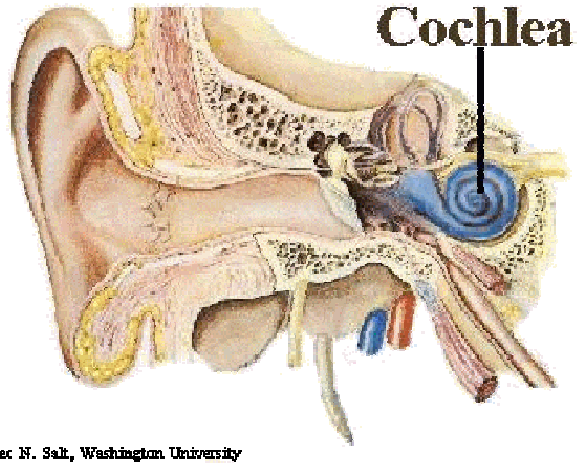
Recuperación de Información Multimedia

Descriptores de Audio

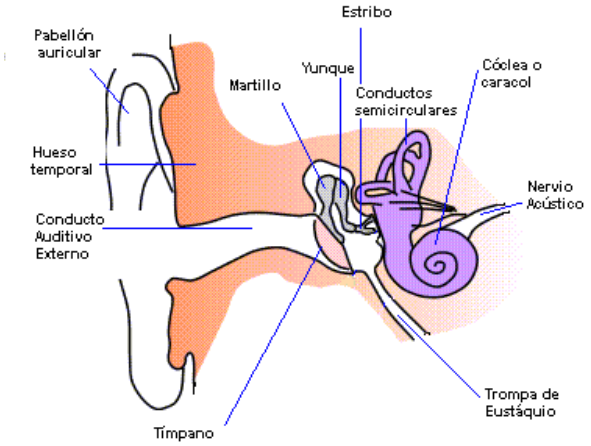
CC5213 – Recuperación de Información Multimedia
Departamento de Ciencias de la Computación
Universidad de Chile
Juan Manuel Barrios – https://juan.cl/mir/ – 2019

Oído Humano

- La onda de aire ingresa por el canal auditivo externo hasta el tímpano, el que mueve una serie de pequeños huesos en el oído medio
- El oído medio golpea el vestíbulo del oído interno moviendo su fluido (perilinfa)
- La Cóclea o Caracol tiene dos ductos por donde va y vuelve la perilinfa y un tercer ducto (rampa coclear) relleno con endolinfa que contiene sensores (estereocilios)
- La Membrana Basilar activa los diferentes estereocilios según las frecuencias del movimiento de la perilinfa.

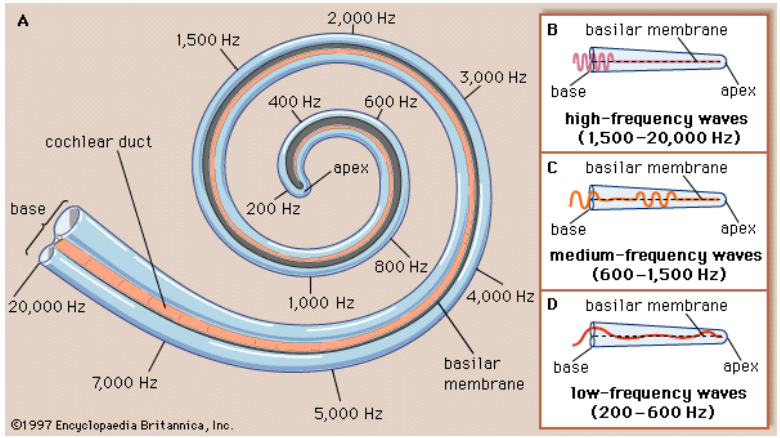


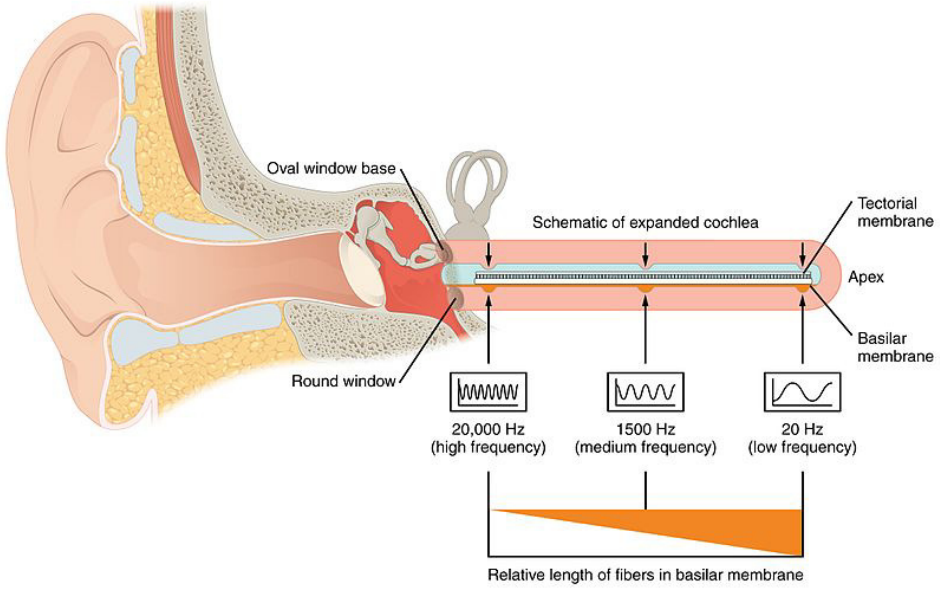
Alec N. Salt, Washington University



Órgano de Corti

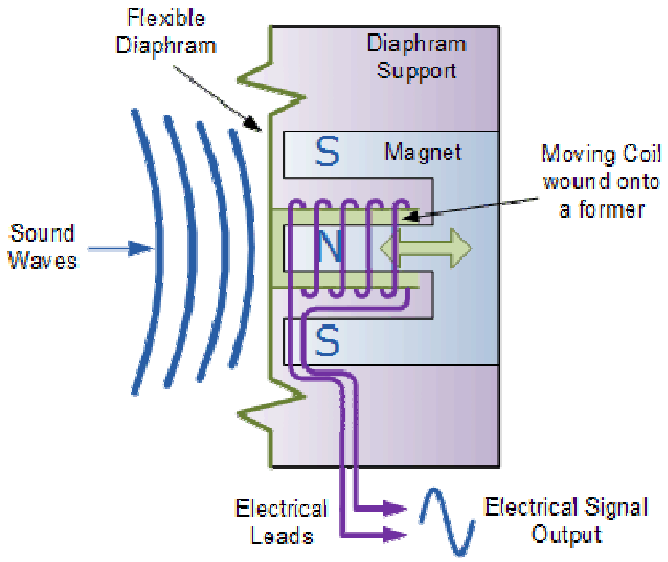
- La Membrana Basilar varía en rigidez y ancho, para tener diferentes frecuencias de resonancia: delgada y rígida (frecuencias altas) hasta ancha y flexible (frecuencias bajas)
- El movimiento de los más de 10 mil estereocilios produce el impulso nervioso que viaja por el nervio auditivo





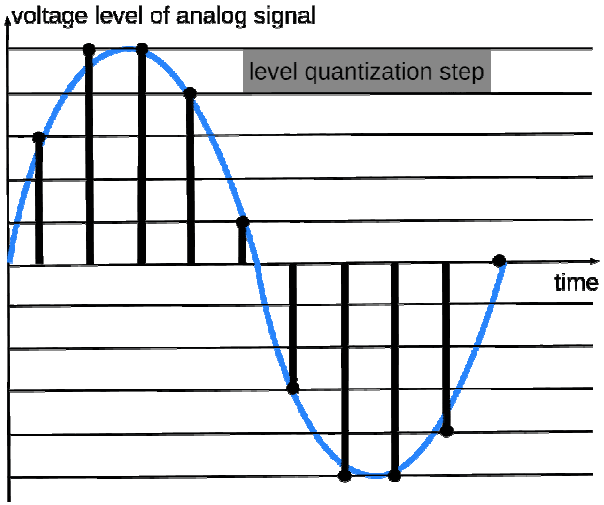
Micrófonos

- La onda de sonido hace vibrar el diafragma del micrófono, que mueve una bobina sobre un imán y produce un impulso eléctrico (inducción electromagnética)



Codificación del audio

- Codificación PCM (pulse-code modulation) consiste en capturar la señal (amplitud) cada cierto tiempo fijo
  - Frecuencia (sampling rate): La tasa a la que se mide la amplitud: 8kHz (voz), 16kHz, 44.1kHz (CD), 48kHz, 96kHz
  - Profundidad (bit depth): La resolución usada para cada valor de amplitud (sample): s16le (entero 16 bits -32768 a 32767), s24le (entero 24 bits), f32le (float 32 bits)

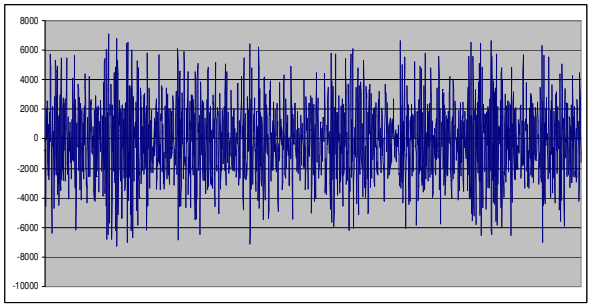


Decodificación del Audio

- Los primeros samples de un audio decodificado:

12	160	207	-36	-651	-976	-796	-642	-458	-50	104	-161	-31	774	1257	1010	535	.....

- Graficando los samples de 1 segundo de audio:



- Convertir un audio en su versión decodificada:

```
ffmpeg -i video.mp4 -ac 1 -vn -sn -ar 16000 -acodec pcm_s16le -f s16le audio.raw
```

- Escuchar un audio decodificado:

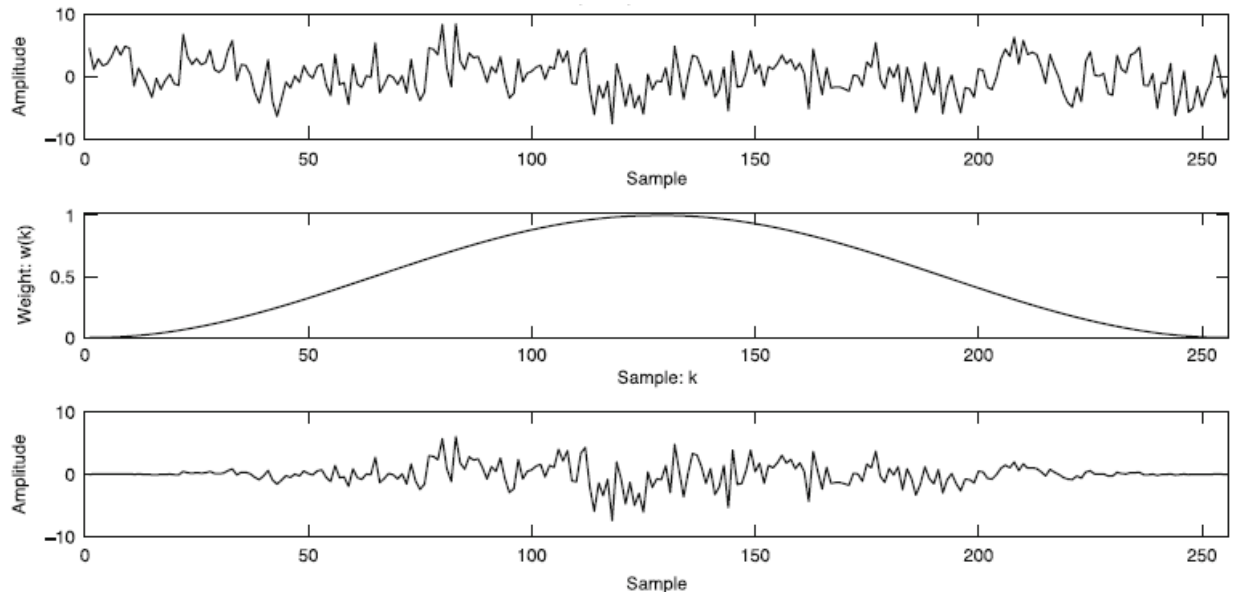
```
ffplay -f s16le -acodec pcm_s16le -ar 16000 audio.raw
```

```
vlc --demux=rawaud --rawaud-channels 1 --rawaud-samplerate 16000 --rawaud-fourcc=s16l audio.raw
```

Windowing

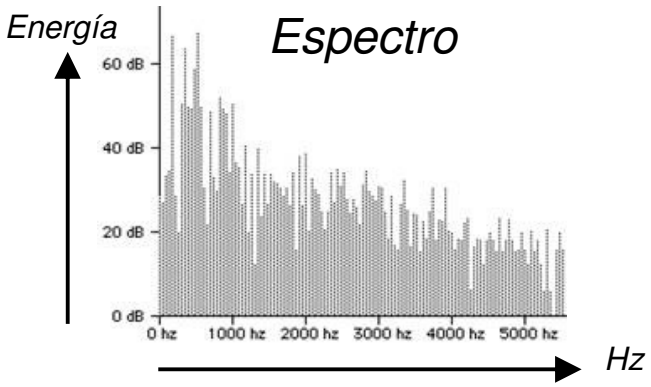
- Dividir pista de audio en ventanas pequeña, usualmente entre 10 a 100 ms, unos 256-8192 samples (potencia 2)
- Se calcula la FFT sobre la ventana de samples para obtener la energía de cada frecuencia
- Para evitar saltos, previo a la FT se multiplica la ventana por una función que suavice bordes (ej. función de Hann)

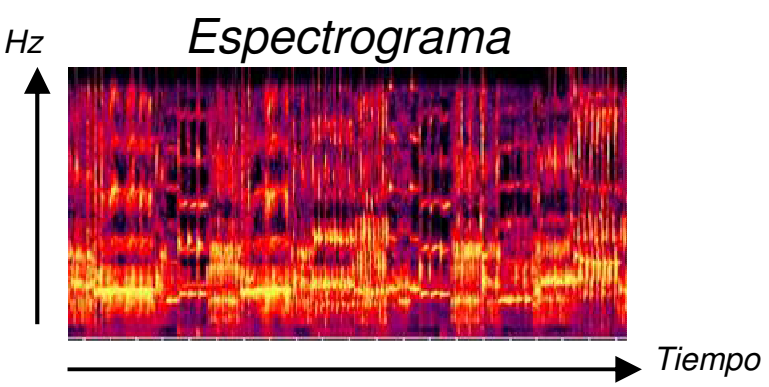
$$w(k) = 0.5 \cdot \left(1 - \cos\left(\frac{2 \cdot \pi \cdot k}{K - 1}\right)\right)$$



Espectro del Audio

- La Transformada de Fourier (FT) sobre la ventana entrega el espectro de frecuencias
  - Cada coeficiente muestra la energía o el aporte de esa frecuencia al audio
  - 300 Hz a 2 kHz es el rango audible más relevante





Descriptores de Bajo Nivel

- Dominio Temporal:
  - **Amplitude Envelope**: Máxima amplitud en la ventana (sample máximo)
    - Representar un ritmo

$$AE_t = \max_{k=t\cdot K}^{(t+1)\cdot K-1} s(k)$$

  - **Root-Mean-Square energy**: Raíz de la suma de los cuadrados de las amplitudes
    - Mide el “loudness” (intensidad del sonido, volumen)

$$RMS_t = \sqrt{\frac{1}{K}\cdot \sum_{k=t\cdot K}^{(t+1)\cdot K-1} s(k)^2}$$

  - **Zero-Crossing Rate**: Cantidad de cambios de positivo a negativo

$$ZCR_t = \frac{1}{2}\cdot \sum_{k=t\cdot K}^{(t+1)\cdot K-1} |\operatorname{sgn}(s(k)) - \operatorname{sgn}(s(k+1))|$$

Ver Knees et al. cap 2.3

Descriptores de Bajo Nivel

- Dominio de Frecuencias:
  - **Band Energy Ratio**: Razón entre energías de frecuencias bajas vs frecuencias altas
  - **Spectral Centroid**: Centro de masa de las energías de las frecuencias
  - **Spectral Spread** o **Bandwidth**: Variación de las energías con respecto al Spectral Centroid
  - **Spectral Flux**: Suma de diferencias de energías entre ventanas consecutivas

$$BER_t = \frac{\sum_{n=1}^{F-1} m_t(n)^2}{\sum_{n=F}^{N} m_t(n)^2}$$

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)}$$

$$BW_t = \frac{\sum_{n=1}^{N} |n - SC_t| \cdot m_t(n)}{\sum_{n=1}^{N} m_t(n)}$$

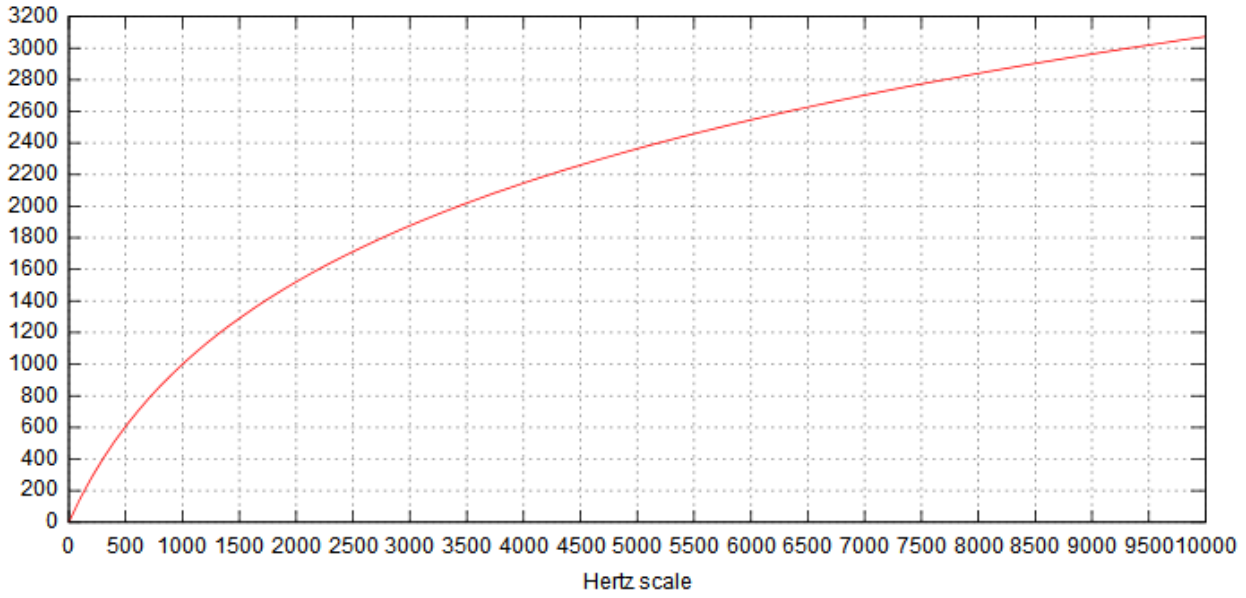
$$SF_t = \sum_{n=1}^{N} (D_t(n) - D_{t-1}(n))^2$$

Ver Knees et al. cap 2.3

Escala Mel

- La sensibilidad del oído no es lineal con las frecuencias
- Convertir frecuencias a escala Mel (escala logarítmica)

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$



Descriptor MFCC

- Mel-frequency cepstral coefficients (MFCC)
  - Convertir cada frecuencia de la FT a la escala Mel
  - Utilizar el logaritmo de los coeficientes de cada frecuencia Mel
  - Codificar los coeficientes usando DCT
  - Los coeficientes de la DCT forman el descriptor
- El “Espectro del Espectro” es llamado el “Cepstro”
- MFCC es el descriptor de audio más usado para reconocimiento de voz y análisis de audio en general

Más Descriptores

- Descriptor OSC (Octave-Based Spectral Contrast) similar al MFCC
- Otros descriptores son las estadísticas en el tiempo de los low-level o MFCC para calcular el Rhythm, Pitch, Chroma
- Se usan como base para problemas de más alto nivel como clasificación de géneros, artistas, grupos musicales, identificar instrumentos, etc.

.

Ver Fu et al, 2011.

Descriptor para Duplicados

- Descriptor binario para detectar duplicados:
  - Tamaño de venta a 25 ms
  - Frecuencias entre 300 Hz a 3 kHz, se calculan 16 frecuencias o sub-bandas
  - Representa el perfil de las frecuencias, si hay aumento o disminución entre bandas consecutivas
  - Descriptor binario de 15 bits
  - Para una ventana *n* y sub-banda *m*, el *m*-ésimo bit del *n*-ésimo descriptor vale 1 si:

$$E(n, m) > E(n, m + 1)$$

Ver Saracoglu et al, 2009.

Descriptor de Audio (Phillips)

- Otro descriptor binario para duplicados:
  - Usa los coeficientes de cada frecuencia Mel
  - Se comparan las diferencias de energía entre canales consecutivos entre dos ventanas consecutivas
  - Descriptor binario 32 bits
  - Para una ventana *n* y sub-banda *m*, el *m*-ésimo bit del *n*-ésimo descriptor vale 1 si la diferencia de energías aumenta:

$$E(n,m) - E(n,m+1) > E(n-1,m) - E(n-1,m+1)$$

Ver Haitsma et al, 2002.

Comparación de series temporales

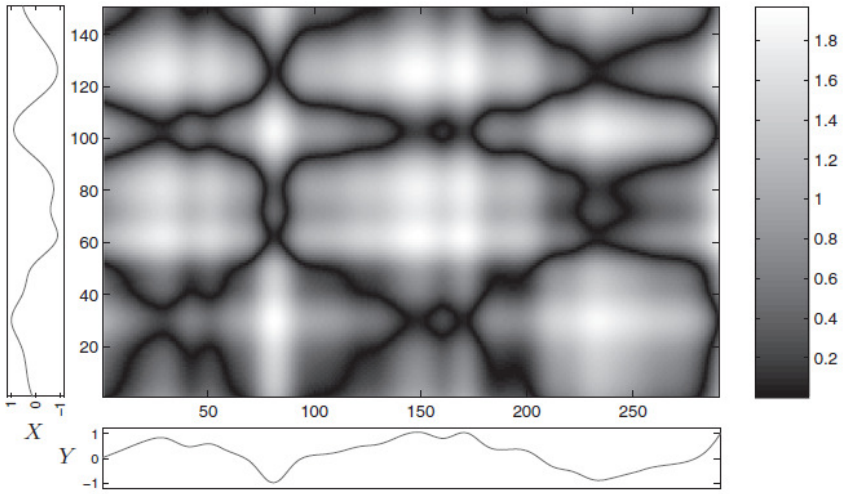
- Dada dos secuencias, la DTW es el costo de la mejor coincidencia entre ellas:

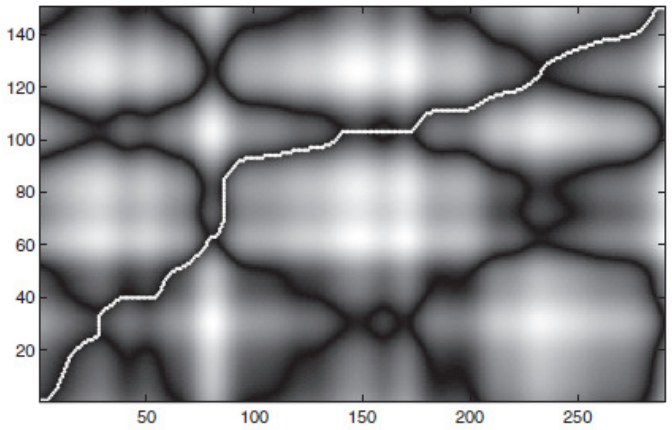
- Busca la mejor alineación entre ambas series
- Puede comparar secuencias de largo distinto
- Definición recursiva (largos n y m):

$$D(n,m) = \min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$$

Dynamic Time Warping

- Calcular matriz de costos entre las componentes de ambas secuencias
  - Distancia entre descriptores
- El valor de la *DTW(x,y)* es el costo del camino óptimo (menor suma total)
- Restricciones:
  - Debe iniciar en (0,0) y finalizar en (*n*,*m*)
  - Camino continuo, sin saltos ni devolverse
  - Se puede fijar un desfase máximo entre secuencias



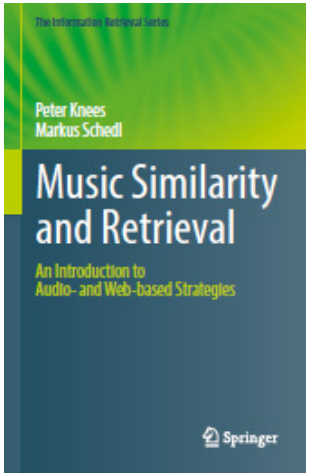


Implementación DTW

- Implementación eficiente usa programación dinámica
- Costo de evaluación: O(*nm*)
- En general, DTW puede tener varios caminos óptimos
- DTW es simétrica si la ground-distance es simétrica
- DTW no es métrica incluso si la ground-distance lo es

Bibliografía

- **Music Similarity and Retrieval**. Knees et al. 2016
  - Cap 2.3 (descriptores low-level)
  - Cap 3 (Mel, MFCC)



Papers

- Z. Fu, G. Lu, K. Ming, and D. Zhang. A **Survey of Audio-Based Music Classification and Annotation**. IEEE Transactions on Multimedia, 2011.
- A. Saracoglu et al. **Content based copy detection with coarse audio-visual fingerprints**. In Proc. of the int. workshop on Content-Based Multimedia Indexing (CBMI), 2009.
- J. Haitsma and T. Kalker. **A highly robust audio fingerprinting system**. In Proc. of the int. symp. on Music Information Retrieval (ISMIR), 2002.

Librerías

- LibROSA: https://librosa.github.io/librosa/
- Marsyas: http://marsyas.info/
- HTK: http://htk.eng.cam.ac.uk/
- Sphynx: https://cmusphinx.github.io/
- Spro: http://www.irisa.fr/metiss/guig/spro/
- CLAM: http://www.clam-project.org
- Calcular MFCC:
  - Versión C: https://github.com/jsawruk/libmfcc
  - Versión C++: https://github.com/tlacael/MFCC