Instituto Tecnológico de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias
ITESM Campus Monterrey

¿Cuál es el nombre de esa canción?

F1009 – Análisis de métodos matemáticos para la física

Autores:

Myrna Dinorah Ortiz Ramirez A01383998
Diana Paola Cadena Nito A01197399
Mariana Robles Zaldívar A01570482
Fernanda Córdova Ramírez A00830674
Lester Garcia Zavala A01721128

Profesor:

Dr. Gerardo Ramón Fox

4 de diciembre de 2021

I. INTRODUCCIÓN

Actualmente, las aplicaciones para identificar el nombre de alguna canción se han vuelto muy populares. Un ejemplo claro de esto es *Shazam*, el cual utiliza un algoritmo muy particular para identificar canciones, en donde se aplica la *transformada de Fourier rápida*.

Lo que se realizó en nuestro caso para crear un algoritmo similiar, fue lo siguiente. Primero, se descargaron algunas canciones y fueron añadidas a nuestra base de datos para tener canciones "base" que pudiéramos identificar con el algoritmo. Al haber descargado algunas canciones realizamos dos archivos *.mlx* para el reconocimiento de la canción, uno que trabaja con el audio y el reconocimiento, mientras que el segundo archivo es referente a la base de datos como tal, al cual le agregamos todas las canciones que descargamos.

A continuación, se demuestra el funcionamiento del código realizado, así como una explicación del mismo. Se resalta que el siguiente algoritmo tiene la capacidad de replicar lo que aplicaciones como *Shazam* realizan; identificar canciones.

II. CÓDIGO

A. Explicación

1. *Archivo 'BASE DD.mlx'*

En este archivo, se procesan las canciones que se encuentren en la biblioteca; estas tienen terminación *.wav* y se busca hacer un archivo *.mat* para realizar el proceso de reconocimiento y demás (más adelante se detallará ésto).

Cada canción dentro de la biblioteca es procesado individualmente. Es decir, para añadir una canción dentro de la biblioteca, se tiene que realizar esta añadidura manualmente. Uno se tiene que egurar que el archivo de la nueva canción a añadir esté en formato .wav y además que se encuentre dentro de la carpeta donde se almacena toda la información base (un audio en formato .wav y su respectivo conjunto de cantidad de máximos y valores frecuenciales por segundo). Dentro de este subcódigo se tendrán que cambiar la entrada manualmente para audioread() con el nombre de la canción nueva a añadir. Ejemplo: audioread(Reto-fismat(SI)-base NOMBRECANCION.wav) Con ello se podrá realizar su procesamiento de FFT, normalización y obtención de máximos para luego capturar toda la información relevante a un conjunto (matriz). Más adelante se explayará en este proceso, el cual se tiene que realizar para el audio a analizar grabado por el usuario.Finalmente, se captura este conjunto como archivo .mat donde se tiene la información base anteriormente mencionada.En este último paso también se tendrá que manipular manualmente en el caso de agregar una nueva canción, como se mencionaba para la función audioread(). Ejemplo: 'peaks-NOMBRECANCION .mat' Notas: se recomienda poner cada espacio del nombre de la nueva canción a añadir con guiones bajos. De igual manera, en el código principal solo se tendrá que duplicar el proceso que se observa para las canciones ya preestablecidas (su lectura, renombre, porcentaje de similitud con el audio del usuario y comparaciones a mayor porcentaje de similitud).Estos últimos pasos serán explicados posteriormente.

La primera parte de este archivo es simplemente la lectura del audio disponible.

```
clc ; clear all; close all
[cancion,Fs2] = audioread('Reto Fismat (SI)\canciones_base\So_High.wav');
info = audioinfo('Reto Fismat (SI)\canciones_base\So_High.wav');

t = floor(info.Duration)
```

FIG. 1: Canción en la biblioteca

+ mono + huella digital

Por último, se crea el archivo *.mat* en la dirección establecida .

```
% Archivo matlab
mat_file = matfile('peaks_So_High.mat','Writable',true);
save('peaks_So_High.mat',"tabla_mono");
movefile('peaks_So_High.mat','Reto Fismat (SI)\canciones_base\');
```

FIG. 2: Canción en la biblioteca

2. *Archivo 'test fismat.mlx'*

Este archivo se divide en dos secciones principales: la primera graba el audio y la segunda hace todo el proceso de reconocimiento. La grabación del audio tiene una duración de 10 segundos ya que, de acuerdo a lo investigado en la primera etapa, este es el tiempo promedio que se utiliza en el algoritmo de *Shazam*.

```
1  clc ; clear all ; close all
2
3  tg = 10;
4  Fs = 48000;
5
6  audio = audiorecorder(Fs,16,1);
7  disp ('Inicia grabación.')
8  recordblocking(audio,tg);
9  disp('Fin de grabación.')
10 grabacion = getaudiodata(audio,'single');
11 audiowrite('Reto Fismat (SI)\audio1.wav',grabacion,Fs)
```

FIG. 3: Grabado de audio.

La variable *tg* guarda el tiempo de la grabación y la variable *Fs* es la frecuencia del muestreo. Se estableció esta frecuencia debido a que otorga una mejor calidad de audio. En la línea 6, MATLAB empieza a grabar bajo el parámetro de la frecuencia previamente establecido así como los bits y el canal (canal el cual es mono para facilitar el proceso *investigar para argumentar).

En la línea 8, se delimita la grabación de acuerdo a los 10 segundos y en la línea 10 se leen los datos de dicha grabación. Por último, se escribe y se guarda el audio en formato *.wav* en la carpeta del proyecto, respetando nuevamente el parámetro de la frecuencia.

En cuanto a la segunda sección de este archivo, esta es más extensa puesto a que contiene el algoritmo de reconocimiento en sí y se puede dividir en 4 fases. En la primera fase, se recupera al audio grabado y se reproduce con fines de comprobar que sí es ese el audio que se quiere analizar.

```
12  clc ; clear all; close all
13
14  [audioG,Fs3] = audioread('Reto Fismat (SI)\audio1.wav');
15  info = audioinfo('Reto Fismat (SI)\audio1.wav');
16  sound(audioG, Fs3)
17  Fs1 = 48000;
```

FIG. 4: Recuperación de audio.

Asimismo, se realizan 3 gráficas distintas respecto al audio: la primera grafica tal y como se grabó el audio, la segunda gráfica el audio con la Transformada Rápida de Fourier aplicada y la tercera gráfica; el audio ya con la normalización de la transformada.

```
19  figure(1)
20  subplot(3,1,1);
21  t = 0:seconds(1/Fs1):seconds(info.Duration);
22  t = t(1:end-1);
23  plot(t,audioG);
24  title('Audio a analizar: Dominio del Tiempo');
25  grid on
26  xlabel('Tiempo') ; ylabel('Amplitud')
```

FIG. 5: Gráfica del audio puro

```
28  audioFFT = abs(fft(audioG));
29  subplot(3,1,2);
30  plot(audioFFT);
31  title('Audio a analizar: Dominio Frecuencial')
32  grid on
33  xlabel('Frecuencia') ; ylabel('Amplitud')
```

FIG. 6: Gráfica del audio con FFT

```
35  audioFFT = audioFFT(1:2500);
36  audioFFT = audioFFT/max(audioFFT);
37  subplot(3,1,3);
38  plot(audioFFT);
39  title('Audio FFT normalizado')
40  grid on
41  xlabel('Frecuencia') ; ylabel('Amplitud normalizada')
```

FIG. 7: Gráfica del audio con FFT normalizado

La fase 2 involucra todo lo que es el analisis de el *hash function* en donde evaluamos por intervalos de 1 segundo lo que se grabó anteriormente dentro de un tramo de 10 segundos. Después, realizamos el mismo proceso realizado en la fase 1 pero ahora lo realizamos para cada intervalo de 1 segundo. Al ya haber realizado esto, se identifican los puntos más altos. Esto quiere decir que se toman solamente las partes con más fuerza o volumen de la canción para poder buscar canciones con los mismos datos de frecuencia y fuerza/volumen en la base de datos. Dichos puntos se obtienen utilizando una función llamada *findpeaks*. Al haber obtenido los puntos se crea un conjunto en donde añadimos los segundos, la cantidad de máximos y por último, la frecuencia del primer máximo hasta el último máximo.

En cuanto a la fase 3, éste se basa en acomodar los datos de tal forma que se grafiquen los datos en nuestro *audio muestra* de forma correcta. Es decir este paso se basa en crear nuestra huella digital de una manera correcta para identificar la canción que se esta buscando.

Por último, la fase 4 simplemente es realizar las comparaciones entre el *audio muestra* y las canciones disponibles en la base de datos. Las comparaciones se basan en corroborar si el *audio muestra* es un subconjunto del conjunto de las canciones de la biblioteca de 0.001. Esto es porque se debe de considerar que la muestra y la canción en la base de datos no van a ser totalmente iguales, si no, destacablemente similares. El resultado de las comparaciones se lee en porcentajes. Entonces, la comparación que obtenga un mayor porcentaje, será la canción que se busca reconocer.

```
154 comparacion1 = mean(mean(ismembertol(muestra,cancion1,0.0001))); % So High
155 comparacion2 = mean(mean(ismembertol(muestra,cancion2,0.0001))); % i want you
156 comparacion3 = mean(mean(ismembertol(muestra,cancion3,0.0001))); % the love that bind us
157
158 if comparacion1 > comparacion2 && comparacion1 > comparacion3
159     disp('La canción es "So High" de Doja Cat')
160 elseif comparacion2 > comparacion1 && comparacion2 > comparacion3
161     disp('La canción es "I want you" de Mitski')
162 elseif comparacion3 > comparacion1 && comparacion3 > comparacion2
163     disp('La canción es "The Love that Bind Us" del soundtrack la serie Violet Evergarden')
164 end
```

FIG. 8: Comparación y reconocmiento final

B. Validación de funcionamiento

A continuación, se muestra el proceso de reconocimiento de una de las canciones que se encuentra en la base de datos. Esto con la finalidad de proporcionar evidencia de que el código funciona como es debido.

1. Se incia el código y se comienza con la grabación del audio

```
Inicia grabación.

Fin de grabación.
```

FIG. 9: Grabado de audio

2. Se muestra las gráficas correspondientes a la fase 1 explicada en la sección anterior.

3. Se muestra la gráfica correspondiente a la fase 2 explicada en la sección anterior.

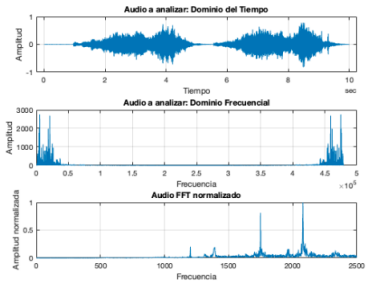

FIG. 10: Gráficas del audio

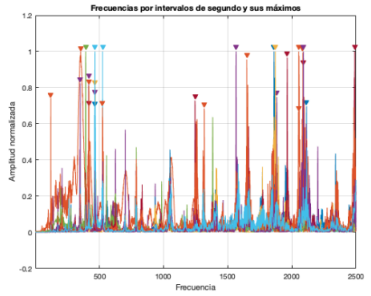

FIG. 11: Gráfica de frecuencias

4. Se muestra la gráfica correspondiente a la fase 3 explicada en la sección anterior.

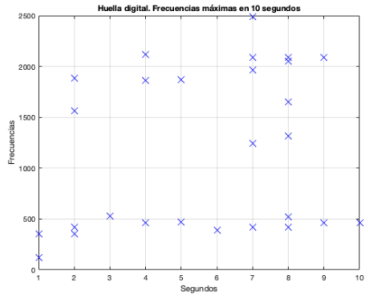

FIG. 12: Huella digital

5. Se muestra el resultado; es decir, se identifica la canción.

```
La canción es "The Love that Bind Us" del soundtrack la serie Violet Evergarden
```

FIG. 13: Reconocimiento de la canción

III. LIMITACIONES Y VENTAJAS

Una de las principales limitaciones de nuestro código es referente al tamaño de la base de datos. Para ser un proyecto, se demuestra el funcionamiento correcto, pero, si se buscara aplicarlo a mayor escala, se tendría que optimizar la manera de hacer los archivos *.mat*.

De igual manera, el proceso de reconocimiento acaba arrojando el porcentaje mayor, pero no identifica si no se encuentran canciones en la base de datos que correspondan a la canción. Es decir, el programa, sea como sea, acaba arrojando un resultado aún y cuando puede que éste no sea verídico.

También una de las desventajas que tenemos, es el reconocimiento del ruido blanco. Éste es una señal aleatoria que se caracteriza por sus valores de señal en un periodo de tiempo, en donde no guarda una correlación estadística. Y aunque se puede separar el ruido blanco del programa, tenemos otros tipos de ruidos (aleatorios) que al momento de correr nuestro programa, se podrían detectar. De esta manera, se pueden añadir sonidos fuertes no deseados al reconocimiento, como una puerta azotandose o un grito. Esto presenta a que el algoritmo no fucnione adecuadamente, ya que los ruidos fuertes aleatories y erráticos son tomados en cuenta, cuando realmente no tienen nada que ver con la canción que se busca.

Una de las ventajes que presenta este código es su funcionamiento. A través de una ingeniosa aplicación de la transformada de Fourier rápida, se puede convertir una señal sonóra análoga en un espectro discreto de frecuencias que sirve como base, para compararse con otro espectro previamente guardado. Es decir, la aplicación de la FFT (La transformada rápida de Fourier) ayuda, de manera muy práctica, a identificar canciones.Este algoritmo es muy eficiente para toda aquella persona interesada en el reconocimiento de canciones.

Otra de las ventajas es la facilidad de identificar una canción ya que solamente se necesitan 10 segundos de la canción para que el algoritmo lo pueda identificar.