Situación problema F1009: Reconocimiento de audio mediante análisis de frecuencias
Profesor Dr.-Ing,Jonathan Montalvo-Urquizo

Miguel Ángel Chávez Robles A01620402

27 de noviembre de 2020

Resumen

El presente trabajo explica el desarrollo de un programa de reconocimiento de audio el cual es capaz de identificar una canción mediante una muestra de la misma. La problemática que se busca resolver es imitar el reconocimiento que nuestro cerebro es capaz de realizar para identificar una canción que ya conocemos total o parcialmente, de manera computacional. Tomando dicha situación se propone la creación de un programa capaz de muestrear un fragmento significativo de una canción y a través de este extraer información importante que permita realizar una comparación y determinar de que canción se trata. Para efectuar esto se apoya de métodos como la Transformada rápida de Fourier; un importante método matemático usado ampliamente en el análisis de señales en la industria, así como funciones de autoría propia y funciones del software Matlab. El trabajo se desarrolló partiendo de la pregunta: ¿Que proceso hace nuestro cerebro para identificar un sonido o conjunto de sonidos que ya conocemos?.

1. Introducción

El objetivo de este trabajo es implementar un programa que sea capaz de identificar de manera eficiente canciones mediante un análisis comparativo de sus propiedades características, buscando obtener resultados similares a software ya existente en el mercado como Shazam.

En concreto, se busca realizar un programa que sea capaz de identificar una canción teniendo una pequeña muestra de la misma. Siendo este un trabajo con un alcance menor al del software similar en el mercado, se limitará a un programa que es capaz de identificar solo un número limitado de canciones. Para lograr los objetivos se utilizará Matlab, así como el toolbox de audio del mismo, para obtener las grabaciones deseadas, además de métodos matemáticos que nos permiten cuantificar las características de las grabaciones.

2. Marco teórico

2.1. Movimiento ondulatorio

En *Física para ciencias e ingeniería*, Serway y Jewett afirman que la esencia del movimiento ondulatorio puede reducirse en:

"La transferencia de energía a través del espacio sin el acompañamiento de transferencia de materia" [Serway et al., 2007]

2.2. Onda mecánica

A partir de la definición dada en la *sección 2.1*, se entiende como onda mecánica un movimiento ondulatorio el cual cuenta con tres elementos descritos por Serway y Jewett en su libro *Física para ciencias e ingeniería volumen 1*:

1. Una fuente de perturbación (Esta fuente por consiguiente debe tener una fuente de energía para generarla).

2. Un medio de transferencia, este medio debe tener elementos que puedan ser perturbados por la fuente mencionada anteriormente.

3. Un mecanismo físico que permita que los elementos del medio de propagación pueden influenciarse mutuamente, por ejemplo, en una perturbación presentada dentro del agua, el mecanismo físico y el medio son ambos el agua, pues esta propaga las perturbaciones y permite que estas se extiendan mediante la misma.

[Serway et al., 2007]

2.2.1. Ondas sonoras

Serway y Jewett definen las ondas sonoras como ondulaciones que pueden viajar mediante cualquier material, siendo su presentación más común el aire, las cuales percibimos como ondas mecánicas, las cuales generan perturbaciones en este medio, logrando la percepción humana de la audición. Estas ondas se clasifican en

- Ondas audibles: se encuentran dentro del rango de sensibilidad del oído humano.
- Ondas infrasónicas: tienen frecuencias que se encuentran debajo del intervalo de audición humana, son usados por algunos animales para comunicarse
- Ondas ultrasónicas: contienen frecuencias que se encuentran por arriba de el espectro de audición humano, entre sus aplicaciones se puede denotar la obtención de imágenes médicas.

[Serway et al., 2007] Las ondas sonoras son ondas con un comportamiento omnidireccional, y somos capaces de percibirlas debido a cambios en la presión del fluido en el que se propagan. Un sonido periódico puede ser representado con la función de onda de desplazamiento:

$$s\left(x,t\right) = s_{max}\cos\left(kx \pm \omega\right) \tag{1}$$

Donde s_{max} representa la máxima posición de un elemento del medio que se mueve armónicamente respecto al equilibrio $\frac{2\pi}{\lambda}$ el número de onda y $\omega = 2\pi f$ la frecuencia angular de la onda y $f = \frac{1}{T}$ el periodo de la onda. La función de onda de presión está dada por:

$$\Delta P(x,t) = P_{max}\sin(kx \pm \omega t) \tag{2}$$

Donde $P_{max} = \rho v \omega s_{max}$.La intensidad de sonido se define como la rapidez con la cual este se propaga por unidad de área, dado que el sonido se propaga onmidireccionalmente se considera esta área como una esfera. El nivel de intensidad es quizá la magnitud que más asociamos al sonido, pues es una magnitud logarítmica la cual se mide en decibeles. [Sánchez,].

2.3. Señales

Oppenheim describe las señales como funciones de una o más variables que tienen información de un fenómeno o algún comportamiento de la naturaleza. [Oppenheim et al., 1998] Uno de los contextos de gran interés para el estudio de las señales es la reconstrucción de una señal que ha sido alterada o degradada. Las señales se pueden presentar en formas continuas o discretas. Para el procesamiento de señales, se usan representaciones discretas de las señales de interés por medio de un muestreo.

2.3.1. Señales de audio

Olmos nos menciona que estas son señales analógicas las cuales se encuentran dentro del espectro que se considera audible por humanos, es decir que sus frecuencias se encuentran en un rango de entre los 20 a los 20,000 Hz. [Olmos Herrera, 2020]

2.4. Muestreo digital

Dado que el sonido es una señal analógica, es necesario realizar un proceso de conversión para poder representarla de manera digital. Este proceso puede lograrse gracias al muestreo digital. Se define como muestreo la cantidad de veces que se cuantificará una señal por unidad de tiempo. Estas mediciones se tomando fragmentos de la señal en un tiempo continuo en instantes de tiempo discreto dados por un

intervalo. [Mendoza, 2016] Una tasa de muestreo de $10{,}000Hz$ quiere decir que por cada segundo se tomaron 10,000 muestras. Lograr reconstruir una señal completamente a partir de sus muestras, es posible si se cumplen las condiciones que establece el Teorema de Nyquist-Shannon.

2.5. Teorema de Nyquist-Shannon

Es un teorema establecido por Harry Nyquist, y demostrado por Claude Shannon, el cual establece que la reconstrucción exacta de una señal mediante sus muestras es teóricamente posible si la tasa de muestreo es de al menos el doble del ancho de banda de la señal, es decir, si quisiéramos reconstruir una señal con un ancho de 1,000 Hz, las muestras deberán tomarse a 2,000 Hz.

2.6. Transformada de Fourier

J.F James acertadamente menciona que, gran parte del análisis llevado a cabo en la física se centra en ondas. Estas se encuentran presentes en una gran cantidad de fenómenos, tales como electromagnetismo, fluidos, sonido, etc. Estos conceptos están muy relacionados con señales, y con su espectro. Las formas de onda de interés pueden ser analizadas para entender sus propiedades, en concreto la frecuencia y amplitud. [James, 2011] Esto último es posible gracias a la transformada de Fourier. Esta transformada se define por la siguiente integral:

$$\hat{f}(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{iwx} dx \tag{3}$$

Sin embargo, esta representación, como se ha establecido en la *sección 2.4*, no es utilizada en la práctica, y se usa en lugar una definición discreta de la ecuación 3.

2.6.1. Transformada discreta de Fourier

Es natural que se requiera una representación discreta para esta transformada, pues para poder tomar muestras se debe hacer en intervalos finitos. Es por esto, que surge la transformada discreta de Fourier, la cual da una buena aproximación a su contraparte continua, es computacionalmente más eficiente. Recordemos que la transformada de Fourier es una integral definida por la ecuación 3, por lo que J.F James tiene un punto muy válido al momento de decir que, en la práctica, el integrando de interés comúnmente estará dado por un conjunto de datos que requerirían una integración individual de cada punto. Por esto se creó la transformada discreta de Fourier. [James, 2011] Entonces, teniendo un vector f que contiene los N puntos muestreados de la señal de interés, la transformada discreta de Fourier está dada por:

$$\hat{f}_n = \sum_{k=0}^{N-1} e^{\frac{-2\pi i n k}{N}}; n = 1, 2, \ldots, N-1 \tag{4}$$

La formulación anterior puede ser expresada de forma matricial, creando así lo que conocemos como la transformada rápida de Fourier.

2.6.2. Transformada rápida de Fourier

La ecuación 4 puede ser pensada como una matriz de un espacio de $N \times N$ de la forma:

$$\hat{f} = A = \begin{bmatrix} A(0) \\ A(1) \\ \vdots \\ A(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & e^{\frac{2\pi i}{N}} & \ldots & e^{\frac{2(N-1)\pi i}{N}} \\ \vdots & \vdots & & \vdots \\ 1 & \ldots & \ldots & e^{\frac{(N-1)^2 2\pi i}{N}} \end{bmatrix} \begin{bmatrix} a(0 \\ )a(1) \\ \vdots \\ a(N-1) \end{bmatrix} \tag{5}$$

Donde a es el vector que contiene los datos de interés(equivalente a f en la ecuación 4). Gracias a esta formulación, se puede simplificar lo suficiente el cálculo como para obtener lo que conocemos como la transformada rápida de Fourier.

El proceso de representar la transformada discreta de Fourier implica tener una matriz cuadrada y un espacio de $N \times N$. Realizar estas operaciones de manera normal implicaría que el total de operaciones a realizar sería de N^2. La transformada rápida de Fourier reduce el total de operaciones a $2n \log_2(N)$. Eso

implica que una transformada rápida de un vector de datos que contiene 10^6 datos requiere de $4.2(10^7)$ operaciones, en lugar de 10^{12}. J.F James describe la esencia de la transformada rápida de Fourier como la factorización de la matriz que contiene los exponenciales para reducir el tiempo que toma computar la operación matricial. Entonces, lo que sucede a continuación es que se buscan 2 números l, k que al multiplicarse sean iguales a N, siempre que $\frac{N}{2n}$ pueda ser factorizado en factores de 2, entonces se repite este procedimiento hasta que solo quedan matrices de 2×2, las cuales son mucho mas fáciles de computar(donde n es el número de la enésima matriz creada con este procedimiento) [James, 2011].Para este procedimiento N debe ser potencia de 2, cuando esto no es así, se rellena el vector de datos con ceros para cumplir con esta condición.

3. Desarrollo

Esta sección se centra en explicar el principio de funcionamiento detrás de el programa creado, está dividida en:

1. Vista general del programa, *sección 3.1*
2. Entendiendo el contexto del problema, *sección 3.2*
3. Explicación del algoritmo, y áreas de oportunidad.

3.1. Vista general del programa

El programa funciona en 3 etapas las cuales se resumen en la *figura 1*:

1. Grabación: En esta etapa se toma una muestra de audio.
2. Comparación: En esta etapa se compara la muestra con la base de datos.
3. Calificación: Esta última fase asigna una calificación numérica mediante una ponderación, con esta se determina cual canción es una mejor coincidencia.

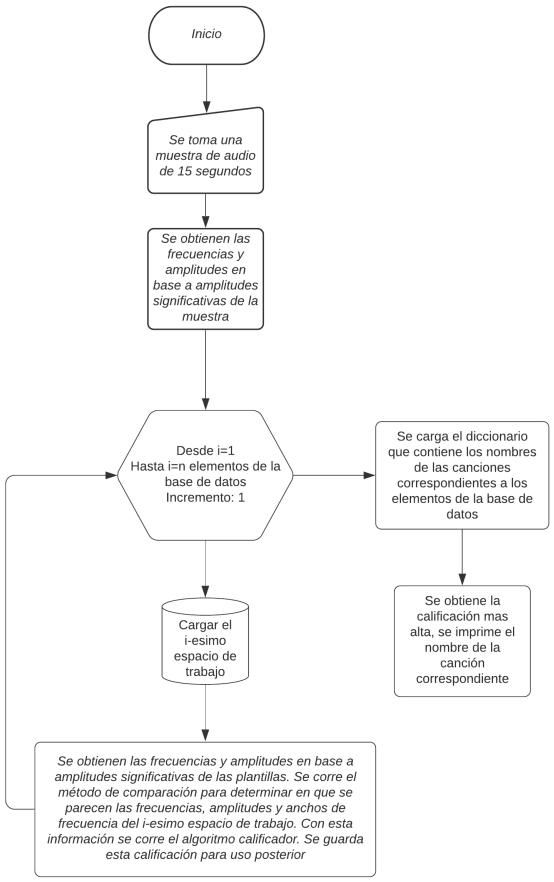

Figura 1: Diagrama de flujo del programa [Creación propia, 2020]

3.2. Entendiendo el contexto del problema

Antes de explicar como crear un programa capaz de reconocer una canción mediante una muestra, es válido preguntarse como es que nuestro cerebro logra el mismo cometido, pues a fin de cuentas, es algo que sucede cotidianamente, pero pocas veces nos cuestionamos como es que sucede. Para saber el nombre de una canción solo con escuchar un fragmento de la misma, lógicamente el primer requisito es haber escuchado total o parcialmente la canción con ante-

rioridad una o más veces, tener presente el nombre de la misma y estar familiarizado con la misma ya sea por la letra, apartado instrumental, o cualquier otra característica que nos permita reconocerla en un futuro al escucharla. Es bajo esta idea que se creó este programa, pues en esencia trabaja similarmente.

Para poder reconocer una canción por medio de una muestra significativa de la misma se requiere:

- La señal que se desea identificar.
- Una muestra representativa de esta misma señal.
- Un algoritmo que permita determinar si la similitudes en las señales son suficientes para poder confirmar que en efecto se trata de la misma señal.

Es bastante sencillo obtener los primeros 2 elementos anteriormente mencionados, el problema radica en en crear el último, pues debe ser un algoritmo robusto, pero eficiente, que sea capaz de diferenciar canciones aún cuando existan parecidos entre las mismas, pero que sea eficiente al momento de computar una respuesta. Como se ha establecido en la *sección 2.3.1*, el espectro audible humano puede llegar hasta los $20kHz$, medir la totalidad del espectro para los propósitos del programa sería innecesario por 2 razones: sería computacionalmente ineficiente, y se estarían tomando frecuencias en un espectro irrelevante para nosotros, pues si bien es cierto que el límite de audición absoluto ronda los $20kHz$, el espectro de mayor sensibilidad es mucho menor, Silvana Serra, Mónica Brizuela y Lorena Baydas mencionan que el rango de mayor sensibilidad se encuentra entre los $500Hz$ y los $800Hz$. [Serra et al., 2018] Aún más importante, mencionan que las notas musicales se encuentran en las frecuencias descritas en la *figura 2*:

Nota	**Frecuencia**
Si	$493.88Hz$
La#	$466.16Hz$
La	$440Hz$
Sol#	$415.3Hz$
Sol	$392Hz$
Fa#	$369.23Hz$
Fa	$349.23Hz$
Mi	$329.63Hz$
Re#	$311.13Hz$
Do#	$277.18Hz$
Do	$261.63Hz$

Figura 2: Frecuencias de las notas musicales, [Serra et al., 2018]

Sabiendo esto, se realizaron varias pruebas de grabaciones cortas de pistas, convirtiéndolas al dominio de frecuencias y amplitudes normalizadas, usando una tasa de muestreo de $40kHz$, pues como se mencionó en la *sección 2.5*, es necesario usar una tasa de muestreo del doble de la señal que se desea reconstruir, en este caso se deseaba reconstruir una señal en todo el espectro audible por humanos. Estas grabaciones arbitrarias tenían el objetivo de determinar un espectro de interés para el análisis, para tener un algoritmo que solo se centrase en frecuencias relevantes, esto reduciría la cantidad de datos a analizar por cada ejecución del programa, lo que se traduce en una eficiencia mayor, pero al hacer esto se debe considerar que recortar un espectro muy grande puede causar que el algoritmo pierda información relevante para computar una respuesta adecuada. La *figura 3* ilustra los resultados obtenidos:

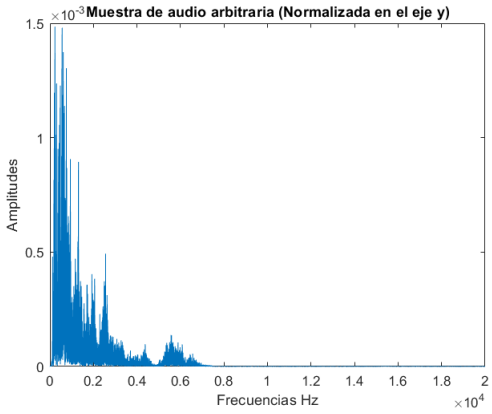

Figura 3: Ejemplo de audio arbitrario muestreado a $40kHz$[Creación propia,2020]

Teniendo esto en cuenta, se tomó la decisión de centrar el estudio en las frecuencias de 0 a $4000Hz$, pues se puede observar que la mayoría de frecuencias se encuentran debajo de los $8000Hz$. A mi consideración, el espectro de entre 4000 y $6000Hz$ es despreciable, ya que la parte significativa de las grabaciones se encuentra aún más abajo(la razón por la cual no se definió $20Hz$ como límite inferior como se definió en la *sección 2.3.1* es por meras limitaciones del audio toolbox de Matlab).

3.3. Explicación del algoritmo, y áreas de oportunidad.

3.3.1. Explicación del algoritmo

El programa funciona de la siguiente manera: se tiene una función que toma una muestra de 15 segundos de audio a una tasa de $8000Hz$ a la cual llamaremos $grabar()$, pues se ha decidido analizar un máximo de $4000Hz$, una función comparadora y otra calificadora que se explicarán mas adelante. Cabe mencionar que la magnitud de las amplitudes de la grabación son normalizadas desde este punto.

La parte más fundamental del programa es una función que se encarga de tomar la grabación deseada(ya sea la muestra o las canciones disponibles en la base de datos), obtener sus datos en forma numérica, aplicar la transformada rápida de Fourier a la misma pasando de un dominio de decibeles a tiempo a uno de frecuencias y amplitudes(tomando el valor absoluto de las últimas, pues estas son números complejos), guardando esta información en los vectores frecuencias y amplitudes, nos referiremos a esta función como $scanner()$. Con esta información, se utiliza la función *findpeaks* de Matlab con el parámetro $minPeakHeight = 2std(amplitudes) + mean(amplitudes)$, esto por que no sería eficiente obtener todos los picos en el dominio de frecuencias y amplitudes, además Avery Wang, quien es uno de los creadores de Shazam, menciona que para el algoritmo que ellos utilizan se centra en crear lo que el describe como una huella digital para la canción, parte de esta huella digital son características que la definen, por ejemplo un punto frecuencia/tiempo. Su algoritmo solo analiza puntos frecuencia/tiempo candidatos, y uno de los criterios que usan para determinar si un punto es candidato o no es que este tenga una amplitud mucho mayor que las frecuencias vecinas [Wang et al., 2003]. El proceso que Wang describe es posible gracias a que Shazam realiza su procesamiento en tiempo real, pero como este no es el caso para este programa se tomó la decisión de crear este parámetro adaptativo definido como 2 desviaciones estándar mas la media de las amplitudes detectadas. Esta métrica no es arbitraria, tras varias pruebas fue la que demostró entregar mejores resultados sin tardar mucho en entregar una respuesta. De esta manera, cada vez que una muestra es grabada, se crea una métrica personalizada en función de las características de la misma, volviéndolo un proceso adaptativo. Tras aplicar la función *findpeaks*, se obtienen vectores que contienen las frecuencias donde suceden amplitudes significativas, la magnitud de las amplitudes y los anchos de las mismas.

En cuanto a la función de comparación, esta funciona con la información obtenida en la función descrita anteriormente, esta función recibe el vector de frecuencias significativas de la muestra, así como de alguna de las canciones disponibles en la base de datos, y mediante un ciclo anidado se determina si hay frecuencias que estas compartan, si así lo es se agregan a un vector que guarda registro de las coincidencias el cual llamaremos $matches$, y en otro vector

se guarda la posición de las coincidencias en el vector de frecuencias de la canción, al que llamaremos $matchWhere$.

La función calificadora a la cual llamaremos $grader()$, toma los vectores $matches, matchWhere$, así como los vectores de ancho, frecuencias y amplitudes tanto de la muestra como de una canción. Usando esta información, se obtiene el la diferencia entre las amplitudes correspondientes a las frecuencias que tanto la canción como la muestra comparten, y se hace lo mismo con los anchos (cada uno de estos resultados, es decir cada error individual en amplitudes y en anchos es dividido entre la longitud del vector de frecuencias significativas de la canción), llamaremos a estos vectores e y w respectivamente. Se crea una constante definida como la longitud del vector de frecuencias que comparten ambas canciones dividido entre la longitud del vector de las frecuencias de la canción, el cual llamaremos a. Finalmente, la calificación final, la cual representa similitud entre la canción a la que corresponden los vectores recibidos, y la muestra se define como:

$$100a + 5(1 - sum(e)(10)^7) + 5(1 - sum(w)(10)^5) \quad (6)$$

En cuanto a las canciones disponibles en la base de datos, estas fueron obtenidas usando una versión modificada del programa que solo contiene la función $grabar()$, adaptando el parámetro de tiempo de muestreo a la duración de la canción en cuestión. Posteriormente, se guardó el espacio de trabajo entero, para su posterior invocación al momento de comparar.

Explicado esto, a continuación se describe el proceso a detalle por el cual pasa el programa:

1. Se corre el programa, al momento de correrlo se carga un archivo CSV con el nombre de las canciones que el programa es capaz de reconocer, así como el nombre del archivo que contiene la información de cada canción, esta información se carga a vectores de strings que llamaremos $names$ y $files$ respectivamente.

2. El programa invoca $grabar()$ y toma la muestra de audio mencionada, y la pasa por la función que obtiene sus frecuencias significativas, amplitudes y anchos.

3. Se crea un vector llamado resultados, este guardará información que será útil más adelante.

4. Se entra a un ciclo for el cual va de 1 a n siendo n el número de canciones disponibles para analizar, en el cual basándose en el elemento n de $files$, carga el archivo correspondiente, y corre $scanner()$, para posteriormente ejecutar $grader()$, pasando la información requerida, se asigna el resultado obtenido en $grader()$ al elemento n de $resultados$.

5. Terminado el ciclo anterior, se obtiene el elemento más alto de $resultados$(no se hace nada con esta información pero por limitaciones de Matlab es necesario obtenerla), así como la posición del mismo, la cual llamaremos $index$.

6. Se despliega el siguiente mensaje: La canción analizada debe ser; seguido del elemento $index$ del vector $names$.

3.3.2. Áreas de oportunidad

A pesar de que este programa ha mostrado ser bastante efectivo, tiene ciertas limitaciones, estas se presentarán a continuación:

- El tiempo de ejecución del programa crece con el tamaño de la base de datos. Utilizando la herramienta *Run and time* se determinó que para analizar 4 canciones toma alrededor de 40 segundos en computar una respuesta (tiempo de grabación incluido). La *figura 4* muestra el tiempo de ejecución según el número de canciones que se permitió cargar:

Canciones	Tiempo de ejecución (s)
1	18.7
2	21.5
3	27.5
4	34.7
5	37.1

Figura 4: Tiempos de ejecución [Creación propia,2020]

Las canciones anteriores son: It's Called: Free Fall, Rainbow Kitten Surprise; Los Malaventurados No Lloran, PXNDX; Cancer, My Chemical Romance; Baba Yetu, Christopher Tin; Everybody Wants to Rule The World, Tears For Fears, todas con duraciones distintas, por lo que la información a analizar varía no solo por número de canciones, también lo hace por el contenido de las mismas.

Si bien existen pequeñas diferencias en los tiempos de ejecución que pueden atribuirse a cosas que no son el enfoque del proyecto, como la carga que sufría el ordenador en el momento de la ejecución, existe evidencia para suponer que el crecimiento en el tiempo de ejecución es aproximadamente lineal respecto al número de canciones, lo cual haría totalmente impráctico implementar este mismo algoritmo a gran escala.

- Una debilidad del programa importante, es que determina sus resultados en función de la mejor coincidencia, el problema es que si se llegara a muestrear una canción fuera de la base de datos, el hecho de que no exista la canción en la base de datos no implicaría que no se de ningún resultado, se correría el algoritmo normalmente, y se devolvería el resultado más cercano. Por lo cual un área de oportunidad es implementar un método que sepa manejar este caso y distinguir una buena coincidencia de una que no tiene nada que ver con los datos disponibles.
- Ligado al punto anterior, el algoritmo actual depende de una métrica numérica, misma que puede ser engañada dadas las condiciones suficientes, forzando al programa a arrojar resultados incorrectos, por ejemplo pensando en un caso extremo podría darse el caso de una muestra que arroje solo una frecuencia significativa, cuyo error en amplitud y ancho sea pequeño al compararlo con una canción. En ese improbable pero posible caso se podría entregar un resultado incorrecto en un sample que no tenga nada que ver con ninguna de las canciones.
- El tiempo de muestreo no es muy alto, pero en teoría podría ser más bajo, sin embargo las pruebas efectuadas mostraron que un tiempo de muestreo menor arrojaba resultados poco precisos, por lo que el algoritmo actual requiere el tiempo designado actualmente. Lograr que el programa entregara los resultados con un tiempo de muestreo menor reduciría el tiempo de ejecución total. Otro problema que un tiempo de muestreo alto puede causar es que, para el momento en el que termina la ejecución, se está reproduciendo otra canción que afectará la lectura deseada.

4. Demostración

Se ha tomado un sample de 15 segundos de audio correspondiente a la canción: Los Malaventurados No Lloran, en condiciones normales, es decir, niveles de ruido normales pero aceptables, con un volumen adecuado, sin interrupciones en la canción. El análisis de frecuencias obtenido fue el mostrado en la *figura 5*.

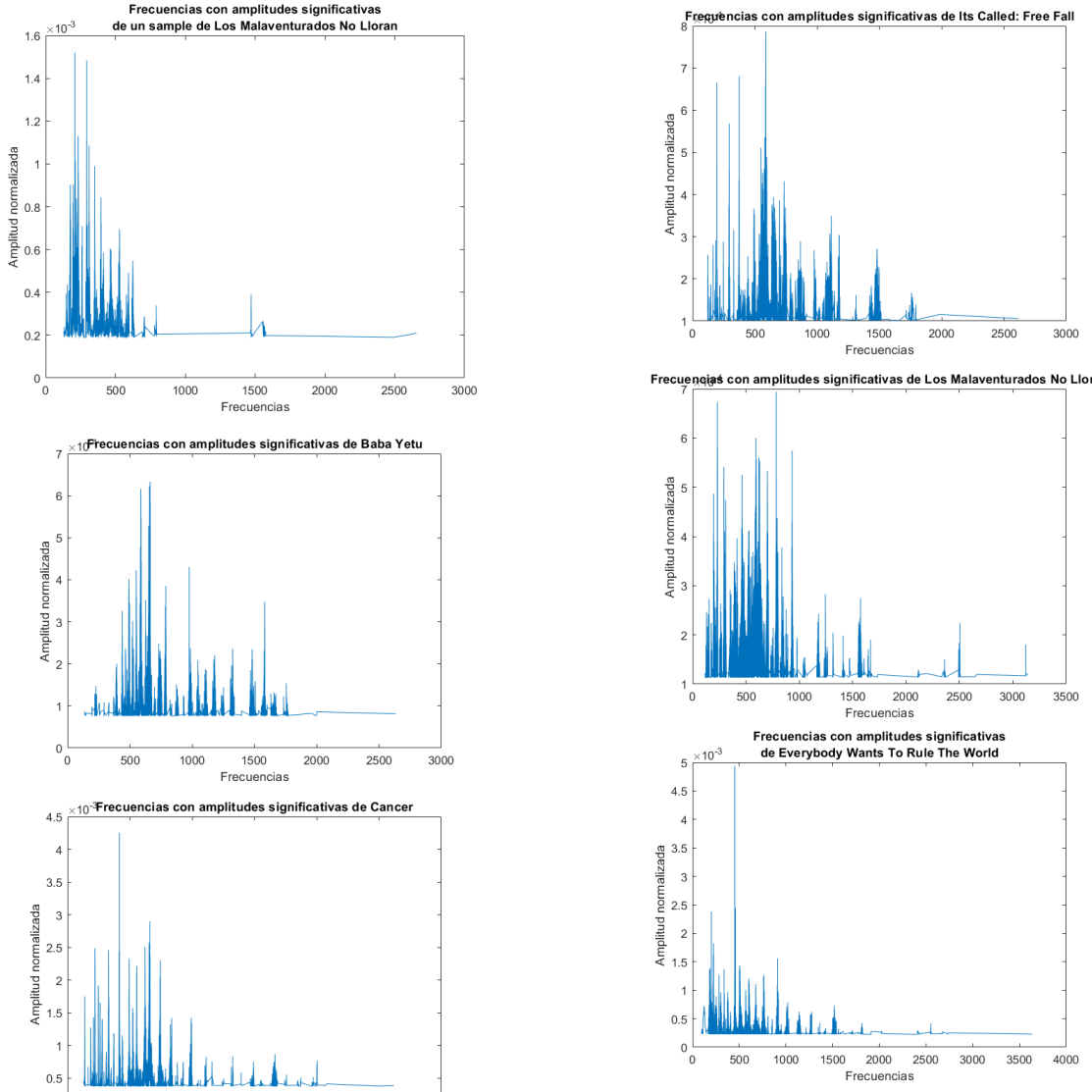

Figura 5: Análisis de frecuencias [Creación propia, 2020]

El programa fue capaz de identificar la canción sin problema, retornando el mensaje de éxito mostrado en la *figura 6*. Este tiene una precisión bastante buena, fallando solo en casos de ruido extremo, o casos atípicos, tales como que el sample sea tomado en los últimos 5 segundos de una canción, y coincida con los primeros 10 de otra que no es la de interés. Sin dudas, se han alcanzado los resultados deseados con este programa, aprovechando las herramientas computacionales disponibles como Matlab, así como algoritmos de creación propia, teoremas que han demostrado su función histórica como el teorema de Nyquist-Shannon, y métodos como la transformada rápida de

Fourier.

Figura 6: Resultado de la ejecución del programa [Creación propia,2020]

5. Conclusión

El presente trabajo ha sido sin dudas satisfactorio, me abrió los ojos en cuanto a la importancia de la eficiencia de un algoritmo, pues se ha logrado imitar la función de software comercial de reconocimiento de audio en cuestión de unos cuantos días, y en el proceso aprendí cosas que no esperaba, como la razón por la cual no se pueden usar transformadas continuas en problemas de la vida real, la importancia de las tasas de muestreo en el muestreo digital y como estas afectan de manera directa la eficacia de un programa. Además, comprendí el por que es natural que a partir de conceptos como la transformada de Fourier, nazca la necesidad de crear su versión discreta, y como al tratar de implementarla en soluciones del día a día nos vimos en la necesidad de crear una versión más eficiente de la misma; la transformada rápida de Fourier.

Antes de empezar este proyecto consideraba el análisis de señales como un tema casi inalcanzable y bastante complejo, que debería dejarse únicamente a personas que dedican toda su vida a las mismas, pero gracias a este trabajo ahora lo veo como algo alcanzable, y muy importante, pues está detrás de muchas cosas cotidianas, tales como cancelación de ruido en micrófonos, compresión de imágenes, escritura por voz, etc.

Referencias

[James, 2011] James, J. F. (2011). *A Student's Guide to Fourier Transforms : With Applications in Physics and Engineering.*, volume 3rd ed. Cambridge University Press.

[Mendoza, 2016] Mendoza, L. J. M. (2016). Teorema del muestreo. *Departamento de Maestría DICIS-UG*.

[Olmos Herrera, 2020] Olmos Herrera, S. (2020). *Estudio y análisis de circuitos analógicos de ecualización y distorsión de señal de audio.* PhD thesis, Universidad Politécnica de Valencia.

[Oppenheim et al., 1998] Oppenheim, A. V., Willsky, A. S., and Nawab, S. H. (1998). *Señales y sistemas.* Pearson Educación.

[Sánchez,] Sánchez, L. P. Ondas sonoras.

[Serra et al., 2018] Serra, S., Brizuela, M., and Baydas, L. (2018). *Manual de la audición.* Colección Fonoaudiología. Editorial Brujas.

[Serway et al., 2007] Serway, R., Jewett, J., Olguín, V., and Rosas, M. (2007). *Fisica: Para ciencias e ingeniera V.1.* Cengage Learning Editores, S.A. de C.V.

[Wang et al., 2003] Wang, A. et al. (2003). An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC.