cos(2\pi) - j \operatorname{sen}(2\pi)] \\ + 6[\cos(3\pi) - j \operatorname{sen}(3\pi)]$$

$$c(2) = -10$$

$$c(3) = \sum_{n=0}^3 x(n) e^{-3\pi j n/2}$$

$$c(3) = x(0) + x(1) e^{-3\pi j/2} + x(2) e^{-3\pi j} \\ + x(3) e^{-9\pi j/2}$$

$$c(3) = -3 + 5[\cos(3\pi/2) - j \operatorname{sen}(3\pi/2)] \\ + 4[\cos(3\pi) - j \operatorname{sen}(3\pi)] \\ + 6[\cos(9\pi/2) - j \operatorname{sen}(4\pi/2)]$$

$$c(3) = -7.00 - 1.00i$$

$$c(k) = [12.00, -7.00 + 1.00i, -10, -7.00 - 1.00i]$$

Ejemplo 1.3.

Hallar la TDFI si $c(k) = [12.00, -7.00 - 1.00i, 4 - 10, -7.00 - 1.00i]$

$$x(n) = \frac{1}{N} * \sum_{k=0}^3 c(k) e^{\frac{-2\pi jkn}{4}}$$

$$x(0) = \frac{1}{4} * \sum_{k=0}^3 c(k) e^0$$

$$x(0) = \frac{1}{4} * [c(0) + c(1) + c(2) + c(3)] = -12/4$$

$$x(0) = -3$$

$$x(1) = \frac{1}{4} * \sum_{k=0}^3 c(k) e^{\frac{\pi jk}{2}}$$

$$x(1) = \frac{1}{4} * [c(0) e^0 + c(1) e^{\pi j/2} + c(2) e^{\pi} + c(3) e^{(3\pi j/4)}]$$

$$x(1) = \frac{1}{4} * [c(0) + c(1) \left[\cos\left(\frac{\pi}{2}\right) + j \operatorname{sen}\left(\frac{\pi}{2}\right) \right] + c(2) [\cos(\pi) + j \operatorname{sen}(\pi)] + c(3) \left[\cos\left(\frac{3\pi}{4}\right) + j \operatorname{sen}\left(\frac{3\pi}{4}\right) \right]]$$

$$\begin{aligned}
 x(1) &= 12 + (-7 + i) * \left[\cos\left(\frac{\pi}{2}\right) - j \operatorname{sen}\left(\frac{\pi}{2}\right) \right] \\
 &\quad - 10[\cos(\pi) - j \operatorname{sen}(\pi)] + (-7 - i) \\
 &\quad * \left[\cos\left(\frac{3\pi}{4}\right) - j \operatorname{sen}\left(\frac{4\pi}{4}\right) \right]
 \end{aligned}$$

$$x(1) = 5$$

$$x(2) = \frac{1}{4} * \sum_{k=0}^3 c(k) e^{\pi j k}$$

$$x(2) = \frac{1}{4} * [c(0) + c(1) e^{\pi j} + c(2) e^{2\pi j} + c(3) e^{3\pi j}]$$

$$\begin{aligned}
 x(2) &= \frac{1}{4} * [12 + (-7 + i) * [\cos(\pi) - j \operatorname{sen}(\pi)] - \\
 &\quad 10[\cos(2\pi) - j \operatorname{sen}(2\pi)] + (-7 - i)[\cos(3\pi) - \\
 &\quad j \operatorname{sen}(3\pi)]]
 \end{aligned}$$

$$x(2) = 4$$

$$x(3) = \frac{1}{4} * \sum_{k=0}^3 x(n) e^{2\pi j k}$$

$$x(3) = \frac{1}{4} * [c(0) + c(1) e^{2\pi j} + c(2) e^{4\pi j} + c(3) e^{6\pi j}]$$

$$x(3) = \frac{1}{4} * [12 + (-7 + i) * [\cos(2\pi) - j \operatorname{sen}(2\pi)] \\ + 10[\cos(4\pi) - j \operatorname{sen}(4\pi)] + (-7 - i) \\ * [\cos(6\pi) - j \operatorname{sen}(6\pi)]]$$

$$x(3) = 6$$

Entonces la transformada de Fourier inversa está dada por:

$$x(n) = [-3, 5, 4, 6]$$

Ejemplo 1.4.

Hallar la TDF de $x(n) = \cos(2\pi n 80)$; para $N=7$ y una frecuencia de muestro de 1Khz.

Utilizando la misma estructura del ejemplo anterior y resolviendo los coeficientes que en este caso son 7, se obtiene que:

$$c(k) = \sum_{n=0}^6 X(n) e^{\frac{-2\pi jkn}{7}}$$

$$c(0) = \sum_{n=0}^6 x(n) e^0 = 3.9421$$

Seguidamente se calcula el valor de $c(1)$..., ..., $c(6)$ y se obtiene:

$$c(k) = [3.9421, \quad -1.6546 - 0.5692i, \\ -0.2390 - 0.1633i, -0.0775 \\ - 0.0449i, \quad -0.0775 + 0.0449i, \\ -0.2390 + 0.1633i, \\ -1.6546 + 0.5692i]$$

Hasta el momento se ha realizado ejercicios que contribuyen a la destreza del cálculo matemática de la TDF.

Ejemplo 1.5.

Si una señal fue muestreada a 8Khz, tiene una única componente de frecuencia, el tiempo de adquisición es de 2 segundos, y la respuesta en frecuencia

muestra que el pico de máxima amplitud está en 300 puntos, ¿qué componente de frecuencia real tiene la señal?

Para dar solución a este tipo de problema, se debe tener en cuenta que la frecuencia de muestreo equivale a longitud máxima. En este caso la longitud del vector es: 16000 puntos, entonces:

$$\begin{array}{l} 16000 \text{ puntos} \rightarrow 8\text{Khz} \\ 300 \text{ puntos} \rightarrow f.\text{real} \end{array}$$

$$f.\text{real} = 150\text{Hz}$$

De esta manera es el análisis verdadero de la transformada de Fourier en datos reales.

Gráficamente sería:

$$F_s = 8\text{Khz}$$

$$t = 0: 1/8000: (2 - 1/8000);$$

$$y = \sin(2 * \pi * t * 150);$$

En la Figura 1.3, se muestra la respuesta gráfica del ejercicio para validar con la teoría aplicada.

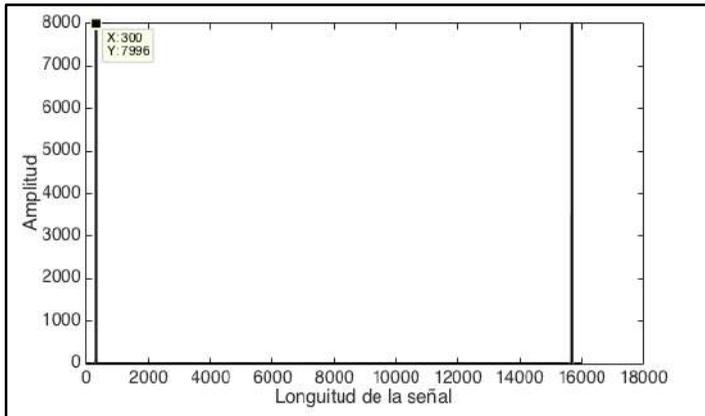


Figura 1.3.
Transformada de Fourier.

Ejemplo 1.6.

Calcular la transformada de Fourier de la suma de 3 señales senoidales con frecuencias: 120Hz, 310Hz y 420Hz. Las señales tienen una longitud de 9000 puntos y fueron adquiridas durante 3 segundos.

Para resolver este punto se calcula inicialmente la frecuencia de muestreo F_s , la cual está consiste en dividir la longitud máxima entre el tiempo. Realizando este cálculo se concluyó que $f_s = 3\text{Khz}$. Entonces la TDF tiene una respuesta, ver Figura 1.4:

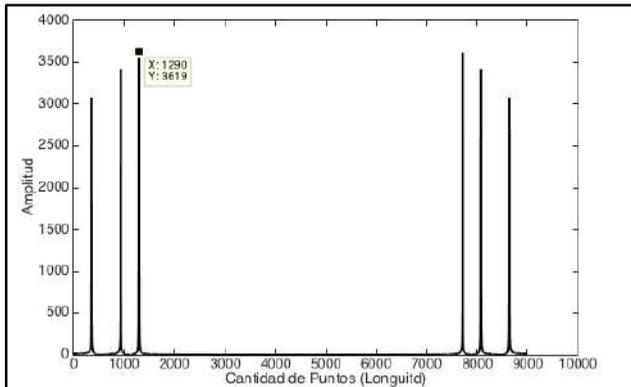


Figura 1.4.
TDF de la suma de 3 señales senoidales.

Aquí el pico ubicado en una posición de 1290, equivale a una frecuencia real de 310 Hz. Es importante mencionar que la transformada de

Fourier es simétrica y por ende se observan picos duplicados, la simetría en este ejercicio se encuentra en la posición 4500, que equivale a tener una frecuencia de 1500hz.

Ejemplo 1.7.

Calcula el rango de frecuencias real de una señal de electrocardiografía (ver Figura 1.5), que fue muestreada a una $f_s = 1Hz$.

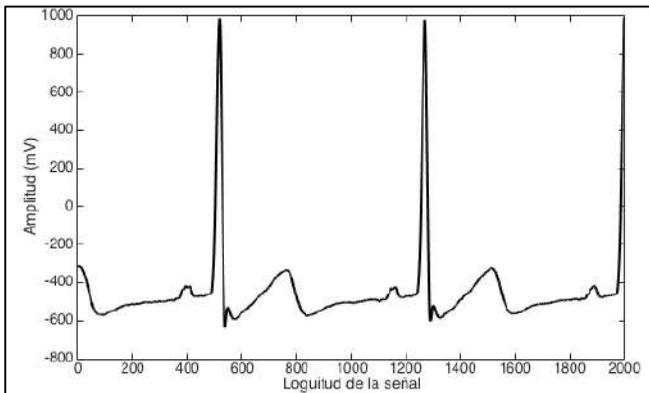


Figura 1.5.
Señal Electrocardiográfica.

La respuesta en frecuencia de señal se observa en la

Figura 1. 6 Nótese como la cantidad de frecuencia con mayor amplitud esta alrededor de cero eso indica que las frecuencias de la señal, tiene valores pequeños.

Entonces para responder que rango de frecuencias se ubica la posición donde los valores de amplitud de las componentes se hacen casi cero. Para este ejercicio se eligió el valor 85, es decir que el rango estará entre 0 y 85 puntos, que equivalente a tener frecuencias entre 0Hz – 127hz, rango que es totalmente coherente con la teoría o rangos de frecuencias que presenta una señal fisiológica de este tipo que oscila entre 0.05Hz y 150Hz.

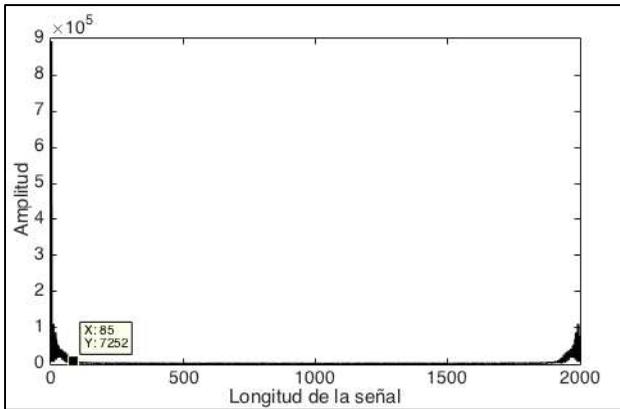


Figura 1. 6.
Transformada de Fourier señal electrocardiográfica.

Es importante resaltar que la transformada de Fourier, además de ejecutar en su mejor expresión las componentes de frecuencia, se puede utilizar para otras aplicaciones tales como: extracción de patrones.

1.4 Estructura matemática de la TDF TC- 1D.

Matemáticamente la TDF TC-1D esta expresada como $c(k)$ [21-27]:

$$c1(k) = \sum_{n=0}^{N-1} x1(n) * e^{-2*pi*n*k/N} * \varnothing(n - k) \quad (1.3)$$

Aquí, $x(n)$, es el vector original, N = longitud del vector y n la posición. $* \varnothing(n)$, se conoce como la ventana de multiplicación que es la encargada de darle resolución en tiempo a la transformada de Fourier. Nótese además que dicha ventana tiene la propiedad de multiplicación y de desplazamiento en el tiempo que está descrita por la letra k . Dicho desplazamiento se realiza según característica de la ventana.

Por otro lado, la estructura matemática de la transformada de Fourier en tiempo corto inversa (**TDFI-TC - 1D**), se describe como $x_1(n)$ [25-27]:

$$x_1(n) = \frac{1}{N} * \sum_{k=0}^{N-1} c1(k) * \varnothing(n - k) * e^{2*pi*k*n/N} \quad (1.4)$$

Aquí es importante mencionar que la aplicación de dichas estructuras matemáticas conduce a pasar del dominio de la resolución tiempo- frecuencia al

domino único del tiempo. A continuación, se presenta un ejemplo grafico del funcionamiento de la TDF y la TDF-TC. Si es usada la TDF la respuesta es una consolidación de las 3 frecuencias, sin conocer posibilidad del tiempo en que se producen dichas frecuencias (ver Figura 1.7). Por otro lado, la Figura 1.8 muestra la aplicación de la TDF-TC usando una ventana de 4500 datos, y desplazamientos de 4500 datos, esto quiere decir que deben aparecer 6 respuestas diferentes de la TDF. Nótese como en alguna de ellas aparecen respuestas diferentes lo cual permite mencionar que, por ejemplo, aparece una única frecuencia con un valor de 100Hz en un intervalo de muestras de 1 a 4500. Finalmente se muestra una componente de frecuencia de 400Hz en un intervalo de muestra de 25500 a 27000. Aquí se debe mencionar que el funcionamiento de la TDF-TC, permite realizar análisis más focalizados que lleven a encontrar

patrones y características importantes a diferencia de la TDF.

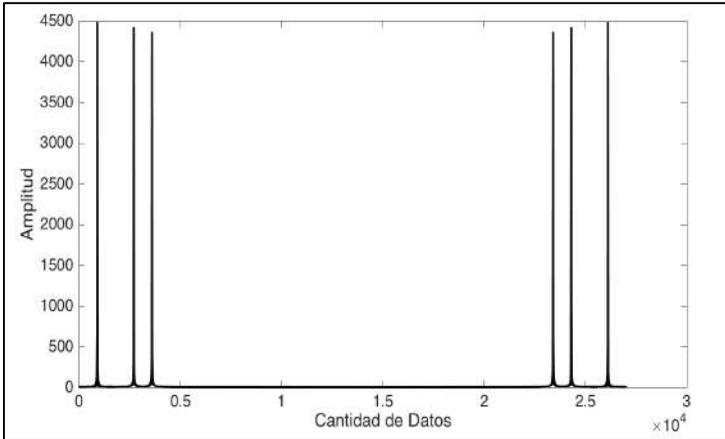


Figura 1.7.
TDF de una señal senoidal con 3 frecuencias diferentes.

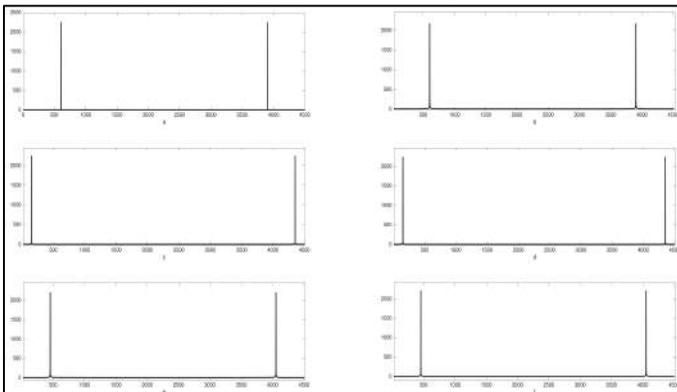


Figura 1.8.

TDF en tiempo corto de una señal senoidal con 3 frecuencias diferentes.

1.5 Estructura de matemática de la TDF- 2D. y TDFI-2D.

Como es común encontrar datos representados en dos dimensiones, se hace importante conocer la estructura matemática de la TDF-2D, la cual es utilizada para los datos representados en forma de matrices [28-33].

Matemáticamente la TDF-2D, esta expresada como $c(k, l)$

$$c(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n, m) * e^{-2*pi*n*k/N} * e^{-2*pi*m*l/M} \quad (1.5)$$

Donde $c(k, l)$, es la TDF-2D de $x(n, m)$, N es el número de filas y M es el número de columnas. Al igual que la TDF-1D, al aplicarse esta estructura matemática a una matriz $x(n, m)$, se obtiene las

componentes de frecuencia de una imagen o matriz. Aquí también se presenta la simetría de la TDF, ya que como se observa en la ecuación (1. 5), $c(k,l)$ es compleja y por ende debe obtenerse el valor absoluto o magnitud de cada dato. Esta estructura matemática contempla dos sumatorias esto es porque una sumatoria depende de las filas de la matriz y la otra depende de las columnas de la matriz [34-38].

Dentro del término transformada, existe la denominación de inversa, lo cual quiere decir que existe una estructura matemática que pasa del espacio de frecuencia al espacio del tiempo. En este caso es representada su estructura matemática de la siguiente manera [34-39].

$$x(n, m) = \frac{1}{N * M} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} c(k, l) * e^{2*pi*n*k/N} * e^{2*pi*m*l/M} \quad (1. 6)$$

La ecuación (1. 6), presenta la forma de cómo pasar del dominio de la frecuencia al dominio del tiempo. Es importante resaltar que se obtiene la señal $x(n, m)$ original solo cuando $c(k, l)$ no se modifica en ningún caso. Finalmente es de mencionar que el tamaño de $x(n, m)$ debe ser igual al tamaño de $c(k, l)$.

A continuación, se presentan algunos ejemplos que explicarán de mejor manera el funcionamiento de la ecuación (1. 5).

1.6 Ejemplos matemáticos de la TDFI-2D y desarrollo de ejercicios en datos reales.

Ejemplo 1.8.

$$\text{si } X(n, m) = \begin{bmatrix} 2 & -1 \\ 3 & 4 \end{bmatrix} \text{ hallar } x(k, l)$$

$$c(k, l) = \sum_{n=0}^1 \sum_{m=0}^1 X(n, m) e^{-\pi j n k} e^{-\pi j n l}$$

$$c(0,0) = \sum_{n=0}^1 \sum_{m=0}^1 X(n, m)$$

$$c(0,0) = \sum_{n=0}^1 [x(n, 0) + x(n, 1)]$$

$$c(0,0) = x(0,0) + x(0,1) + x(1,0) + x(1,1)$$

$$c(0,0) = 8$$

$$c(0,1) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi jm}$$

$$c(0,1) = \sum_{n=0}^1 [x(n, 0) e^0 + x(n, 1) e^{-\pi j}]$$

$$c(0,1) = x(0,0) + x(0,1)e^{-\pi j} + x(1,0) + x(1,1) e^{-\pi j}$$

$$c(0,1) = 5 + 3 [\cos \pi - j \sin \pi]$$

$$c(0,1) = 2$$

$$c(1,0) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi jn}$$

$$c(1,0) = \sum_{n=0}^1 [(X(n, 0) + X(n, 1)) e^{-\pi jn}]$$

$$c(1,0) = x(0,0) + x(0,1) + [x(1,0) + x(1,1)] e^{-\pi j}$$

$$c(1,0) = 1 + [7](\cos \pi - j \sin \pi)$$

$$c(1,0) = -6$$

$$c(1,1) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi jn} e^{-\pi jm}$$

$$c(1,1) = \sum_{n=0}^1 (x(n, 0) + x(n, 1) e^{-\pi j}) e^{-\pi jn}$$

$$c(1,1) = x(0,0) + x(0,1) e^{-\pi j} + (x(1,0) + x(1,1) e^{-\pi j}) e^{-\pi j}$$

$$c(1,1) = 2 - e^{-\pi j} + (3 + 4e^{-\pi j})e^{-\pi j}$$

$$c(1,1) = 4$$

Entonces $c(k, l) = \begin{bmatrix} 8 & 2 \\ -6 & 4 \end{bmatrix}$, que equivale a resolver la transformada de Fourier en 2D.

Ejemplo 1.9.

si $x(n, m) = \begin{bmatrix} 1 & 3 \\ 9 & 5 \end{bmatrix}$ hallar $c(k, l)$

$$c(k, l) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi j n k} e^{-\pi j n l}$$

$$c(0,0) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m)$$

$$c(0,0) = \sum_{n=0}^1 [x(n, 0) + x(n, 1)]$$

$$c(0,0) = x(0,0) + x(0,1) + x(1,0) + x(1,1)$$

$$c(0,0) = 18$$

$$c(0,1) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi j m}$$

$$c(0,1) = \sum_{n=0}^1 [x(n, 0) e^0 + x(n, 1) e^{-\pi j}]$$

$$c(0,1) = x(0,0) + x(0,1)e^{-\pi} + x(1,0) + x(1,1) e^{-\pi}$$

$$c(0,1) = 5 + 3 [\cos \pi - j \sin \pi]$$

$$c(0,1) = 2$$

$$c(1,0) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi j n}$$

$$c(1,0) = \sum_{n=0}^1 [(x(n, 0) + x(n, 1))] e^{-\pi j n}$$

$$c(1,0) = x(0,0) + x(0,1) + [x(1,0) + x(1,1)] e^{-\pi j}$$

$$c(1,0) = 1 + [7](\cos \pi - j \sin \pi)$$

$$c(1,0) = -10$$

$$c(1,1) = \sum_{n=0}^1 \sum_{m=0}^1 x(n, m) e^{-\pi j n} e^{-\pi j m}$$

$$c(1,1) = \sum_{n=0}^1 (x(n, 0) + x(n, 1) e^{-\pi j}) e^{-\pi j n}$$

$$c(1,1) = x(0,0) + x(0,1) e^{-\pi j} + (x(1,0) + x(1,1) e^{-\pi j}) e^{-\pi j}$$

$$c(1,1) = 2 - e^{-\pi j} + (3 + 4e^{-\pi j})e^{-\pi j}$$

$$c(1,1) = -6$$

Entonces $c(k, l) = \begin{bmatrix} 18 & 2 \\ -10 & -6 \end{bmatrix}$, que equivale a resolver la transformada de Fourier en 2D.

Ejemplo 1.10.

Hallar TDF-2D de $x(n, m) = \begin{bmatrix} -1 & 2 & 7 \\ 2.5 & 8 & -6 \\ 1.6 & 9 & 12 \end{bmatrix}$

Para resolver este ejercicio se procede de manera similar al ejercicio anterior solo se debe tener en cuenta el tamaño de la matriz.

$$c(k, l) = \sum_{n=0}^2 \sum_{m=0}^2 X(n, m) e^{-2\pi jnk/3} e^{-\pi 2jnl/3}$$

$$c(0,0) = \sum_{n=0}^2 \sum_{m=0}^2 X(n, m)$$

$$c(0,0) = \sum_{n=0}^2 [x(n, 0) + x(n, 1) + x(n, 2)]$$

$$c(0,0) = x(0,0) + x(0,1) + x(0,2) + x(1,0) + x(1,1) + x(1,2) + x(2,0) + x(2,1) + x(2,2)$$

$$c(0,0) = 8$$

$$c(0,1) = \sum_{n=0}^2 \sum_{m=0}^2 X(n, m) e^{-\pi 2jm/3}$$

$$c(0,2) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-\pi 4jm/3}$$

$$c(1,0) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-2n\pi j/3}$$

$$c(1,1) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-2\pi jn/3} e^{-2\pi jm/3}$$

$$c(1,2) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-2\pi jn/3} e^{-4\pi jm/3}$$

$$c(2,0) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-4\pi jm/3}$$

$$c(2,1) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-4\pi jn/3} e^{-2\pi jm/3}$$

$$c(2,2) = \sum_{n=0}^2 \sum_{m=0}^2 x(n, m) e^{-4\pi jn/3} e^{-4\pi jm/3}$$

Calculando matemáticamente los 9 coeficientes se obtiene:

$$c(k, l) = \begin{bmatrix} 35.1 & -12.90 - 5.19i & -12.90 + 5.19i \\ -5.55 + 15.67 & -14.55 + 0.086i & 10.95 - 18.09 \\ -5.55 - 15.65i & 10.95 + 18.09i & -14.44 - 0.086i \end{bmatrix}$$

Nótese que la respuesta es una transformada con valores complejos. Hasta el momento se ha repasado como resolver la estructura matemática.

Ejemplo 1.11.

$$\text{si } c(k, l) = \begin{bmatrix} 18 & 2 \\ -10 & -6 \end{bmatrix} \text{ hallar } x(n, m)$$

$$x(n, m) = \frac{1}{4} * \sum_{k=0}^1 \sum_{l=0}^1 c(k, l) e^{\pi jnk} e^{\pi jnl}$$

$$x(0,0) = \frac{1}{4} * \sum_{k=0}^1 \sum_{l=0}^1 c(k, l)$$

$$x(0,0) = \frac{1}{4} * \sum_{k=0}^1 [c(k, 0) + c(k, 1)]$$

$$x(0,0) = \frac{1}{4} * [c(0,0) + c(0,1) + c(1,0) + c(1,1)]$$

$$x(0,0) = 1$$

$$x(0,1) = \frac{1}{4} * \sum_{k=0}^1 \sum_{l=0}^1 c(k, l) e^{\pi jl}$$

$$x(0,1) = \frac{1}{4} * \sum_{k=0}^1 [c(k, 0) e^0 + c(k, 1) e^{\pi j}]$$

$$x(0,1) = \frac{1}{4} * [c(0,0) + c(0,1)e^{\pi} + c(1,0) + c(1,1) e^{\pi}]$$

$$x(0,1) = 3$$

$$x(1,0) = \frac{1}{4} * \sum_{k=0}^1 \sum_{l=0}^1 c(k, l) e^{\pi jk}$$

$$x(1,0) = \frac{1}{4} * \sum_{k=0}^1 [(c(k, 0) + c(k, 1)) e^{\pi jk}]$$

$$x(1,0) = \frac{1}{4} * [c(0,0) + c(0,1) + [c(1,0) + c(1,1)] e^{\pi j}]$$

$$x(1,0) = 9$$

$$x(1,1) = \frac{1}{4} * \sum_{k=0}^1 \sum_{l=0}^1 c(n, m) e^{\pi jk} e^{\pi jl}$$

$$x(1,1) = \frac{1}{4} * \sum_{k=0}^1 (c(k, 0) + c(k, 1) e^{\pi j}) e^{\pi jk}$$

$$x(1,1) = \frac{1}{4} * [c(0,0) + c(0,1) e^{\pi j} + (c(1,0) + c(1,1) e^{\pi j}) e^{\pi j}]$$

$$x(1,1) = 20/4 = 5$$

Entonces $x(n, m) = \begin{bmatrix} 1 & 3 \\ 9 & 5 \end{bmatrix}$, que equivale a resolver la transformada de Fourier inversa en 2D. Lo que concluye que es posible conseguir los datos de la imagen original haciendo uso de la TDFI-2D.

Ejemplo 1.12.

Calcular la transformada de Fourier en 2D de una imagen que contenga una frecuencia única de 81HZ, (ver Figura 1.9).

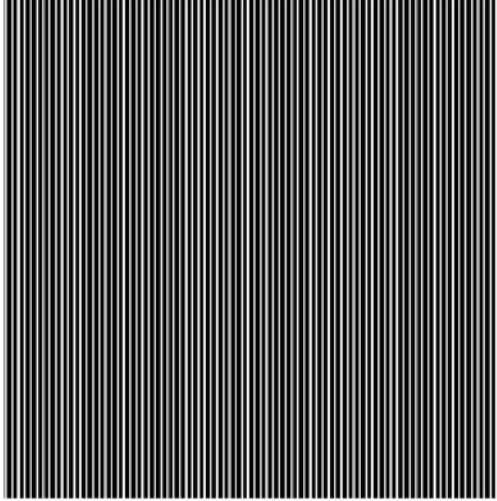


Figura 1.9.
Imagen de 80Hz, tamaño 512*512.

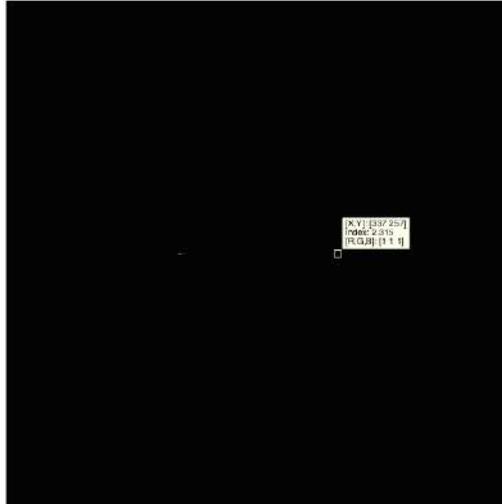


Figura 1.10.
Transformada de Fourier de una señal senoidal.

La

Figura 1.10 muestra la transformada de la una señal real con frecuencia de 81Hz. Para conocer la frecuencia real basta con restar el valor donde está el punto color blanco menos el número de columnas/2. Entonces si la imagen tiene 512 columnas y el punto se encuentra en la posición 337 como se observa, para calcular la frecuencia se resta 337-256, entonces la frecuencia de la señal que se

tiene es de 81Hz, tal como es el caso de la limitante incluida en el párrafo del ejercicio. Con esto se puede mencionar que la TDF-2D, calcula las componentes de frecuencia de una imagen senoidal.

Ejemplo 1.13.

Calcular la transformada de Fourier una imagen que contenga dos frecuencias: 75hz y de 100hz.

La

Figura 1.11 muestra la imagen original, con frecuencias 75 y 100hz, a la cual se le calca la transformada de Fourier en 2D.

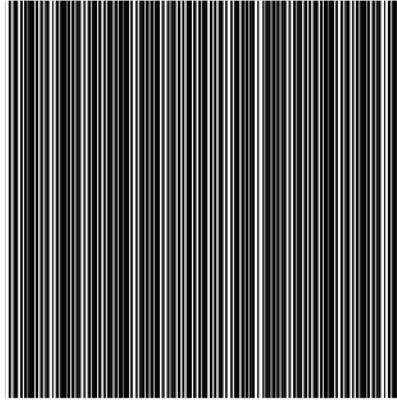


Figura 1.11.
Imagen senoidal, con frecuencias de 75hz y 100hz.

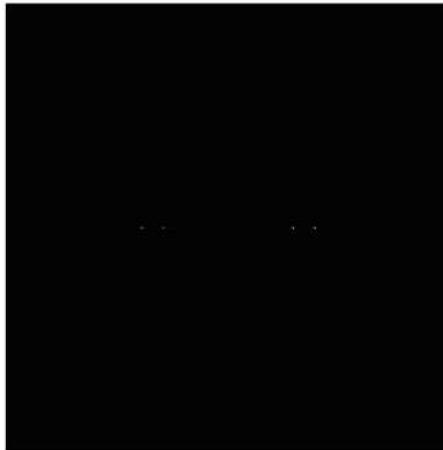


Figura 1.12.
Transformada de Fourier de señales senoidales con 2
frecuencias 75 y 100Hz.

En la

Figura 1.12 se muestra la transformada de una imagen con frecuencias 75Hz y 100hz. Nótese como existen 4 puntos de color blanco, aquí se observa la propiedad de simetría. Para el análisis solo se toman los puntos que se observan a la derecha del eje de simetría, el cual es el número de columnas dividido entre 2.

En los ejercicios anteriores se ha explicado como es el funcionamiento de la transformada de Fourier matemáticamente y así mismo en imágenes controladas en frecuencia. Los ejercicios a continuación son imágenes tomadas en cámaras digitales (ver Figura 1.13).

Ejemplo 114.

Calcular la Transformada de Fourier de imagen capturada conocida como Leonor.



Figura 1.13.
Imagen a nivel de gris Leonor.

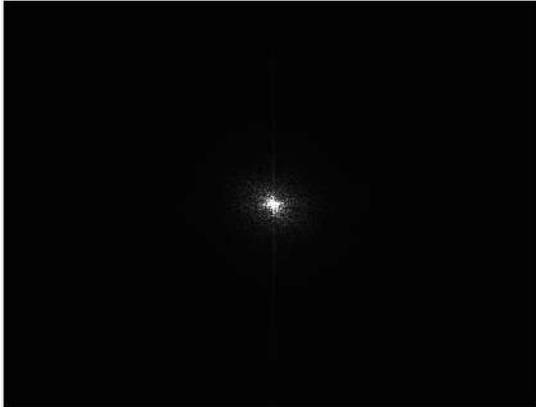


Figura 1.14.
Transforma de Fourier imagen Leonor.

En la

Figura 1.14, se observa la transformada de Fourier de una imagen a nivel de gris conocida como Leonor, nótese como las componentes de frecuencia son muy diferentes a una imagen con frecuencias únicas, aquí se puede observar que la imagen Leonor contiene componentes de frecuencias variantes. Sin embargo, no es posible hacer el cálculo respecto porque no se cuenta con los datos necesarios como por ejemplo tiempo y frecuencia de muestreo. Aquí se desea mostrar cómo es la transformada de Fourier de una imagen cualquiera.

Ejemplo 1.15.

Hallar la transformada de Fourier de una imagen que contenga una figura geométrica (ver Figura 1.15).

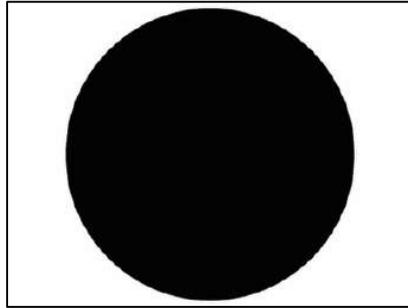


Figura 1.15.
Imagen figura geométrica.

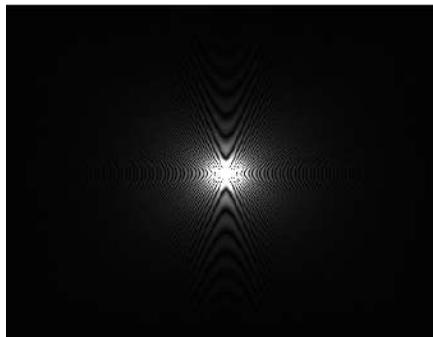


Figura 1.16.
Transformada de Fourier figura geométrica.

La
Figura 1.16, muestra la TDF-2D de la figura geométrica, este ejemplo permite identificar la forma de onda de la TDF-2D de esta figura, con el objetivo

de conocer de manera real el comportamiento frecuencia de imágenes reales.

Con los ejercicios anteriormente presentados, se puede observar que la TDF-2D, es diferente para todo el conjunto de datos debido al comportamiento frecuencial de cada imagen. Por otro lado, es importante mencionar que con el uso de la transformada se puede realizar diferentes aplicaciones como se menciona a continuación.

1.7 Aplicaciones en el procesamiento de datos

En esta sección se presenta, una serie de ejemplos y aplicaciones reales de la TDF. En este caso las aplicaciones están concentradas en: Eliminación de ruido, Compresión de voz, y Extracción de Patrones para clasificación de datos. Se trabajarán imágenes de angiografía coronaria, de resonancia magnética y señales de voz. Además, se muestran resultados de imágenes inéditas de los autores, donde se aplica la TDF. Así mismo este capítulo muestra los códigos de cada ejemplo, esto con el objetivo de presentar a

los lectores la oportunidad de simular y obtener los mismos resultados [20-41].

1.7.1 Extracción de patrones

El objetivo es buscar componentes de frecuencia que logran diferenciar un conjunto de datos o que permitan mapear el conjunto de datos iniciales que no son linealmente separables a un conjunto de datos linealmente separable de ser el caso. El espacio de la transformada debe descubrir elementos que se consideren esenciales o únicos en cada conjunto de datos.

Ejemplo 1.16:

A continuación, se muestra un ejemplo de la transformada de Fourier para dos señales de voz: casa y carro. Ver

Figura 1.17 y

Figura 1.18.

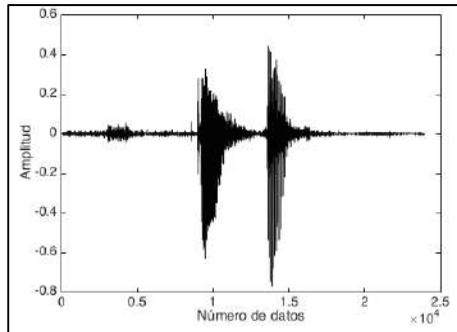


Figura 1.17.
Señal de voz palabra 'CASA'.

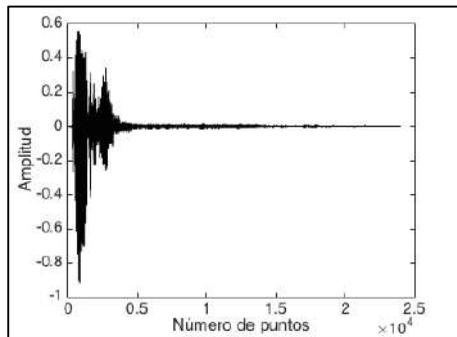


Figura 1.18.
Señal de voz palabra 'CARRO'.

A continuación, se muestran las TDF de las señales

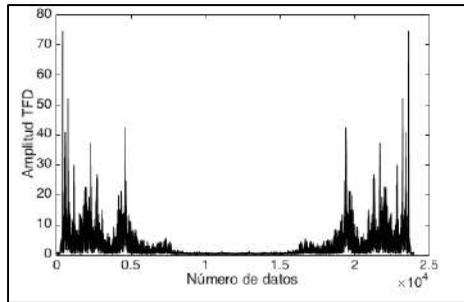


Figura 1.19.
Transformada de Fourier señal de voz CASA.

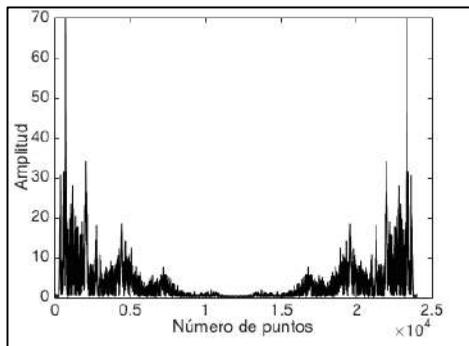


Figura 1.20.
Transformada de Fourier señal de voz CARRO.

Realizando un análisis visual de la
Figura 1.19 y la

Figura 1.20, se puede concluir que la TDF de la palabra Carro, tiene mucha más energía en la zona ubicada entre los puntos 2000 y 4000, aquí se podría ubicar en primera instancia un patrón. Es importante mencionar que para tener un patrón con mayor precisión se deben tener por lo menos 50 señales de cada grupo, es decir 50 señales de la palabra **CASA** y 50 señales de la palabra **CARRO**. En este libro solo se hace el énfasis en cómo obtener patrones diferenciadores usando TDF.

Ejemplo 1.17.

Extracción patrones: Huellas





Figura 1.21.
TDF-2D imágenes de huellas digitales.

La

Figura 1.21, muestra las TDF, para imágenes de huellas digitales, nótese como, analizando las TDF, se muestra que tiene componentes de frecuencias diferentes, esto permiten evidenciar que es posible utilizar esta herramienta matemática para extraer patrones característicos de las imágenes mostradas. Este es otro ejemplo de que es posible extraer patrones en datos en dos dimensiones. Cabe aclarar que al igual en que los ejemplos anteriores es importante tener un conjunto de datos amplio, en este libro se propone tener más de 50 imágenes de

cada grupo, y así conseguir un patrón confiable para caracterizar cada huella digital. El objetivo de la extracción es utilizar una zona donde identifique cada huella, realizar un recorte que sea el patrón característico.

Otro ejemplo se da en el caso que se desee realizar extracción del patrón para clasificar figuras geométricas. Nótese en la Figura 1.22, como cada geometría tiene una caracterización en frecuencia diferente. Se observa de manera clara que cada figura geométrica tiene un comportamiento en el espacio de Fourier diferente.

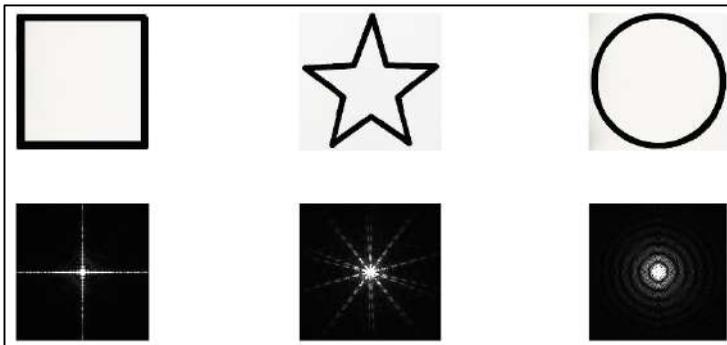


Figura 1.22.
TDF de figuras geométricas.

Ejemplo 1.18.

Eliminación de ruidos y reconstrucción de imágenes

El proceso de reconstrucción de imágenes o eliminación de información poco relevantes basado en la transformada de Fourier discreta, está basada en la teoría que se condensa en los siguientes pasos:

1. Selección de la imagen a procesar, identificando en el espacio del tiempo que se desea realizar, la identificación de la información tiene que ver con la distribución de frecuencias en la TDF.
2. Aplicar la transformada discreta de Fourier.
3. Identificar las zonas que se desean modificar en el espacio de la transformada, ya sea haciéndolas cero o aplicando algún método como promedio, máximo, mínimo, desviación estándar entre otras técnicas que permiten

modificar información teniendo en cuenta los datos presentes en las imágenes. Se recomienda que la modificación esté basada en ventanas de tamaño mínimo 8×8 .

4. Se aplica la transformada de Fourier en tiempo discreto, con el fin de lograr observar los cambios realizados en el espacio de la frecuencia y su efecto en el espacio del tiempo o espacio de la imagen original.

1.7.2 Eliminación de información

A continuación, se muestra un ejemplo real de filtrado de imágenes basado en TDF.

La

Figura 1.23, muestra la imagen a filtrar, nótese como esta imagen tiene unas componentes horizontales que se denominan ruido y así mismo alteran el espacio original de la imagen.



Figura 1.23.
Figura imagen Lena con líneas.

La

Figura 1. 24a, representa la TDF de la imagen de líneas, y la

Figura 1. 24b, es la modificación de componentes de frecuencia de la TDF original. Aquí se realizó un proceso donde las posiciones comprendidas en el rango de fila 50 hasta la fila 463, y la totalidad de las columnas se ponen a cero. Notese que realizando este proceso se conserva la teoría de simetría, que trae consigo la transformada de Fourier. Es

importante resaltar que este teorema en cualquier procesos de reconstrucción o filtrado se debe mantener la simetría de la TDF.

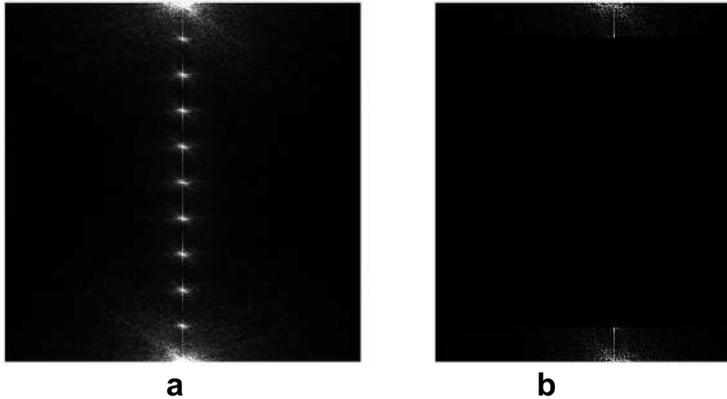


Figura 1. 24.

- a) TDF imagen Lena con líneas.
 b) TDF con eliminación de componentes de frecuencia.

La

Figura 1.25, Muestra el resultado de aplicar la transformada de Fourier discreta inversa a la imagen de la figura 1.24b. Nótese que el efecto producido de poner a cero ciertas posiciones en dicha imagen hacen que en un gran porcentaje las líneas

presentes en la imagen original se hicieran prácticamente nulas, el efecto del por qué no se conserva la nitidez de la imagen está dada a y que la imagen contiene información importante en las zonas que se colocaron acero, esto puede concluir que se eliminaron componentes de frecuencia importantes que hacen que la resolución de la imagen se altere. Para evitar esto se debe identificar muy bien las zonas cuyo valor será cero.



Figura 1.25.
Imagen Lena reconstruida.

La

Figura 1.26, muestra la TDF, de la misma imagen de la

Figura 1.23, pero las zonas que fueron puestas a cero han sido menos, fila 100 hasta la fila 412 y la totalidad de las columnas se ponen a cero. Esto indica que menos información se ha modificado.

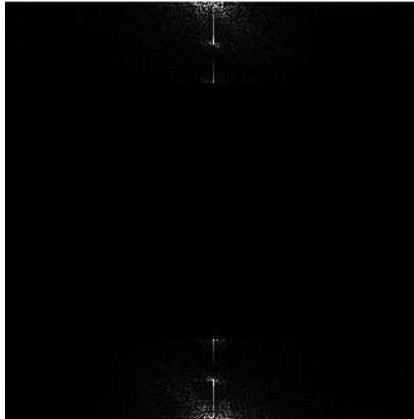


Figura 1.26.
Figura imagen Lena reconstruida.

Finalmente, la

Figura 1.27 muestra el resultado de la reconstrucción de la imagen usando la transformada de Fourier inversa de la

Figura 1.26. Nótese que por el hecho de que las zonas fueron eliminadas en menor proporción las líneas horizontales, se han difuminado en una medida no tan notoria como en la

Figura 1.25. El trabajo de investigación está enfocado en encontrar las zonas en la TDF que serán modificadas o alteradas para conseguir ya sea eliminar ruido o reconstruir imágenes.

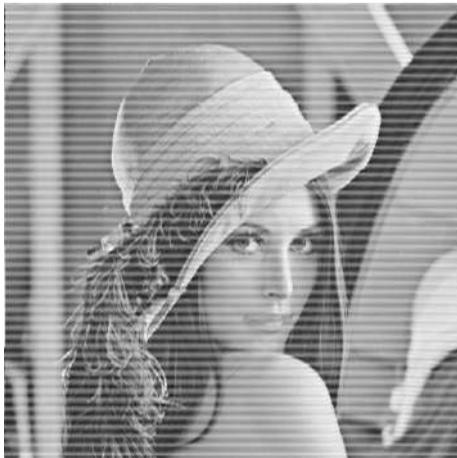


Figura 1.27.

Figura imagen Lena reconstruida.

Se recomienda el uso adecuado de la TDF, con el fin de preservar la simetría, identificar las zonas a modificar, elegir o probar las técnicas posibles en el proceso de modificación de la imagen en el espacio de la frecuencia. De esta manera se garantizará un ejercicio mucho más eficiente y con resultados justificables en el desarrollo y uso de la transformada de Fourier para aplicaciones diferentes a calcular las componentes de frecuencia.

Para finalizar este capítulo y consolidar el conocimiento y aplicabilidad de la TDF, se presentan algunos ejemplos propuestos.

1.8 Ejercicios propuestos

- 1) Hallar la TDF de $x(n) = [1, 4, 9, 3, 4]$;
- 2) hallar la TDF de $x(n) = [1, 1, 1, 1, 1, 1,]$;
- 3) Hallar la TDF de $x(n) = [1, 4, -9, -3, 4, 3 + 5j]$;

4) Hallar la TDFI de $x(k) = [2 + 4j, 3j, 4 - 7j, 5, 6 - 9j]$;

5) Hallar la TDFI de $x(k) = [3, 5 - 6j, 1 - 0j, 3.4 + 4j - 9j]$;

3) Hallar la TDFI de $x(k) = [-4j, 5 - 2j]$;

4) Hallar gráficamente la TDF de una señal de voz, con una $f_s = 8\text{Khz}$, una duración de 2.5 segundos. Si desea tener esta señal escribanos a luis.mendoza@unipamplona.edu.co

5) Hallar gráficamente la TDF, de una señal ECG, la cual tiene una frecuencia de muestreo de 1Khz. Y su duración de adquisición de 20 segundos. Si desea tener esta señal escribanos a luis.mendoza@unipamplona.edu.co

6) Hallar gráficamente la TDF de una señal metabólica*. Analizar el rango de frecuencias más significativas

7) Hallar gráficamente la TDF de una señal de electroforesis capilar*. Analizar el rango de frecuencias más significativas.

8) Hallar la TDFI de una señal, en la cual el 20% de los datos se colocaron a cero, la selección de datos a poner a cero está dado por las componentes de frecuencia más alta posible.

9) Hallar TFD-2D de $c(n, m) = \begin{bmatrix} 3 & -2 & -1 \\ -8 & 6 & 2 \\ -4 & 9 & 5 \end{bmatrix}$

10) Hallar TFD-2D de $c(n, m) = \begin{bmatrix} 1 & 4 & -1.4 \\ -6 & -2 & 8.3 \\ 7 & -3 & 10.4 \end{bmatrix}$

11) Hallar TFD-2D de $c(k, l) = \begin{bmatrix} 3 & 9 \\ 4 & -5 \\ -2 & 1 \end{bmatrix}$

12) Encontrar la transformada de Fourier de una imagen que contenga una frecuencia única de valor 120Hz, frecuencias de muestreo 512Hz y el tamaño 512*512 pixeles.

13) Demostrar la propiedad de linealidad usando la TDF-2D, utilice frecuencias de 320Hz para una imagen y 450Hz para la segunda imagen, y una frecuencia de muestreo de 2.5Khz.

14) Sumar dos señales con frecuencias diferentes. Utilice TDF-1D, para realizar un filtro que elimine la frecuencia de valor menor. Grafique la respuesta.

Referencias

1. Stark, R. W., & Heckl, W. M. (2000). Fourier transformed atomic force microscopy: tapping mode atomic force microscopy beyond the Hookian approximation. *Surface Science*, 457(1-2), 219-228.
2. Rockley, M. G. (1979). Fourier-transformed infrared photoacoustic spectroscopy of polystyrene film. *Chemical Physics Letters*, 68(2-3), 455-456.

3. Ozaktas, H. M., & Kutay, M. A. (2001, September). The fractional Fourier transform. In *2001 European Control Conference (ECC)*(pp. 1477-1483). IEEE.
4. Ozaktas, H. M., Arikan, O., Kutay, M. A., & Bozdagt, G. (1996). Digital computation of the fractional Fourier transform. *IEEE Transactions on signal processing*, 44(9), 2141-2150.
5. Weisstein, E. W. (2015). Fast fourier transform.
6. Weller, H. (2015). Fourier analysis.
7. Croft, A. (2017). *Engineering mathematics*. Pearson education.
8. Antoniou, A. (2016). *Digital signal processing*. McGraw-Hill.
9. Howell, K. B. (2016). *Principles of Fourier analysis*. CRC Press.
10. Sogge, C. D. (2017). *Fourier integrals in classical analysis* (Vol. 210). Cambridge University Press.
11. Bracewell, R. N., & Bracewell, R. N. (1986). *The Fourier transform and its*

applications (Vol. 31999). New York: McGraw-Hill.

12. Osgood, B. G. (2019). *Lectures on the Fourier Transform and Its Applications* (Vol. 33). American Mathematical Soc..
13. Campos, R. G. (2019). The Ordinary Discrete Fourier Transform. In *The XFT Quadrature in Discrete Fourier Analysis* (pp. 3-37). Birkhäuser, Cham.
14. Grigoryan, A. M., & Grigoryan, M. M. (2016). *Brief notes in advanced DSP: Fourier analysis with MATLAB*. CRC Press.
15. Olson, T. (2017). The Fourier Transform. In *Applied Fourier Analysis* (pp. 121-148). Birkhäuser, New York, NY.
16. Wang, S., Yang, M., Zhang, Y., Li, J., Zou, L., Lu, S., ... & Zhang, Y. (2016). Detection of left-sided and right-sided hearing loss via fractional Fourier transform. *Entropy*, 18(5), 194.
17. Satsangi, S., & Patvardhan, C. (2016). Application of Genetic Algorithm for Evolution of Quantum Fourier Transform Circuits. In *Proceedings of the Second International*

Conference on Computer and Communication Technologies (pp. 773-782). Springer, New Delhi.

18. Sundararajan, D. (2018). The Discrete Fourier Transform. In *Fourier Analysis—A Signal Processing Approach* (pp. 31-55). Springer, Singapore.
19. Yu, H., Lu, R., Han, S., Xie, H., Du, G., Xiao, T., & Zhu, D. (2016). Fourier-transform ghost imaging with hard X rays. *Physical review letters*, 117(11), 113901.
20. Turitsyn, S. K., Prilepsky, J. E., Le, S. T., Wahls, S., Frumin, L. L., Kamalian, M., & Derevyanko, S. A. (2017). Nonlinear Fourier transform for optical data processing and transmission: advances and perspectives. *Optica*, 4(3), 307-322.
21. Turitsyn, S. K., Prilepsky, J. E., Le, S. T., Wahls, S., Frumin, L. L., Kamalian, M., & Derevyanko, S. A. (2017). Nonlinear Fourier transform for optical data processing and transmission: advances and perspectives. *Optica*, 4(3), 307-322.

22. Hussein, H. J., Hadi, M. Y., & Hameed, I. H. (2016). Study of chemical composition of *Foeniculum vulgare* using Fourier transform infrared spectrophotometer and gas chromatography-mass spectrometry. *Journal of Pharmacognosy and Phytotherapy*, 8(3), 60-89.
23. Langel, W. (2016). Analysis of perturbed H₂O vibrations beyond Fourier transform. *arXiv preprint arXiv:1601.05007*.
24. Bülow, H. (2015). Experimental demonstration of optical signal detection using nonlinear Fourier transform. *Journal of Lightwave Technology*, 33(7), 1433-1439.
25. Koç, A., Bartan, B., Gundogdu, E., Çukur, T., & Ozaktas, H. M. (2017). Sparse representation of two-and three-dimensional images with fractional Fourier, Hartley, linear canonical, and Haar wavelet transforms. *Expert Systems with Applications*, 77, 247-255.
26. Ly, H. B., Monchiet, V., & Grande, D. (2016). Computation of permeability with Fast Fourier Transform from 3-D digital images of porous microstructures. *International Journal of Numerical Methods for Heat & Fluid Flow*, 26(5), 1328-1345.

27. Liu, J., Bai, T., Shen, X., Dou, S., Lin, C., & Cai, J. (2017). Parallel encryption for multi-channel images based on an optical joint transform correlator. *Optics Communications*, 396, 174-184.
28. Durande, M., Tlili, S., Homan, T., Guirao, B., Graner, F., & Delanoë-Ayari, H. (2019). Fast determination of coarse-grained cell anisotropy and size in epithelial tissue images using Fourier transform. *Physical Review E*, 99(6), 062401.
29. Yoshimasu, T., Kawago, M., Hirai, Y., Ohashi, T., Yata, Y., Fusamoto, A., ... & Nishimura, Y. (2017). P3. 13-012 Fast Fourier Transform Analysis for the Outline of Pulmonary Nodules on Computed Tomography Images. *Journal of Thoracic Oncology*, 12(11), S2320.
30. Wen, D., Yue, F., Ardron, M., & Chen, X. (2016). Multifunctional metasurface lens for imaging and Fourier transform. *Scientific reports*, 6, 27628.
31. Zhang, Y. D., Wang, S. H., Liu, G., & Yang, J. (2016). Computer-aided diagnosis of abnormal breasts in mammogram images by weighted-type fractional Fourier

- transform. *Advances in Mechanical Engineering*, 8(2), 1687814016634243.
32. Durande, M., Tlili, S., Homan, T., Guirao, B., Graner, F., & Delanoë-Ayari, H. (2019). Fast determination of coarse-grained cell anisotropy and size in epithelial tissue images using Fourier transform. *Physical Review E*, 99(6), 062401.
 33. Zhang, Y., Hu, Q., Guo, Z., Xu, J., & Xiong, K. (2018, June). Multi-Class Brain Images Classification Based on Reality-Preserving Fractional Fourier Transform and Adaboost. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)* (pp. 444-447). IEEE.
 34. McAndrew, A. (2015). *A computational introduction to digital image processing*. Chapman and Hall/CRC.
 35. Vyas, A., Yu, S., & Paik, J. (2018). *Multiscale Transforms with Application to Image Processing*. Springer Singapore.
 36. Singh, A. K., Kumar, B., Singh, G., & Mohan, A. (Eds.). (2017). *Medical image watermarking: techniques and applications*. Springer.

37. Burger, W., & Burge, M. J. (2016). *Digital image processing: an algorithmic introduction using Java*. Springer.
38. Grigoryan, A. M., & Grigoryan, M. M. (2016). *Brief notes in advanced DSP: Fourier analysis with MATLAB*. CRC Press.
39. Tripathy, R. K., Mendez, A. Z., de la O, S., Arrieta Paternina, M. R., Arrieta, J. G., & Naik, G. R. (2018). Detection of life threatening ventricular arrhythmia using digital taylor fourier transform. *Frontiers in physiology*, 9, 722.
40. Bahaz, M., & Benzid, R. (2018). Efficient algorithm for baseline wander and powerline noise removal from ECG signals based on discrete Fourier series. *Australasian physical & engineering sciences in medicine*, 41(1), 143-160.
41. Czerwinski, D., & Powroznik, P. (2018, November). Human Emotions Recognition with the Use of Speech Signal of Polish Language. In *2018 Conference on Electrotechnology: Processes, Models, Control and Computer Science (EPMCCS)*(pp. 1-6). IEEE.

42. Broughton, S., & Kurt, B. (2018). *Discrete Fourier Analysis and Wavelets: Applications to Signal and Image Processing*. Hoboken: Wiley.
43. Demeter, C. (2020). *Fourier Restriction, Decoupling, and Applications (Cambridge Studies in Advanced Mathematics)*. Indiana: Cambridge University Press.
44. Hsu, T. (2020). *Fourier Series, Fourier Transforms, and Function Spaces: A Second Course in Analysis*. Rhode Island: MAA PRESS American Mathematical Society.
45. Osgood, B. (2019). *Lectures on the Fourier Transform and Its Applications*. Rhode Island: American Mathematical Society.
46. Radożycki, T. (2020). *Solving Problems in Mathematical Analysis, Part III: Curves and Surfaces, Conditional Extremes, Curvilinear Integrals, Complex Functions, ... Fourier Series*. USA: Springer.

CAPÍTULO II

TRANSFORMADA DISCRETA DEL COSENO (TDC)

Luis Enrique Mendoza,
Ingeniería en Telecomunicaciones,

Universidad de Pamplona.

Leonor Jaimes Cerveleón,
Ingeniería Industrial,
Universidad de Pamplona.

La TDC, ha venido evolucionando en el uso del procesamiento de datos, gracias a su diversidad y la morfología que presentan los resultados, es así, que es una herramienta que permite la transformación sparse.

2.1 Introducción de la TDC en el procesamiento de datos

La transformada discreta del coseno (**TDC**), es una herramienta matemática que permite realizar aplicaciones como la transformada de Fourier, con la gran diferencia en que la TDC es real, no tiene simetría y tiene la característica de compactar la

energía de los datos originales. Compactar la energía se puede definir como la representación en pocos puntos de la información importante de la señal original, esto no significa que se tenga menor cantidad de puntos, como es el caso de compresión de datos. En la compactación se mantiene la relación en cantidad de puntos, sólo que la TDC permite tener una zona que contiene gran cantidad de información original.

Por otro lado, la TDC es una herramienta matemática que permite representar la señal original en un conjunto de datos que son combinaciones lineales de una función coseno real. También, permite realizar análisis frecuencial, y así realizar aplicaciones de gran envergadura como la compresión de datos, específicamente en imágenes. Esta herramienta es la utilizada para comprimir los datos en el estándar de televisión digital en Colombia. En este libro se presenta dos versiones de la transformada discreta del coseno, la primera es la versión 1D, que es aplicada a vectores, y la segunda

es la versión 2D, que es aplicada a matrices o imágenes. La TDC, es un poco más fácil de utilizar que la TDF, debido a su no complejidad ni simetría, aunque las dos se deben saber dominar para realizar aplicaciones más avanzadas que las componentes de frecuencia. A continuación, se describe la TDC-1D y la TDC-2D [1-9].

2.2 Estructura matemática de la TDC-1D

$$Q(k) = \alpha(k) \sum_{n=0}^{N-1} X(n) \cos\left(\frac{(2n+1)\pi k}{2N}\right) \quad (2.1)$$

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{Para } k = 0 \\ \sqrt{\frac{2}{N}} & \text{para } \neq \text{de } 0 \end{cases} \quad (2.2)$$

$$x(n) = \sum_{k=0}^{N-1} \alpha(k) Q(k) \cos\left(\frac{(2n+1)\pi k}{2N}\right) \quad (2.3)$$

Aquí $Q(k)$ se define como la transformada discreta del coseno TDC, y $x(n)$ se define como la transformada discreta del coseno inversa TDCI. El coeficiente $\alpha(k)$ define un factor de normalización o de ajuste, el cual está dado por un escalar en función del valor de k , como se observa en su definición matemática (Ecuación (2. 1)) [8-12].

A continuación, se muestra un ejemplo del funcionamiento real de la TDC.

En la

Figura 2.1 y

Figura 2.2 se muestra una porción de una señal de voz real y su representación de la TDC. Aquí se observa la longitud del vector (7000 muestras).

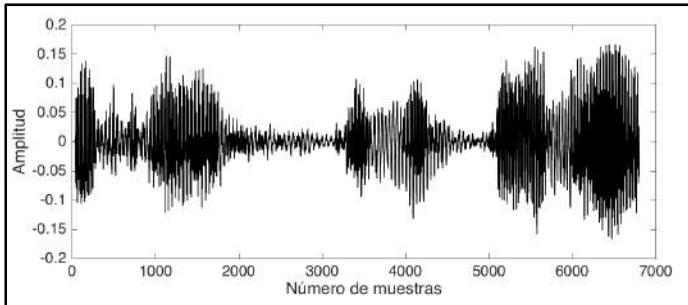


Figura 2.1.
Señal de voz en el tiempo

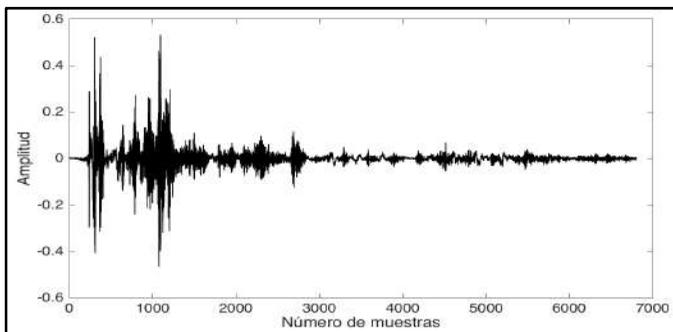


Figura 2.2.
TDC-1D, de la señal de voz.

Nótese como la señal en frecuencia, tiene también una longitud de 7000 datos o muestras. Se debe resaltar que los números del eje x, no indican valores

en frecuencia. Para conocer los valores reales de frecuencia se debe tener en cuenta: frecuencia de muestreo y tiempo de adquisición, ya que en la Transformada Discreta del Coseno la longitud máxima de puntos, es decir 7000 puntos, equivale a la frecuencia máxima que se puede obtener a partir de la frecuencia de muestreo, que para este ejemplo se tomó una frecuencia de muestreo de 8Khz. En este ejemplo se puede mencionar que las componentes más representativas en amplitud están alrededor de los puntos 800 - 1000, pero no indica que son las frecuencias. Para indicar la frecuencia real se concluye que 7000 puntos, es 4Khz, entonces 1000 puntos equivalen a 571hz.

Por otro lado, todo término transformada trae consigo la definición de su inversa, en el caso de la TDC-1D se denomina transformada discreta del coseno inversa (TDCI-1D), la cual transforma los datos del dominio de la frecuencia al dominio del tiempo. Esto es relevante ya que en el espacio $Q(k)$,

la resolución de la información original se pierde, es decir, que la forma de onda de los datos originales no se mantiene.

La estructura de TDCI-1D ver ecuación (2. 2), consigue obtener los datos originales si y solo si, ningún coeficiente de $Q(k)$ modifica sus características. El hecho que al menos uno solo sea modificado, el resultado final sería $X'(n)$, donde la ' , quiere decir que el resultado es una aproximación de $X(n)$ original. Dependiendo el **número** de datos que se modifiquen y **cuales** datos sean modificados, se obtendrá una representación muy aproximada o muy diferentes de $X(n)$ [12-17].

2.3 Ejemplos matemáticos TDC

Con el propósito de afianzar la teoría en la práctica y el buen desarrollo matemático, se muestran los siguientes ejemplos:

Ejemplo 2.1.

Si $X(n) = [1 \ 3 \ 5]$,

hallar la transformada discreta del coseno

$Q(k), N = 3$, Entonces.

$$Q(k) = \alpha(k) \sum_{n=0}^2 x(n) \cos\left(\frac{(2n+1)\pi k}{6}\right)$$

$$Q(0) = \alpha(0) \sum_{n=0}^2 X(n) \cos(0) = \sqrt{\frac{1}{3}} [\sum_{n=0}^2 X(n)]$$

$$Q(0) = \sqrt{\frac{1}{3}} [X(0) + X(1) + X(2)]$$

$$Q(0) = \sqrt{\frac{1}{3}} [9] = 5.196$$

$$Q(1) = \alpha(1) \sum_{n=0}^2 X(n) \cos\left(\frac{(2n+1)\pi}{6}\right)$$

$$Q(1) = \sqrt{\frac{2}{3}} \left[X(0) \cos\left(\frac{\pi}{6}\right) + X(1) \cos\left(\frac{\pi}{2}\right) + X(2) \cos\left(\frac{5\pi}{6}\right) \right]$$

$$Q(1) = \sqrt{\frac{2}{3}} \left[\cos\left(\frac{\pi}{6}\right) + 3 \cos\left(\frac{\pi}{2}\right) + 5 \cos\left(\frac{5\pi}{6}\right) \right]$$

$$Q(1) = -2.82$$

$$Q(2) = \alpha(2) \sum_{n=0}^2 X(n) \cos\left(\frac{(2n+1)\pi}{3}\right)$$

$$Q(2) = \sqrt{\frac{2}{3}} \left[X(0) \cos\left(\frac{\pi}{3}\right) + X(1) \cos(\pi) + X(2) \cos\left(\frac{5\pi}{3}\right) \right]$$

$$Q(2) = \sqrt{\frac{2}{3}} \left[\cos\left(\frac{\pi}{3}\right) + 3 \cos(\pi) + 5 \cos\left(\frac{5\pi}{3}\right) \right]$$

$$Q(2) = 0$$

$$Q(k) = [5.195 \quad -2.82 \quad 0]$$

Ejemplo 2.2.

Si $X(n) = [-4, 7, -1, 6]$
hallar $Q(k)$

$$N = 4$$

$$X(k) = \alpha(k) \sum_{n=0}^3 x(n) \cos\left(\frac{(2n+1)\pi k}{8}\right)$$

$$Q(0) = \alpha(0) \sum_{n=0}^3 X(n) \cos(0) = \sqrt{\frac{1}{4}} [\sum_{n=0}^3 X(n)]$$

$$Q(0) = \sqrt{\frac{1}{4}} [X(0) + X(1) + X(2) + X(3)]$$

$$Q(0) = \sqrt{\frac{1}{4}} [9] = 5.196$$

$$Q(1) = \alpha(1) \sum_{n=0}^2 X(n) \cos\left(\frac{(2n+1)\pi}{8}\right)$$

$$Q(1) = \sqrt{\frac{2}{4}} \left[X(0) \cos\left(\frac{\pi}{8}\right) + X(1) \cos\left(\frac{3\pi}{8}\right) + X(2) \cos\left(\frac{5\pi}{8}\right) \right]$$

$$Q(1) = \sqrt{\frac{2}{4}} \left[\cos\left(\frac{\pi}{8}\right) + 3 \cos\left(\frac{3\pi}{8}\right) + 5 \cos\left(\frac{5\pi}{8}\right) \right]$$

$$Q(1) = -2.82$$

$$Q(2) = \alpha(2) \sum_{n=0}^2 X(n) \cos\left(\frac{(2n+1)\pi}{4}\right)$$

$$Q(2) = \sqrt{\frac{2}{3}} \left[X(0) \cos\left(\frac{\pi}{4}\right) + X(1) \cos\left(\frac{3\pi}{4}\right) + X(2) \cos\left(\frac{5\pi}{4}\right) \right]$$

$$Q(2) = \sqrt{\frac{2}{3}} \left[\cos\left(\frac{\pi}{4}\right) + 3 \cos\left(\frac{3\pi}{4}\right) + 5 \cos\left(\frac{5\pi}{4}\right) \right]$$

$$Q(2) = 0$$

$$Q(k) = [5.195 \quad -2.82 \quad 0]$$

A continuación, se presenta un ejemplo matemático para el cálculo de la transformada discreta del coseno inversa.

Ejemplo 2.3.

Si $Q(k) = [2 \ -1 \ 3]$,
hallar $x[n]$

$$X(n) = \sum_{k=0}^2 \alpha(k) Q(k) \cos\left(\frac{(2n+1)\pi k}{6}\right)$$

$$X(0) = \sum_{k=0}^2 \alpha(k) Q(k) \cos\left(\frac{\pi k}{6}\right)$$

$$X(0) = \alpha(0) X(0) \cos(0) + \alpha(1) X(1) \cos\left(\frac{\pi}{6}\right) + \alpha(2) X(2) \cos\left(\frac{\pi}{3}\right)$$

$$X(0) = \sqrt{\frac{1}{3}} (2) + \sqrt{\frac{2}{3}} (-1) \cos\left(\frac{\pi}{6}\right) + \sqrt{\frac{2}{3}} (3) \cos\left(\frac{\pi}{3}\right)$$

$$X(0) = 1.67$$

$$X(1) = \sum_{k=0}^2 \alpha(k) X(k) \cos\left(\frac{\pi k}{2}\right)$$

$$X(1) = \alpha(0) X(0) \cos(0) + \alpha(1) X(1) \cos\left(\frac{\pi}{2}\right) + \alpha(2) X(2) \cos(\pi)$$

$$X(1) = \sqrt{\frac{1}{3}} (2) + \sqrt{\frac{2}{3}} (-1) \cos\left(\frac{\pi}{2}\right) + \sqrt{\frac{2}{3}} (3) \cos(\pi)$$

$$X(1) = -1.29$$

$$X(2) = \sum_{k=0}^2 \alpha(k) X(k) \cos\left(\frac{5\pi k}{6}\right)$$

$$X(2) = \alpha(0) X(0) \cos(0) + \alpha(1) X(1) \cos\left(\frac{5\pi}{6}\right) + \alpha(2) X(2) \cos\left(\frac{5\pi}{3}\right)$$

$$X(2) = \sqrt{\frac{1}{3}} (2) + \sqrt{\frac{2}{3}} (-1) \cos\left(\frac{5\pi}{6}\right) + \sqrt{\frac{2}{3}} (3) \cos\left(\frac{5\pi}{3}\right)$$

$$X(2) = 3.08$$

$$x[n] = [1.67 \quad -12.9 \quad 3.08]$$

2.4 Ejemplos TDC en señales de longitudes altas

Ejemplo 2.4.

Si una señal fue muestreada a la 8Khz, tiene una única componente de frecuencia, el tiempo de adquisición es de 4 segundos, y la respuesta de

frecuencia muestra que el pico de máxima amplitud está en 1000 puntos, ¿qué componente de frecuencia real tiene la señal?

Para dar solución a este tipo de datos, se debe tener en cuenta que la frecuencia de muestreo equivale a longitud máxima. En este caso la longitud del vector es: 16000 puntos, entonces:

$$32000 \text{ puntos} \rightarrow 4 \text{ KHz}$$

$$1000 \text{ puntos} \rightarrow f.\text{real}$$

$$f.\text{real} = 125\text{Hz}$$

De esta manera es el análisis verdadero de la transformada de Fourier en datos reales.

Gráficamente sería:

$$Fs = 8 \text{ KHz}$$

$$t = 0:1/8000:(2 - 1/8000);$$

$$y = \sin(2 * \pi * t * 125);$$

En la

Figura 2.3, se muestra la respuesta gráfica del ejercicio para validar con la teoría aplicada.

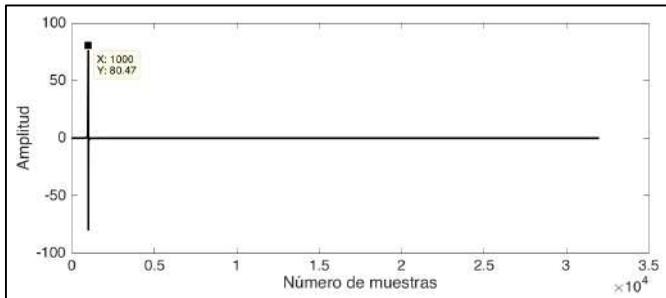


Figura 2.3.
Transformada discreta del coseno.

Ejemplo 2.5.

Calcular la transformada discreta del coseno de la suma de 3 señales senoidales con frecuencias: 100Hz, 300Hz y 400Hz. Las señales tienen una longitud de 9000 puntos y fueron adquiridas durante 3 segundos.

Para resolver este punto se calcula inicialmente la frecuencia de muestreo F_s , la cual se calcula

dividiendo longitud máxima en el tiempo. Realizando este cálculo se concluyó que $F_s=3\text{KHz}$. Entonces, la TDC-1D tiene como respuesta: (ver Figura 2.4) [18-20]:

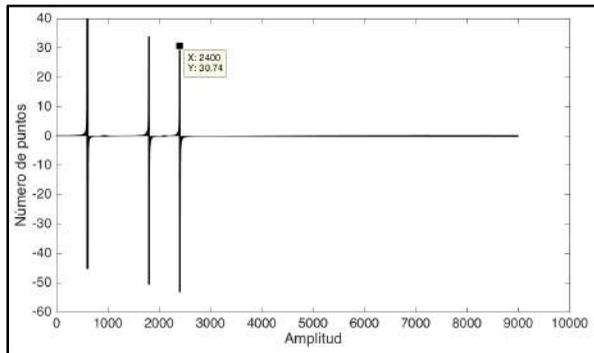


Figura 2.4.
Transformada discreta del coseno de la suma de 3 señales senoidales.

Aquí el pico ubicado en una posición de 2400, equivale a una frecuencia real de 400 Hz. Es importante mencionar que la transformada discreta del coseno no es simétrica y por ende no se observan picos duplicados. Para comprobar que la

posición 2400 equivale a 400 Hz, se procede de la siguiente manera:

1. La frecuencia de muestreo dividido en 2 equivale a la longitud máximo que es 9000.
2. Utilizando la relación anterior se dice que 2400 puntos tiene una frecuencia o equivale a una frecuencia de 400Hz.

Ejemplo 2.6.

Calcula el rango de frecuencias real usando TDC en una señal de electroencefalografía, que fue muestreada a $F_s=1\text{Hz}$, ver

Figura 2.5.

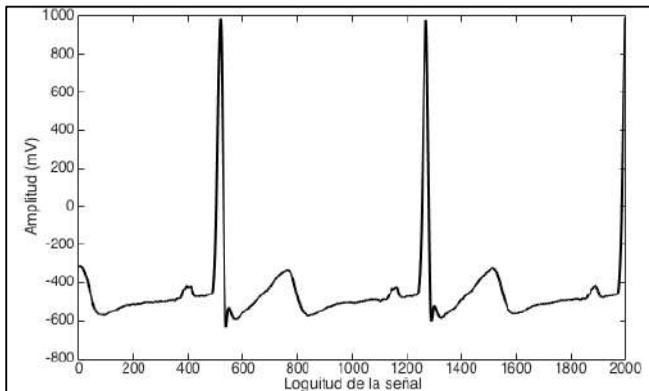


Figura 2.5.
Señal Electrocardiográfica.

La respuesta en frecuencia de señal se muestra en la

Figura 2.6 Nótese como la cantidad de frecuencia con mayor amplitud está alrededor de cero, eso indica que las frecuencias de la señal tiene valores pequeños, y que efectivamente la TDC permite realizar un proceso que se conoce como compactación de la energía, que no es más que poder representar la mayor cantidad de información o energía de la señal en un número mucho más pequeño de datos que su estado original.

Entonces, para encontrar el rango de frecuencias se ubica la posición donde las componentes los valores de amplitud se hacen casi cero. Para este ejercicio se eligió el valor 139, es decir, que el rango estará entre 0 y 139 puntos, que equivale a tener frecuencias entre 0Hz – 35Hz aproximadamente, rango que es totalmente coherente con la teoría o

rangos de frecuencias que presenta una señal fisiológica de este tipo que oscila entre 0.05Hz y 150Hz.

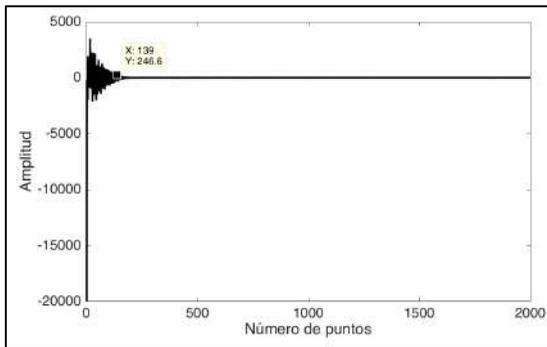


Figura 2.6.

Transformada discreta del Coseno señal Electrocardiográfica.

Es importante resaltar que la TDC, además de ejecutar en su mejor expresión las componentes de frecuencia, se puede utilizar para otras aplicaciones tales como: extracción de patrones, que se explicará más adelante en este mismo capítulo.

2.5 Estructura de matemática de la TDC-2D.

$$Q(K, L) = \alpha_1(K) \alpha_2(L) \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} X(n, m) \cos\left(\frac{(2n+1)\pi k}{2N}\right) \cos\left(\frac{(2m+1)\pi l}{2M}\right) \quad (2.4)$$

$$\begin{aligned} \alpha_1(k) &= \begin{cases} \sqrt{\frac{1}{N}} & \text{para } K = 0 \\ \sqrt{\frac{2}{N}} & \text{para } K \neq 0 \end{cases} & \alpha_2(L) &= \begin{cases} \sqrt{\frac{1}{M}} & \text{para } L = 0 \\ \sqrt{\frac{2}{M}} & \text{para } L \neq 0 \end{cases} \end{aligned} \quad (2.5)$$

$$X(n, m) = \sum_{K=0}^{N-1} \sum_{L=0}^{M-1} \alpha_1(K) \alpha_2(L) Q(K, L) \cos\left(\frac{(2n+1)\pi K}{2N}\right) \cos\left(\frac{(2m+1)\pi L}{2M}\right) \quad (2.6)$$

Q denota la TDC-2D, los coeficientes descritos como α_1 y α_2 , se conocen como coeficientes de normalización, los cuales son dependientes de los valores que tome K y L, es decir, las posiciones del coeficiente de la TDC-2D. Finalmente X es la imagen original. La ecuación (2.4), se conoce como la TDC-2D y la ecuación (2.6), se conoce como la transformada discreta del coseno inversa TDCI-2D [18-25].

2.6 Ejemplos matemáticos de la TDC-2D

Ejemplo 2.7.

Si $X(n, m) = \begin{bmatrix} 2 & 4 \\ 3 & 6 \end{bmatrix}$ Hallar $X(K, L)$

$$Q(K, L) = \alpha_1(K)\alpha_2(L) \sum_{n=0}^1 \sum_{m=0}^1 X(n, m) \cos\left(\frac{(2n+1)\pi K}{4}\right) \cos\left(\frac{(2m+1)\pi L}{4}\right)$$

$$Q(0,0) = \alpha_1(0)\alpha_2(0) \sum_{n=0}^1 \sum_{m=0}^1 [X(n, m) \cos(0) \cos(0)]$$

$$Q(0,0) = \sqrt{\frac{1}{N}} \sqrt{\frac{1}{M}} \sum_{n=0}^1 [X(0,0) + X(n, 1)]$$

$$Q(0,0) = \sqrt{\frac{1}{2}} \sqrt{\frac{1}{2}} [X(0,0) + X(0,1) + X(1,0) + X(1,1)]$$

$$Q(0,0) = \frac{1}{2} [2 + 4 + 3 + 6]$$

$$Q(0,0) = \frac{15}{2}$$

$$Q(0,1) = \alpha_1(0)\alpha_2(1) \sum_{n=0}^1 \sum_{m=0}^1 X(n, m) \cos(0) \cos\left(\frac{(2m+1)\pi}{4}\right)$$

$$Q(0,1) = \sqrt{\frac{1}{N}} \sqrt{\frac{2}{M}} \sum_{n=0}^1 \left[X(n, 0) \cos\left(\frac{\pi}{4}\right) + X(n, 1) \cos\left(\frac{3\pi}{4}\right) \right]$$

$$Q(0,1) = \sqrt{\frac{1}{2}} (1) [X(0,0) \cos\left(\frac{\pi}{4}\right) + X(0,1) \cos\left(\frac{3\pi}{4}\right) + X(1,0) \cos\left(\frac{\pi}{4}\right) + X(1,1) \cos\left(\frac{3\pi}{4}\right)]$$

$$Q(0,1) = \sqrt{\frac{1}{2}} [(2 + 3) \cos\left(\frac{\pi}{4}\right) + (4 + 6) \cos\left(\frac{3\pi}{4}\right)]$$

$$Q(0,1) = -2.5$$

$$Q(1,0) = \alpha_1(1)\alpha_2(0) \sum_{n=0}^1 \sum_{m=0}^1 X(n, m) \cos\left(\frac{(2n+1)\pi}{4}\right) \cos(0)$$

$$Q(1,0) = \sqrt{\frac{2}{N}} \sqrt{\frac{1}{M}} [\sum_{n=0}^1 (X(n, 0) + X(n, 1)) \cos\left(\frac{(2n+1)\pi}{4}\right)]$$

$$Q(1,0) = 1 \sqrt{\frac{1}{2}} [(X(0,0) + X(0,1) \cos\left(\frac{\pi}{4}\right) + (X(1,0) + X(1,1)) \cos\left(\frac{3\pi}{4}\right)]$$

$$Q(1,0) = \sqrt{\frac{1}{2}} [6x \cos\left(\frac{\pi}{4}\right) + 9x \cos\left(\frac{3\pi}{4}\right)]$$

$$Q(1,0) = -1.5$$

$$Q(1,1) = \alpha_1(1)\alpha_2(1) \sum_{n=0}^1 \sum_{m=0}^1 X(n, m) \cos\left(\frac{(2n+1)\pi}{4}\right) \cos\left(\frac{(2m+1)\pi}{4}\right)$$

$$Q(1,1) = \sqrt{\frac{2}{N}} \sqrt{\frac{2}{M}} \sum_{n=0}^1 [X(n, 0) \cos\left(\frac{\pi}{4}\right) + X(n, 1) \cos\left(\frac{3\pi}{4}\right)] \cos\left(\frac{(2n+1)\pi}{4}\right)$$

$$Q(1,1) = \sqrt{1}\sqrt{1} [X(0,0) \cos\left(\frac{\pi}{4}\right) + X(0,1) \cos\left(\frac{3\pi}{4}\right)] \cos\left(\frac{\pi}{4}\right) + [X(1,0) \cos\left(\frac{\pi}{4}\right) + X(1,1) \cos\left(\frac{3\pi}{4}\right)] \cos\left(\frac{3\pi}{4}\right)$$

$$Q(1,1) = [2 \cos\left(\frac{\pi}{4}\right) + 4 \cos\left(\frac{3\pi}{4}\right)] \cos\left(\frac{\pi}{4}\right) + [3 \cos\left(\frac{\pi}{4}\right) + 6 \cos\left(\frac{3\pi}{4}\right)] \cos\left(\frac{3\pi}{4}\right)$$

$$Q(1,1) = 0.5$$

En conclusión, la TDC-2D está expresada como:

$$Q(K, L) = \begin{bmatrix} 7.5 & -2.5 \\ -1.5 & 0.5 \end{bmatrix}$$

Ejemplo 2.8.

Encontrar $Q(K, L)$ si $X(n, m) = \begin{bmatrix} 5 & -3 & 8 \\ 4 & 1 & -14 \\ 7 & 0 & 40 \end{bmatrix}$

$$Q(K, L) = \alpha_1(K)\alpha_2(L) \sum_{n=0}^2 \sum_{m=0}^2 X(n, m) \cos\left(\frac{(2n+1)\pi K}{6}\right) \cos\left(\frac{(2m+1)\pi L}{6}\right)$$

$$Q(0,0) = \alpha_1(0)\alpha_2(0) \sum_{n=0}^2 \sum_{m=0}^2 [X(n,m) \cos(0) \cos(0)]$$

$$Q(0,0) = 15$$

Realizando el mismo procedimiento anterior se puede concluir que:

$$Q(K,L) = \begin{bmatrix} 16 & -7.34 & 12.72 \\ -15.67 & 15 & -8.08 \\ 17.67 & -20.78 & 15 \end{bmatrix}$$

Es importante resaltar que el procedimiento mostrado en el ejemplo anterior, se repite en este ejemplo, sin embargo, existe una diferencia significativa que es el tamaño de la matriz o imagen, que para este caso se calculan 9 coeficientes [23-28].

Ejemplo 2.9.

Calcular la Transformada discreta del coseno inversa si:

$$Q(K,L) = \begin{bmatrix} 7.5 & -2.5 \\ -1.5 & 0.5 \end{bmatrix}$$

Entonces la TDCI está dado por:

$$X(n, m) = \sum_{K=0}^{N-1} \sum_{L=0}^{M-1} \alpha_1(K) \alpha_2(L) Q(K, L) \cos\left(\frac{(2n+1)\pi K}{2N}\right) \cos\left(\frac{(2m+1)\pi L}{2M}\right)$$

$$\begin{aligned} X(0,0) &= \sum_{K=0}^1 \sum_{L=0}^1 \alpha_1(K) \alpha_2(L) Q(K, L) \cos\left(\frac{\pi K}{4}\right) \cos\left(\frac{\pi L}{4}\right) \end{aligned}$$

$$\begin{aligned} X(0,0) &= \sum_{K=0}^1 \alpha_1(K) \cos\left(\frac{\pi K}{4}\right) * [\alpha_2(0) Q(K, 0) + \\ &\alpha_2(1) * Q(K, 1) \cos\left(\frac{\pi}{4}\right)] \end{aligned}$$

$$\begin{aligned} X(0,0) &= \alpha_1(0) \cos(0) * [\alpha_2(0) Q(0,0) + \cos\left(\frac{\pi}{4}\right) * \\ &\alpha_2(1) * Q(0,1)] + \alpha_1(1) * \cos\left(\frac{\pi}{4}\right) * [\alpha_2(0) Q(1,0) + \\ &\cos\left(\frac{\pi}{4}\right) * \alpha_2(1) * Q(1,1)] \end{aligned}$$

$$\begin{aligned} X(0,0) &= \sqrt{\frac{1}{2}} * \left[\sqrt{\frac{1}{2}} * Q(0,0) + \cos\left(\frac{\pi}{4}\right) * Q(0,1) \right] + \sqrt{\frac{1}{2}} * \\ &\cos\left(\frac{\pi}{4}\right) * \left[\sqrt{\frac{1}{2}} * Q(1,0) + \cos\left(\frac{\pi}{4}\right) * Q(1,1) * \cos\left(\frac{\pi}{4}\right) \right] \end{aligned}$$

$$\begin{aligned} X(0,0) &= \sqrt{\frac{1}{2}} * \left[\sqrt{\frac{1}{2}} * 7.5 + \cos\left(\frac{\pi}{4}\right) * -2.5 \right] + \sqrt{\frac{1}{2}} * \\ &\cos\left(\frac{\pi}{4}\right) * \left[\sqrt{\frac{1}{2}} * -1.5 + \cos\left(\frac{\pi}{4}\right) * 0.5 * \cos\left(\frac{\pi}{4}\right) \right] \end{aligned}$$

$$X(0,0) = 2$$

$$X(0,1) = \sum_{K=0}^1 \sum_{L=0}^1 \alpha_1(K) \alpha_2(L) Q(K,L) \cos\left(\frac{\pi K}{4}\right) \cos\left(\frac{3\pi L}{4}\right)$$

$$X(0,1) = \sum_{K=0}^1 \alpha_1(K) \cos\left(\frac{\pi K}{4}\right) * [\alpha_2(0) Q(K,0) + \alpha_2(1) * Q(K,1) \cos\left(\frac{3\pi}{4}\right)]$$

$$X(0,1) = \alpha_1(0) \cos(0) * [\alpha_2(0) Q(0,0) + \cos\left(\frac{3\pi}{4}\right) * \alpha_2(1) * Q(0,1)] + \alpha_1(1) * \cos\left(\frac{\pi}{4}\right) * [\alpha_2(0) Q(1,0) + \cos\left(\frac{3\pi}{4}\right) * \alpha_2(1) * Q(1,1)]$$

$$X(0,1) = \sqrt{\frac{1}{2}} * \left[\sqrt{\frac{1}{2}} * Q(0,0) + \cos\left(\frac{3\pi}{4}\right) * Q(0,1) \right] + \sqrt{\frac{1}{2}} * \cos\left(\frac{\pi}{4}\right) * \left[\sqrt{\frac{1}{2}} * Q(1,0) + \cos\left(\frac{\pi}{4}\right) * Q(1,1) * \cos\left(\frac{\pi}{4}\right) \right]$$

$$X(0,1) = \sqrt{\frac{1}{2}} * \left[\sqrt{\frac{1}{2}} * 7.5 + \cos\left(\frac{\pi}{4}\right) * -2.5 \right] + \sqrt{\frac{1}{2}} * \cos\left(\frac{\pi}{4}\right) * \left[\sqrt{\frac{1}{2}} * -1.5 + \cos\left(\frac{\pi}{4}\right) * 0.5 * \cos\left(\frac{3\pi}{4}\right) \right]$$

$$X(0,1) = 4$$

$$X(1,0) = \sum_{K=0}^1 \sum_{L=0}^1 \alpha_1(K) \alpha_2(L) Q(K, L) \cos\left(\frac{3\pi K}{4}\right) \cos\left(\frac{\pi L}{4}\right)$$

$$X(1,0) = \sum_{K=0}^1 \alpha_1(K) \cos\left(\frac{3\pi K}{4}\right) * [\alpha_2(0) Q(K, 0) + \alpha_2(1) * Q(K, 1) \cos\left(\frac{\pi}{4}\right)]$$

$$X(1,0) = \alpha_1(0) * \cos(0) * [\alpha_2(0) Q(0,0) + \cos\left(\frac{\pi}{4}\right) * \alpha_2(1) * Q(0,1)] + \alpha_1(1) * \cos\left(\frac{3\pi}{4}\right) * [\alpha_2(0) Q(1,0) + \cos\left(\frac{\pi}{4}\right) * \alpha_2(1) * Q(1,1)]$$

$$X(1,0) = \sqrt{\frac{1}{2}} * [\sqrt{\frac{1}{2}} * Q(0,0) + \cos\left(\frac{\pi}{4}\right) * Q(0,1)] + \sqrt{\frac{1}{2}} * \cos\left(\frac{3\pi}{4}\right) * [\sqrt{\frac{1}{2}} * Q(1,0) + \cos\left(\frac{\pi}{4}\right) * Q(1,1) * \cos\left(\frac{\pi}{4}\right)]$$

$$X(1,0) = \sqrt{\frac{1}{2}} * [\sqrt{\frac{1}{2}} * 7.5 + \cos\left(\frac{\pi}{4}\right) * -2.5] + \sqrt{\frac{1}{2}} * \cos\left(\frac{3\pi}{4}\right) * [\sqrt{\frac{1}{2}} * -1.5 + \cos\left(\frac{\pi}{4}\right) * 0.5 * \cos\left(\frac{\pi}{4}\right)]$$

$$X(1,0) = 3$$

$$X(1,1) = \sum_{K=0}^1 \sum_{L=0}^1 \alpha_1(K) \alpha_2(L) Q(K, L) \cos\left(\frac{3\pi K}{4}\right) \cos\left(\frac{3\pi L}{4}\right)$$

$$X(1,1) = \sum_{K=0}^1 \alpha_1(K) \cos\left(\frac{3\pi K}{4}\right) * [\alpha_2(0) Q(K, 0) + \alpha_2(1) * Q(K, 1) \cos\left(\frac{3\pi}{4}\right)]$$

$$X(1,1) = \alpha_1(0) * \cos(0) * [\alpha_2(0) Q(0,0) + \cos\left(\frac{3\pi}{4}\right) * \alpha_2(1) * Q(0,1)] + \alpha_1(1) * \cos\left(\frac{3\pi}{4}\right) * [\alpha_2(0) Q(1,0) + \cos\left(\frac{\pi}{4}\right) * \alpha_2(1) * Q(1,1)]$$

$$X(1,1) = \sqrt{\frac{1}{2}} * [\sqrt{\frac{1}{2}} * Q(0,0) + \cos\left(\frac{3\pi}{4}\right) * Q(0,1)] + \sqrt{\frac{1}{2}} * \cos\left(\frac{3\pi}{4}\right) * [\sqrt{\frac{1}{2}} * Q(1,0) + \cos\left(\frac{3\pi}{4}\right) * Q(1,1) * \cos\left(\frac{3\pi}{4}\right)]$$

$$X(1,1) = \sqrt{\frac{1}{2}} * [\sqrt{\frac{1}{2}} * 7.5 + \cos\left(\frac{\pi}{4}\right) * -2.5] + \sqrt{\frac{1}{2}} * \cos\left(\frac{3\pi}{4}\right) * [\sqrt{\frac{1}{2}} * -1.5 + \cos\left(\frac{3\pi}{4}\right) * 0.5 * \cos\left(\frac{3\pi}{4}\right)]$$

$$X(1,1) = 3$$

Entonces la TDCI está da por:

$$X(n, m) = \begin{bmatrix} 2 & 4 \\ 3 & 6 \end{bmatrix}$$

Ya finalizando los ejemplos matemáticos se procede a describir los ejemplos en datos reales.

2.7 Ejemplos TDC-2D en imágenes reales

Ejemplo 2.10.

En la figura 2.8 se muestra el cálculo la transformada discreta del coseno 2D de una imagen con forma geométrica, ver

Figura 2.7.

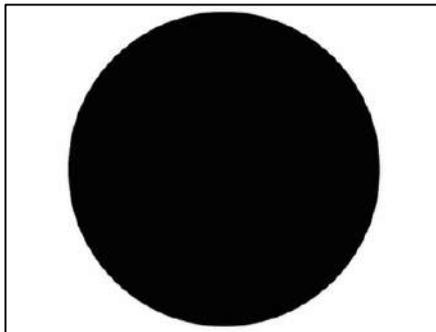


Figura 2.7.
Imagen figura geométrica.



Figura 2.8.
Transformada discreta del coseno figura geométrica.

Nótese que la

Figura 2.8 muestra la respuesta de la transformada discreta del coseno 2D. Nótese como las componentes de frecuencia se ubican en la parte superior izquierda de la gráfica (parte derecha), esto permite evidenciar que existe efectivamente un proceso de compactación de energía usando la TDC. Se debe tener en cuenta que las componentes de frecuencia en TDC-2D se ubican en su mayoría en la parte superior izquierda de la imagen.

Dependiendo de la imagen analizada se ubican las componentes de frecuencia.

Ejemplo 2.11.

TDC-2D de la imagen conocida como Leonor
Figura 2.9.



Figura 2.9.
Imagen a nivel de gris. Leonor.



Figura 2.10.
Transformada discreta del coseno imagen nivel de gris.

Nótese en la

Figura 2.10 como, gran parte de la información importante, se ubican en la parte superior izquierda, de modo que permite ejemplificar que siempre la TDC-2D concentra su información importante en zonas conocidas como segundo cuadrante.

2.8 Aplicaciones en el procesamiento de datos

En esta sección se presenta, una serie de ejemplos y aplicaciones reales de la TDC-2D. En este caso las aplicaciones están concentradas en: Compresión de voz, y Extracción de Patrones para clasificación de

datos. Se trabajarán imágenes de angiografía coronaria, de resonancia magnética y señales de voz. Además, se muestran resultados de imágenes inéditas de los autores, donde se aplicada la TDC-2D [15-42].

2.8.1 Compresión

A continuación, se presenta un ejemplo de la TDC, utilizada para realizar procesos de extracción de patrones en señales de voz. Así mismo, se explica el proceso que se resumen en 4 pasos, descritos como:

1. Registro de señal de voz original.
2. Se obtiene de la TDC.
3. Selección de la zona donde se concentra la mayor cantidad de energía.
4. Recortar las zonas de mayor concentración de energía
5. Se aplica la TDCI.

La

Figura 2.11 muestra la señal original registrada, que en este ejemplo representa la palabra *Luis*. Nótese

que dicha señal tiene una longitud igual a 24000 datos, la cual fue muestreada a 8Khz y por un tiempo de 3 segundos.

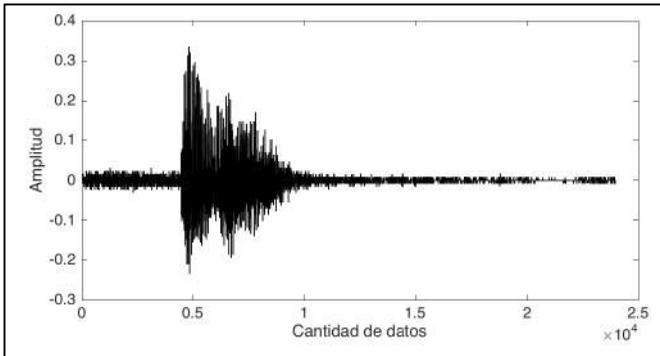


Figura 2.11.
Señal de voz, palabra Luis.

Por otro lado, la

Figura 2.12, muestra la TDC señal registrada, aplicando los pasos 2 y 3, se menciona que la zona donde se concentra la mayor cantidad de energía fue seleccionada entre 1 y 1200 datos, esto quiere decir que el recorte de dicha zona, se realiza bajo estas dos cantidades 1 y 12000. La figura 2.12 es la TDC

de la señal de voz, nótese como se concentra la energía en zonas de 1 a 5000 muestras.

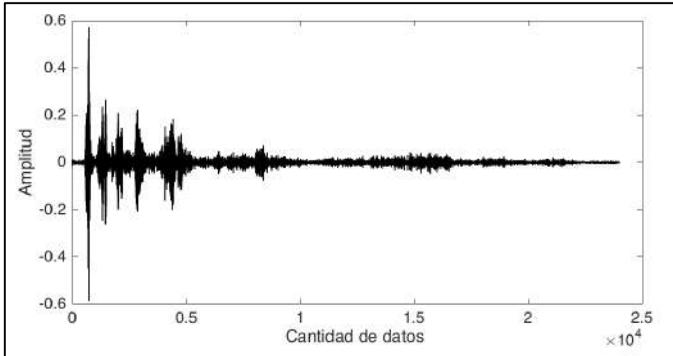


Figura 2.12.
TDC señal de voz, palabra Luis.

Finalmente, se procede a la aplicación de la TDCI, que permite pasar del espacio de la frecuencia al espacio del tiempo [30-36]. Aquí se puede observar que la señal reconstruida tiene una longitud de 12000 puntos (ver

Figura 2.13), así mismo, se puede observar que la forma o morfología de la señal reconstruida comparada con la morfología de la señal original, presenta un grado de similitud bastante alto,

haciendo énfasis en que tiene 24000 datos y la señal resultante 12000 datos, el cual es la diferencia más significativa. Usando TDC, se ha podido demostrar que es posible mantener el mensaje de voz, en este caso la palabra Luis, en la señal reconstruida con menos datos que la original. Así mismo, es importante denotar que es posible tener mayor grado de compresión dependiendo del tiempo de la señal a analizar. Se recomienda realizar procesos de compresión al 50%, 75% y así mismo validar cuál de las compresiones mantiene gran parte del mensaje original. La compresión de información se puede aplicar tanto en señales en 1D y en 2D o en cualquier tipo de información que está representada con variables reales.

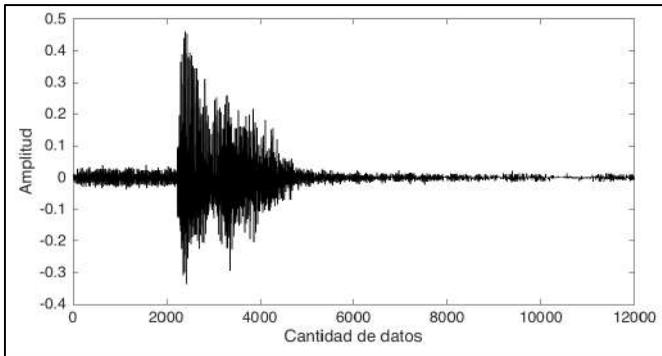


Figura 2.13.
Señal comprimida al 50%,
palabra Luis.

2.8.2 Detección de contornos.

A continuación, se presenta un ejemplo de cómo es el procedimiento utilizando transformada discreta del coseno para detección de contornos, es importante mencionar que su funcionamiento tiene mayor relevancia si la imagen es binaria, sin embargo, el ejemplo se presenta en una imagen a nivel gris.

Los pasos para la detección de contorno se describen a continuación:

- 1) Imagen a nivel de gris
- 2) Transformada discreta del coseno de la imagen
- 3) Ubicar zona del espectro de la frecuencia que se coloca a cero, tener en cuenta la teoría de compactación de la energía
- 4) Aplicar la transformada discreta del coseno inversa
- 5) Resta de la imagen original a nivel de gris y la imagen reconstruida.

Ejemplo 2.12.

Calcular el contorno usando TDC-2D

En este caso se usó una imagen con una figura geométrica, la imagen tiene una dimensión de 304 filas y 400 columnas, una vez aplicada la TDC-2D, solo se conservaron las componentes de frecuencias ubicada en la zona que comprenden de la fila 1 a la 75, y de la columna 1 a la columna 100. La demás información se asignó un valor de cero, la Figura 2.14, muestra los resultados de lo descrito anteriormente (imagen izquierda imagen original, imagen derecha TDC-2D y zona puesta a cero).

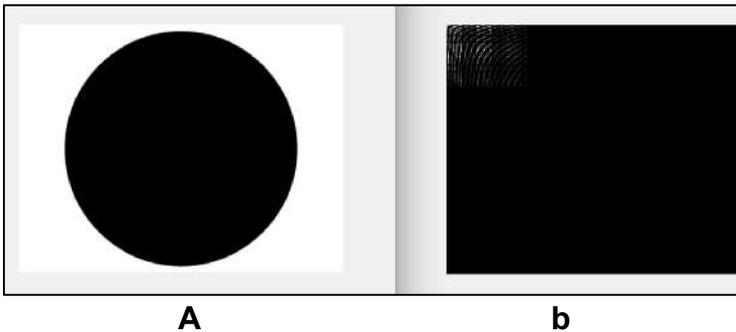


Figura 2.14.
a) Imagen original. b) TDC y paso 3
de contornos.

Realizados los pasos 1 al 3, se aplica la transformada discreta del coseno inversa. La Figura 2.15, muestra el resultado de la TDCI-2D, como se observa conserva una similitud con la imagen original, esto significa que el proceso de eliminación o puesta de información a cero fue satisfactorio. Aplicando el paso 5, se obtiene la respuesta que consigue el contorno del objeto inicial. Este proceso se puede aplicar a todo tipo de imágenes y usar como técnica para realzar contornos.

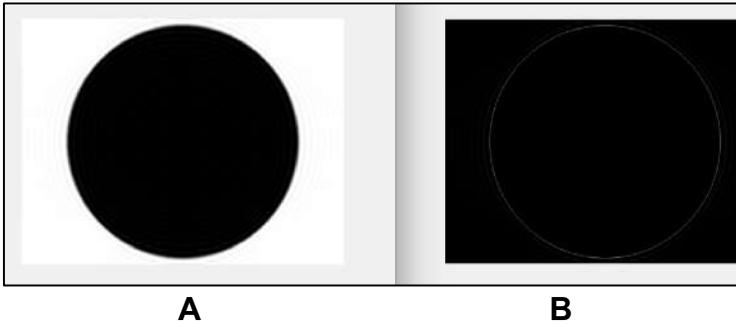


Figura 2.15.
a) TDCI 2D. b) paso 5 contornos
(resta imágenes).

Ejemplo 2.13.

A continuación, se muestra el resultado a una imagen mamográfica (ver Figura 2.16). Nótese como se presenta diferentes contornos que contiene la imagen. Es importante mencionar que entre más información sea a cero mayor información se pierde en el proceso, sin embargo, con los resultados mostrados se recomienda que por lo menos el 80% de la información sea colocada a cero con el fin de obtener

un resultado importante en cuanto a reconocimiento de contornos se refiere.

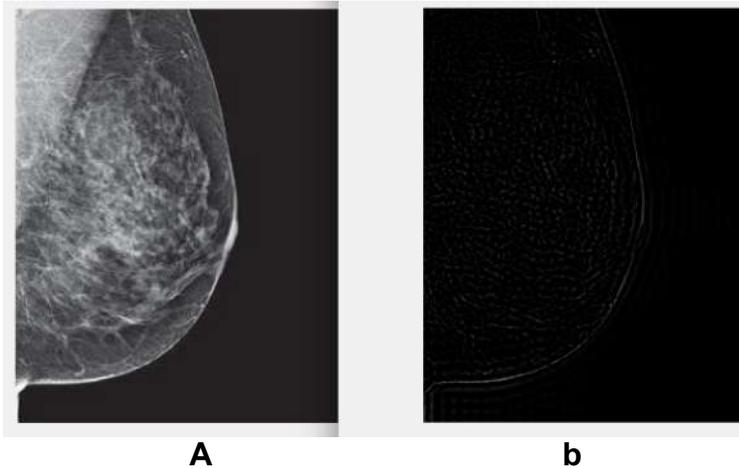


Figura 2.16.

a) Imagen original. b) Contornos de imagen usando TDC 2D.

Para este caso, se aplicó el paso 3, donde el 97% de la TDC-2D, fue puesta a cero, lo que significa que gran parte de la información de la imagen tiene componentes de baja frecuencia.

2.8.3 Extracción de patrones

Con el objetivo de proporcionar un ejemplo de cómo se observa el patrón característico de diferentes imágenes, se incluye un ejemplo de imágenes con lenguaje de señas, nótese como se muestra la imagen original y la respectiva TDC-2D de cada una de las imágenes. El objetivo es identificar las zonas donde las componentes de frecuencia se hacen diferentes entre las imágenes originales.

La

Figura 2.17, muestra diferentes imágenes relacionadas con el lenguaje de señas, aquí se demuestra que usando la TDC 2D, se pueden encontrar patrones característicos de cada imagen.

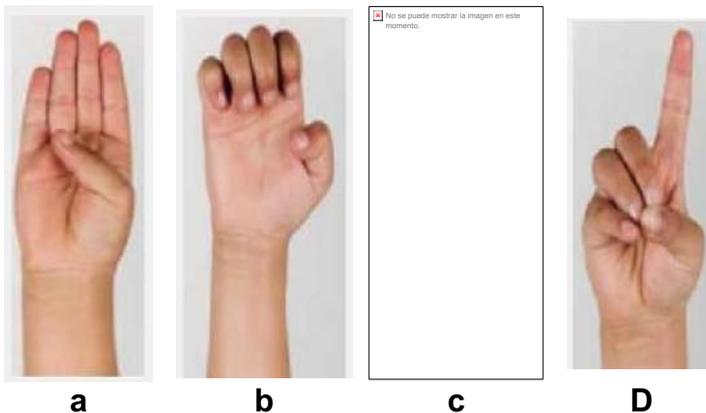


Figura 2.17.
a, b, c y d) Imagen original.

Por otro lado, la Figura 2.18, muestra la TDC-2D de las imágenes de las señas. Nótese que los componentes de frecuencia importantes representada por los colores cercanos a 255, es decir, blancos, están concentrados en la parte superior izquierda. El objetivo es lograr ubicar las diferencias entre cada TDC y que en un contexto se puede clasificar información basados en los patrones encontrados.

Para lograr visualizar de mejor manera la forma de la TDC de cada imagen, se ha realizado un zoom (ver

Figura 2.19), de las dos primeras imágenes de señas (a, b). Nótese como las formas en que se presentan las componentes de frecuencia para cada imagen es diferente, lo que permitiría evidenciar que, si hacemos el proceso con por lo menos 50 imágenes de la misma señal, se presentaría una distribución similar de cada una. Se debe tener en cuenta que la

toma de imágenes y el cálculo de la TDC depende de la intensidad de gris que tiene cada imagen, tema que se debe tener en cuenta para el proceso de validación y efectividad de un patrón encontrado.

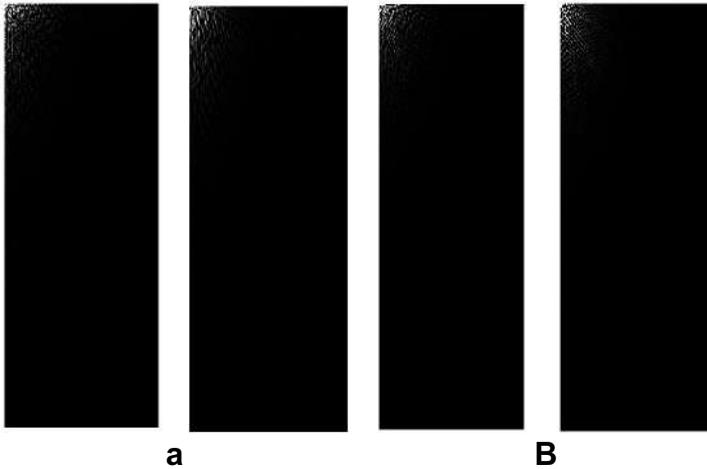


Figura 2.18.
a) Imagen original. b) TDC 2D y paso 3 de contornos.

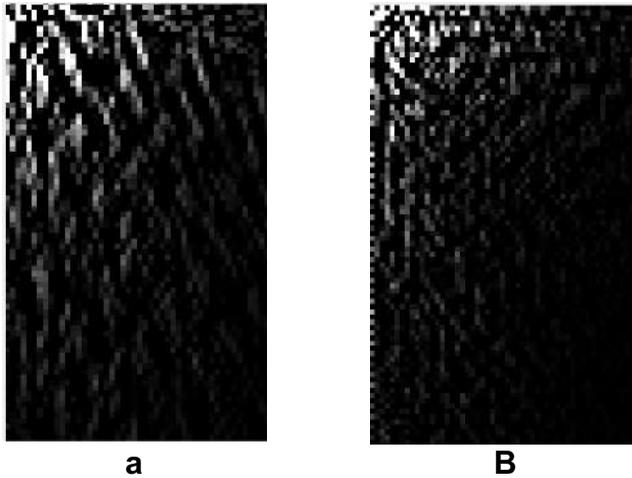


Figura 2.19.

a) Imagen original. b) TDC 2D y paso 3 de contornos.

Para la detección de patrones se recomienda realizar un estudio de la distribución de los niveles de gris en cada una de las imágenes de la TDC-2D. Como las imágenes originales son significativamente diferentes, se debe realizar un estudio de la distribución de nivel de gris.

La extracción de patrones usando TDC en 1D y 2D, ha venido tomando fuerza debido a las particularidades y a la forma como se presenta los

resultados de la técnica. Así mismo, se viene aprovechando el proceso de compactación de la energía, elemento importante que permite condensar información en espacios mucho más pequeños que en el espacio original. Es un proceso que se ha utilizado en datos como: señales electrocardiográficas, señales electromiográficas, imágenes biomédicas, señales sísmicas, entre otras. Para finalizar este capítulo, a continuación se describe la sesión de ejercicios propuestos.

2.8.4 Eliminación de Información

A continuación, se muestra un ejemplo usando la TDC -2D para eliminación de componentes de frecuencia anormales en una imagen. El proceso de reconstrucción de imágenes o eliminación de información poco relevante usando la transformada discreta del coseno, está basada en la teoría que se condensa en los siguientes pasos:

1. Selección de las imágenes a procesar, identificando en el espacio del tiempo que se desea realizar, la identificación de la información tiene que ver con la distribución de frecuencias en la TDC.
2. Aplicar la transformada discreta del coseno,
3. Identificar las zonas que se desean modificar en el espacio de la transformada, ya sea haciéndolas cero o aplicando algún método como promedio, máximo, mínimo, desviación estándar entre otras técnicas que permiten modificar información teniendo en cuenta los datos presentes en las imágenes. Se recomienda que la modificación esté basada en ventanas de tamaño mínimo 8×8 .
4. Se aplica la transformada discreta del coseno inversa, con el fin de lograr observar los cambios realizados en el espacio de la frecuencia y su efecto en el espacio del tiempo o espacio de la imagen original.

La

Figura 2.20 muestra una imagen en niveles de gris (tamaño 512*512) con un comportamiento en donde aparece líneas horizontales color blanco, entonces, el objetivo es eliminar en cierta medida las líneas horizontales usando TDC, proceso que se realizó usando TDF capítulo I.



Figura 2.20.

Imagen nivel de gris con información de líneas horizontales.

Por otro lado, la

Figura 2.21 es la TDC, la cual, como se observa, presenta la compactación de la energía en el

extremo superior, y existen componentes de frecuencia en zonas muy cercanos a la primera columna.

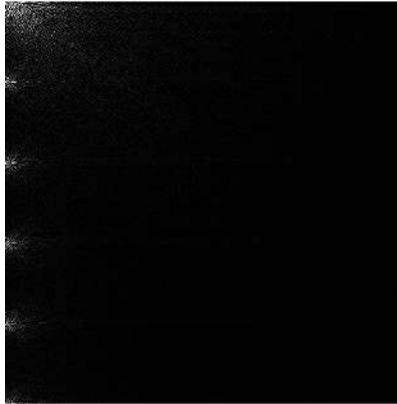


Figura 2.21.
Transformada discreta del coseno de la
Figura 2.20

La

Figura 2.22, muestra el proceso de eliminación (poner ciertas posiciones de la imagen con un valor igual a cero) de información, con el fin de eliminar las líneas horizontales. Para este ejemplo solo se conservaron los valores que estuvieran entre las filas 1 a 64, y las columnas 1 a 64, la demás información fue colocada

a cero. En este caso, las componentes de frecuencia que producen las líneas se encuentran en las posiciones puestas a cero. Sin embargo, es importante mencionar que la zona que se desea modificar es susceptible a cambios, y ver el efecto que se obtiene al modificar dicha zona.

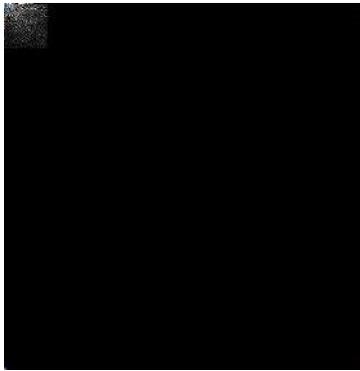


Figura 2.22.

Transformada discreta del coseno modificada para filtrado.

Finalmente, en la Figura 2.23 se observa la TDCI-2D o reconstrucción de la imagen. Nótese como se han eliminado las líneas horizontales, sin embargo, el proceso ha llevado a que la imagen tenga una distorsión notable.

Aquí el objetivo principal está en eliminar las líneas horizontales. Así mismo, se menciona que, para evitar distorsiones en alto grado, se debe estudiar muy bien que componentes de frecuencia o que porciones de la TDC se colocan a cero.



Figura 2.23.

Figura nivel de gris con información de líneas horizontales.

2.9 Ejercicios propuestos TDC 1D y 2D

- 1) Hallar la TDC de $x = [3, -4, 2, 5, 7, 8]$;
- 2) Hallar la TDC de $x = [0, 1, 0, 0, 0, 1]$;
- 3) Hallar la TDC de $x = [1 - 4j, -9, 2, 4, -1 + 8j]$;

4) Hallar la TDCI de $Q(k) = [2 + 4j, 3j, 4 - 7j, 5, 6 - 9j]$;

5) Hallar la TDCI de $Q(k) = [12, -5, 4, 4 - 1]$;

3) Hallar la TDCI de $Q(k) = [1, 1, 1, 1, 1]$;

4) Hallar gráficamente la TDC de una señal de voz sumada con una señal senoidal de 120Hz. Las señales fueron tomadas con una $F_s = 8\text{KHz}$ y una duración de 2.5 segundos. Obtenga conclusiones de la respuesta.

5) Hallar gráficamente la TDC de una señal de electromiografía, la cual tiene una frecuencia de muestreo de 1KHz, y su duración de adquisición de 30 segundos. Si desea tener esta señal escribanos a luis.mendoza@unipamplona.edu.co.

6) Hallar TDC2D de $x(n, m) = \begin{bmatrix} 2 & 9 & -1 \\ 8 & 3 & -2 \\ -2 & -7 & -4 \end{bmatrix}$

7) Hallar TDC2D de $x(n, m) = \begin{bmatrix} 3.2 & 4.5 & 0 \\ 9 & -2 & 1 \\ 7 & -8 & 1.4 \end{bmatrix}$

8) Hallar TDC2D de $c(k, l) = \begin{bmatrix} 2 & 3 \\ 3 & -5 \\ -2 & 8 \end{bmatrix}$

9) Encontrar la TDC2D de una imagen que contenga una frecuencia única de valores de 120Hz y 70Hz, frecuencias de muestreo 512Hz y el tamaño 512*512 pixeles.

10) Demostrar la propiedad de linealidad y la propiedad de desplazamiento en frecuencia usando la TDC 2D, utilice frecuencias de 800Hz y 450Hz para cada una de las imágenes y una frecuencia de muestreo de 2.5Khz.

11) Demostrar que es posible eliminar frecuencias utilizando TDC, realice un filtro paso bajo con frecuencia de corte de 400Hz. Realice un mapa de entrada - salida y concluya que tan importante es realizar filtrado de señales basado en TDC.

12) realice un registro de voz de la oración “mi casa está vacía”, aplicar la transformada discreta del coseno y comprima a una razón del 75%. Reproduzca la señal resultante y realice una comparación del mensaje con la señal original, concluya si una señal de voz se puede comprimir al 75% usando discreta del coseno y conserva el mensaje original.

13) Aplique la TDC-2D, a una imagen cualquiera”, aplique compresión a una razón del 87.5%. Concluya si es posible a nivel de imagen comprimir a esta razón, qué información se pierde y qué información se preserva en el ejercicio.

Referencias

1. Vyas, A., Yu, S., & Paik, J. (2018). *Multiscale Transforms with Application to Image Proc*
2. Joshi, M. A. (2018). *Digital image processing: An algorithmic approach*. PHI Learning Pvt. Ltd..
3. López, R. R. (2016). *Identificación de la fuente en vídeos de dispositivos móviles*.
4. Aguirre Martín, F. (2017). Desarrollo y análisis de clasificadores de señales de audio.
5. Peris, P. B., & González, M. S. (2017). *Autenticación de Imágenes Digitales Mediante Patrones Locales de Texturas*. UNIVERSIDAD COMPLUTENSE DE MADRID.
6. Huamán, C. Q. (2016). *Técnicas Anti-Forenses para Vídeos de Dispositivos Móviles*.

7. Lezama, J. (2015, October). Image compression by Johnson graphs. In *2015 XVI Workshop on Information Processing and Control (RPIC)* (pp. 1-6). IEEE.
8. Cruz, K. J. A. (2017). *Desarrollo de un algoritmo de compresión de datos optimizado para imágenes satelitales* (Bachelor's thesis).
9. Mondragón Contreras, S. (2018). Sistema de inteligencia artificial para el control de androides autónomos.
10. Torres, D. F. M. (2016). *Pronóstico de vida útil restante en rodamientos, con base en datos de vibraciones y sistemas de inferencia estocástica con degradación no lineal* (Doctoral dissertation, Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Maestría en Ingeniería Eléctrica.).
11. Lezama, J. (2017). COMPRESIÓN DE IMÁGENES: FORMATO JPEG. *Revista de Educación Matemática*, 32(2).
12. Martínez-Aponte, J. M., & Stivenson-Pinto, S. (2015). Design of a communication system

- between deaf people and hearing people. *Iteckne*, 12(2), 138-145.
13. Martínez-Aponte, J. M., & Stivenson-Pinto, S. (2015). Diseño de un sistema de comunicación entre personas sordas y personas oyentes. *ITECKNE*, 12(2), 138-145.
 14. Benítez López, J. (2016). El sistema de compresión JPEG. Un pequeño paseo por la transformada discreta de Fourier y la coseno. *Gaceta de la Real Sociedad Matemática Española*, 19(1), 25-45.
 15. Cruz Rodríguez, V. (2012). *Diseño de un codificador de imágenes adaptativo multitransformada mediante el uso de la transformada Karhunen-Loève* (Bachelor's thesis).
 16. Amer, I., Hishmat, P., Badawy, W., & Jullien, G. (2010). Comparisons and Analysis of DCT-based Image Watermarking Algorithms. In *Advanced Techniques in Computing Sciences and Software Engineering* (pp. 55-58). Springer, Dordrecht.

17. Soria Lorente, A., Cumbreira González, R. A., & Fonseca Reyna, Y. (2016). Algoritmo esteganográfico de clave privada en el dominio de la transformada discreta del coseno. *Revista Cubana de Ciencias Informáticas*, 10(2), 116-131.
18. Lloris, A., Fernández, P. G., & Ramírez, J. (2001). Procesamiento de imágenes utilizando la transformada discreta coseno. *Revista española de electrónica*, (558), 72-75.
19. Ramos, A. I. C., Riverón, E. M. F., Ramírez, P. M., & Pogrebnyak, O. B. (2016). Filtro de restauración de imágenes basado en la transformada discreta del coseno y el análisis de componentes principales. *Research in Computing Science*, 120, 169-178.
20. Portocarrero Rodríguez, M. A. (2018). Diseño de la arquitectura de transformada discreta directa e inversa del coseno para un decodificador HEVC.
21. Checa, H., & Andrés, M. (2017). *Diseño e implementación de un prototipo para adquisición y compresión de señales ECG con filtros coseno modulado* (Bachelor's thesis, Universidad de las Fuerzas Armadas

ESPE. Carrera de Ingeniería en Electrónica, Automatización y Control.).

22. Reyes Rodriguez, V. (2015). *Study of accuracy and hardware performance in discrete transforms and their fast algorithms* (Doctoral dissertation).
23. Avila-Domenech, E. (2018). Marca de agua digital basada en DWT-DCT para imágenes de documentos manuscritos: optimización contra ataques de compresión JPEG. *Revista Cubana de Ciencias Informáticas*, 12(2), 30-43.
24. Moya, S., Hadad, M., Funes, M., Donato, P., & Carrica, D. (2017, September). Different alternatives for the use of Cosine Transform in OFDM systems. In *2017 XVII Workshop on Information Processing and Control (RPIC)* (pp. 1-5). IEEE.
25. Hernández, J. L., Bautista, C. V., Miyatake, M. N., & Meana, H. P. (2015). Algoritmo Esteganografico Robusto a Compresión JPEG Usando DCT. *Instituto Politécnico Nacional*, 6.

26. García-Pinzón, J. A., Mendoza, L. E., & Flórez, E. G. (2015). Electronic control arm using electromyographic signals. *Facultad de Ingeniería*, 24(39), 71-84.
27. García-Pinzón, J. A., Mendoza, L. E., & Flórez, E. G. (2015). Control de brazo electrónico usando señales electromiográficas. *Facultad de Ingeniería*, 24(39), 71-84.
28. Gamboa Córdova, R. G. (2017). *Diseño e implementación de un sistema MIMO Fast OFDM en módulos NI-USRP* (Bachelor's thesis).
29. Alfonte Zapana, R. (2018). Reducción de la dimensionalidad de series temporales climáticas usando Deep Multi-Layer Autoencoder.
30. Moreno-Alvarado, R., Pérez-Meana, H., Nakano-Miyatake, M., & Robles-Camarillo, D. (2019). Método de compresión de electrocardiogramas basado en muestreo compresivo.
31. Avila-Domenech, E. (2018). Marca de agua digital basada en DWT-DCT para imágenes de documentos manuscritos: optimización

- contra ataques de compresión JPEG. *Revista Cubana de Ciencias Informáticas*, 12(2), 30-43.
32. Coy, L., Orjuela, L., & Jiménez, F. (2018). Compresión de video en Streaming usando transformadas Wavelet y DCT. *Infometric@-Serie Ingeniería, Básicas y Agrícolas*, 1(2).
33. Alkawaz, M. H., Sulong, G., Saba, T., & Rehman, A. (2018). Detection of copy-move image forgery based on discrete cosine transform. *Neural Computing and Applications*, 30(1), 183-192.
34. Mahmood, T., Mehmood, Z., Shah, M., & Saba, T. (2018). A robust technique for copy-move forgery detection and localization in digital images via stationary wavelet and discrete cosine transform. *Journal of Visual Communication and Image Representation*, 53, 202-214.
35. Siddiqui, M., Siddiqi, I., & Khurshid, K. (2018, March). Feature Extraction for Cursive Language Document Images: Using Discrete Cosine Transform, Discrete Wavelet Transform and Gabor Filter. In *Proceedings of the 2nd Mediterranean Conference on Pattern*

Recognition and Artificial Intelligence (pp. 84-87). ACM.

36. Gong, L., Deng, C., Pan, S., & Zhou, N. (2018). Image compression-encryption algorithms by combining hyper-chaotic system with discrete fractional random transform. *Optics & Laser Technology*, 103, 48-58.
37. Li, X. Z., Chen, W. W., & Wang, Y. Q. (2018). Quantum image compression-encryption scheme based on quantum discrete cosine transform. *International Journal of Theoretical Physics*, 57(9), 2904-2919.
38. Alotaibi, R. A., & Elrefaei, L. A. (2018). Text-image watermarking based on integer wavelet transform (IWT) and discrete cosine transform (DCT). *Applied Computing and Informatics*.
39. Miri, A., Sharifian, S., Rashidi, S., & Ghods, M. (2018). Medical image denoising based on 2D discrete cosine transform via ant colony optimization. *Optik*, 156, 938-948.
40. Smith, J. S., & Wilamowski, B. M. (2018, June). Discrete Cosine Transform Spectral Pooling Layers for Convolutional Neural Networks. In *International Conference on*

Artificial Intelligence and Soft Computing (pp. 235-246). Springer, Cham.

41. Prabukumar, M., Sawant, S., Samiappan, S., & Agilandeewari, L. (2018). Three-dimensional discrete cosine transform-based feature extraction for hyperspectral image classification. *Journal of Applied Remote Sensing*, 12(4), 046010.
42. Jain, A., Pandey, N., & Jain, P. (2019). FPGA-Based Architecture for Implementation of Discrete Sine Transform. In *Advances in System Optimization and Control* (pp. 13-22). Springer, Singapore.
43. Tan, E. L., & Gan, W. S. (2015). *Perceptual Image Coding with Discrete Cosine Transform*. New York: Springer-Verlag Singapur.
44. Rao, K. R., & Ochoa-Dominguez, H. (2019). *Discrete Cosine Transform*. CRC Press.

CAPÍTULO III

TRANSFORMADA WAVELET DISCRETA (TWD)

Luis Enrique Mendoza,
Ingeniería en Telecomunicaciones,
Universidad de Pamplona.

La **TWD**, es la herramienta que en los últimos años ha sido utilizada para realizar procesos que exigen análisis con resolución tiempo - frecuencia. TWD, es una técnica que permite tener diferentes escalas, representaciones o resoluciones de una misma señal o señal original.

3.1 Introducción análisis Wavelet

La transformada wavelet tiene como característica principal la descomposición de señales o imágenes en un conjunto de sub señales o imágenes llamadas coeficientes de detalle o aproximación, que permiten realizar análisis detallado basado en componentes de frecuencia, esto es conocido comúnmente como análisis multi-resolucional, o análisis multi-nivel.

La transformada wavelet discreta tiene una particularidad, y es que en su desarrollo realiza un proceso de compresión de información conocido como downsamplig (eliminación de muestras). Esto se realiza con el objeto de conservar la propiedad del

teorema de Shannon y Nyquist de reconstrucción de datos. Así mismo, la transformada wavelet se descompone en transformada wavelet continua y transformada wavelet discreta. En este libro se hará énfasis en la transformada wavelet discreta y sus múltiples aplicaciones que son: compresión, análisis de frecuencia, extracción de patrones, eliminación de ruido, realce de contornos, y detección de cambios rápidos en las señales. Algo interesante de la TWD es que permite realizar análisis tiempo-frecuencia, lo que indica que es posible conocer de manera más detallada características de los datos que se procesan.

Dentro de wavelet es importante recomendar las siguientes notaciones:

1. **Down-sampling**
2. **Up-sampling**
3. **Análisis multiresolución**
4. **Coeficientes de detalle**
5. **Coeficientes de aproximación**
6. **Coeficientes de descomposición**
7. **Coeficientes de reconstrucción**

Matemáticamente wavelet discreta 1D (TWD-1D) está definida como:

$$T_{x,y} = T(2^{-j}t - 2^{-j}r) \quad (3.1)$$

$$C_{x,y} = \sum x(t) * T_{x,y} \quad (3.2)$$

Donde T se denomina wavelet madre y C la transformada wavelet discreta en 1D.

Nótese que la transformada depende de dos factores: una de desplazamiento (r) y otro de escala 2^j . Estos parámetros permiten que se puedan obtener diferentes resoluciones de la misma señal o imagen a procesar.

La

Figura 3.1, muestra de manera gráfica el funcionamiento de las variables escala y traslación. Nótese como se traslada una wavelet madre (señal roja), con escalada de 50. Aquí se desplaza la

wavelet madre hasta el final de la señal original.

Seguidamente, en la

Figura 3.2 se muestra la misma wavelet madre pero a escala de $\frac{1}{2}$, equivalente a escala 25, y así mismo, se desplaza la señal hasta el final de señal original.

Cada escala y cada recorrido de toda la wavelet madre por la señal original se considera como una resolución wavelet, en el ejemplo anterior se tiene dos resoluciones de la señal original.

En la transformada wavelet se dice que a menor resolución en el tiempo de la wavelet madre, es decir, escalas pequeñas, se consigue representaciones de alta frecuencia, y si la wavelet madre tiene una resolución alta o duración en tiempo alta o escala alta, se representa las componentes de frecuencia de menor valor o componentes de baja frecuencia. Esto debido a la teoría: a menor longitud de onda, mayores son las frecuencias encontradas.

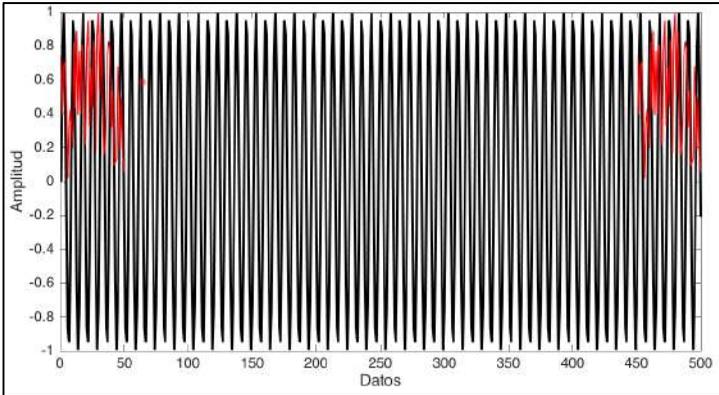


Figura 3.1.
Análisis wavelet escala 50

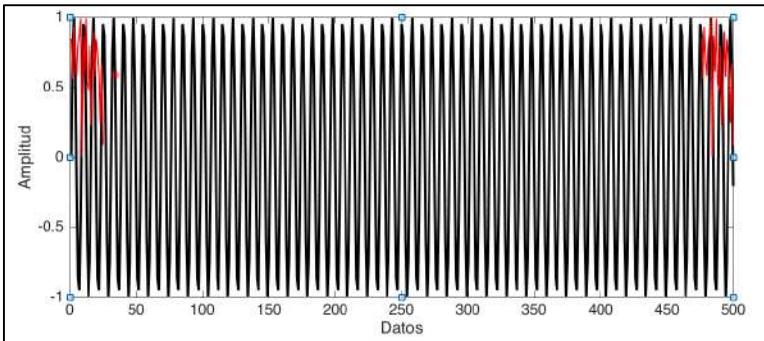


Figura 3.2.
Análisis wavelet escala 25.

Finalmente, la

Figura 3.3, muestra el proceso desplazamientos y escalados para el análisis de frecuencia basado en wavelet. Nótese como la escala trabaja en un factor diádico y se realizan desplazamientos hasta la longitud final o tamaño del vector.

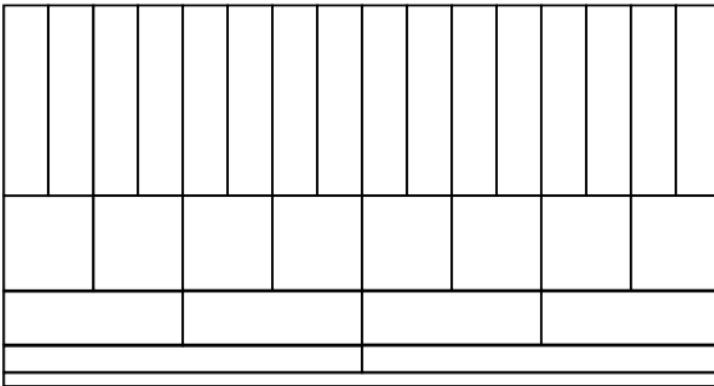


Figura 3.3.
Cálculo de resoluciones usando wavelet.

Por otro lado, es importante mencionar que wavelet en 1D, trabaja dos tipos de coeficientes: coeficiente de aproximación y coeficiente de detalle. El coeficiente de aproximación es el encargado de representar frecuencias bajas es decir escalas altas de la wavelet madre y el coeficiente de detalle es el

encargado de representar componentes de frecuencia altas alertas, es decir, escalas de la wavelet madre bajas. Siguiendo esta línea, es decir, que la transformada wavelet es una descomposición de la señal en múltiples frecuencias, lo cual puede ser demostrado o expresado de la siguiente forma.

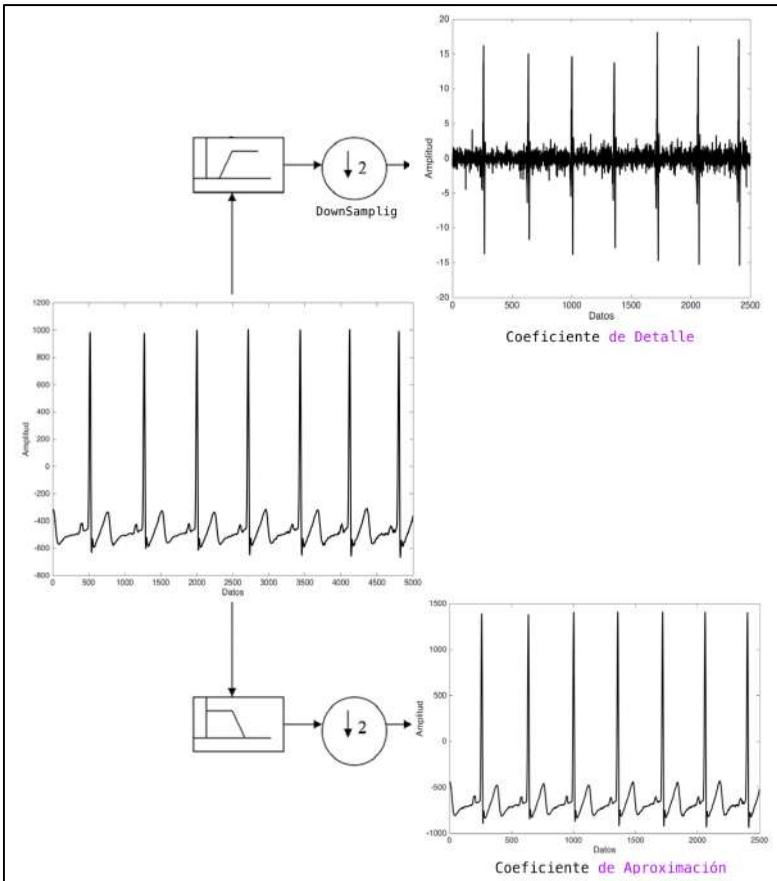


Figura 3.4.
Desarrollo transformada wavelet discreta 1D.

En la

Figura 3.4 se puede observar los coeficientes de aproximación (**CA**) y coeficiente de detalle (**CD**), nótese que el coeficiente de detalle contiene las componentes de alta frecuencia de la señal electrocardiográfica, es decir, la señal ha pasado por un filtro paso alto, así mismo, el coeficiente de aproximación contiene las componentes de baja frecuencia de la señal original, es decir, se obtiene a partir de pasar la señal por un filtro paso bajo. Aquí también se observa el efecto del término down-samplig, quien es el encargado de eliminar información redundante para el proceso de reconstrucción. Aquí, la señal original tiene 5000 puntos y los coeficientes hallados muestran una longitud de 2500 puntos, lo que es posible para reconstruir una señal con estas características. Es importante entender que este es el primero proceso de descomposición, si se desea mayor número de coeficientes se debe descomponer ya sea el coeficiente de detalle o el de aproximación según la aplicación a realizar.

En conclusión, la transformada wavelet, trabaja en función de procesos de filtro paso bajo y paso alto con descomposición diádica, lo que indica que es posible tener diferentes resoluciones de la misma señal cumpliendo con el proceso basado en filtros.

3.2 Transformada wavelet discreta inversa 1D

La transformada wavelet discreta inversa 1D está definida como:

$$x(t) = \sum C_{x,y} * T_{x,y} \quad (3.3)$$

El propósito del proceso inverso o de reconstrucción, es lograr obtener los datos originales a partir de las diferentes resoluciones o descomposiciones obtenidas.

Gráficamente el proceso de descomposición y de reconstrucción está definido como (ver

Figura 3.5):

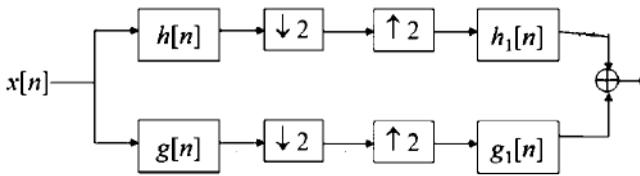


Figura 3.5.
Proceso de descomposición y reconstrucción.

Es importante mencionar que cada wavelet madre tiene relacionada un conjunto de coeficientes que sustituyen la wavelet madre con un conjunto de banco de filtros, que tiene una relación directa con la wavelet madre. Por ejemplo, el filtro paso alto de descomposición está relacionada con la wavelet madre utilizada para descomposición y el banco de filtro paso bajo está relacionada con los coeficientes de la wavelet madre utilizada.

Por otro lado, el proceso de descomposición tiene relación con el proceso de reconstrucción, siendo así que los coeficientes en los dos procesos tienen una relación de espejo. Esto quiere decir que si los coeficientes de la wavelet madre db1 para un

proceso de descomposición paso alto son $[-0.70 \ 0.70]$, los coeficientes para el proceso de reconstrucción serán $[0.70 \ -0.70]$, aquí se demuestra la relación directa. Entiéndase que wavelet trabaja en un banco de filtros relacionados con las wavelets madres definidas.

La **wavelet madre** es aquella función que permite realizar el proceso de escala y traslación para lograr encontrar los niveles de descomposición y las características en cada tipo de señal. Existen actualmente más de 70 tipos de wavelet madre entre lo más representativas están db y sym, las cuales han sido utilizadas en diferentes aplicaciones como: señales fisiológicas, imágenes, señales de voz, entre otras. Una wavelet madre debe ser ortogonal y ortonormal. El uso del tipo de wavelet madre, depende de la forma de onda de la señal a procesar y de la aplicación a realizar, por ejemplo, en las señales de electromiografías ha sido utilizado la wavelet madre Daubechies 3 (db3), ya que el

comportamiento de los potenciales de acción de unidad motora tiene una correlación alta con la db3. Cada wavelet genera una respuesta diferente en los niveles de aproximación y detalle, por ende, el uso de la misma no está definido estrictamente de ninguna manera.

Para finalizar el desarrollo matemático y teórico de la transformada wavelet en 1D, se muestran a continuación algunos ejemplos gráficos de la wavelet madres (ver Figura 3.6). Nótese como las formas de ondas de cada una de ellas es diferente, lo que conlleva a concluir que usar distintas wavelets madres permite tener diferentes respuestas en función de la misma señal a procesar.

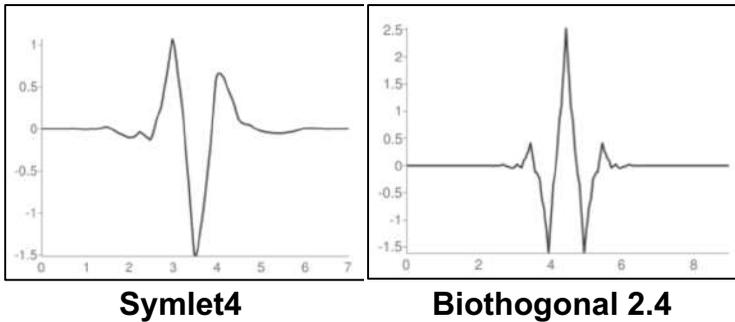


Figura 3.6.
Ejemplos de wavelet madre.

3.3 Ejemplos transformada wavelet 1D

A continuación, se muestran ejemplos del procesamiento de señales usando transformada wavelet discreta. Dentro de los ejemplos a presentar se encuentran descomposición, eliminación de ruido, detección de cambios rápidos en el tiempo y extracción de patrones.

3.3 1 Descomposición

En la

Figura 3.7, se muestra la descomposición wavelet a un nivel de descomposición (para una señal electrocardiográfica (ECG) con una wavelet madre db2, nótese como la señal original

Figura 3.7a, contiene 10000 puntos y en el primer proceso de descomposición se obtiene un coeficiente de detalle (

Figura 3.7c) y un coeficiente de aproximación (

Figura 3.7b). Además, nótese como estos coeficientes muestran una cantidad de datos reducida a la mitad de la señal original lo que significa que se ha aplicado el downsamplig, de este contexto se depende el tema de compresión de señales, es decir, el coeficiente de aproximación tiene características en morfología o en forma que la señal original. La pregunta a responder para poder trabajar con una señal más reducida en tamaño que la origina es: ¿se puede realizar el mismo proceso con la señal original que con la señal del coeficiente de aproximación?, si la respuesta es sí, entonces se puede decir que se utilizó wavelet para disminuir la

resolución o comprimir la señal y tener un sistema que permita realizar la tarea usando menos datos que los originales.

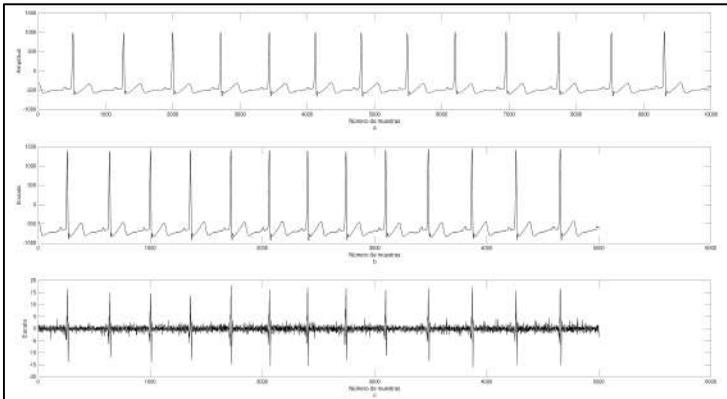


Figura 3.7.
Transformada wavelet nivel 1 y db2.

Así mismo,

Figura 3.8, muestra un segundo nivel de descomposición de la señal electrocardiográfica. Nótese como esta señal en primera estancia contiene $\frac{1}{4}$ del tamaño de la señal original, sin

embargo, se observa que el coeficiente de aproximación (Figura 3.8a), es similar en forma a la señal original. Así mismo, existe o se obtiene otro coeficiente de detalle (Figura 3.8b) que contiene diferentes frecuencias. Por ejemplo, si la tarea fuera contar el número de ondas R que existe en la señal ECG, seguro se pediría utilizar la señal del coeficiente de aproximación en el segundo nivel descomposición ya que mantiene esta información a pesar de que tiene 4 veces menos datos. La respuesta para esta aplicación sería 13 picos u ondas R.

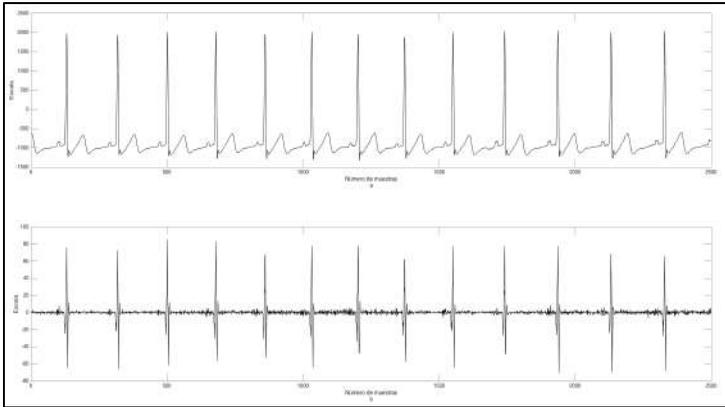


Figura 3.8.
Transformada wavelet nivel 2 y db2.

Para la compresión de datos usando wavelet, se debe tener en cuenta el proceso de downsampling, quien es el encargado de eliminar muestras conservando el teorema de Nyquist. En este caso es importante mencionar que entre mayor es el nivel de descomposición mayor es el nivel de compresión. Ejemplo, para un nivel de descomposición existe un factor de compresión del 50%, es decir la señal de salida tiene la una longitud de $L/2$ que la señal original. Si existe un nivel de descomposición 4 el

nivel de compresión es de aproximadamente el 93.5% lo cual indica que el proceso de compresión es bastante amplio. Debe tenerse en cuenta que wavelet es un sistema de compresión con pérdida, ya que en general el nivel de selección o señal comprimida es el coeficiente de aproximación del nivel de descomposición seleccionado. Aunque no es el detalle relevante o la aplicación más importante en wavelet, se debe considerar si la aplicación a desarrollar lo permite. Hay que recordar que dependiendo de los datos se podría usar una etapa de compresión para disminuir costo computacional, y si los resultados son importantes justifican el proceso.

A continuación, se explica el proceso de compresión usando la transformada wavelet discreta de manera práctica y gráfica. Se debe hacer bastante énfasis en la longitud de la señal y así mismo la morfología de la señal comprimida, que en su efecto, la TWD a diferencia de otras herramientas de compresión

mantiene en cierto porcentaje la morfología de la señal original. Por último, se debe mencionar que es posible reconstruir la señal de un tamaño igual al original, siempre y cuando se utilice de la manera correcta el proceso de reconstrucción, donde sin importar que cambio exista en cada nivel, se deben utilizar todas las resoluciones o imágenes que se encontraron en el proceso de descomposición.

3.3 2 Eliminación de ruido

Para eliminar ruido basado en wavelet se necesita inicialmente el concepto de filtro suave y filtro duro. En la ecuación (3. 4), se muestran los dos tipos de umbrales que se utilizan para esta aplicación.

$$D_s(u, \lambda) = \begin{cases} \emptyset & \text{si } |u| \leq \lambda \\ u - \lambda & \text{si } u > \lambda \\ u + \lambda & \text{si } u < -\lambda \end{cases} \quad (3.4)$$

$$D_h(u, \lambda) = \begin{cases} \emptyset & \text{si } |u| \leq \lambda \\ u & \text{si } u > \lambda \end{cases}$$

Dónde: D_s denota filtrado suave, D_h denota filtrado fuerte y $\lambda = \log\sqrt{N}$ umbral universal

Nótese que adicionalmente se incluye una variable umbral (λ), que es la encargada de realizar la comparación con cada dato de la señal, entienda que es la señal a filtrar usando wavelet. Esta definición se debe aplicar a cada uno de los coeficientes de ser el caso y dependiendo de las componentes de frecuencia a filtrar o eliminar. Para esta aplicación debe entender que cada nivel de descomposición o resolución tiene características de frecuencia diferentes, por ende, es prácticamente relevante realizar filtrado usando wavelet.

A continuación, se explica el proceso de filtrado usando wavelet de manera práctica y gráfica (ver Figura 3.9).

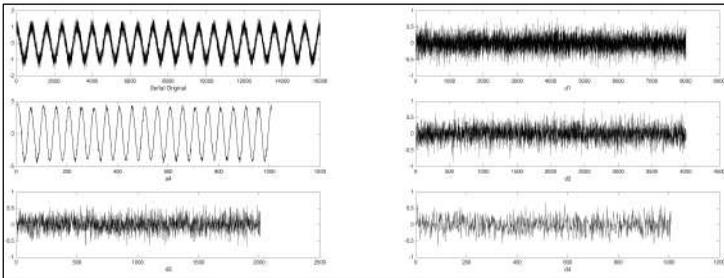


Figura 3.9.
Descomposición wavelet par filtrado de una señal.

En el ejercicio se aplicó el filtrado en donde todos los coeficientes de detalles se llevan a un valor cero. Siguiendo esta directriz, se tiene una reconstrucción de la señal de la forma ver Figura 3.10.

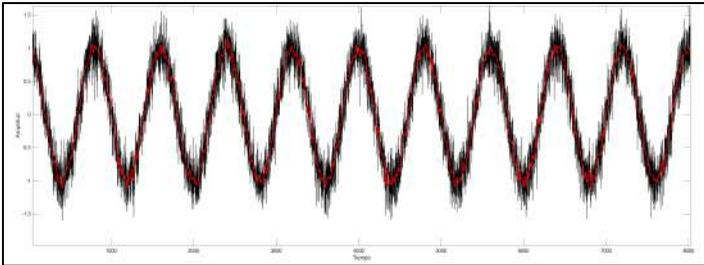


Figura 3.10.
Resultado proceso de filtrado.

En este ejemplo se tomaron 4 niveles de descomposición y una wavelet madre db5. Cabe resaltar que se debe seleccionar otros tipos de wavelet madre para observar el funcionamiento de cada una de ellas.

3.3.3 Extracción de patrones

La extracción de patrones es un proceso relevante ya que permite identificar o caracterizar una señal o un conjunto de señales. Esta caracterización es conocida como patrones característicos de un conjunto de datos o de una señal. La transformada wavelet discreta es una técnica que permite extraer

patrones significativos basados en componentes de frecuencia. En la Figura 3.11 y la Figura 3.12, se muestra un ejemplo con db2, y 5 niveles de descomposición para las palabras registradas '**Casa**' y '**Mamá**'. Con el propósito de observar cómo los diferentes niveles de descomposición permiten evidenciar con mejor resolución las características que diferencian estos dos grupos de señales. El objetivo es identificar cuál de los diferentes niveles obtenidos hace que estas dos palabras tengan una diferencia significativa. Es importante mencionar que en este campo de la extracción de patrones se debe incluir técnicas como: desviación estándar, varianza, las cuales permiten tener un análisis cuantitativo del proceso realizado, o en otras palabras verificar que tan relevantes son los patrones. En este ejercicio se eligieron 5 niveles. Realizando un análisis de cada nivel usando desviación estándar y análisis de la varianza, se concluyó que el nivel d4 es el más

adecuado para realizar el proceso de extracción de patrones de estos dos tipos de palabras. En conclusión, o en otras palabras, se puede decir que si desea realizar un proceso de clasificación se recomienda que se utilice el nivel de descomposición 5 y una db4, ya que las diferencias se hacen más significativas en este nivel que en cualquier otro sumado a la señal original. Por otro lado, se recomienda que estos análisis no se realicen en una sola señal de cada palabra, se recomienda que por lo menos existan 15 palabras por cada grupo, es decir 15 señales de la palabra *Casa* y 15 de la palabra *Mamá*, esto para mejorar la confiabilidad del proceso.

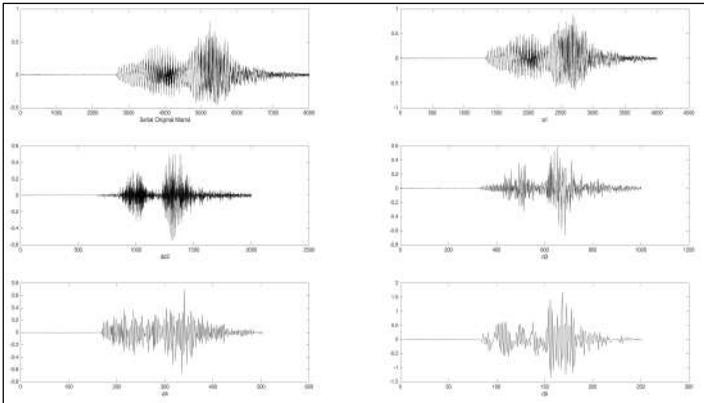


Figura 3.11.
Resultado procesos de extracción de patrones Casa.

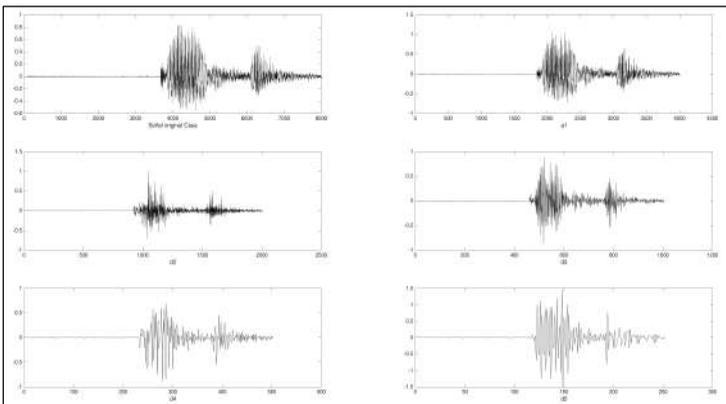


Figura 3.12.
Resultado procesos de extracción de patrones Mamá.

3.3.4 Compresión de señales

El proceso de compresión usando wavelet se consolida en el uso de los coeficientes de aproximación, es así como en la

Figura 3.13, se muestra un ejemplo gráfico del proceso de compresión usando una wavelet madre db5 y 2 niveles de descomposición. En este ejemplo se observa una compresión al 50% y una compresión al 75%, es decir, se pasa de tener 6800 datos a tener 3400 y 1700 respectivamente. Nótese como las formas de onda o morfología de la señal se conserva de una manera importante, incluso cuando las longitudes son diferentes. Es importante mencionar que wavelet es una técnica de compresión con pérdida y se debe lograr un análisis donde la pérdida de información sea la menos posible, esto buscando la wavelet madre más adecuada para el proceso que se desea hacer y la señal a comprimir.

Finalmente, cada nivel de descomposición wavelet permite una compresión de longitud $L/2$, es decir que

los factores de compresión son diádicos y esto constituye una fortaleza en cuanto a procesos de compresión se refiere, por las cantidades de información que en ocasiones maneja un conjunto de señales o datos.

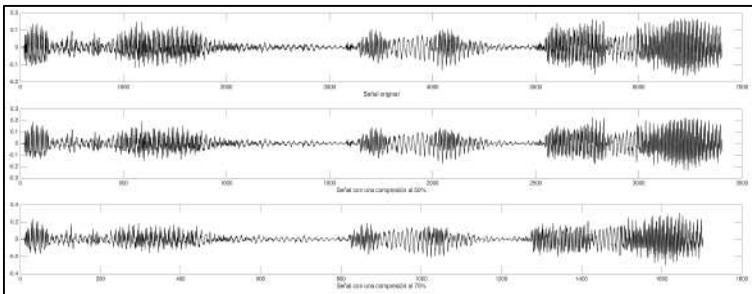


Figura 3.13.
Compresión de información 2 niveles de descomposición.

3.4 Transformada wavelet en 2D

Matemáticamente wavelet discreta 2D está definida como:

$$W_{x,y}^{i,j} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) * \phi_{x,y}^{i,j} \quad (3.5)$$

Donde ϕ se define como wavelet madre 2D, que al igual que en una dimensión tiene propiedad de desplazamiento (i), y escalado (j), y $f(x, y)$ es la imagen original.

La transformada wavelet inversa está definida como:

$$f(x, y) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} W_{x,y}^{i,j} * \phi_{x,y}^{i,j}$$

(3. 6)

La descripción grafica en bloques de la Transformada wavelet discreta 2D se muestra en la Figura 3. 14, Nótese como para un solo nivel de descomposición aparecen 4 imágenes que representan a la imagen original. Es de resaltar que cada descomposición o resolución tiene componentes de frecuencias diferentes. Aquí aparecen los coeficientes de aproximación ($x_{1,H3}$), y de detalle: horizontal ($x_{1,H2}$), vertical ($x_{1,H1}$) y diagonal ($x_{1,l}$). Si se desea encontrar otro nivel de

descomposición, por lo general, se recomienda que la descomposición se realice al coeficiente de aproximación, donde se obtienen 4 resoluciones de la imagen de aproximación.

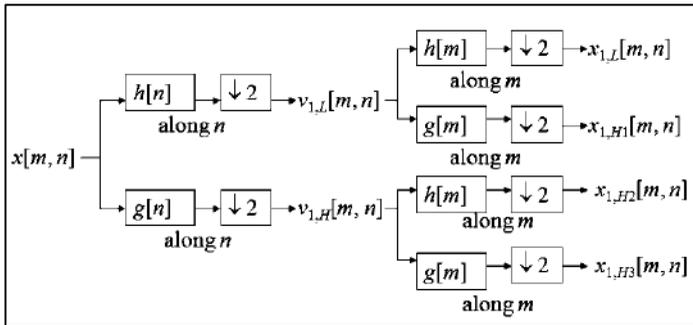


Figura 3. 14.
Diagrama bloques TWD-2D.

A continuación, se muestran dos ejemplos de la transformada wavelet 2D a dos niveles de descomposición, con una wavelet madre symlet.

3.4.1 Descomposición en 2D

Ejemplo 1.

En la Figura 3.15, se presenta un ejemplo del uso de la TWD-2D. Nótese como para dos niveles de descomposición se consigue 7 imágenes que representan la imagen original, aparecen coeficientes de aproximación, coeficientes de detalles verticales, diagonales, y horizontales (ver Figura 3.15b). Cada uno de ellos con unas componentes de frecuencia diferentes. Nótese como el coeficiente de aproximación 2, mantiene información importante de la imagen original.



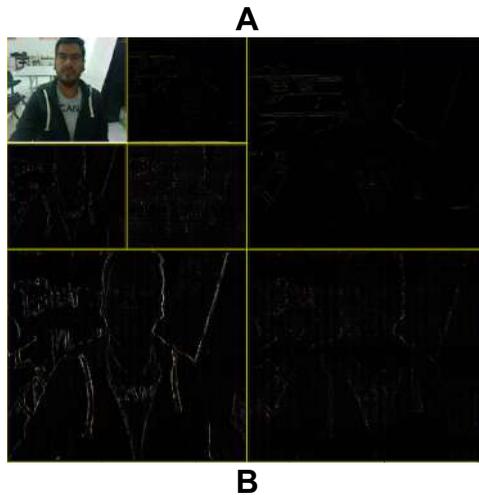


Figura 3.15.
Transformada wavelet 2D. 2 niveles de descomposición
imagen a color.

Ejemplo 2.

En el ejemplo de la Figura 3. 16, y Figura 3. 17, se muestra una descomposición a un solo nivel de una imagen a nivel gris, nótese que se obtienen 4 imágenes (ver Figura 3. 17b) que son las resoluciones de la imagen original.



Figura 3. 16.
Imagen a nivel de gris original.



A



B



C



D

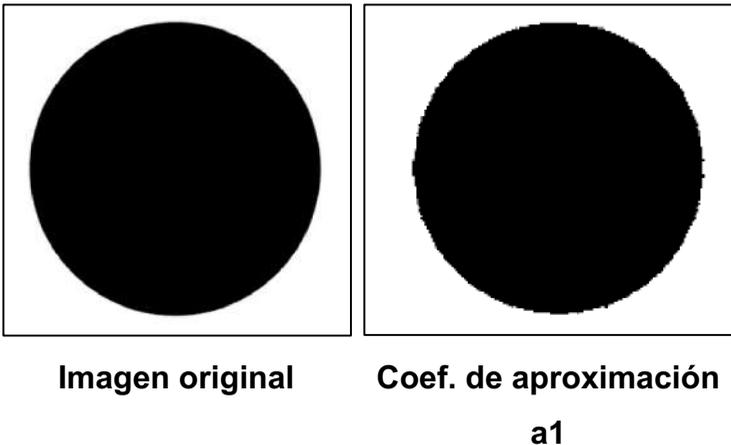
Figura 3. 17.

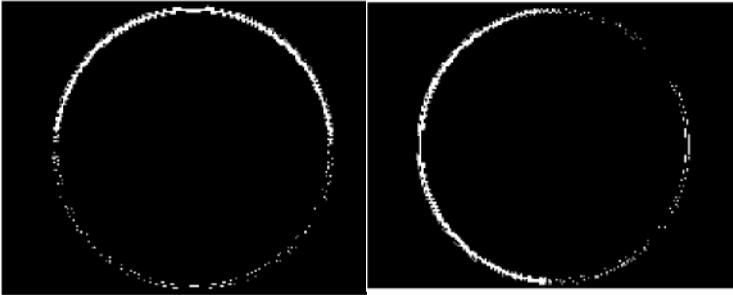
Transformada wavelet 2D con 2 niveles de descomposición imagen a nivel de gris. a) aproximación. b) d. vertical. c) d. horizontal. d) d. Diagonal.

Una vez obtenido las resoluciones y niveles de descomposición, y la wavelet madre, se procede a realizar las aplicaciones posibles. En este libro se prestarán aplicaciones interesantes: detección o realce de contornos y, compresión y extracción de patrones, con fines de aplicar procesamiento de imágenes usando wavelet.

Ejemplo 3. Selección de información relevante (contornos). Ya definiendo la estructura matemática en 2D, a continuación, se presenta un ejemplo sencillo usando una figura geométrica que contiene dos estados: blanco o negro (ver Figura 3.18). En este ejemplo se muestra una comparación de resultados en utilizar una wavelet madre db1, y una wavelet madre db4, con el objetivo de tener un panorama claro de los resultados en

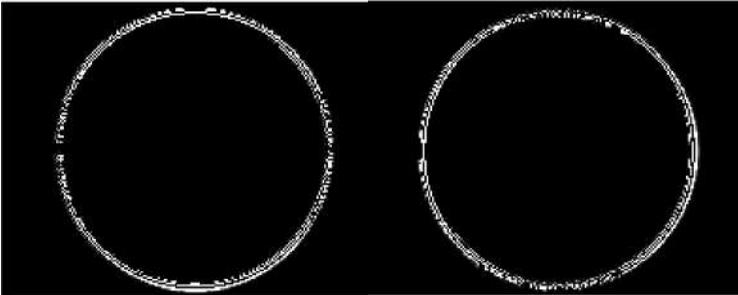
función de la selección de la wavelet madre. Nótese como los coeficientes de detalles muestran los contornos de la figura o imagen original, sin embargo, los resultados son más eficientes si utiliza la wavelet madre db4. Esto indica que la TWD permite calcular los contornos de una imagen binaria, y que permite obtener componentes de frecuencia diferentes en cada uno de los coeficientes de detalle.





**Coef. de detalle 1,
madre db1**

**coef. detalle 2. Madre
db1**



**Coef. de detalle 1.
Madre db4**

**Coef. detalle 2. Madre
db4**

Figura 3.18.
Ejemplo TW 2D.

3.4.2 Compresión 2D

La compresión tiene como objetivo disminuir la cantidad de información de un conjunto de datos, sin embargo, esta disminución de información debe mantener por lo menos el 85% de la información importante de la imagen original. En el caso de TWD, se utilizan los coeficientes de aproximación para realizar este proceso. En el ejemplo (ver Figura 3.19), se muestran diferentes resultados de compresión usando una wavelet madre daubechies y symlet, en 1 y 2 niveles de descomposición. Es decir, que se comprime a 75% y 87.25%. Sin embargo, es de apuntar que las imágenes obtenidas mantienen la información importante de la imagen original. En la Figura 3.19c, Figura 3.19d y Figura 3.19e, se muestra compresión con un nivel de descomposición 1 y nivel de descomposición 2, para una wavelet madre db5, y el mismo procedimiento para una wavelet madre symlet6. Así mismo, se

puede notar que el proceso de compresión basado en TWD, es eficiente y permite consolidar un proceso de extracción de patrones, ya que se obtiene información relevante de la imagen original, pero con mucho menos cantidad de información.



a) Imagen original



b) Compresión con un nivel de descomposición db5



**c) Compresión con dos niveles de descomposición
db 5**



**d) Compresión con un nivel de descomposición
sym6**



**e) Compresión con dos niveles de descomposición
sym6**

Figura 3.19.
Compresión de datos.

Con el fin de identificar, y observar que las representaciones mostradas en la Figura 3.19 mantienen un patrón relevante, se vectorizan las imágenes, y se muestran con el fin de tener una mejor resolución de las longitudes. Nótese como la morfología de las señales 1D se mantiene, a pesar de la gran diferencia en longitud. Además, se debe mencionar que el espacio wavelet se define como un espacio escala–tiempo, pero por su comportamiento en descomposición, y por banco de filtros se considera una aproximación importante de la imagen o señales originales, y por ende, se menciona el término compresión de datos. Nótese como la longitud de cada vector disminuye en un factor de 4, esto conlleva a concluir, que si la imagen original tiene un tamaño de 720×1280 , tendrá un vector de tamaño 921600, y que su compresión en un nivel de descomposición tiene un tamaño de

365*645, y una longitud de 235425, que son los datos que aparece en la Figura 3.20 y Figura 3.21. Finalmente, la imagen de dos niveles de descomposición tiene un tamaño de 188*328, y una longitud de vectorización de 61664, para el caso de una wavelet madre db5.

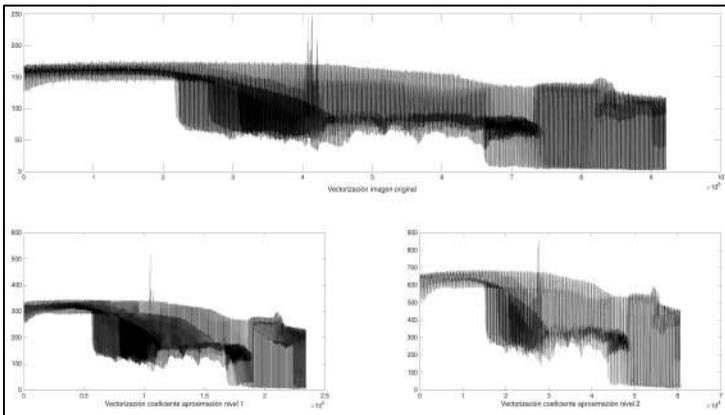


Figura 3.20.
Vectorización wavelet db5.

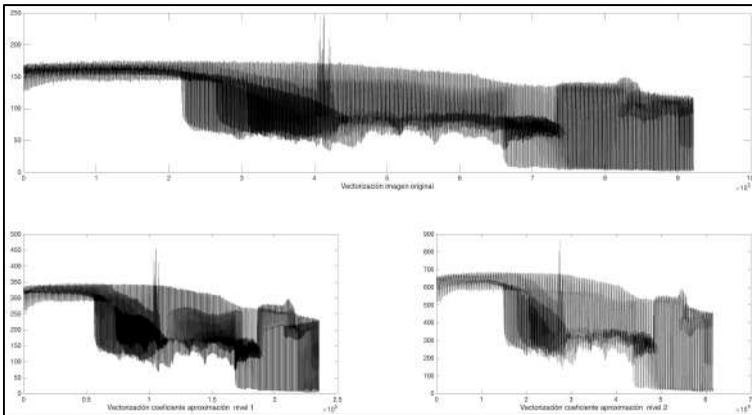


Figura 3.21.
Vectorización wavelet db5.

3.4.3 Extracción de patrones

La

Figura 3.22, muestra un ejemplo de las letras a, b y m. Se demostrará que por medio de TWD-2D, es posible tener un patrón característico que represente las 3 letras de una manera diferente. El propósito es obtener un nivel de descomposición, una wavelet madre y un coeficiente que permita tener diferencias significativas entre las letras presentadas. Aquí se usó una wavelet madre db4 y un nivel de

descomposición 3, y se seleccionó el nivel de aproximación 3.

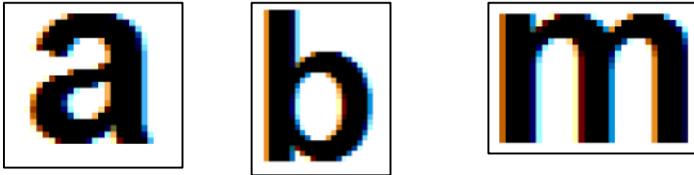


Figura 3.22.
Imágenes características de letras a, b y m.

Nótese como la
Figura 3.23, la
Figura 3.24, y la
Figura 3.25, presentan patrones diferentes que son
significativos, y seguro representan de manera
importante cada una de las letras. Aquí el patrón de
la letra **a** es muy poco denso en comparación con los
patrones de la letras **b** y **m**.

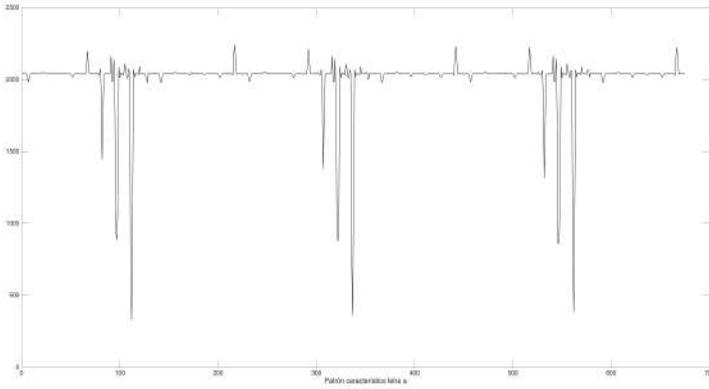


Figura 3.23.
Patrón vectorizado letra a.

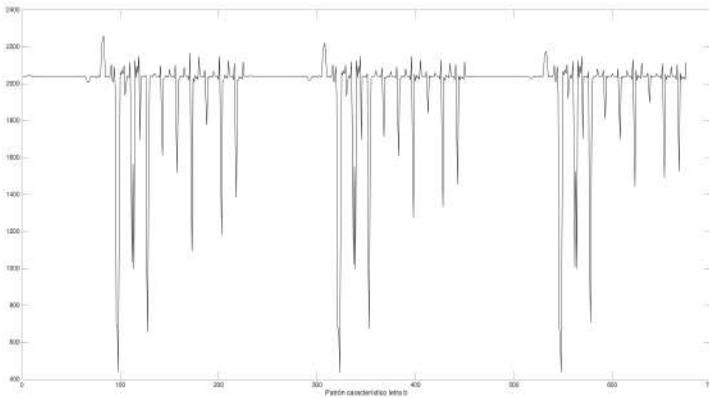


Figura 3.24.
Patrón vectorizado letra b.

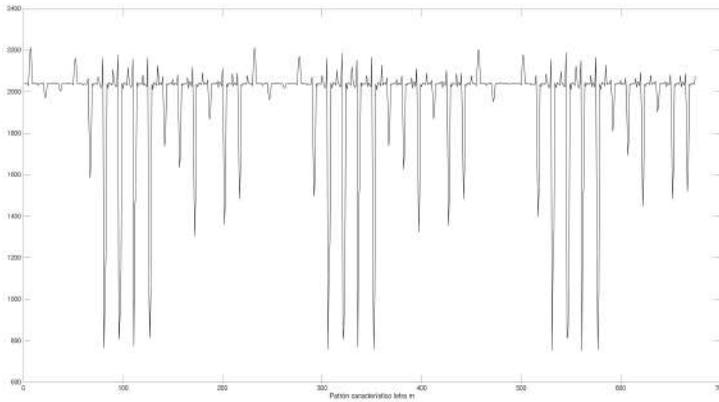


Figura 3.25.
Patrón vectorizado letra m.

Finalmente, se menciona que estos patrones deben ser susceptibles a validación usando técnicas de clasificación supervisada o no supervisadas, también, se menciona que no basta con solo una imagen para garantizar que el patrón es muy relevante, se recomienda tener como mínimo 5 imágenes de la misma letra.

Referencias

1. Chui, C. K. (2016). *An introduction to wavelets*. Elsevier.
2. Navarro J. F, Martinez D. El (2010). *Introducción a la transformada wavelet continua*. Ahmad, K. (2018).
3. Applications in Image Processing. In *Wavelet Packets and Their Statistical Applications* (pp. 203-224). Springer, Singapore.
4. Addison, P. S. (2017). *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press.
5. Ahmad, K. (2018). Applications in Image Processing. In *Wavelet Packets and Their Statistical Applications* (pp. 203-224). Springer, Singapore.
6. Zhang, D. (2019). Wavelet transform. In *Fundamentals of Image Data Mining* (pp. 35-44). Springer, Cham.
7. Chatterjee, P. (2015). *Wavelet analysis in civil engineering*. CRC Press.

8. Baleanu, D. (Ed.). (2015). *Wavelet Transform and Some of Its Real-World Applications*. BoD–Books on Demand.
9. Radhakrishnan, S. (Ed.). (2018). *Wavelet Theory and Its Applications*. BoD–Books on Demand.
10. Wang, S. H., Zhang, Y. D., Dong, Z., & Phillips, P. (2018). Wavelet Families and Variants. In *Pathological Brain Detection*(pp. 85-104). Springer, Singapore.
11. Kolekar, M. K. H., Raja, G. L., & Sengupta, S. (2018). An Introduction to Wavelet-Based Image Processing and Its Applications. In *Computer Vision: Concepts, Methodologies, Tools, and Applications* (pp. 110-128). IGI Global.
12. Subasi, A., & Yaman, E. (2019, May). EMG Signal Classification Using Discrete Wavelet Transform and Rotation Forest. In *International Conference on Medical and Biological Engineering*(pp. 29-35). Springer, Cham.

13. Krivoshein, A., Protasov, V., & Skopina, M. A. (2016). *Multivariate wavelet frames* (p. 182). Singapore: Springer.
14. Du, R. (2019). Engineering monitoring and diagnosis using wavelet transforms. In *Computer-Aided Design, Engineering, and Manufacturing* (pp. 312-341). CRC Press.
15. Haldorai, A., & Ramu, A. (2018). An intelligent-based wavelet classifier for accurate prediction of breast cancer. In *Intelligent Multidimensional Data and Image Processing* (pp. 306-319). IGI Global.
16. Meng, A., Ge, J., Yin, H., & Chen, S. (2016). Wind speed forecasting based on wavelet packet decomposition and artificial neural networks trained by crisscross optimization algorithm. *Energy Conversion and Management*, 114, 75-88.
17. Zhang, Y., Dong, Z., Wang, S., Ji, G., & Yang, J. (2015). Preclinical diagnosis of magnetic resonance (MR) brain images via discrete wavelet packet transform with Tsallis entropy and generalized eigenvalue proximal support vector machine (GEPSSVM). *Entropy*, 17(4), 1795-1813.

18. Wang, S., Li, Y., Shao, Y., Cattani, C., Zhang, Y., & Du, S. (2017). Detection of dendritic spines using wavelet packet entropy and fuzzy support vector machine. *CNS & Neurological Disorders-Drug Targets (Formerly Current Drug Targets-CNS & Neurological Disorders)*, 16(2), 116-121.
19. Bai, Y., Li, Y., Wang, X., Xie, J., & Li, C. (2016). Air pollutants concentrations forecasting using back propagation neural network based on wavelet decomposition with meteorological conditions. *Atmospheric pollution research*, 7(3), 557-566.
20. Le Douget, J. E., Fouad, A., Filali, M. M., Pyrzowski, J., & Le Van Quyen, M. (2017, July). Surface and intracranial EEG spike detection based on discrete wavelet decomposition and random forest classification. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 475-478). IEEE.
21. Wozniak, M., Napoli, C., Tramontana, E., Capizzi, G., Sciuto, G. L., Nowicki, R. K., & Starczewski, J. T. (2015, July). A multiscale image compressor with rbfnn and discrete

- wavelet decomposition. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-7). IEEE.
22. Alickovic, E., Kevric, J., & Subasi, A. (2018). Performance evaluation of empirical mode decomposition, discrete wavelet transform, and wavelet packed decomposition for automated epileptic seizure detection and prediction. *Biomedical Signal Processing and Control*, 39, 94-102.
 23. Ghassemian, H. (2016). A review of remote sensing image fusion methods. *Information Fusion*, 32, 75-89.
 24. Zheng, Y., Blasch, E., & Liu, Z. (2018). *Multispectral Image Fusion and Colorization*. SPIE Press.
 25. Singh, A. K., Kumar, B., Dave, M., & Mohan, A. (2015). Multiple watermarking on medical images using selective discrete wavelet transform coefficients. *Journal of Medical Imaging and Health Informatics*, 5(3), 607-614.
 26. Sudarshan, V. K., Mookiah, M. R. K., Acharya, U. R., Chandran, V., Molinari, F., Fujita, H., & Ng, K. H. (2016). Application of wavelet

- techniques for cancer diagnosis using ultrasound images: A Review. *Computers in biology and medicine*, 69, 97-111.
27. Lai, Z., Qu, X., Liu, Y., Guo, D., Ye, J., Zhan, Z., & Chen, Z. (2016). Image reconstruction of compressed sensing MRI using graph-based redundant wavelet transform. *Medical image analysis*, 27, 93-104.
 28. Nayak, D. R., Dash, R., & Majhi, B. (2016). Brain MR image classification using two-dimensional discrete wavelet transform and AdaBoost with random forests. *Neurocomputing*, 177, 188-197.
 29. Li, C., Huang, Y., & Zhu, L. (2017). Color texture image retrieval based on Gaussian copula models of Gabor wavelets. *Pattern Recognition*, 64, 118-129.
 30. Mughal, B., Muhammad, N., Sharif, M., Saba, T., & Rehman, A. (2017). Extraction of breast border and removal of pectoral muscle in wavelet domain. *Biomedical Research*, 28(11), 5041-5043.
 31. Das, D. K., & Dutta, P. K. (2019). Efficient automated detection of mitotic cells from breast histological images using deep convolution neural network with wavelet

- decomposed patches. *Computers in biology and medicine*, 104, 29-42.
32. Bascoy, P. G., Quesada-Barriuso, P., Heras, D. B., & Argüello, F. (2019). Wavelet-Based Multicomponent Denoising Profile for the Classification of Hyperspectral Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.
 33. Li Vigni, M., Prats-Montalban, J. M., Ferrer, A., & Cocchi, M. (2018). Coupling 2D-wavelet decomposition and multivariate image analysis (2D WT-MIA). *Journal of Chemometrics*, 32(1), e2970.
 34. Ahuja, S., & Mehan, A. (2018, April). Design of Orthogonal Wavelet for Human Palmprint Recognition. In *2018 International Conference on Intelligent Circuits and Systems (ICICS)* (pp. 265-270). IEEE.
 35. Sivakumar, R., & Mohan, E. (2018). High Resolution Satellite Image Enhancement Using Discrete Wavelet Transform. *International Journal of Applied Engineering Research*, 13(11), 9811-9815.
 36. Chaddad, A., Daniel, P., & Niazi, T. (2018). Radiomics evaluation of histological heterogeneity using multiscale textures

derived from 3D wavelet transformation of multispectral images. *Frontiers in oncology*, 8, 96.

37. Huang, Y., De Bortoli, V., Zhou, F., & Gilles, J. (2018). Review of wavelet-based unsupervised texture segmentation, advantage of adaptive wavelets. *IET Image Processing*, 12(9), 1626-1638.
38. Phinyomark, A., Nuidod, A., Phukpattaranont, P., & Limsakul, C. (2012). Feature extraction and reduction of wavelet transform coefficients for EMG pattern classification. *Elektronika ir Elektrotechnika*, 122(6), 27-32.
39. de A Berger, P., Francisco, A. D. O., do Carmo, J. C., & da Rocha, A. F. (2006). Compression of EMG signals with wavelet transform and artificial neural networks. *Physiological measurement*, 27(6), 457.
40. Subasi, A., Yaman, E., Somaily, Y., Alynabawi, H. A., Alobaidi, F., & Altheibani, S. (2018). Automated EMG Signal Classification for Diagnosis of Neuromuscular Disorders

Using DWT and Bagging. *Procedia Computer Science*, 140, 230-237.

41. Ryan, Ø. (2019). *Linear Algebra, Signal Processing and Wavelets – a unified Approach. Python Version*. Oslo, Norway: Springer.
42. Hariharan. (2019). *Wavelet Solutions for Reaction–Diffusion Problems in Science and Engineering*. Tamil Nadu, India: Springer International Publishing.
43. Abood, S. (2020). *Digital Signal Processing: A Primer with MATLAB*. CRC Press.
44. Panja, M. M., & Mandal, B. N. (2020). *Wavelet Based Approximation Schemes for Singular Integral Equations*. Kolkata, India: CRC Press.
45. Krantz, S. (2020). *Differential Equations: A Modern Approach with Wavelets*. New York: Chapman and Hall/CRC.

CAPÍTULO IV

DESARROLLO EN PYTHON

Víctor Alberto Lizcano,
Ingeniería Electrónica
Universidad de Pamplona.

Ronald de Jesús Torres,
Ingeniería Electrónica,
Universidad de Pamplona.

Luis Enrique Mendoza,
Ingeniería en Telecomunicaciones.
Universidad de Pamplona.

El llevar a la práctica los conocimientos adquiridos de forma teórica permite la apropiación de los mismos de una forma más profunda, ayudando al estudiante a formular y afrontar nuevos retos,

generando así nuevo conocimiento; por ello, el presente capítulo propone al estudiante realizar algunos problemas propuestos, así como guiar el proceso de instalación y manejo del lenguaje de programación Python enfocado al procesamiento de señales en 1D y 2D aprendidos en capítulos anteriores, también se podrán encontrar ejercicios prácticos sobre algunos problemas específicos que se pueden presentar en el momento de acondicionar y procesar señales en 1D y 2D.

4.1 ¿Qué es Python?

Es un lenguaje de programación de alto nivel interpretado, multiparadigma, lo que permite al desarrollador diseñar el software utilizando el paradigma que mejor se adapte al problema que se desea solucionar, soporta la programación orientado a objetos, es multiplataforma por lo que se puede utilizar en diferentes sistemas operativos como Windows, Mac y Linux. Ya viene instalado de forma predeterminada en los sistemas operativos Mac y

Linux, para Windows. Se muestra a continuación como instalarlo. [1]

4.2 ¿Qué es PyCharm?

Es un entorno de desarrollo integrado (IDE) diseñado exclusivamente para Python, permite crear códigos con extensión .py, es multiplataforma disponible para los sistemas operativos Windows, Mac y Linux, para este curso se trabajará con la versión community puesto que es libre. [2].

4.3 Instalación de Python

El enlace que se muestra a continuación contiene los archivos necesarios para la instalación de Python 3.7.3, así como su entorno de desarrollo integrado (IDE) PyCharm Community 2019.1.1. [3]

https://drive.google.com/file/d/14SteFmCC_tNA39NY6SJoEt3DdIKMTrUy/view?usp=sharing

Luego debe dirigirse a la ubicación del archivo descargado con extensión .rar, y con ayuda del programa *Winrar* u otro con función similar se

procede a descomprimir, posteriormente haga clic sobre la carpeta generada para acceder a las carpetas llamadas Python y PyCharm.

Dentro de la carpeta llamada Python se encuentra el archivo ejecutable ***python-3.7.3.exe***, y sobre este haga clic derecho y ejecutar como administrador (Figura 4.1), luego acepte los permisos.

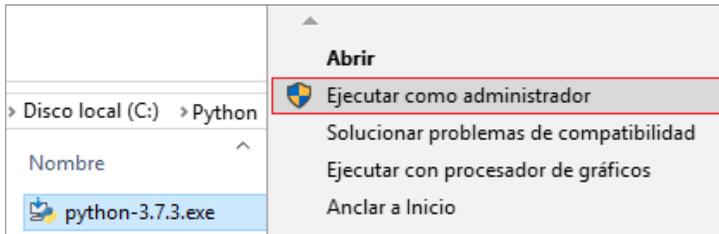


Figura 4.1.

Ejecución de Python-3.7.3.exe como administrador.

Seleccione las opciones ***Install launcher for all users (recommended)*** y ***Add Python 3.7 to PATH***, esta última opción se debe seleccionar para evitar a futuro agregar la ruta de instalación de Python a la variable de entorno (*path*) de forma manual. Posteriormente debe hacer clic en la opción ***Install now***, como se muestra en la

Figura 4.2. Luego espere mientras termina la Instalación (Figura 4.3).

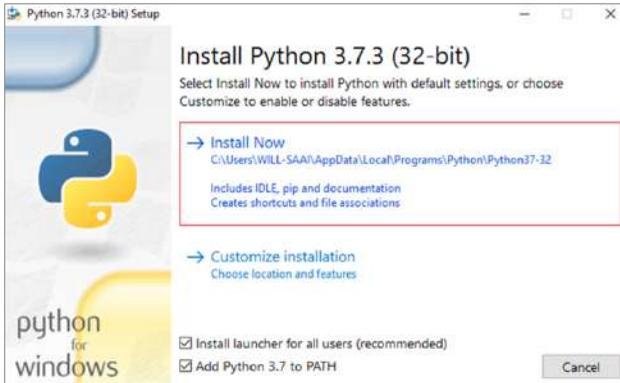


Figura 4.2.
Ventana de instalación.

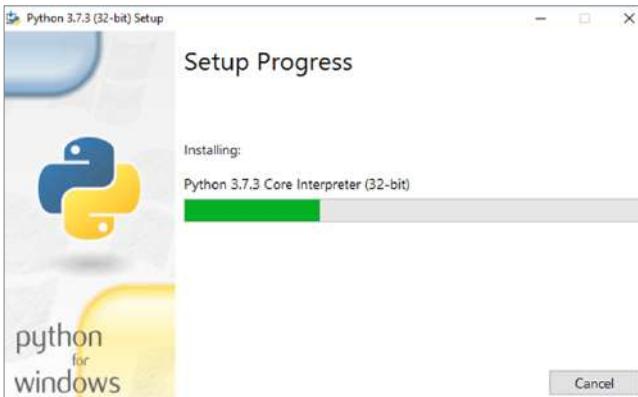


Figura 4.3.
Progreso del proceso de instalación.

Cierre la ventana del asistente de instalación de Python haciendo clic en el botón **Close**, como muestra la Figura 4.4.



Figura 4.4.
Cierre de la ventana del asistente de instalación.

Verifique la correcta instalación de Python realizando los siguientes pasos:

Ejecute la aplicación **Símbolo del sistema**. Puede encontrarla escribiendo en el buscador de Windows su nombre, como se muestra en la

Figura 4.5, o presionando la combinación de teclas Win + R. La figura Y muestra el entorno gráfico de la aplicación ***Símbolo del sistema***.

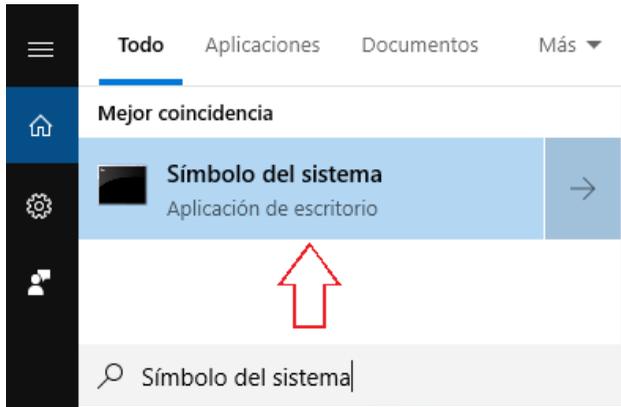


Figura 4.5.
Búsqueda de la aplicación (Símbolo del sistema).

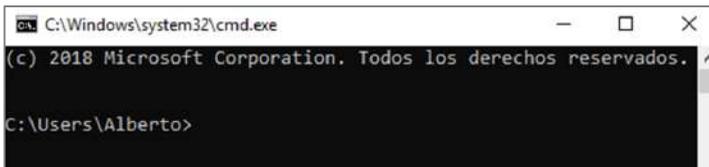
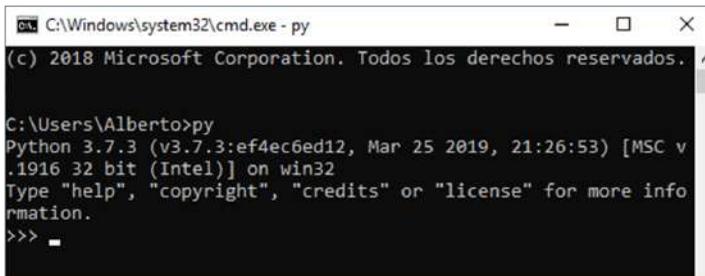


Figura 4.6.
Aplicación Símbolo del sistema.

Para validar la correcta instalación se debe ejecutar el comando *py* o *python* y presionar la tecla *Enter*.

De manera emergente deben mostrarse algunos parámetros de Python, como su versión, fecha de instalación, entre otros, como lo muestra la Figura 4.7.



```
C:\Windows\system32\cmd.exe - py
(c) 2018 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Alberto>py
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Figura 4.7.
Verificación de correcta instalación de Python.

4.4 Instalación y configuraciones iniciales de PyCharm

Concluida la correcta instalación de Python, se procede a instalar su IDE PyCharm. Nuevamente dirijase a la ubicación de la carpeta descomprimida luego de ser descargada [4]. Dentro de la carpeta llamada PyCharm se encuentra el archivo ejecutable

pycharm-community-2019.1.1.exe, y sobre este haga clic derecho y ejecutar como administrador (Figura 4.8), luego acepte los permisos.

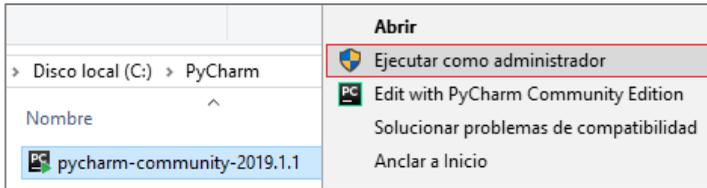


Figura 4.8.

Ejecución de PyCharm Community como administrador.

La

Figura 4.9 muestra la ventana del entorno de instalación de PyCharm, en ella haga clic en el botón **Next** para continuar con el proceso.

La

Figura 4.10 muestra una ventana donde encontrará la ruta de instalación del IDE PyCharm, si desea instalarlo en una carpeta diferente a la ruta preestablecida por defecto, seleccione la ruta haciendo clic en el botón **Browse**. Luego haga clic en el botón **Next** para continuar.

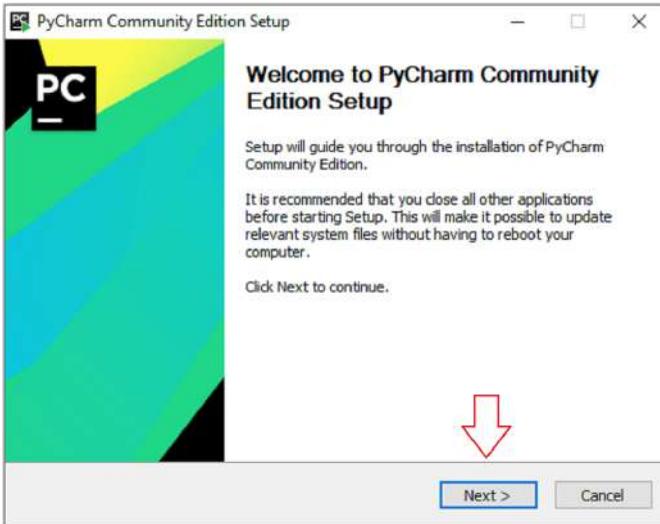


Figura 4.9.
Ventana de inicio de instalación.

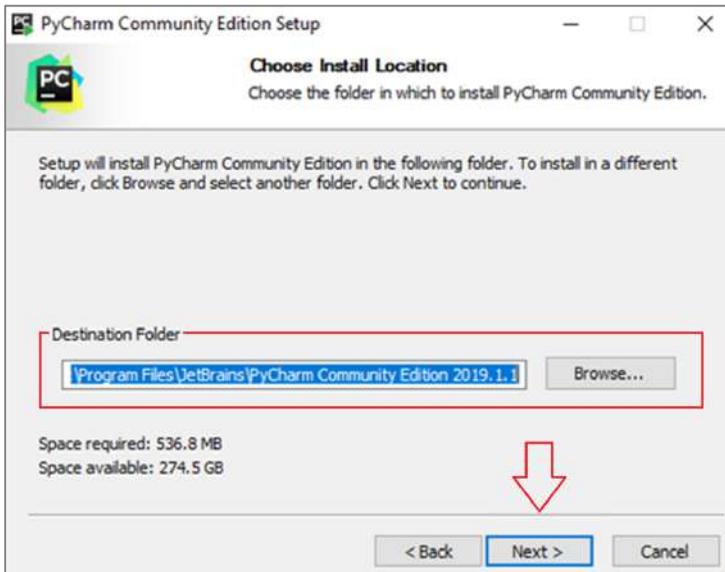


Figura 4.10.
Selección de la ruta de instalación.

Se debe seleccionar las opciones de instalación: **64-bit launcher** para crear un acceso directo en el escritorio, **Add “Open Folder as Project”** para crear una carpeta para los proyectos dentro de la ruta ya definida en el paso anterior, **.py** para asociar los archivos .py con PyCharm, y **Add launchers dir to the PATH** para agregar la ruta a la variable de

entorno PATH la cual necesitaremos para poder direccionar todos los archivos de Python (Figura 4.11).

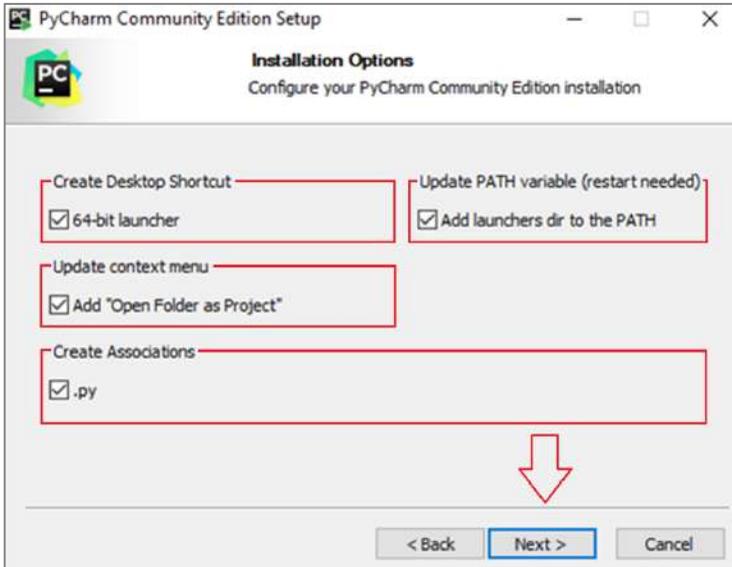


Figura 4.11.
Opciones de Instalación.

Una vez se han seleccionado las opciones de instalación, haga clic en el botón **Install** para comenzar, como muestra la Figura 4.12. Espere mientras termina el proceso (Figura 4.13).

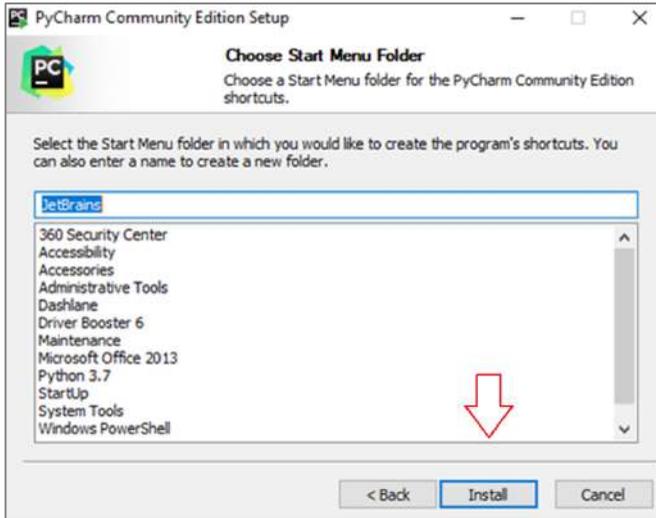


Figura 4.12.
Terminación de configuraciones e inicio de instalación.

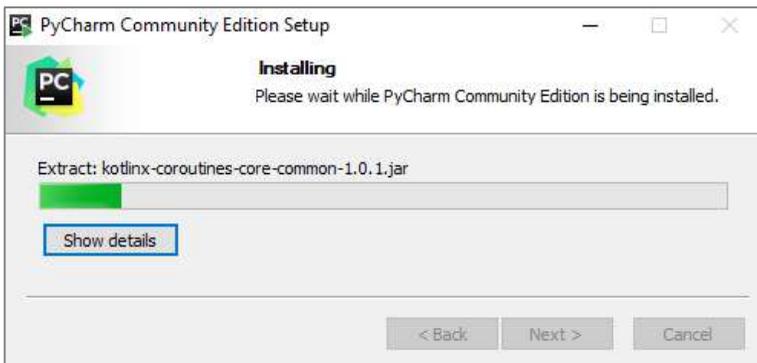


Figura 4.13.

Progreso de instalación.

Una vez terminada la instalación, seleccione la opción ***I want to manually reboot later*** para reiniciar la computadora manualmente en otro momento, o si lo desea, puede hacerlo de manera automática en el mismo instante con la opción ***Reboot Now***. Finalmente haga clic en el botón ***Finish*** (

Figura 4.14). Se requiere reiniciar la computadora para actualizar las librerías .DLL ((Dynamic Link Library).

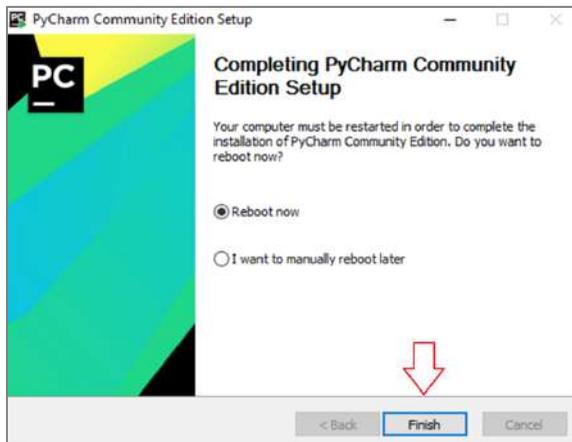


Figura 4.14.
Selección de reinicio de la computadora.

Configuraciones Iniciales

Terminada la instalación, debe aparecer en el escritorio el icono de PyCharm como muestra la Figura 4.15, abra la aplicación haciendo doble clic sobre su icono para iniciar con las configuraciones iniciales.



Figura 4.15.
Icono de PyCharm.

Iniciada la aplicación se mostrará en primer instante la ventana que se muestra en la Figura 4.16; mantenga la opción **Do not import settings**, la cual viene por defecto (no importar configuraciones), y haga clic en el botón **OK**.

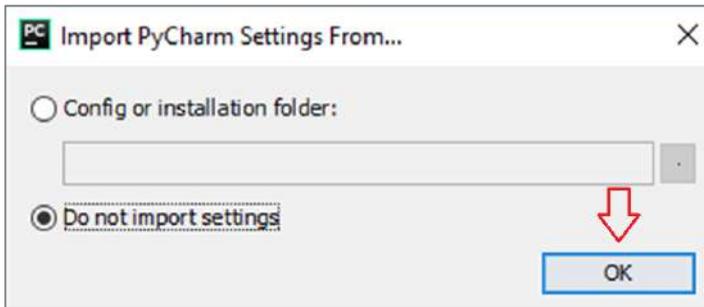


Figura 4.16.
Opciones de importar configuraciones.

Elija el tema de interfaz de usuario de PyCharm a su preferencia y haga clic en el botón **Next Featured plugins** para continuar con la configuración de las siguientes características como muestra la Figura 4.17.

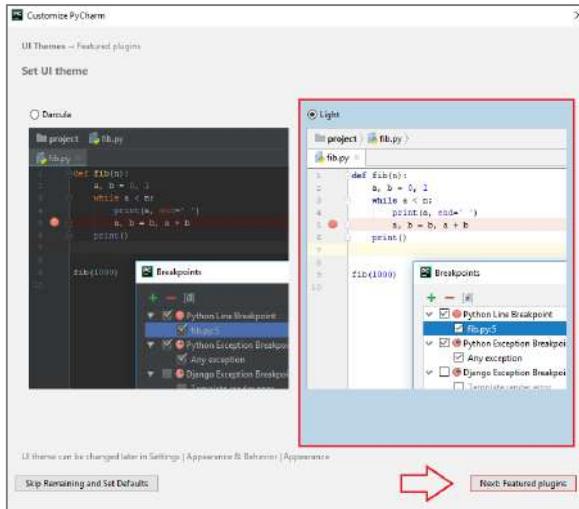


Figura 4.17.
Selección de temas de interfaz
de usuario.

Posteriormente haga clic en **Start using PyCharm** para comenzar a utilizar PyCharm como muestra la Figura 4.18.

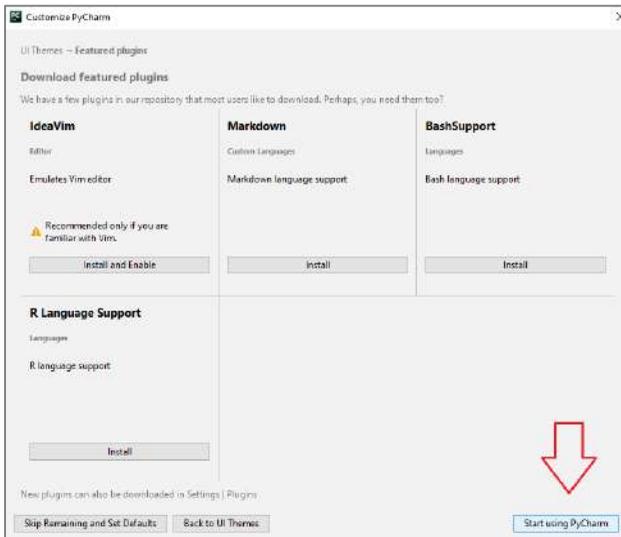


Figura 4.18.
Ventana de instalación
de complementos e inicio
de uso de PyCharm.

Realizando esto, PyCharm estará listo para ser usado. Debe aparecer una pantalla como la que se muestra a continuación (Figura 4.19).



Figura 4.19.
Ventana para abrir o crear un nuevo proyecto.

4.5 Crear un nuevo proyecto en Python

Para crear un nuevo proyecto haga clic en el botón

Create New Project como muestra la

Figura 4.19; esta acción lo dirige a una nueva

ventana, en la cual debe seleccionar la ubicación de

la carpeta que almacena los archivos que contiene

el código de lenguaje de programación Python, para

ello, haga clic en el botón  y luego en el botón

Create, como muestra la

Figura 4.20. Para este caso la carpeta lleva por nombre **Python**, que se encuentra dentro de la ruta **D:\Documents**

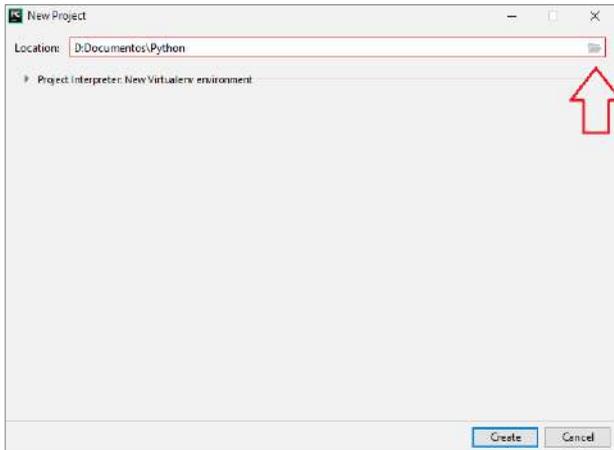


Figura 4.20.

Selección de la carpeta contenedora de los archivos del nuevo proyecto.

La acción anterior lo dirige al entorno de trabajo de PyCharm, como muestra la

Figura 4.21. En esta ventana se pueden identificar la barra de herramientas, algunos botones de acceso rápido, la carpeta de trabajo donde se guardan los

archivos generados, algunos atajos de teclado, entre otros.

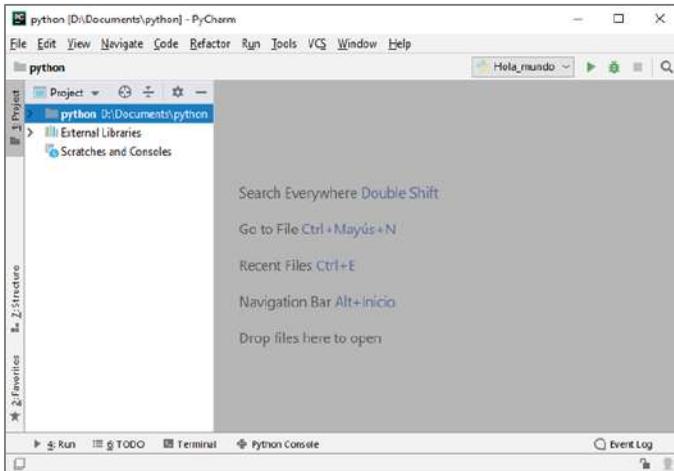


Figura 4.21.
Entorno de PyCharm.

Haga clic derecho sobre la carpeta creada en los pasos anteriores, despliegue la lista de opciones que contiene la opción **New**, y entre ellas elija **Python File** para crear un nuevo archivo con extensión **.py** (Archivo que contiene un código escrito en el lenguaje de programación Python) (

Figura 4.22), posteriormente, en el cuadro de diálogo (New Project file) (Figura 4.23) debe introducir el nombre de su proyecto (para este caso **Hola_mundo**), y presionar el botón **Ok**.

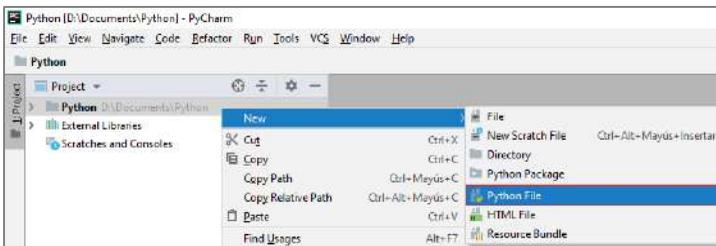


Figura 4.22.
Creación de archivo con extensión .py.

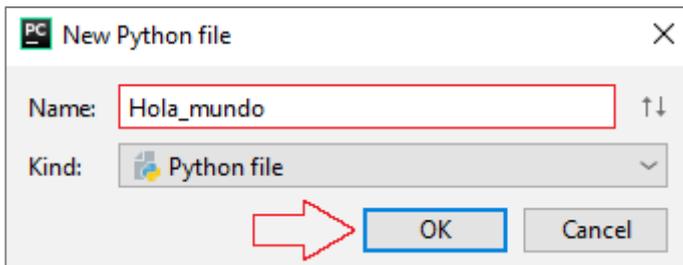


Figura 4.23.
Introducción del nombre del archivo con extensión .py.

La

Figura 4.24 muestra el entorno para iniciar a escribir código en el lenguaje de programación python.

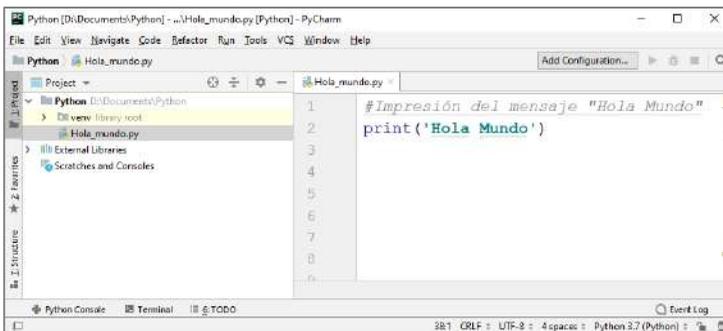


Figura 4.24.
Entorno de PyCharm (Con escritura de código).

Para ejecutar el código creado, haga clic derecho dentro del entorno de PyCharm y luego en la opción **Run 'Hola_mundo'** (Figura 4.25). Este mismo proceso puede realizarse haciendo clic en el botón **Run** ► asegurando de que el nombre de su proyecto se encuentre al lado izquierdo del botón, de no ser así, elija el nombre de su proyecto desplegando la lista de opciones (

Figura 4.26). El resultado de la ejecución se muestra en la consola (

Figura 4.27), así como el total de errores y/o advertencias con su respectiva descripción.

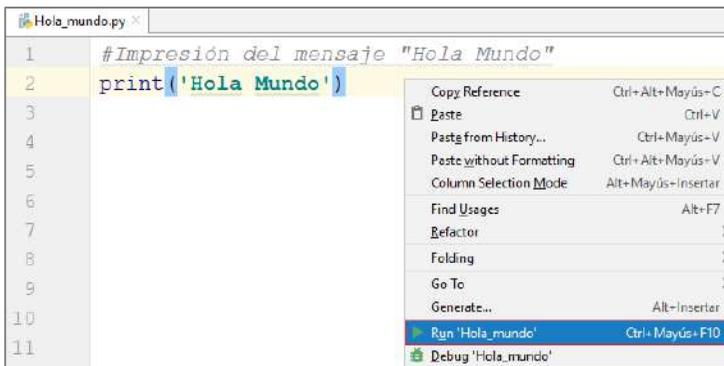


Figura 4.25.
Opción 1 para ejecutar código.

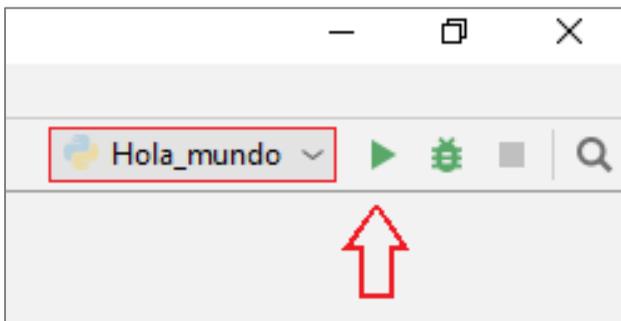


Figura 4.26.

Opción 2 para ejecutar código.



Figura 4.27.
Entorno de PyCharm con panel de resultados.

4.6 Algunas librerías importantes y su instalación

Para el procesamiento digital de señales en 1D y 2D en python, es necesario instalar algunas librerías que contienen funciones específicas para realizar determinada tarea. A continuación, se describe cada una de ellas a ser utilizadas en este capítulo.

4.6.1 Numpy

Es una librería o paquete de python encargado de trabajar con vectores de N dimensiones, está

constituido por una gran variedad de funciones matemáticas que involucran vectores y matrices, su gran versatilidad permite al usuario realizar operaciones complejas, por ejemplo, la transformada rápida de fourier en 1D, con tan solo llamar una extensión específica de numpy en python. [5]

4.6.2 Matplotlib

Es un paquete diseñado para la elaboración de gráficas en el lenguaje de programación python, permitiendo al desarrollador visualizar el comportamiento de las señales sobre las cuales se encuentra trabajando. [6]

4.6.3 OpenCV

Es un paquete utilizado comúnmente para el procesamiento de imágenes, permite al desarrollador crear una infinidad de aplicaciones enfocadas a su procesamiento y visión artificial, cuenta con funciones ya diseñadas para normalizar,

cambiar a escala de grises, cambiar los espacios de color entre otros muchas aplicaciones. [7, 8]

A continuación, se muestra el proceso para agregar paquetes a python a través de PyCharm. Como primer paso diríjase a la barra de herramientas y en ella haga clic sobre la opción **archivo** y luego en **configuraciones** (Figura 4.28).

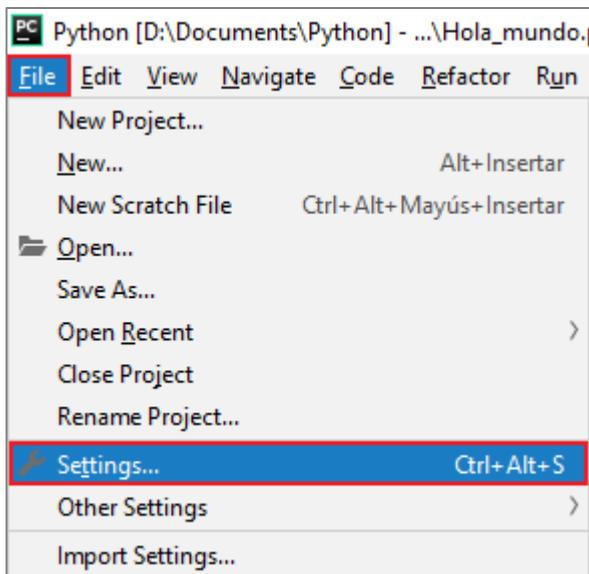


Figura 4.28.

Opciones de la pestaña File (Archivo).
 En las opciones de configuración, haga clic en **Project: Python** y luego seleccione **Project Interpreter** como ilustra la Figura 4.29.

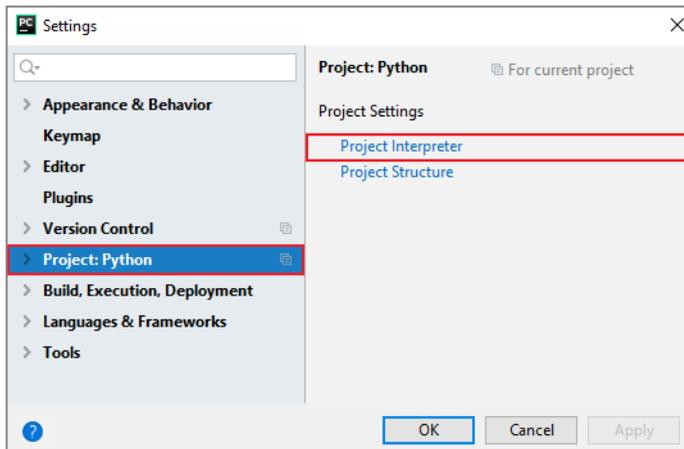


Figura 4.29.
 Configuraciones (Project Python).

Debe aparecer el intérprete del proyecto con la lista de paquetes instalados hasta ese momento (Figura 4.30), haga clic en el botón  para agregar nuevos paquetes a Python. Esta acción debe mostrar el gestor de paquetes de Python (Figura 4.31) en el que

se encuentra la barra de búsqueda rápida y en ella escriba el nombre del paquete a instalar. Seleccione la librería y a su lado derecho se muestra la descripción de la misma, (su función, versión, autor, etc.). En el botón **Install Package**, hacer click para iniciar la instalación. Una vez terminado el proceso, el paquete quedará listo para ser utilizado.

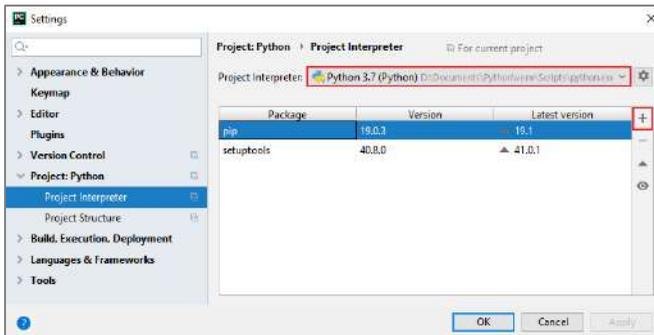


Figura 4.30.
Configuración (Project Interpreter).

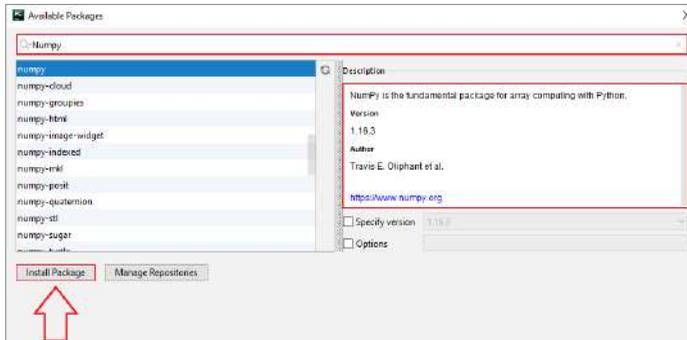


Figura 4.31.
Instalar paquetes a Python.

4.7 Ejemplos usando Python

El capítulo 1 habla del análisis de Fourier en 1D y 2D como herramienta de gran ayuda al momento de realizar extracción de patrones, filtrado de señales entre otras aplicaciones.

Para aplicar la transformada de Fourier en 1D en Python, es necesario tener instaladas las librerías Numpy y Matplotlib. [9, 10]

4.7.1 Filtrado de señales en 1D

El filtrado de señales digitales es una tarea que encontramos a diario en un sin número de

aplicaciones, como, por ejemplo, el determinar la frecuencia del ritmo cardíaco del feto eliminando la frecuencia del ritmo cardíaco de la madre (sin tener presente otros factores fisiológicos). [11]

La

Figura 4.32 (a) muestra una señal senoidal ($f_1(t)$) de 20 Hz ($f(t) = A \text{sen}(2\pi ft)$) muestreada a una frecuencia de 2 KHz.

$$f_1(t) = \text{sen}(40\pi t)$$

Sea ($f(t)$) la suma de $f_1(t)$ con una señal de 400 Hz ($f_2(t)$) la cual llamaremos ruido, la

Figura 4.32 (b) muestra la representación gráfica de la señal con ruido ($f(t)$).

$$f_2(t) = \text{sen}(800\pi t)$$

$$f(t) = f_1(t) + f_2(t)$$

Para identificar las frecuencias presentes en una señal ($f(t)$) se hace uso de la transformada de Fourier como se indicó en el capítulo 1, nótese que la señal de la

Figura 4.32 (c) muestra el valor absoluto de la transformada de Fourier de $f(t)$, para visualizar las componentes frecuenciales de la señal ruido, la cual se entiende que está compuesta por valores altos de frecuencia.

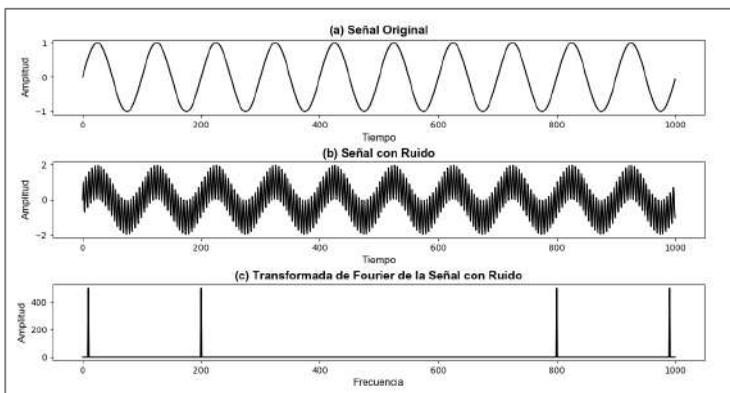


Figura 4.32.

Representación gráfica: (a) Señal senoidal $f_1(t)$, (b) Señal con ruido $f(t)$, (c) Transformada de Fourier de la señal $f(t)$.

Para filtrar la señal ($f(t)$) (

Figura 4.33 (a)) haga uso de los pasos mencionados en el capítulo 1. Para este ejemplo, los puntos que corresponden a la frecuencia a filtrar (400 Hz) se

encuentran en el punto 200 y su correspondiente valor simétrico 800 (Figura 4.33 (b)). Cabe mencionar que se puede hacer filtrado en una frecuencia específica o en un rango de frecuencias. Es importante mencionar que el proceso de filtrado se realiza en la transformada de Fourier y no en su correspondiente valor absoluto (utilizado únicamente para efectos visuales), ya que para analizar los efectos del filtrado sobre la señal original es necesario regresar al dominio del tiempo aplicando la transformada inversa de Fourier. Eliminados los puntos correspondientes mediante la puesta a cero de los mismos, se aplica la transformada inversa de Fourier para obtener la señal filtrada en el dominio del tiempo (Figura 4.33 (c)), y su transformada de Fourier se muestra en la Figura 4.33(d).

Las señales ruidosas están presentes en cualquier registro obtenido en un sistema de adquisición, por

ejemplo, en un registro de señales electroencefalográficas [12], se puede definir las señales oculográficas como señales de interferencias o ruido [13], otro claro ejemplo puede evidenciarse en la toma de ritmo cardíaco de un feto, donde el ritmo cardíaco de la madre podría definirse como ruido, o viceversa, dependiendo de la particularidad de la aplicación, cabe mencionar que en muchas ocasiones la frecuencia de las señales ruido son conocidas por estudios previos u otros trabajos realizados.

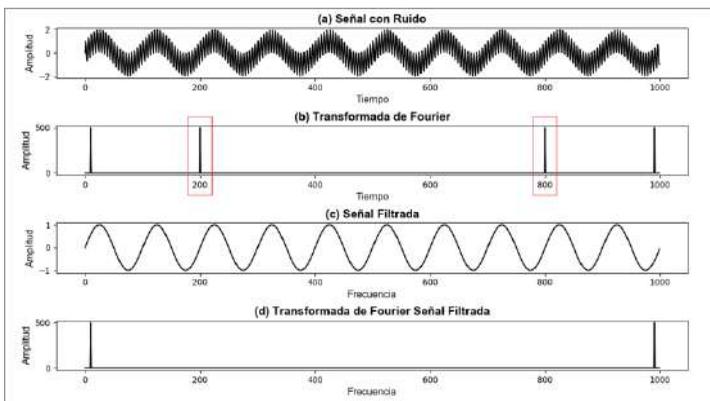


Figura 4.33.

Representación gráfica: (a) Señal con Ruido $f(t)$, (b) Transformada de la señal ruido con puntos localizados (enmarcados en color rojo) de la frecuencia a eliminar, (c) señal filtrada $f(t)$ en el dominio del tiempo, (d) Transformada de Fourier de la señal filtrada $f(t)$.

A continuación se muestra el código implementado en Python:

```
#importación de librerías
import numpy as np
import matplotlib.pyplot as plt
#
font1 = {'family': 'Arial',
         'weight': 'bold',
         'size': 14,
        }
font = {'family': 'Arial',
        'weight': 'normal',
        'size': 12,
       }

Pi = np.pi
Tiempo = np.arange(0,0.5,(1/2000))
Frecuencia_1 = 20
Frecuencia_2 = 400
Señal_Original = np.sin(2*Pi*Frecuencia_1*Tiempo)
Ruido = np.sin(2*Pi*Frecuencia_2*Tiempo)
Señal_con_Ruido = Señal_Original+Ruido
Fourier = np.fft.fft(Señal_con_Ruido)
Valor_Absoluto=np.abs(Fourier)
```

```
Filtro = Fourier
Filtro[150:250]=0
Filtro [750:850]=0
Señal_Filtrada=np.fft.ifft(Filtro)
TRansformada_de_la_Señal_filtrada=abs(np.fft.fft(S
eñal_Filtrada))
```

```
plt.figure(1)
```

```
plt.subplots_adjust( hspace=0.62 )
plt.subplot(311)
plt.plot(Señal_Original,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(a) Señal Original',fontdict=font1)
```

```
plt.subplot(312)
plt.plot(Señal_con_Ruido,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(b) Señal con Ruido',fontdict=font1)
```

```
plt.subplot(313)
plt.plot(Valor_Absoluto,'k')
plt.xlabel('Frecuencia',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(c) Transformada de Fourier de la Señal
con Ruido',fontdict=font1)
```

```
plt.figure(2)
```

```
plt.subplots_adjust(
hspace=0.95,bottom=0.1,top=0.95 )
plt.subplot(411)
plt.plot(Señal_con_Ruido,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(a) Señal con Ruido',fontdict=font1)
```

```
plt.subplot(412)
plt.plot(Valor_Absoluto,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(b) Transformada de
Fourier',fontdict=font1)
```

```
plt.subplot(413)
plt.plot(Señal_Filtrada,'k')
plt.xlabel('Frecuencia',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(c) Señal Filtrada',fontdict=font1)
```

```
plt.subplot(414)
plt.plot(TRansformada_de_la_Señal_filtrada,'k')
plt.xlabel('Frecuencia',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(d) Transformada de Fourier Señal
Filtrada',fontdict=font1)
```

```
plt.show()
```

4.7.2 Extracción de patrones en señales 1D

La extracción de patrones es una tarea que suele realizarse principalmente en tareas de clasificación, siendo a la vez una de las más difíciles, ya que si esta no se realiza de la manera apropiada repercute en las otras.

En este ejemplo se demuestra que la transformada de Fourier tiene potencial en extracción de patrones. Para ello se busca encontrar diferencias significativas en el dominio de la frecuencia entre los sonidos vocálicos de la letra A y la letra E (Figura 4.34 (a) y Figura 4.34 (b) respectivamente) los cuales fueron adquiridas durante un lapso de tiempo de 4 segundos a una frecuencia de muestreo de 6.8 KHz (Teorema de Nyquist) [14] representados por un vector de 27200 puntos para cada uno. Nótese que las señales en el dominio del tiempo no presentan un patrón que las haga distintivas mientras que las señales en el dominio de la frecuencia permiten ver

con mayor claridad las diferencias que existen entre estos dos sonidos como muestran la Figura 4.34 (c) y la Figura 4.34(d).

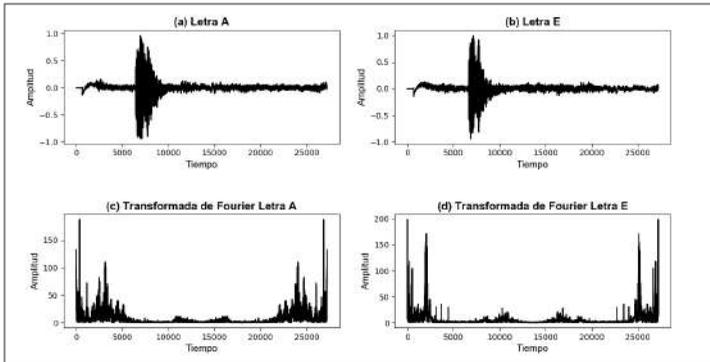


Figura 4.34.

Representación gráfica: (a) Señal sonido vocálico letra A, (b) Señal sonido vocálico letra E, (c) Valor absoluto transformada de Fourier del sonido vocálico letra A, (d) Valor absoluto transformada de Fourier del sonido vocálico letra E.

En el capítulo 2 se enseña todo lo relacionado con la transformada discreta de coseno demostrando que permite realizar tareas de extracción de patrones, compresión de información, entre otros. Para su implementación en Python es necesario tener instaladas las librerías Scipy y Matplotlib.

A continuación se muestra el código implementado en Python:

```
import numpy as np
import matplotlib.pyplot as plt

#
font1 = {'family': 'Arial',
        'weight': 'bold',
        'size': 14,
        }
font = {'family': 'Arial',
       'weight': 'normal',
       'size': 12,
       }

A = np.loadtxt('Letra_A.txt')
E = np.loadtxt('Letra_E.txt')
TDF_A = np.fft.fft(A)
Valor_Absoluto_A = np.abs(TDF_A)
TDF_E = np.fft.fft(E)
Valor_Absoluto_E = np.abs(TDF_E)
print (len(Valor_Absoluto_E))
plt.figure(1)

plt.subplots_adjust( hspace=0.62 )
plt.subplot(221)
plt.plot(A, 'k')
plt.xlabel('Tiempo', fontdict=font)
plt.ylabel('Amplitud', fontdict=font)
plt.title('(a) Letra A', fontdict=font1)
```

```
plt.subplot(222)
plt.plot(E, 'k')
plt.xlabel('Tiempo', fontdict=font)
plt.ylabel('Amplitud', fontdict=font)
plt.title('(b) Letra E', fontdict=font1)
```

```
plt.subplot(223)
plt.plot(Valor_Absoluto_A, 'k')
plt.xlabel('Tiempo', fontdict=font)
plt.ylabel('Amplitud', fontdict=font)
plt.title('(c) Transformada de Fourier Letra
A', fontdict=font1)
```

```
plt.subplot(224)
plt.plot(Valor_Absoluto_E, 'k')
plt.xlabel('Tiempo', fontdict=font)
plt.ylabel('Amplitud', fontdict=font)
plt.title('(d) Transformada de Fourier Letra
E', fontdict=font1)
```

```
plt.show()
```

4.7.3 Extracción de patrones usando la transformada discreta del coseno TDC.

Al igual que la transformada de Fourier, la transformada discreta del coseno permite extraer patrones en el dominio de la frecuencia. La

Figura 4.35 (a) y la Figura 4.35 (b) muestran la representación gráfica del sonido vocálico de la letra A y E respectivamente. Nótese que para este ejemplo la TDC muestra una diferencia significativa en el dominio frecuencial con respecto al dominio del tiempo, representando un patrón característico para los sonidos vocálicos A y E. El patrón puede evidenciarse visualmente entre los puntos 3000 y 12000 (Figura 4.35 (c) y Figura 4.35 (d)), donde la diferencia radica en la presencia de mayor componentes frecuenciales para la vocal A, y un lóbulo característico para la vocal E.

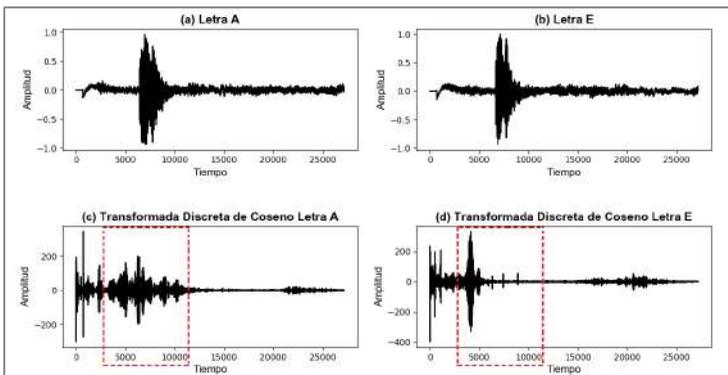


Figura 4.35.

Representación gráfica: (a) Señal sonido vocálico letra A, (b) Señal sonido vocálico letra E, (c) Transformada discreta del coseno del sonido vocálico letra A, (d) Transformada discreta del coseno sonido vocálico letra E.

A continuación se muestra el código implementado en Python:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import dct,idct
#
font1 = {'family': 'Arial',
         'weight': 'bold',
         'size': 14,
        }
font = {'family': 'Arial',
        'weight': 'normal',
        'size': 12,
        }

A = np.loadtxt('Letra_A.txt')
E = np.loadtxt('Letra_E.txt')
TDC_A = dct(A)
TDC_E = dct(E)

print (len(TDC_E))
plt.figure(1)

plt.subplots_adjust( hspace=0.62 )
plt.subplot(221)
plt.plot(A,'k')
```

```
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(a) Letra A',fontdict=font1)
```

```
plt.subplot(222)
plt.plot(E, 'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(b) Letra E',fontdict=font1)
```

```
plt.subplot(223)
plt.plot(TDC_A, 'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(c) Transformada Discreta de Coseno
Letra A',fontdict=font1)
```

```
plt.subplot(224)
plt.plot(TDC_E, 'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(d) Transformada Discreta de Coseno
Letra E',fontdict=font1)
```

```
plt.show()
```

4.7.4 Compresión de información

La TDC permite realizar compresión de información en el dominio de la frecuencia, concentrando la

mayor cantidad información en una pequeña zona, logrando así procesar y reconstruir señales utilizando el menor número de puntos posibles. Para este ejemplo se utiliza el sonido vocálico de la letra A (

Figura 4.36 (a)), aplicando TDC se obtiene la misma señal pero en el dominio de la frecuencia (

Figura 4.36 (b)), nótese como la mayor cantidad de información está representada entre los primero 10.000 puntos por tanto se realiza una extracción de la zona de interés (

Figura 4.36 (c)), aplicando la TDC inversa sobre la zona de interés se puede llegar a la señal original pero representada con menos puntos (

Figura 4.36 (d)).

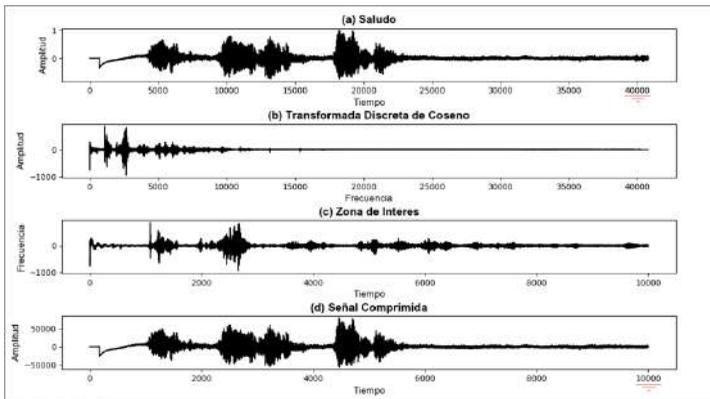


Figura 4.36.

Representación gráfica: (a) Señal sonido vocálico letra A, (b) Transformada discreta del coseno letra A, (c) Zona de interés, (d) Reconstrucción del sonido vocálico A, a partir de la zona de interés.

A continuación se muestra el código implementado en Python:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import dct,idct
#
font1 = {'family': 'Arial',
         'weight': 'bold',
         'size': 14,
        }
font = {'family': 'Arial',
        'weight': 'normal',
        'size': 12,
```

```

    }
    Saludo = np.loadtxt('Saludo.txt')
    TDC_Saludo = dct(Saludo)

    Zona_de_Interes=TDC_Saludo[0:10000]
    Señal_Compresida=idct(Zona_de_Interes)

    plt.figure(1)

    plt.subplots_adjust( hspace=0.80,
                        bottom=0.08,top=0.95 )
    plt.subplot(411)
    plt.plot(Saludo,'k')
    plt.xlabel('Tiempo',fontdict=font)
    plt.ylabel('Amplitud',fontdict=font)
    plt.title('(a) Saludo',fontdict=font1)

    plt.subplot(412)
    plt.plot(TDC_Saludo,'k')
    plt.xlabel('Frecuencia',fontdict=font)
    plt.ylabel('Amplitud',fontdict=font)
    plt.title('(b) Transformada Discreta de
    Coseno',fontdict=font1)

    plt.subplot(413)
    plt.plot(Zona_de_Interes,'k')
    plt.xlabel('Tiempo',fontdict=font)
    plt.ylabel('Frecuencia',fontdict=font)
    plt.title('(c) Zona de Interes',fontdict=font1)

    plt.subplot(414)

```

```
plt.plot(Señal_Comprimida,'k')  
plt.xlabel('Tiempo',fontdict=font)  
plt.ylabel('Amplitud',fontdict=font)  
plt.title('(d) Señal Comprimida',fontdict=font1)  
plt.show()
```

4.7.5 Análisis tiempo-frecuencia en 1D usando la transformada Wavelet Estacionaria.

La transformada wavelet a diferencia de la transformada discreta del coseno y la transformada de Fourier, permite conocer en el dominio del tiempo el instante en el que ocurre una frecuencia [15] como se mencionó en el capítulo 3. Para efectos de aprendizaje se diseñó una señal senoidal (muestreada a una frecuencia de 1000 Hz) que contiene un rizo en cierto instante de tiempo como muestra la

Figura 4.38 (a). La

Figura 4.37 representa el número de niveles de descomposición de wavelet utilizados.

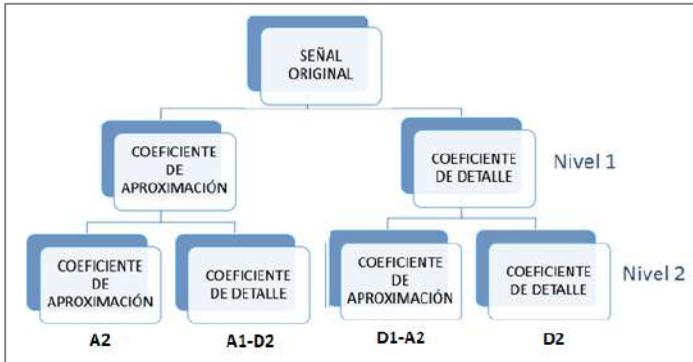


Figura 4.37.

Árbol de descomposición wavelet para dos niveles.

En el nivel dos del árbol de descomposición de la transformada wavelet, se puede decir que la señal de los coeficientes de aproximación (A2) contiene las frecuencias más baja de la señal tratada, (Señal senoidal con rizado), es decir, en el rango de 0-250 HZ, por lo tanto también contiene la señal filtrada (Figura 4.38 (b)), ya que el rizado se representa como una frecuencia alta inmersa en su gran mayoría en los coeficientes de detalle D2 , rango 750-1000 Hz (Figura 4.38 (c)).

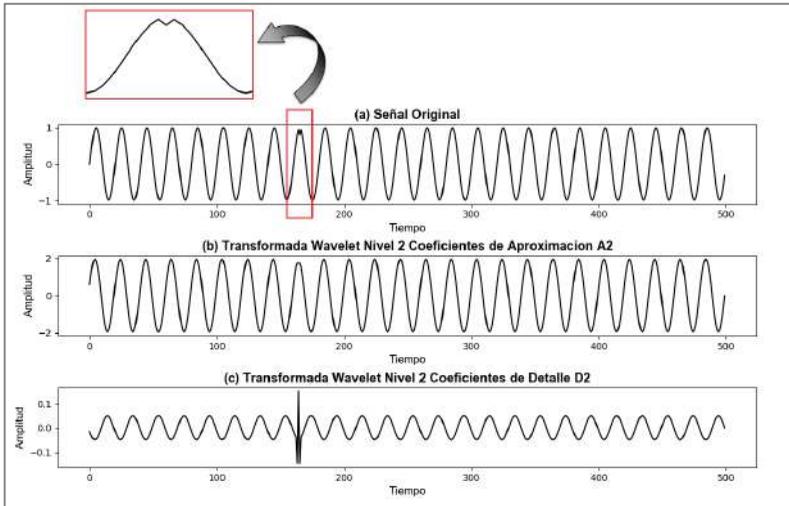


Figura 4.38.

Representación gráfica: (a) Señal senoidal con presencia de rizo, (b) Coeficientes de aproximación A1 (Nivel 2), (c) Coeficientes de detalles D2 (Nivel 2).

A continuación se muestra el código implementado en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pywt
```

```
font1 = {'family': 'Arial',
         'weight': 'bold',
         'size': 14,
         }
font = {'family': 'Arial',
```

```

'weight': 'normal',
'size': 12,
}

```

```

Pi = np.pi
Tiempo = np.arange(0,0.5,(1/1000))
Frecuencia = 50
Señal = np.sin(2*Pi*Frecuencia*Tiempo)
Señal [165]=0.8
TW_Nivel_1 = pywt.swt(Señal, 'db1', 1)
A1 = TW_Nivel_1 [0][0]
D1 = TW_Nivel_1 [0][1]

TW_Nivel_2_A1 = pywt.swt(A1, 'db1', 1)
A2 = TW_Nivel_2_A1 [0][0]
D1_A1 = TW_Nivel_2_A1 [0][1]

TW_Nivel_2_D1 = pywt.swt(D1, 'db1', 1)
A1_D1 = TW_Nivel_2_D1 [0][0]
D2 = TW_Nivel_2_D1 [0][1]
plt.figure(1)

plt.subplots_adjust( hspace=0.65)
plt.subplot(311)
plt.plot(Señal,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(a) Señal Original',fontdict=font1)

plt.subplot(312)
plt.plot(A2,'k')

```

```
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(b) Transformada Wavelet Nivel 2
Coeficientes de Aproximacion A2',fontdict=font1)
```

```
plt.subplot(313)
plt.plot(D2,'k')
plt.xlabel('Tiempo',fontdict=font)
plt.ylabel('Amplitud',fontdict=font)
plt.title('(c) Transformada Wavelet Nivel 2
Coeficientes de Detalle D2',fontdict=font1)
plt.show()
```

4.7.6 Compresión y extracción de patrones utilizando TDC en 2D.

Al igual que en 1D la TDC en 2D suele utilizarse para realizar tareas de compresión de información [16] y extracción de patrones. Este ejemplo tiene como finalidad encontrar un patrón característico para los números del 1 al 5 (

Figura 4.39 (a)) utilizando el menor número de puntos posibles, en la

Figura 4.39 (b) se muestra la TDC en 2D evidenciando que la mayor concentración de información se encuentra presente en la esquina

superior izquierda de la misma, se realizó un proceso de vectorización con el propósito de visualizar de una mejor manera las características de las señales provenientes de las TDC en 2D, usando un recorte de la zona inicial de la imagen, puesto que hay se encuentra concentrada la mayor cantidad de información, la vectorización del patrón encontrado se visualiza en la

Figura 4.40 (color rojo) en el que se puede apreciar de una mejor manera.

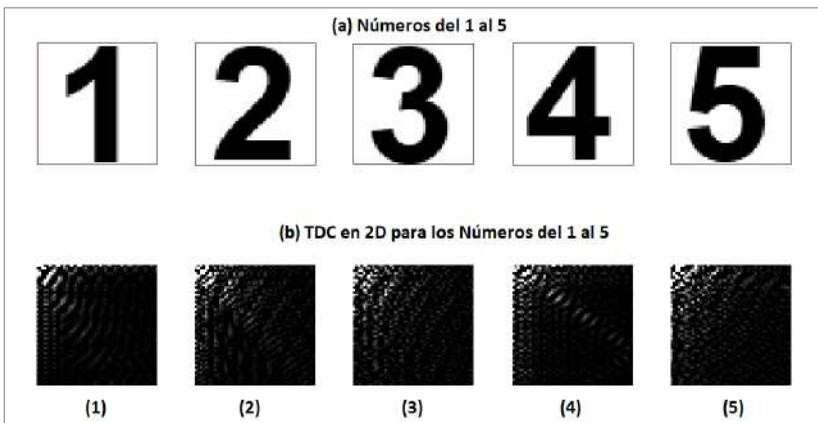


Figura 4.39.

Representación gráfica: (a) Números del 1 al 5, (b)
Transformada TDC en 2D para los números del 1 al 5

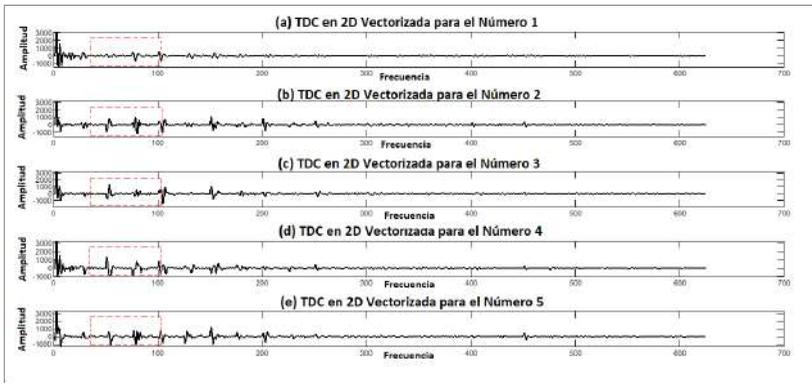


Figura 4.40.

Representación Gráfica: (a) Vector de la TDC en 2D para el Número 1, (b) Vector de la TDC en 2D para el Número 2, (c) Vector de la TDC en 2D para el Número 3, (d) Vector de la TDC en 2D para el Número 4, (e) Vector de la TDC en 2D para el Número 5.

En este capítulo se mostraron algunas aplicaciones y ejemplos básicos utilizando la transformada discreta del coseno, la transformada discreta de Fourier, y la transformada wavelet estacionaria, en futuras versiones se implementarán ejemplos más complejos con aplicaciones enfocadas a sistemas de

clasificación, segmentación de imágenes y filtrado entre otras.

A continuación se muestra el código implementado en Python:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from scipy.fftpack import dct,idct

def vectorizar(x):
    (a,b)=np.shape(x)
    v = []
    for i in range(a):
        for j in range(b):
            v.append(x[i][j])
    return (v)
def desvec(y):
    aux=0
    m = []
    for i in range(50):
        inicial = aux+1
        final = aux+50
        if aux == 0:
            xx=y[0:final]
        else:
            xx = y[inicial:final]
        m = np.array([xx])
        aux=aux+50
```

```

    return (m)
font1 = {'family': 'Arial',
        'weight': 'bold',
        'size': 14,
        }
font = {'family': 'Arial',
        'weight': 'normal',
        'size': 12,
        }
Uno =
cv2.imread('1.png',cv2.IMREAD_GRAYSCALE)
Dos =
cv2.imread('2.png',cv2.IMREAD_GRAYSCALE)
Tres =
cv2.imread('3.png',cv2.IMREAD_GRAYSCALE)
Cuatro =
cv2.imread('4.png',cv2.IMREAD_GRAYSCALE)
Cinco =
cv2.imread('5.png',cv2.IMREAD_GRAYSCALE)

```

```

TUno = dct(vectorizar(Uno))
TDos = dct(vectorizar(Dos))
TTres = dct(vectorizar(Tres))
TCuatro = dct(vectorizar(Cuatro))
TCinco = dct(vectorizar(Cinco))

```

```

aaa = vectorizar(Uno)
bbb =desvec(aaa)

```

```
print (np.shape(bbb),bbb)
print (np.shape(Uno))
plt.figure(1)

plt.subplot(3,5,1)
plt.imshow(cv2.cvtColor(np.uint8(Uno),cv2.COLOR_
GRAY2RGB))
plt.subplot(3,5,2)
plt.imshow(cv2.cvtColor(np.uint8(Dos),cv2.COLOR_
GRAY2RGB))
plt.subplot(3,5,3)
plt.imshow(cv2.cvtColor(np.uint8(Tres),cv2.COLOR
_GRAY2RGB))
plt.subplot(3,5,4)
plt.imshow(cv2.cvtColor(np.uint8(Cuatro),cv2.COLO
R_GRAY2RGB))
plt.subplot(3,5,5)
plt.imshow(cv2.cvtColor(np.uint8(Cinco),cv2.COLO
R_GRAY2RGB))
plt.subplot(3,5,6)
plt.imshow(cv2.cvtColor(np.uint8(TUno),cv2.COLO
R_GRAY2RGB))
plt.subplot(3,5,7)
plt.imshow(cv2.cvtColor(np.uint8(TDos),cv2.COLOR
_GRAY2RGB))
plt.subplot(3,5,8)
plt.imshow(cv2.cvtColor(np.uint8(TTres),cv2.COLO
R_GRAY2RGB))
plt.subplot(3,5,9)
plt.imshow(cv2.cvtColor(np.uint8(TCuatro),cv2.COL
OR_GRAY2RGB))
```

```
plt.subplot(3,5,10)
plt.imshow(cv2.cvtColor(np.uint8(TCinco),cv2.COLOR_GRAY2RGB))
plt.subplot(3,5,11)
plt.plot(aaa)
plt.subplot(3,5,12)
plt.imshow(np.uint8(bbb))
plt.subplot(3,5,13)
plt.plot(aaa)
plt.subplot(3,5,14)
plt.plot(aaa)
plt.subplot(3,5,15)
plt.plot(aaa)

key = cv2.waitKey(0)
cv2.destroyAllWindows()
plt.show()
```

Referencias

- [1] S. Bird, E. Klein y E. Loper, Natural Language Processing with Python, O'Reilly Media, 2009.
- [2] Q. Nafiul Islam, Mastering PyCharm, Packt Publishing, 2015.
- [3] «Python Software Foundation,» 2001-2019. [En línea]. Available: <https://www.python.org/downloads/>. [Último acceso: 05 Abril 2019].
- [4] JetBrains s.r.o, «JetBrains s.r.o,» 2000-2019. [En línea]. Available: <https://www.jetbrains.com/pycharm/download/#section=windows>. [Último acceso: 05 Abril 2019].
- [5] Travis E., Guide to Numpy, Oliphant , 2006.
- [6] S. Tosi, Matplotlib for Python Developers, Packt Publishing, 2009.
- [7] G. Bradski y A. Kaebler, Learning OpenCV Computer Vision with the OpenCV Library, M. Loakides, Ed., O'Reilly, 2008.
- [8] A. Mordvintsev y A. K., «OpenCV-Python Tutorial's documentation,» 2013. [En línea]. Available: <https://opencv-python-tutroals.readthedocs.io/en/latest/>. [Último acceso: 05 Abril 2019].

- [9] G. Gonzales, «Series de Fourier, Transformadas de Fourier y Aplicaciones,» Divulgaciones Matemáticas , vol. 5, nº 1, pp. 43-67, 1997.
- [1 P. Athanasios, Sistemas Digitales y Analógicos
0] Transformadas de Fourier, Estimación Espectral, Barcelona- Mexico: Marcombo Boixareu Editores, 1986.
- [1 A. Fournié y G. Boog, «Estudio del Ritmo
1] Cardíaco fetal,» El Sevier, vol. 40, pp. 1-21, 2004.
- [1 F. Alarid Escudero, Solís Escalante, E. Melgar,
2 R. Valdés Cristerna y Yañez Suarez, «Registro de señales de EEG para aplicaciones de Interfaz Cerebro Computadora (ICC) basado en Potenciales Evocados Visuales de Estado Estacionario (PEVEE),» Bioengineering Solutions for Latin American Health , vol. 18, pp. 87-90, 2007.
- [1 A. Quintero Rincon , M. Risk y S. Liberezuk,
3] «Procesamiento de EEG con Filtros Hampel,» Argencon , vol. 2012, nº 89, 2012.
- [1 Á. de la Torre Vega, Procesamiento de voz,
4] Universidad de Granada, 2007.
- [1 D. M. Ballesteros Larrotta, «Aplicación de la
5] transformada wavelet discreta en el filtrado de señales bioeléctricas,» Umbral Científico, nº 5, pp. 92-98, 2004.
- [1 J. López Hernández, C. Velasco Bautista , M.
6] Nakano Miyatake y H. Pérez Meana, «Algoritmo Esteganografico Robusto a Compresión JPEG

Usando DCT,» San Francisco Culhuacan, Mexico D.F.

- [1 Van der Walt, S., Schönberger, J. L., Nunez-7] Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., ... & Yu, T. (2014). scikit-image: image processing in Python. PeerJ, 2, e453.
- [1 Canty, M. J. (2014). Image analysis, 8 classification and change detection in remote sensing: with algorithms for ENVI/IDL and Python. Crc Press.
- [1 Van Rossum, G., & Drake, F. L. (2011). The 9 python language reference manual. Network Theory Ltd..
- [2 Lynch, S. (2018). Image Processing with Python. 0 In Dynamical Systems with Applications using Python (pp. 471-489). Birkhäuser, Cham.
- [2 Loredo, T., & Scargle, J. (2019, March). Time 1] series exploration in Python and MATLAB: Unevenly sampled data, parametric modeling, and periodograms. In AAS/High Energy Astrophysics Division (Vol. 17).
- [2 Tuck, J. (2018). Estimating the Discrete Fourier 2 Transform using Deep Learning.
- [2 Pine, D. J. (2019). Introduction to Python for 3 Science and Engineering. CRC Press.
- [2 Arias Páez, A. S., & Rubiano Venegas, D. A. 4 (2018). Método automático de reconocimiento de voz para la clasificación de vocales al lenguaje de señas colombiano.

- [2 Agustí Melchor, M. (2019). DFT vs DCT: un
5 ejemplo visual de uso mediante OpenCV.
- [2 CONGO PASTRANA, J. W. (2018).
6 APLICACIONES DEL SOFTWARE LIBRE
PYTHON PARA PRÁCTICAS DE
LABORATORIO APLICADO A LA
ASIGNATURA DE TRATAMIENTO DIGITAL DE
SEÑALES DE LA UNIVERSIDAD
TECNOLÓGICA ISRAEL (Bachelor's thesis,
Quito).
- [2 Grinberg, M. (2018). Flask web development:
7 developing web applications with python. "
O'Reilly Media, Inc.".

Procesamiento de Datos Discretos en 1D y 2D: Fourier, Coseno y Wavelet

Aplicaciones 1.0



Formando líderes para la
construcción de un nuevo
país en paz

*Facultad de Ingenierías y Arquitectura
Universidad de Pamplona
2020*