MEMORIA PRÁCTICA 4: Procesamiento audio

Aris Vazdekis Soria & Alejandra Ruiz de Adana Fleitas

Ejercicio 1

Durante este ejercicio, nos enfocamos en el análisis de frecuencias en señales de audio utilizando Python. Implementamos la Transformada Rápida de Fourier (FFT) con la biblioteca NumPy, lo que nos permitió descomponer la señal en sus componentes de frecuencia. Esta técnica es la indicada por el profesor para identificar las notas musicales presentes en una grabación.

Para llevar a cabo el proyecto, utilizamos varias funciones clave:

- **FFT**: Aplicamos `fft()` para realizar la Transformada Rápida de Fourier, que transforma la señal de audio del dominio del tiempo al dominio de la frecuencia. Luego nos quedamos con la mitad de la FFT como nos indico el profesor, y calculamos posteriormente las frecuencias correspondientes a los resultados de la FFT (tambien lo recortaremos para mantener solo la parte positiva). Esto sirve para identificar y visualizar las diferentes componentes de frecuencia presentes en la señal.
- **Detección de picos**: Utilizamos `scipy.signal.find_peaks()` para identificar picos en el espectro de frecuencias. Esta función nos ayudó a localizar la frecuencia dominante, con el fin de poder conocer su nota correspondiente.
- **Visualización**: Implementamos `matplotlib.pyplot` para graficar el espectro de frecuencias, lo que nos permitió observar las frecuencias detectadas y sus amplitudes, proporcionando una representación visual clara de los datos.

Diccionarios de Frecuencias

Una parte importante de nuestro análisis fue la creación de diccionarios de frecuencias, que establecen una relación entre las notas musicales y sus frecuencias correspondientes. Estos diccionarios fueron fundamentales para identificar acordes y notas individuales en la señal. Estos fueron los diccionarios adicionales al de notas_frecuencias para desarrollar las partes optativas:

- **Diccionario de Acordes**: Creamos un diccionario que incluye las frecuencias de los acordes más comunes, como los acordes mayores y menores. Este recurso nos permitió detectar no solo notas individuales del diccionario de notas de piano, sino también combinaciones de notas que forman acordes, facilitando el análisis armónico de la música.
- **Diccionario para Trompeta**: Además, desarrollamos un diccionario específico para las frecuencias características de la trompeta. Esto nos permitió centrarnos en la instrumentación y reconocer las notas tocadas por la trompeta en la señal. Al comparar las frecuencias detectadas con las de nuestro diccionario, pudimos identificar con precisión cuándo y qué notas se estaban tocando, así como los acordes generados en conjunto.

A lo largo de este proceso, logramos detectar no solo notas individuales, sino también acordes de tres notas musicales simultaneas. Una dificultad muy marcada en el proyecto fue el conseguir notas musicales de calidad, puesto que en internet la calidad de instrumentacion para la grabacion de notas musicales suelen ser de dispositivos moviles o en espacios no insonorizados, lo que afecta directamente a la frecuencia de la pieza a analizar por la transformada rapida de fourier. Tras mucha investigacion por varias paginas web y por varias piezas de sonido, procederemos a dejar una bibliografia con los enlaces utilizados con notas musicales de calidad. Un problema muy comun en las octavas bajas es que al ser de menor frecuencia, menos perceptible llega a ser la nota musical, llegando hasta notas muy bajas como la C1 con apenas 32hz. Los microfonos estandares del mercado normalmente suelen estar fabricandos para capturar voz por lo que suelen oscilar en el rango de 40hz hasta los 20.000Hz por lo que en octavas altas no hay problema pero en octavas bajas se complica mucho la deteccion.

La visualización del espectro de frecuencias, junto con la detección de picos, nos ha brindado a su vez conocimiento sobre el analisis de frecuencias en sonido, asi como sobre la estructura de la música. Cada vez que se detectaba un acorde, lo registrábamos y lo comparábamos con nuestros diccionarios, lo que nos permitio comprender y mejorar nuestro conocimiento sobre notas musicales.

Pese a que este ejercicio fue tedioso por la dificultad de encontrar archivos de sonido (muy tedioso en nuestra experiencia) no quita que el trabajo nos gustase muchisimo, combinando en lineas generales conceptos teoricos con piezas musicales comunes. Esto nos hace percibir la musica de una manera un tanto distinta e interesante.

Ejercicio 2

El segundo ejercicio se centró en el desarrollo de una aplicación de filtrado de audio utilizando una interfaz gráfica construida con Tkinter. Este ejercicio nos permitió profundizar en el procesamiento de señales de audio, aplicando diversos tipos de filtros a una señal de audio y visualizando los resultados.

Descripción de Funciones

1. **Filtros Digitales**: Implementamos varios tipos de filtros digitales: pasa-bajos, pasa-altos, pasa-banda y rechaza-banda. Estas funciones se basan en el diseño de filtros Butterworth, lo que garantiza que sean suaves y sin oscilaciones.
2. **Aplicación de Filtros**: La función `apply_filter()` permite aplicar el filtro seleccionado a la señal de audio. Esta función utiliza el método `filtfilt()` de SciPy para aplicar el filtro de manera que se minimicen las distorsiones de fase.
3. **Interfaz Gráfica**: Usamos Tkinter para crear una interfaz gráfica que permite al usuario seleccionar el tipo de filtro, ajustar los valores de corte y el orden del filtro. Además, los resultados se visualizan en tiempo real, mostrando tanto la señal original como la filtrada.
4. **Visualizacion de señal original y filtrada:** Tambien se empleo una funcion llamada `plot_signals()` en donde se grafican los datos originales y los filtrados en tiempo

directo, actualizando las graficas si se cambia el filtro en la interfaz grafica durante su ejecucion.

5. **Reproducción de Audio**: La aplicación a su vez tambien realiza la reproducción del audio filtrado utilizando Pygame, ejecutando la reproducción en un hilo separado para no bloquear la interfaz gráfica. Esta separacion en hilos fue resultado de una serie de errores, congelamientos de pantalla y problemas que nos surgieron con la ejecucion de la funcion `pygame.mixer.music.play()`

Este ejercicio nos ayudó mucho a entender de forma práctica cómo los filtros cambian las señales de audio, y escuchar los resultados en tiempo real nos permitió comprender mejor la teoría sobre frecuencias graves y agudas. Con la interfaz, pudimos ver de forma clara cómo distintas configuraciones de filtros afectaban el contenido espectral de la señal y cómo el sonido cambiaba según el filtro que aplicábamos y los parámetros que ajustábamos.

Además, la aplicación en tiempo real nos dio una experiencia mucho más cercana y útil sobre el filtrado digital y cómo se usa en el procesamiento de audio. Crear la interfaz gráfica también nos enseñó habilidades para el desarrollo de aplicaciones prácticas, lo que hizo que el aprendizaje fuera más accesible y visual. Al final, ver cómo cada decisión sobre el tipo de filtro y los parámetros influía en el sonido que obteníamos fue realmente interesante.

Bibliografia practica 1

- https://freesound.org/search/?f=grouping_pack%3A%222489_Piano+FF%22&s=Date+added+%28newest+first%29&g=1&page=1#sound (notas de calidad)

- https://es.wikipedia.org/wiki/Frecuencias_de_afinaci%C3%B3n_del_piano (para conocer notas y sus frecuencias)

- https://freewavesamples.com/sample-type/synthesizer/piano?page=1 (para las octavas bajas)

- https://freesound.org/search/?f=grouping_pack%3A%2229677_Piano+Chords%22&s=Date+added+%28newest+first%29&g=1&page=1#sound (para los acordes)