Guía de Laboratorio

Procesamiento Digital de Señales

Carlos Andrés Mesa Roldan

2023-1

1. La transformada discreta de Wavelet (TDW)

1.1 Introducción

La transformada discreta de Wavelet es una herramienta para el análisis de señales en tiempo continuo y discreto, a diferentes niveles de detalle. Esta se basa en el concepto de Wavelets, las cuales son estructuras matemáticas que tienen como finalidad tomar y analizar ciertas características de una señal a diferentes escalas. Particularmente la TDW tiene cierta afinidad por las señales de audio y video, permitiendo así expresar estas señales en términos de ellas mismas, descomponiéndolas en diferentes bandas de frecuencia y permitiendo así calcular los coeficientes asociados a cada nivel.

El proceso a grandes rasgos de descomposición tiene como punto de partida una división de la señal, en diferentes niveles de resolución frecuencial, permitiendo así observar con detalle características presentes en la señal, sea en alta o baja frecuencia, todo depende de la escala que presente mayor información con respecto a las demás y claramente de la señal a tratar, pues es claro que ciertas señales presentan un comportamiento de mayor interés para su estudio, en cierto rangos de frecuencia.

1.2 Cargue la señal de audio (tambores.wav) adjunta en la carpeta. Grafique la señal de audio normalizada en el dominio del tiempo.

1.3 Identifique la frecuencia a la que se muestrea la señal.

1.4 Escuche e identifique a que instrumento pertenece la señal presentada.

2. Análisis en el dominio de la frecuencia.

2.1 Para la señal tratada en el numeral anterior, encuentre los coeficientes de la TDW.
Utilice el siguiente bloque de código para encontrar los coeficientes de la transformada de Wavelet.

```
audio, fs = librosa.load("ruta_del_audio.wav")

wavelet = 'db4'

# Realizar la TDW de la señal de audio
coeffs = pywt.wavedec(audio, wavelet)
```

2.2 Grafique los coeficientes de la TDW, encontrados anteriormente. Puede utilizar el siguiente bloque de código, para graficar con mayor claridad cada una de las escalas frecuenciales.

```
plt.figure(figsize=(30,50))


for i in range(len(coeffs_filtered)):

    plt.subplot(len(coeffs_filtered) + 1, 1, i + 2)
    plt.plot(np.arange(len(coeffs_filtered[i])), coeffs_filtered[i]
    plt.title('Escala {}'.format(i + 1), size=16)
    plt.xlabel('Frecuecnia [hz]', fontsize=16)
    plt.ylabel('Amplitud', fontsize=16)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)
    plt.grid()


plt.tight_layout()
plt.show()
```

2.3 ¿Qué puede observar conforme se presenta una escala superior? ¿Como se comporta la señal de interés?

2.4 ¿En qué rangos de frecuencia se logra tener mayor amplitud y menor amplitud?

2.5 ¿Qué ventajas encuentra en la TDW con respecto a la transformada discreta de Fourier al momento de analizar el espectro de la señal? Realice la transformada discreta de Fourier del audio anterior, grafique sus componentes espectrales para un mejor análisis y concluya.

2.6 Basado en su experiencia, ¿son coherentes los resultados obtenidos anteriormente?

3. Energía de la señal

3.1 Utilizando el siguiente bloque código, calcule la energía de la señal analizada previamente con la TDW.

```python
def tdw_energy(coeffs):
    tdw_energy = 0
    for i in range(len(coeffs)):
        tdw_energy += np.sum(np.square(coeffs[i]))

    return tdw_energy
```

3.2 Utilizando la transformada discreta de Fourier encuentre la energía de la señal anteriormente analizada con la TDW. Utilice el siguiente bloque de código.

```python
def fourier_energy(audio):
    dft = np.fft.fft(audio)
    energy = np.sum(np.square(np.abs(dft))) / len(dft)
    return energy
```

3.3 Vuelva a calcular la energía de la señal, esta vez utilizando el teorema de Parseval. Utilice el siguiente bloque de código.

```python
def energy(signal):

    return np.sum(signal**2)
```

3.4 ¿La energía de la señal cambio entre los diferentes métodos? Si es así, ¿por qué se da este hecho?

4. Transformada inversa de Wavelet

4.1 Utilizando el valor de los coeficientes encontrados anteriormente con la TDW, reconstruya la señal original en el dominio del tiempo. Utilice el siguiente código.

```python
mode = 'symmetric'

# Reconstruir la señal a partir de los coeficientes de TDW
reconstructed_signal = pywt.waverec(coeffs, wavelet, mode=mode)
```

4.2 ¿Qué puede concluir acerca de la reconstrucción de la señal, percibe diferencias considerables con respecto a la señal original?

5. Filtros y la TDW

5.1 Tome la señal de audio, adjunta en la guía (Guitarra.wav) y observe con mucho detalle su comportamiento en frecuencia mediante la TDW.

5.2 Diseñe un filtro FIR pasa-bajas con la banda de transición de 200, frecuencia de corte de 2000Hz y un ripple de 60 dB. Utilice el siguiente bloque de código:

```
import scipy.signal as signal
from scipy.signal import kaiserord, lfilter, firwin, freqz
nyq_rate = fs / 2.0
roll_off = 200
cutoff_hz = 2000
width = roll_off/nyq_rate
ripple_db = 60
N,_= kaiserord(ripple_db, width)
taps = firwin(N, cutoff_hz/nyq_rate, pass_zero=True)
w, h = signal.freqz(taps, [1], worN=2000)
```

5.3 Que puede percibir en el audio al variar la frecuencia de corte a 100 Hz, 1000Hz y a 10000Hz.

5.4 Que reflejan los coeficientes de la TDW al tener una frecuencia de corte de 100Hz o incluso inferior.

6. Conclusiones.

Referencias:

Bibliografía AZOR MONTOYA, J. R.: WAVELETS CON MATHCAD, Trabajo Final, Universidad de Mendoza Facultad de Ingeniería, 1998