[[1]](#footnote-1)

Eider Triviño Trejos. Author

eider.trivivo@utp.edu.co

Voice Identifier

Identificador De Voz

***Abstract*— In the following document we want to propose the operation of a program that recognizes different voices, takes advantage of the microphone to be able to record an audio sample that is being reproduced. An acoustic fingerprint is created from the sample and compared to a list of audios in a predefined database to find matches. Once the relationship is made, the user can receive information such as the name of the person to whom the voice belongs.**

***Resumen*—En el siguiente documento se desea plantear el funcionamiento de un programa que reconozca diferentes voces, aprovecha el micrófono para poder grabar una muestra de audio que se esté reproduciendo. Una huella digital acústica se crea a partir de la muestra y se compara con una lista de audios en una base de datos predefinida para encontrar coincidencias. Una vez hecha la relación, el usuario puede recibir información tal como el nombre de la persona a la cual le pertenece la voz.**

***Palabras claves: shazam, Fourier, Musica, Transformada, Huella***

***keywords: shazam, Fourier, Music, Transformed, Footprint***

# Algoritmo

Oyes una voz familiar. ¡Has escuchado esta voz miles de veces, pero no recuerdas su nombre! Afortunadamente, en nuestro increíble mundo futurista, tienes un teléfono con software de reconocimiento de voz instalado. Puedes relajarte, ya que el Software te dijo el nombre de la persona.

**Grabando - Capturando el Sonido:**

Grabar una señal de audio muestreado es fácil. Ya que las tarjetas de sonido modernas ya vienen con convertidores analógicos-digitales, sólo tienes que seleccionar un lenguaje de programación, encontrar una biblioteca adecuada y establecer la frecuencia de la muestra, el número de canales (mono o estéreo) y normalmente, el tamaño de la muestra (por ej. Muestras de 16 bits). Después abre la línea de tu tarjeta de sonido, al igual que cualquier flujo de entrada, y escribe en una matriz de bytes. Así es cómo puedes hacerla en Python:



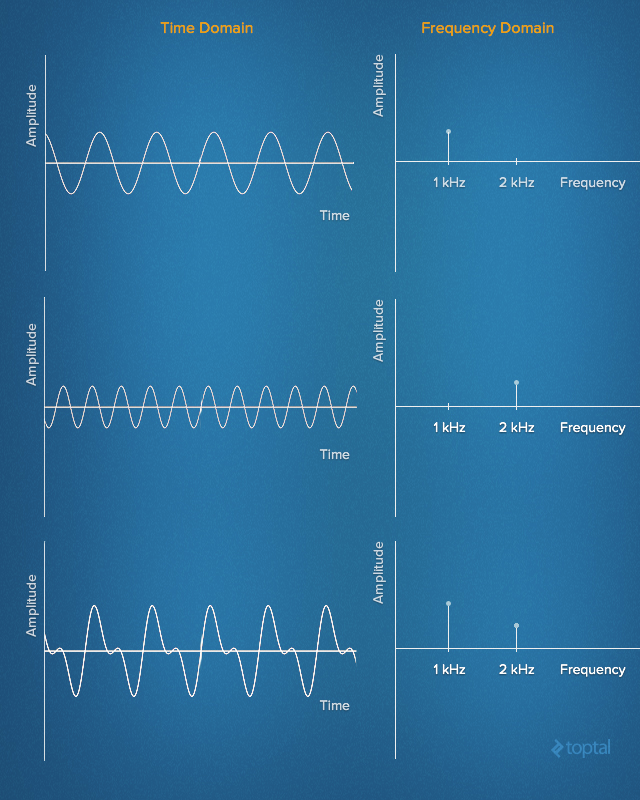


**Dominio de Tiempo y Dominio de Frecuencia:**

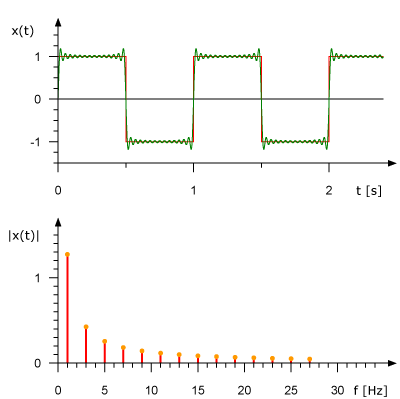
Lo que tenemos en esta matriz de bytes es la señal grabada en el tiempo del dominio. La señal de dominio de tiempo representa el cambio de amplitud de la señal a lo largo del tiempo.

En las primeras décadas del siglo XIX, Jean-Baptiste Joseph Fourier hizo el notable descubrimiento que ninguna señal en el dominio del tiempo es equivalente a la suma de algunos (posiblemente infinito) números de simples señales sinusoidales, dado que cada componente tiene una cierta sinusoidal de frecuencia, amplitud y fase. La serie de las sinusoides que juntos forman la señal de dominio de tiempo original se conoce como serie de Fourier.

En otras palabras, es posible representar cualquier señal en el dominio del tiempo simplemente dando el conjunto de frecuencias, amplitudes y fases correspondientes a cada sinusoide que compone la señal. Esta representación de la señal es conocida como la frecuencia de dominio. En cierto modo, el dominio de la frecuencia actúa como un tipo de huella dactilar o firma para la señal de dominio de tiempo, proporcionando una representación estática de una señal dinámica.



La siguiente imagen muestra la serie de Fourier de una onda cuadrada de 1 HZ y cómo una onda cuadrada (aproximada) puede ser generada a partir de componentes sinusoidales. La señal se muestra en el dominio del tiempo anterior y en el dominio de la frecuencia a continuación.



Analizar una señal en el dominio de la frecuencia simplifica muchas cosas inmensamente. Es más cómodo en el mundo del procesamiento de señal digital porque el ingeniero puede estudiar el espectro (la representación de la señal en el dominio de la frecuencia) y determinar las frecuencias que están presentes y las que faltan. Después de eso, uno puede hacer el filtrado, aumentar o disminuir algunas frecuencias, o simplemente reconocer el tono exacto de las frecuencias.

**La transformada discreta de Fourier:**

Por lo tanto, necesitamos encontrar una manera de convertir nuestra señal desde el momento de dominio para el dominio de la frecuencia. Aquí lo llamamos la transformación discreta de Fourier o Discrete Fourier Transform (DFT). La DFT es un método matemático para realizar Análisis de Fourier en una muestra de la señal. Convierte una lista finita de muestras equidistantes de una función en la lista de los coeficientes de una combinación finita de complejas sinusoides, ordenadas por sus frecuencias, considerando si las sinusoides habían sido muestreadas en la misma proporción.

Uno de los algoritmos numéricos más populares para el cálculo de la DFT es la **Fast Fourier transform (FFT)**. El más utilizado es la variación de FFT Cooley–Tukey algorithm. Este algoritmo es del tipo divide-y-vencerás que recursivamente divide una DFT en varias y pequeñas DFTs. Mientras que la evaluación de un DFT requiere directamente O(n2) operaciones, con un Cooley-Tukey FFT el mismo se computará en iO (n log n) operaciones.

No es difícil encontrar una biblioteca adecuada para la FFT. Aquí están algunos de ellos:

C – FFTW

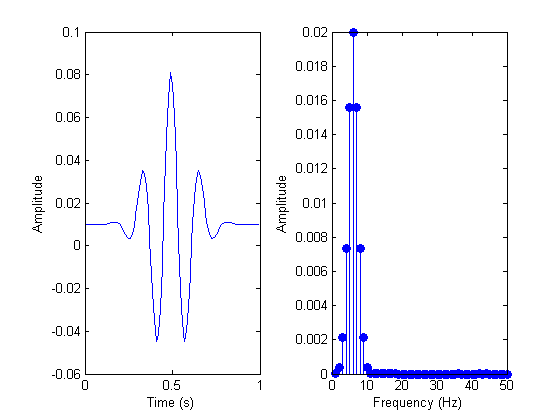
C++ – EigenFFT

Java – JTransform

Python – NumPy

Ruby – Ruby-FFTW3 (Interface to FFTW)

Aquí está un ejemplo de una señal antes y después del análisis FFT:



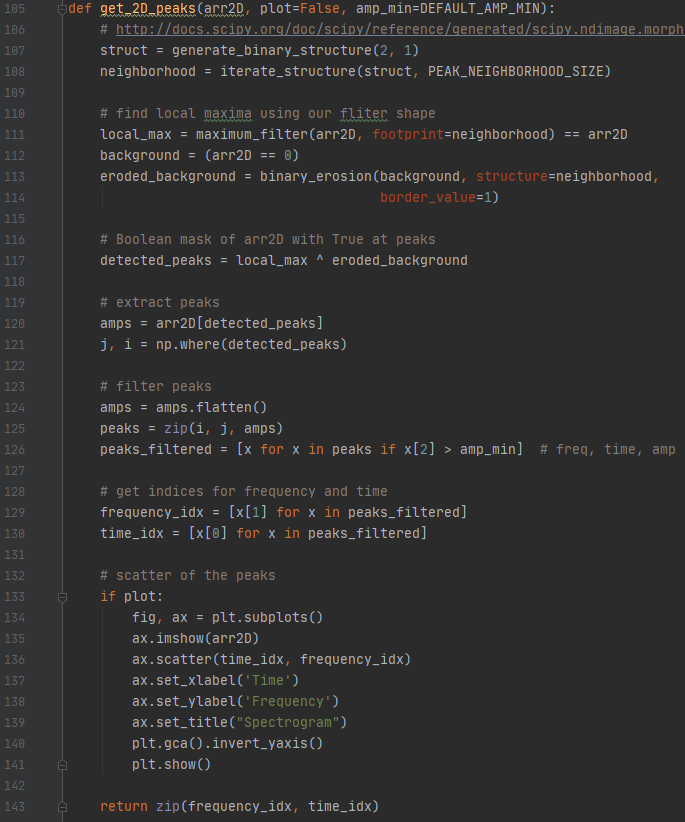
**Reconocimiento de Audios: Las Huellas Digitales en Audios.**

Un desafortunado efecto colateral de FFT es que perdemos una gran cantidad de información acerca de la sincronización (aunque teóricamente esto puede ser evitado, el rendimiento de gastos generales es enorme) para un audio de 3 minutos, podemos ver todas las frecuencias y sus magnitudes, pero no tenemos una pista cuando aparece en el audio. ¡Pero esta es la clave de la información que hace que el audio sea así! De alguna manera tenemos que saber en qué momento aparece cada frecuencia.

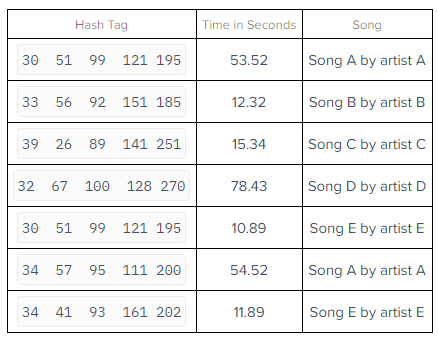
Es por eso que introducimos el tipo de ventana deslizante, o fragmento de datos, así como la transformación de esta parte de la información. El tamaño de cada fragmento puede determinarse de distintas formas. Por ejemplo, si queremos grabar el sonido en estéreo, con muestras de 16 bits, a 44.100 Hz, un segundo de ese sonido será de 44.100 muestras \* 2 bytes \* 2 canales ≈ 176 kB. Si cogemos 4 kB para el tamaño de un segmento, tendremos 44 porciones de datos para analizar en cada segundo del audio. Esa es una buena densidad suficiente para el análisis detallado.

Una vez que tengamos la información acerca de la frecuencia de la señal, podemos empezar a formar nuestra huella digital de audio. Esta es la parte más importante de todo el proceso de reconocimiento de audios de un algoritmo. El principal desafío es cómo distinguir, en el océano de las frecuencias capturadas, las frecuencias que son las más importantes. Intuitivamente, podemos buscar las frecuencias con mayor magnitud (comúnmente llamadas picos).

Ahora, dentro de cada intervalo podemos determinar la frecuencia con la mayor magnitud. Esta información constituye una firma para esta parte del audio y esta firma se convierte en parte de la huella digital del sonido como un todo.



Para facilitar la búsqueda, esta firma se convierte en la clave en una tabla de hashtag. El valor correspondiente es el tiempo en el que este conjunto de frecuencias apareció en el audio, junto con el ID de cada audio (título del sonido y nombre del locutor). Aquí está un ejemplo de cómo pueden aparecer estos registros en la base de datos.



Si ejecutamos toda una biblioteca de audios a través de este proceso, podemos construir una base de datos completa con una huella digital de cada Audio en la biblioteca.

**Eligiendo una grabación:**

Para identificar un audio que se está reproduciendo actualmente, graba el audio en tu teléfono y ejecuta la grabación mediante el mismo proceso de rastreo digital como se hizo anteriormente. Entonces puedes iniciar la búsqueda coincidente en la base de datos de hashtags.

Como sucede, muchos de los hashtags corresponden a varios audios. Por ejemplo, puede ser que alguna pieza de un audio suena exactamente como algún fragmento de otro audio E. Por supuesto, esto no es sorprendente algunas frecuencias en ciertas palabras son muy parecidas entre sí. Cada vez que hemos partido de un hashtag, el número de coincidencias posibles se hace más pequeño, pero es probable que esta información por sí sola no va a reducir el match a un solo audio. Así que hay una cosa más que tenemos que verificar con nuestro algoritmo de reconocimiento de audios, y esa es la distribución.

La muestra que se registró en el micrófono puede ser cualquier punto del audio, así que simplemente no podemos coincidir con el Timestamp del hashtag coincidente con la marca de nuestra muestra. Sin embargo, con varios algoritmos de hashtags que coinciden, podemos analizar el tiempo relativo de la combinación, y, por tanto, aumentar nuestra seguridad.

Por ejemplo, si buscas en la tabla de arriba, verás que el hashtag ‘30 51 99 121 195” corresponde a tanto al audio A, como E. Si un segundo más tarde hacemos coincidir el hash ‘34 57 95 111 200”, que es más una coincidencia para un audio, pero en este caso sabemos que tanto los hashes coinciden y las diferencias horarias (diferencias entre frecuencias a través del tiempo) también.

Bibliografía

<https://matplotlib.org>

<https://pypi.org/project/termcolor/>

<https://www.scipy.org>

<https://pypi.org/project/PyAudio/>

1. This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported in part by the U.S. Department of Commerce under Grant BS123456.”

   The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

   S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

   T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp). [↑](#footnote-ref-1)