DISEÑO DE SOFTWARE IDENTIFICADOR DE SEÑALES MUSICALES

Castelli, Corina[1] y Passano, Nahuel[2]

[1]Universidad Nacional de Tres de Febrero
castellicr2111@gmail.com
[2]Universidad Nacional de Tres de Febrero
n.passano@hotmail.com

Resumen — *En este informe se detalla el desarrollo de un algoritmo, programado en Python, capaz de identificar señales musicales dentro de una base de datos. El programa permite que el usuario grabe una muestra de audio para su posterior procesamiento e identificación. Para describir las señales, se implementa el método de* $audio\ fingerprinting$ *utilizando una función que obtiene la matriz de máximos locales de la STFT de la señal. Este procedimiento se aplica tanto a las canciones de la base de datos como al audio ingresado por el usuario y luego se evalúa si hay o no coincidencia o similitud aplicando la noción de correlación cruzada. El algoritmo demuestra ser eficiente y veloz al mismo tiempo, con un buen funcionamiento incluso en condiciones de SNR bajas.*

1. INTRODUCCIÓN

La propuesta para este trabajo consiste en hallar un método eficiente y veloz para el reconocimiento de un cierto patrón o canción dentro de una base de datos dada. En el mismo se explica qué algoritmos son necesarios y qué herramientas teóricas se deben comprender en profundidad. La base de datos consta de 6 canciones de géneros distintos, donde la comparación del patrón registrado con la base de datos se hace a través del sistema de $audio\ fingerprinting$.

El $audio\ fingerprinting$ es un procedimiento que comprende la obtención de una matriz característica de una señal de audio, tal que contenga la suficiente información clave como para corresponderse con la señal original únicamente (en el caso ideal). De esta manera se puede establecer la igualdad perceptual de dos señales sin compararlas de manera directa, sino que comparando sus huellas asociadas, las cuales son mucho más sencillas y livianas. Un sistema de $audio\ fingerprinting$ consiste de un método para extraer la $fingerprint$ de una señal y buscar huellas coincidentes dentro de una base de datos generada.

El programa diseñado consta de cuatro algoritmos elementales: un algoritmo encargado de obtener la $fingerprint$ de la señal de audio ingresada, uno que graba la señal que se desea identificar, uno que desarrolla la base de datos, y por último un algoritmo que realiza las comparaciones y determina el resultado. El funcionamiento del software está coordinado mediante una interfaz gráfica.

2. MARCO TEÓRICO

Para el desarrollo del trabajo, se necesitan comprender los siguientes conceptos teóricos:

2.1. Correlación cruzada

En procesamiento de señales, la correlación cruzada es una medida de la similitud entre dos señales, frecuentemente usada para encontrar características relevantes en una señal desconocida por medio de la comparación con otra que sí se conoce. Formalmente se la define como función del tiempo relativo entre las señales, a veces también se la llama producto escalar desplazado, y tiene aplicaciones en el reconocimiento de patrones y en criptoanálisis.

$$r_{xy} = \sum_{k=-\infty}^{+\infty} y[k]x[k-n] \tag{1}$$

Por medio de herramientas computacionales, se puede extender dicha definición y aplicar su uso en señales de dos dimensiones, por ejemplo.

2.2. Transformada discreta de Fourier

La transformada discreta de Fourier o DFT (Discrete Fourier Transform) es un tipo de transformada discreta

utilizada en el análisis de Fourier. Transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. La DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Dichas secuencias se suelen generar a partir del muestreo de una función continua. En particular, la DFT se utiliza comúnmente en procesado digital de señales y otros campos relacionados dedicados a analizar las frecuencias que contiene una señal muestreada. Un factor muy importante para este tipo de aplicaciones es que la DFT puede ser calculada de forma eficiente en la práctica utilizando el algoritmo de la transformada rápida de Fourier o FFT (Fast Fourier Transform).

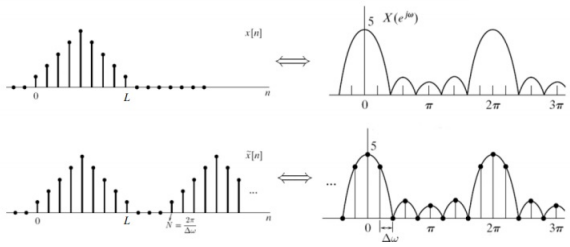

Figura 1: Representación gráfica de la transformada discreta de Fourier

2.3. Ventanas

Una ventana es una función matemática que se emplea en el análisis y muestreo de señales por computadora. Dado que no podemos analizar una señal analógica ya que en teoría es infinita, previamente debemos aplicar la DFT a la señal, y pasar del dominio del tiempo al dominio de frecuencia. Una vez hecho esto, multiplicamos la señal por la función ventana y obtenemos de este modo los primeros segundos de ese intervalo, lo que nos permite analizar detalladamente los componentes de dicha señal. Dicho proceso se denomina windowing. Las ventanas tienen tres propiedades: un ancho de lóbulo principal, un nivel de lóbulos laterales y una pendiente de caída. Cuanto mayor es el lóbulo principal, menor va a ser el nivel de lóbulos laterales. Los lóbulos laterales de gran magnitud producen efectos de rizado en la respuesta frecuencial del tramo de señal ventaneado.

La ventana mas típica y sencilla es la ventana rectangular, que se define en dominio temporal como:

$$h[n] = \begin{cases} 1 & 0 \leq n \leq M-1 \\ 0 & \text{Caso contrario} \end{cases} \qquad (2)$$

donde M es el tamaño de la ventana.

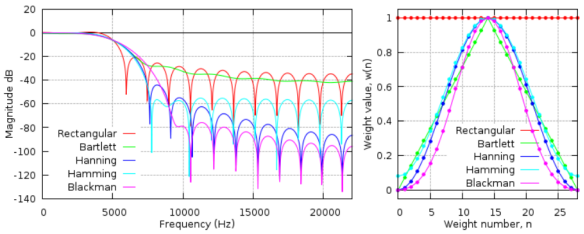

Figura 2: Comparación entre varias ventanas en dominio frecuencial y temporal

2.4. Transformada corta de Fourier

La Transformada corta de Fourier o STFT (Short-time Fourier transform, STFT) está relacionada con la transformada de Fourier usada para determinar el contenido en frecuencia sinusoidal y de fase en secciones locales de una señal así como sus cambios con respecto al tiempo. En el caso del tiempo discreto, la información a ser transformada podría ser dividida en pedazos o tramas (que usualmente se traslapan unos con otros, para reducir irregularidades en la frontera). Cada segmento es una transformación de Fourier, y el resultado complejo se agrega a una matriz, que almacena magnitud y fase para cada punto en tiempo y frecuencia. Esto se puede expresar como:

$$STFT\{x[n]\} = X[l,k]$$

$$X[l,k] = \sum_{k=0}^{M-1} x[n+lL] \cdot w[n] \cdot e^{-j\frac{2\pi kn}{M}} \qquad (3)$$

Donde, $x[n]$ es la señal, $w[n]$ es la ventana, M es el tamaño de la ventana y L es el tamaño del salto temporal entre segmento y segmento.

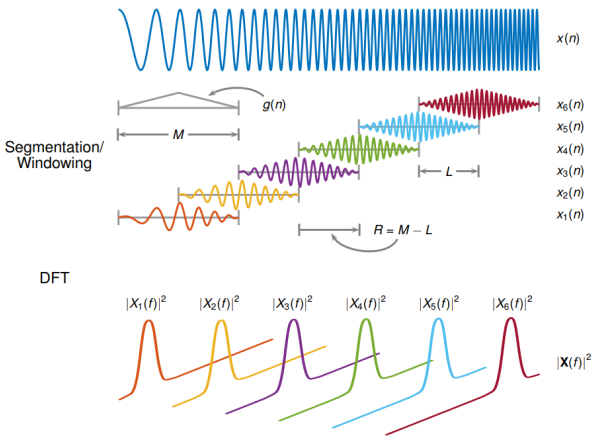

Figura 3: Descripción gráfica de la STFT.

La STFT al igual que la DFT son frecuentemente usadas para analizar música. El espectrograma es el resultado de calcular el espectro de una señal por ventanas de tiempo de la misma. Matematicamente se define como

$|STFT\{x[n]\}|^2$. La representacion mas convencional es en dos dimensiones, donde los ejes representan los dominios temporales y frecuenciales, y cada punto de la gráfica se colorea caracterizando la energía en dicho punto de tiempo y frecuencia, como se ve en la Figura (4).

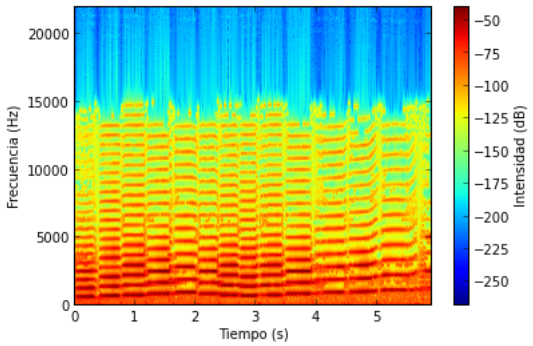

Figura 4: Espectrograma en dos dimensiones

3. PROCEDIMIENTO

El funcionamiento del software se centra en la aplicación de cuatro algoritmos puntuales articulados por una interfaz gráfica de usuario (GUI). Cada algoritmo cuenta con funcionalidades distintas que aportan al resultado final.

3.1. Algoritmo de extracción de información

En primer lugar, se toma como descriptor representativo de una señal de audio, su transformada corta de Fourier (STFT), ya que la misma brinda información tanto temporal como frecuencial de la señal, además de establecer un nexo entre frecuencia y tiempo de la señal, característica que en la música es fundamental para reconocer una melodía, por ejemplo.

La STFT a su vez, posee distintos parámetros a asignar para operar, como lo son la ventana y su respectivo largo M. Por su buena diferenciación en frecuencia, se elige usar una ventana del tipo rectangular de tamaño $M = 1000$, también se debe tener en cuenta el solapamiento que ocurre en los diferentes saltos. Para cumplir con el criterio COLA[3], el factor de solapamiento debe ser de (1/2). Los audios a utilizar como referencia en la búsqueda, son cargados y re-sampleados a $f_s = 8\ kHz$, además de pasarlos a formato mono. Esta modificación se toma en cuenta por lo siguiente: el software se implementa en laptops o teléfonos móviles, es decir, se capta la señal a analizar a través de un micrófono de celular o laptop, los cuales están diseñados para percibir las frecuencias mas importantes en la inteligibilidad de la palabra (desde $1\ kHz$ hasta $4\ kHz$ aproximadamente), por lo tanto, no serán de mucha utilidad frecuencias por encima de lo mencionado, además de que tanto las laptops como los celulares tienen sistema de captación mono, entonces es conveniente a la hora de comparar, que tanto la base de las canciones como el audio registrado sean en formato mono. Además, ahorra procesamiento computacional, menos información harán mas ágil el código.

Una vez cargados los audios se procede a buscar las zonas relevantes de la STFT, para ello, se analizan los máximos locales del módulo de la matriz (ya que la STFT es una matriz compleja, se analiza el módulo de cada elemento). A esta matriz con los máximos locales también se la conoce como *Constellation Map*. Para hallar dichos máximos se utilizan librerías especializadas en procesamiento digital de imágenes, como lo es el módulo `ndimage` de la librería `scipy`. La información de la ubicación temporal y frecuencial de dichos máximos locales se almacena en otra matriz, la cual se la suele llamar $fingerprint$. Dicha matriz es la que se utiliza para comparar los audios entre si.

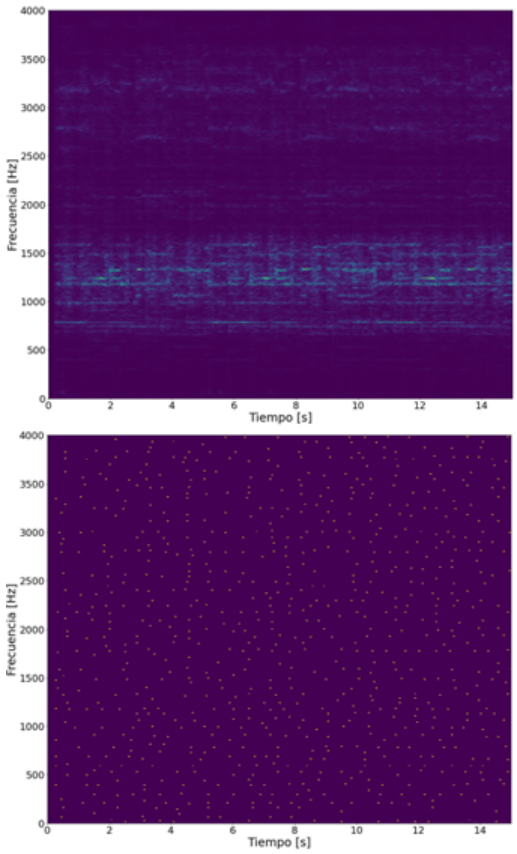

Figura 5: STFT y su respectiva $fingerprint$

3.2. Algoritmo de la base de datos

El algoritmo encargado del desarrollo de la base de datos, genera una conexión entre las $fingerprints$ y los títulos de las canciones. Para guardar las variables generadas por este algoritmo y luego cargarlas para el proceso de identificación se utiliza la librería `pickle`. Para el

caso estudiado, el tamaño en almacenamiento de la base de datos no es un problema o inconveniente, ya que son pocas canciones, pero en una situación mas amplia, la extracción de información debe ser mas minuciosa para ahorrar espacio.

3.3. Algoritmo de grabación

Para llevar a cabo la grabación de la canción a analizar, se desarrolla un algoritmo capaz de registrar la canción, pudiendo seleccionar la entrada de audio, con una frecuencia de muestreo de $f_S = 8\,kHz$ en formato mono, ya que los celulares y laptops tienen sistema de microfoneo mono. La frecuencia de muestreo seleccionada se argumenta anteriormente.

3.4. Algoritmo de identificación

Para identificar si una canción pertenece o no a la base de datos, y especificar concretamente cual de todas es, se utiliza el concepto de correlación cruzada. Como bien se menciona anteriormente, la correlación cruzada analiza la similitud entre dos señales. En este caso, se aplica entre la $fingerprint$ del audio a analizar, con cada una de las $fingerprints$ pertenecientes a la base de datos. El resultado de cada una de estas correlaciones será un vector, donde cada componente será la correlación o similitud que haya en esa marca temporal. Calculando el máximo del vector se encuentra el punto donde el audio es mas parecido al audio de la base de datos analizado. Comparando uno a uno el audio de prueba con las canciones de la base de datos, se obtienen seis vectores de correlación, donde el valor máximo de cada vector será el punto donde mas parecida fue la muestra a las canciones de la base de datos. La correlación, o la máxima correlación depende del tiempo de grabación del audio, ya que, a más segundos grabados, más posibilidades de que haya coincidencias entre ambas $fingerprints$.

Para definir el criterio de decisión, se analizan los máximos de cada vector de correlación. Si el valor mas grande de correlación es mayor a 20, o es mayor o igual a 2.5 veces el segundo mayor valor de correlación detectado, se puede afirmar que el audio analizado corresponde al vector donde se encuentra dicho máximo de correlación.

3.5. Interfaz gráfica de usuario

Para articular el funcionamiento del código, se implementó una GUI mediante la librería `PyQt5`.

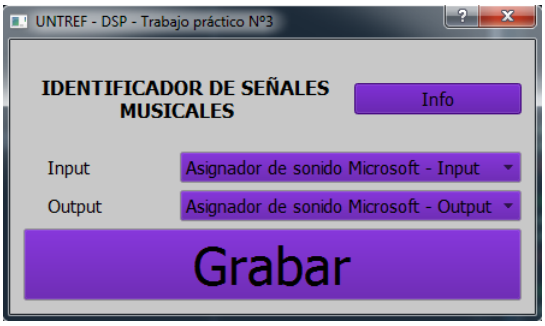

Figura 6: Interfaz gráfica

La interfaz cuenta con opciones para configurar la entrada y salida de audio. Presionando el botón "Grabar", se ejecuta una función que activa el algoritmo de grabación. De esta manera se obtiene un audio de 3 segundos como muestra de la canción a identificar. Este audio pasa posteriormente por el algoritmo de extracción de información y luego es comparado con las canciones de la base de datos mediante el algoritmo de identificación tal como se detalló anteriormente. Si se detecta una correlación mayor al criterio establecido, se despliega una ventana informando el título de la canción. Además, dicha ventana cuenta con un botón de 'Play' para que el usuario pueda reproducir la canción original identificada.

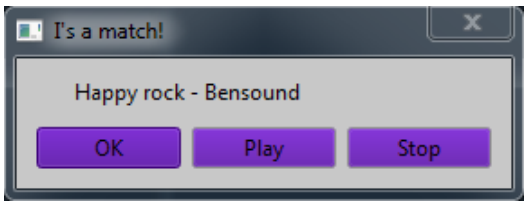

Figura 7: Ejemplo cuando hay una coincidencia

4. RESULTADOS Y ANÁLISIS

Para analizar el tiempo necesario que necesita el software para detectar correctamente, se realizó la prueba reproduciendo las canciones y configurando la grabación en 0.5, 0.75, 1, 2 y 3 segundos. Las grabaciones se realizaron con un micrófono condenser MXL 991, preamplificado por una placa de sonido Audient iD 44, y las canciones se reprodujeron a través del parlante de un celular. Los resultados se muestran en la Tabla 1.

Grabación [s]	Correctos/Intentos	Eficacia
0.5	15/30	50.00 %
0.75	21/30	70.00 %
1	28/30	93.33 %
2	29/30	96.67 %
3	30/30	100.00 %

Tabla 1: Eficiencia del algoritmo a pruebas sin ruido.

Por lo tanto, tomando una muestra de audio de 3 segundos o mas, se garantiza una efectividad del 100 %.

La cantidad de segundos que tarda el software en emitir una respuesta se mide con ayuda de la librería `timeit`, cuyo resultado se presenta en la Tabla 2.

Grabación [s]	Tiempo de detección [s]
0.50	0.1969
0.75	0.2632
1	0.3206
2	0.6005
3	0.8383

Tabla 2: Promedio de la velocidad de detección.

Para analizar las probabilidades de que ocurra un falso positivo, es decir, una canción que no está en la base de datos sea identificada y catalogada como perteneciente a la misma y arroje un resultado, se pone a prueba el algoritmo grabando canciones externas a la base de datos. Los resultados fueron muy buenos, ya que independientemente del tiempo de grabación, no se detectaron falsos positivos, todas las canciones analizadas que no estaban en la base de datos fueron rechazadas, es decir, no hubo coincidencia alguna. En las mediciones se puede constatar que ninguna correlación para los audios externos superó un valor máximo de 13, lo cual respalda el criterio de decisión tomado para la ejecución del algoritmo.

Por otro lado, se sometió al software a captar una muestra de audio con dos canciones de la base de datos sonando a la vez, lo cual genera dos correlaciones de gran valor, es por eso que se toma como parámetro de decisión alternativo que el máximo de correlación tiene que ser 2.5 veces mayor al segundo máximo de correlación para poder otorgar con seguridad la respuesta.

En un contexto real, las grabaciones no son en un ambiente tranquilo, por lo tanto, la relación señal ruido es un factor importante que afecta la eficacia del algoritmo. Para su puesta a prueba, se generan los mismos audios de la base de datos, pero con la suma de ruido de distribución normal. Dichos ruidos se diseñan tal que tal que las SNR (del ingles signal-noise ratio) sean de 6, 9, 12 y 18 dB respectivamente. Los audios generados con el ruido añadido, se reproducen por el parlante de un celular, y se registran a través del micrófono mencionado anteriormente. Los resultados se muestran en la Tabla 3.

SNR [dB]	Eficacia
6	50.00 %
9	80.00 %
12	96.66 %
18	100.00 %

Tabla 3: Eficiencia en función de la SNR.

A modo de comparación empírica, se probó el mismo algoritmo, pero utilizando otra ventana en el calculo de la STFT. La ventana a analizar fue la ventana de Blackman de largo M=2400, con un factor de solapamiento de (2/3), cumpliendo con el criterio COLA. Se realizaron las mediciones de la eficiencia en función del tiempo de grabación y la eficiencia en función de la relación señal ruido. Los resultados se muestran en las tablas 4 y 5.

Grabación [s]	Correctos/Intentos	Eficacia
0.50	12/30	40.00 %
0.75	19/30	63.33 %
1	25/30	83.33 %
2	30/30	100.00 %
3	30/30	100.00 %

Tabla 4: Eficiencia del algoritmo con ventana Blackman.

SNR [dB]	Eficacia
6	53.33 %
9	80.00 %
12	100.00 %
18	100.00 %

Tabla 5: Eficiencia en función de la SNR con ventana Blackman.

Como se puede ver, no hay una diferencia notable o significativa. Los tiempos de ejecución de ambos también son muy similares, por lo tanto se decidió utilizar el algoritmo con ventana rectangular. Por último, se puede comparar directamente el algoritmo dependiendo la ventana como se ve en la Figura 8.

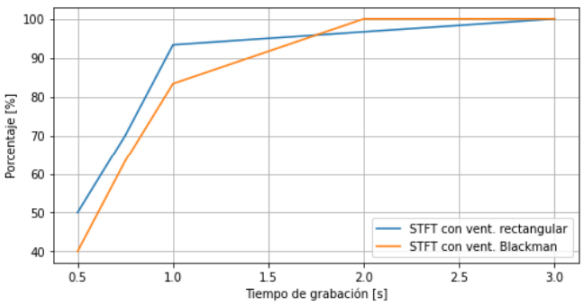

Figura 8: Comparación de eficiencia entre las ventanas

5. CONCLUSIONES

Con los resultados y análisis realizados, se puede decir que se cumplieron las expectativas con respecto al funcionamiento del software, tanto su eficacia y velocidad de predicción, como su comportamiento ante relaciones señal ruido bajas.

En las mediciones de la relación señal ruido para el track 'Funky element', se observó que cuando las señales tienen mucho factor de cresta, es decir, una diferencia muy grande entre su valor pico y su valor RMS, el ruido se hace mucho mas presente, y por consecuente, mas difícil de detectar.

El criterio de decisión es acertado y eficiente, pero se aplicó sabiendo que las canciones en la base de datos son bastante distintas entre sí. En un caso con una base de datos más extensa y que contenga audios con cierta similitud entre ellos, el criterio se debería refinar un poco y quizás contemplar otros parámetros de comparación más allá de la correlación.

La comparación entre ventanas (rectangular y Blackman) no resultó como se esperaba, teóricamente el ventaneo con Blackman debería tardar mas en detectar (por su largo mas extenso) pero mejorar significativamente las mediciones de SNR (por su alta atenuación a los lóbulos secundarios).

En lineas generales los algoritmos en conjunto con la GUI se ejecutan correctamente y cumplen con los requisitos solicitados.

6. REFERENCIAS

[1] A Highly Robust Audio Fingerprinting System - Jaap Haitsma, Ton Kalker.

[2] An Industrial-Strength Audio Search Algorithm - Avery Li-Chun Wang, Shazam Entertainment, Ltd.

[3] Documentación de librería Scipy - Función `scipy.signal.check_COLA`