**Proyecto Shazam**

Laura González y Dafne Castellanos

Procesamiento de Señales – Universidad del Rosario

Contenido

[Orden y análisis de funcionamiento de los archivos 2](#_Toc132134367)

[1. fingerprints.m 2](#_Toc132134368)

[2. convert\_to\_pairs.m 2](#_Toc132134369)

[3. add\_to\_table.m 3](#_Toc132134370)

[4. getMp3List.m 3](#_Toc132134371)

[5. make\_database.m 3](#_Toc132134372)

[6. simple\_hash.m 4](#_Toc132134373)

[7. match\_segment.m 4](#_Toc132134374)

[8. my\_shazam 5](#_Toc132134375)

[Diagramas de flujo 6](#_Toc132134376)

[Parte 1 6](#_Toc132134377)

[Parte 2 7](#_Toc132134378)

[Solución a las preguntas planteadas en el laboratorio 8](#_Toc132134379)

[Parte 1 8](#_Toc132134380)

[Parte 2 17](#_Toc132134381)

[3.2 Test Shazam (Demostración de funcionamiento y conclusiones) 19](#_Toc132134382)

# Orden y análisis de funcionamiento de los archivos

## **fingerprints.m**

La función "fingerprints" toma una señal de sonido y su frecuencia de muestreo y devuelve una matriz binaria que indica la ubicación de los picos en el espectrograma. Un espectrograma es una representación visual de cómo varía la energía de una señal en diferentes frecuencias a lo largo del tiempo.

Primero, se definen algunas variables de configuración, como la tasa de muestreo deseada, el tamaño de la cuadrícula para la búsqueda de picos, la tasa de píxeles por segundo deseada y la superposición para la creación del espectrograma. Luego, se realiza un preprocesamiento de la señal. Se promedia la señal en ambos canales (en caso de que sea estéreo), se sustrae la media y se cambia la tasa de muestreo a la tasa deseada.

A continuación, se crea el espectrograma utilizando la función "spectrogram" en MATLAB. Se utiliza una ventana Hamming de duración "time\_res" (que fue definida anteriormente) para la creación del espectrograma. Después, se buscan los picos locales en el espectrograma utilizando un algoritmo que compara cada píxel con los píxeles vecinos en una cuadrícula de tamaño "gs" y marca los píxeles que son mayores que todos sus vecinos.

Luego, se calcula un umbral para los picos. Se establece un umbral para todo el segmento basado en la tasa deseada de picos por segundo "desiredPPS". Este umbral se calcula a partir de la magnitud de los picos ordenados en orden descendente. Finalmente, se aplican los umbrales a los picos y se devuelve la matriz binaria de picos. Opcionalmente, se puede trazar el espectrograma y los picos detectados en una figura.

## convert\_to\_pairs.m

Este código de Matlab toma una matriz binaria llamada "peaks" como entrada y devuelve una lista de picos próximos. Hay tres parámetros en el código para cambiar el rango de búsqueda y el número de pares permitidos por pico. Además, hay una trama opcional que muestra los picos y las líneas que conectan los pares.

El código comienza definiendo los parámetros "del\_t", "del\_f" y "fanout". Luego, la función "find" se utiliza para encontrar las coordenadas de los elementos no cero en la matriz "peaks". El número de picos se almacena en la variable "peakCount". Se crea una matriz de tuplas con tamaño igual al número máximo de pares permitidos por pico multiplicado por el número de picos.

A continuación, se itera sobre cada pico en "peaks" y se buscan los picos próximos. La primera búsqueda se realiza en la dimensión de la frecuencia, donde se buscan picos en un rango de "del\_f" alrededor del pico actual. Se agrega cada par de picos encontrados a la matriz de tuplas, siempre que el número máximo de pares permitidos por pico no se haya alcanzado todavía. Si se alcanza el número máximo de pares permitidos, la búsqueda se detiene.

Luego, se realiza otra búsqueda en la dimensión del tiempo, donde se buscan picos en un rango de "del\_t" alrededor del pico actual. Para cada pico encontrado en esta búsqueda, se realiza otra búsqueda en la dimensión de la frecuencia para buscar picos adicionales que puedan formar un par con el pico actual. Si se encuentra un par, se agrega a la matriz de tuplas. Si se alcanza el número máximo de pares permitidos por pico, la búsqueda se detiene.

Finalmente, se devuelve la matriz de tuplas con solo las filas que contienen datos, es decir, las filas llenas de información. Si la variable "optional\_plot" es verdadera, se traza una trama que muestra los picos y las líneas que conectan los pares.

## add\_to\_table.m

Este código define una función llamada add\_to\_table que toma una lista de tuplas y el número de identificación de una canción como argumentos y actualiza una tabla hash global llamada hashtable

La tabla hash se usa para almacenar información sobre las tuplas. Cada tupla se hash utilizando una función llamada simple\_hash que toma como argumentos la frecuencia, el tiempo y la diferencia de tiempo de la tupla. La salida de esta función se utiliza como índice en la tabla hash.

Si la entrada de la tabla hash correspondiente a ese índice está vacía, la función agrega el número de identificación de la canción y el tiempo de la tupla a la entrada de la tabla hash. Si la entrada no está vacía, la función agrega la información a las listas existentes de números de identificación y tiempos.

La variable maxCollisions se utiliza para contar el número máximo de colisiones en la tabla hash para una determinada función hash. La función devuelve este valor, aunque no se utiliza en otra parte del código.

## getMp3List.m

Este código es una función de MATLAB llamada "getMp3List" que devuelve una lista de nombres de archivos que contienen ".mp3" en un directorio especificado.

Primero, la función utiliza la función "dir" de MATLAB para obtener una estructura de información sobre el contenido del directorio especificado y se almacena en la variable "dirObject".

Luego, se crea una celda de tamaño igual a la estructura "dirObject" y se recorre la estructura para asignar los nombres de archivo de la estructura a esta celda "fullList".

Después, se crea otra celda "songList" del mismo tamaño que "fullList". La función recorre "fullList" y verifica si cada nombre de archivo contiene la cadena ".mp3". Si el nombre de archivo contiene ".mp3", se agrega a "songList" y se incrementa el contador. Si no, se omite y se continúa verificando el siguiente nombre de archivo.

Finalmente, la función comprueba si no se encontraron archivos MP3. Si se encontraron algunos archivos, se reduce la lista a la longitud correcta y se devuelve como "songList". Si no se encontraron archivos, la función imprime "No se encontraron archivos MP3" y devuelve una lista vacía.

## make\_database.m

Este código lee todos los archivos MP3 en la carpeta 'songs' y los agrega a la base de datos de entrenamiento si aún no están en 'songid'. Comienza definiendo la variable 'dir' como la ruta a la carpeta de canciones y llama a la función 'getMp3List' para obtener una lista de todas las canciones MP3 en esa carpeta.

A continuación, define una variable 'hashTableSize' que determina el tamaño de la tabla hash que se utilizará para almacenar las huellas digitales de las canciones. Luego, comprueba si ya existe una base de datos de canciones en el espacio de trabajo. Si no existe, la función verifica si hay archivos de base de datos previamente guardados y los carga. Si no se encuentran archivos de base de datos previos, crea una nueva base de datos vacía.

El índice de la canción se define como la longitud actual de la matriz 'songid', lo que significa que la primera canción agregada tendrá un índice de 1. El código recorre todas las canciones en la lista 'songs' y verifica si ya están en la base de datos 'songid'. Si una canción no está en la base de datos, se agrega a la base de datos, se procesa mediante la llamada a las funciones 'fingerprints', 'convert\_to\_pairs' y 'add\_to\_table', y se guarda en 'songid'. Finalmente, el código guarda las matrices 'songid' y 'hashtable' en archivos de base de datos para futuras ejecuciones del programa.

## simple\_hash.m

La función simple\_hash toma cuatro argumentos: f1, f2, deltaT y size. Estos argumentos se utilizan para generar un índice para una tabla hash. La fórmula para generar este índice utiliza una serie de operaciones matemáticas simples para producir una salida "caótica" que debe tener una distribución uniforme sobre el rango.

La función comienza calculando el logaritmo natural de f1 y agregando 2, luego calcula el logaritmo natural de f2 y agrega 2, y luego calcula el logaritmo natural de deltaT y agrega 3. Luego suma los tres resultados y los multiplica por size y 1000000. Después de redondear el resultado, lo reduce a un índice válido dentro de la tabla hash, sumando 1 para evitar índices que comiencen desde 0. Este índice resultante es devuelto por la función como la variable hash.

En resumen, simple\_hash es una función que toma tres parámetros de frecuencia y un parámetro de tiempo, y devuelve un índice para una tabla hash utilizando una fórmula matemática caótica que produce una distribución uniforme sobre el rango de la tabla hash.

## match\_segment.m

La función match\_segment toma como entrada un clip de audio y su frecuencia de muestreo, y busca una coincidencia en la base de datos de huellas dactilares de audio utilizando la tabla hash hashtable y la cantidad total de canciones numSongs.

Primero, la función convierte el clip de audio en tuplas de pares de picos llamando a la función fingerprints, que se define en otro lugar del código. Luego, para cada tupla de pares de picos, la función calcula su hash utilizando la función simple\_hash. Si hay alguna entrada en la tabla hash para ese hash en particular, se buscan coincidencias en las canciones utilizando la información almacenada en la tabla hash. Para cada coincidencia, se calcula la diferencia de tiempo entre los pares de picos del clip y los pares de picos de la canción, y se almacena esta información en una celda matches correspondiente a esa canción.

Después de calcular todas las coincidencias posibles, la función encuentra el conteo de la moda de la matriz de desplazamiento de tiempo para cada canción. La canción con el conteo de moda más alto se considera la mejor coincidencia y se devuelve su identificador bestMatchID. Además, la función calcula la confianza en la coincidencia, que se define como la proporción del número de coincidencias encontradas con respecto al número total de tuplas de pares de picos del clip, y se devuelve como confidence.

Finalmente, si la variable optional\_plot se establece en 1, la función genera un histograma de las desviaciones de tiempo para cada canción.

## my\_shazam

El código es una implementación de un algoritmo de reconocimiento de canciones basado en hashing acústico. Primero, el código carga una base de datos de canciones y una tabla hash correspondiente. Luego, el código genera una grabación de audio de 10 segundos o selecciona un segmento aleatorio de una canción de la base de datos y, opcionalmente, agrega ruido a la grabación. Finalmente, el código utiliza la función "match\_segment" para comparar el segmento de audio con los segmentos de audio de la base de datos y devuelve la canción más cercana y un nivel de confianza en la coincidencia.

El código también incluye algunas líneas para limpiar la memoria y cerrar todas las figuras antes de comenzar la ejecución, así como para reproducir el segmento de audio después de que se encuentra la canción coincidente.

# Diagramas de flujo

## Parte 1

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

*Figura 1: Diagrama de flujo para la primera parte del proyecto*

## Parte 2

*Figura 2: Diagrama de flujo para la segunda parte del proyecto*

# Solución a las preguntas planteadas en el laboratorio

## Parte 1

2.2 Fingerprint

2.2.1 Preprocessing

**Q1: Deseamos eliminar el sesgo de CC de la señal de sonido. La polarización de CC es el valor promedio de la señal, que no es audible, pero puede afectar el espectrograma. Elimine esto restando la media de la señal. ¿Por qué podría ser una buena idea volver a centrar nuestra señal?**

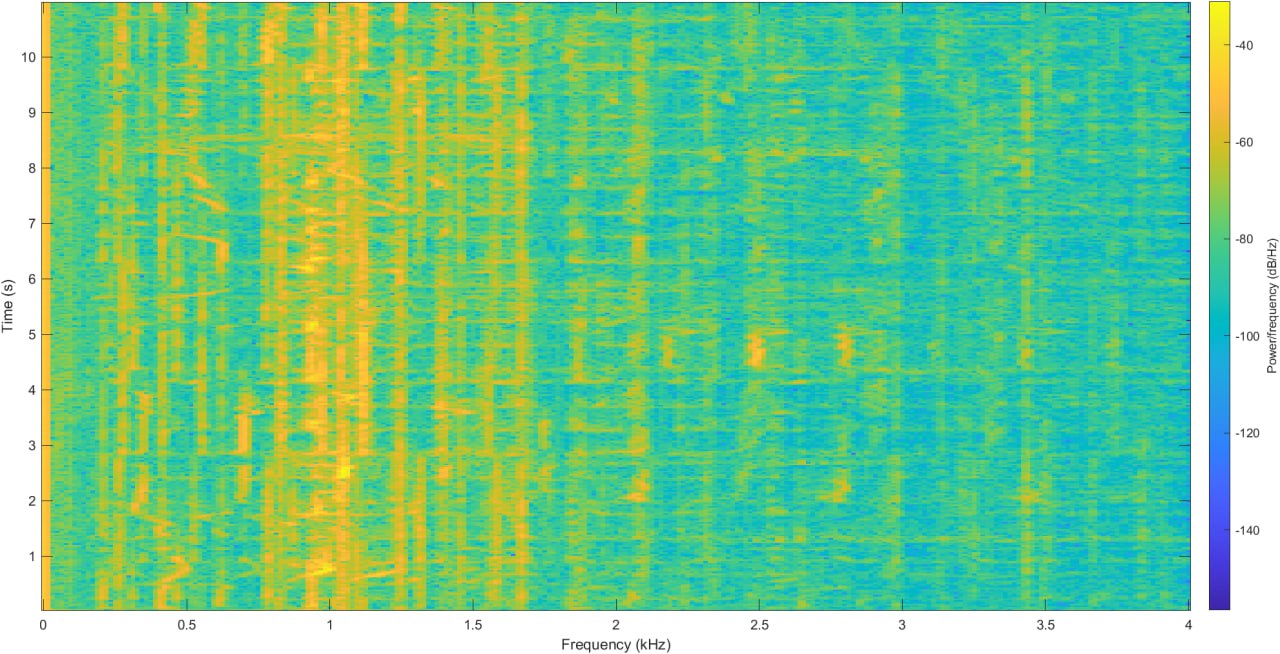
Eliminar el sesgo de CC de una señal sonora es una buena idea porque puede afectar el análisis del espectrograma y otras representaciones espectrales de la señal. El sesgo de CC puede ser causado por varios factores, como la variación en el nivel de la señal, la falta de simetría en la forma de onda, o incluso la elección de la escala de amplitud. Este sesgo puede enmascarar o afectar la detección de ciertos patrones o características en el espectrograma, y puede dificultar la comparación de espectrogramas entre diferentes señales.

Al eliminar el sesgo de CC, se elimina el componente de la señal que no tiene información de frecuencia. De esta manera, el espectrograma se centra en las componentes de la señal que varían en el tiempo y la frecuencia. Por lo tanto, se puede obtener una mejor representación de las características espectrales de la señal y una mejor comparación entre diferentes señales.

Para ilustrar cómo la eliminación del sesgo de CC afecta el espectrograma, se puede trazar el espectrograma de una señal con y sin la media eliminada. La figura resultante mostrará la diferencia en la distribución de energía en el espectrograma antes y después de la eliminación del sesgo de CC. Si el sesgo de CC es significativo, se puede observar una gran diferencia en la escala de amplitud entre el espectrograma antes y después de la eliminación de la media.

2.2.2 Spectrogram

**M1: Trazar la magnitud del espectrograma de la canción con ejes debidamente etiquetados. También trace el registro de la magnitud del espectrograma con ejes debidamente etiquetados.**



*Figura 3: Espectrograma sin restar la media*

*Imagen que contiene Gráfico

Descripción generada automáticamente*

*Figura 4: Espectrograma restándole la media. Note que la barra amarilla del inicio ya no se encuentra*

**Q2: Es común estudiar visualmente el registro de la magnitud del espectrograma. ¿Por qué podría ser esto una buena idea?**

Es habitual estudiar visualmente el logaritmo de la magnitud del espectrograma porque puede mejorar la visualización de las características de frecuencia de la señal y reducir la influencia de las componentes de alta amplitud en la representación del espectrograma.

El espectrograma muestra cómo la energía de la señal está distribuida en el tiempo y la frecuencia. Sin embargo, debido a que la energía en la señal puede variar ampliamente en magnitud en diferentes frecuencias y momentos en el tiempo, los espectrogramas a menudo tienen una gran amplitud en algunos puntos y una amplitud muy baja en otros. Esto puede hacer que sea difícil apreciar visualmente las características de la señal y la variación de energía en el tiempo y la frecuencia.

Al tomar el logaritmo de la magnitud del espectrograma, se reduce la influencia de las componentes de alta amplitud en la representación visual del espectrograma, lo que permite una mejor visualización de las características de frecuencia de la señal y las variaciones de energía en el tiempo. Además, el logaritmo de la magnitud se utiliza comúnmente en el procesamiento de señales de audio porque la percepción humana del sonido es logarítmica. Por lo tanto, un espectrograma en escala logarítmica puede estar más en línea con la percepción humana del sonido y, por lo tanto, puede resultar más fácil de interpretar visualmente.

2.2.3 Local peaks

**M2: Calcule el mapa de la constelación para gs=4, es decir, 4 puntos en cada dirección. Calcula cuántos picos hay y anota tu respuesta.**

Para contar el número de picos, podemos simplemente sumar los valores en la matriz peaks, ya que es una matriz booleana donde los valores verdaderos corresponden a la presencia de un pico.

num\_peaks = sum(peaks(:));

Esto nos da un total de 835 picos.

Patrón de fondo

Descripción generada automáticamente con confianza media

*Figura 5: Picos con g=4*

**D1: ¿Cuántos picos hay por segundo en promedio?**

Para calcular el número de picos por segundo de media, primero debemos calcular la duración total de la señal de audio. Podemos hacerlo dividiendo la longitud de la señal por la frecuencia de muestreo:

duracion = length(sound) / fs;

Esto nos da una duración de aproximadamente 11.0379 segundos. Luego, podemos dividir el número total de picos por la duración de la señal:

picos\_por\_segundo = num\_peaks / duracion;

Esto nos da un resultado de aproximadamente 75.6486 picos por segundo.

Patrón de fondo

Descripción generada automáticamente

*Figura 6: Picos con g=1*

*Imagen que contiene Calendario

Descripción generada automáticamente*

*Figura 7: Picos con g=10*

**D2: Si el tiempo lo permite, puede intentar localizar valles de frecuencia de tiempo en lugar de picos. ¿Crees que tomar huellas dactilares de la canción usando picos proporciona alguna ventaja inherente sobre el uso de valles?**

La elección de utilizar picos o depresiones para la identificación de canciones depende de muchos factores, como la aplicación específica, las características de la señal y las preferencias personales del usuario.

En términos generales, ambos enfoques pueden proporcionar información útil para la identificación de canciones. La identificación de picos puede destacar las regiones de la señal con mayor energía o prominencia, mientras que la identificación de depresiones puede resaltar las transiciones de frecuencia más abruptas en la señal.

En última instancia, la elección del enfoque dependerá de la naturaleza de la señal y de la información que se busque. En algunos casos, puede ser útil utilizar ambos enfoques y comparar los resultados para obtener una comprensión más completa de la señal.

A nivel de código solo es cambiar en la linea

peaks = peaks.\* (magS > shiftedMag);

Por:

peaks = peaks.\* (magS < shiftedMag);

Patrón de fondo

Descripción generada automáticamente

*Figura 8: Localización valles de frecuencia en vez de picos*

2.2.4 Thresholding

**Q3: Queremos usar solo los picos más grandes. ¿Por qué?**

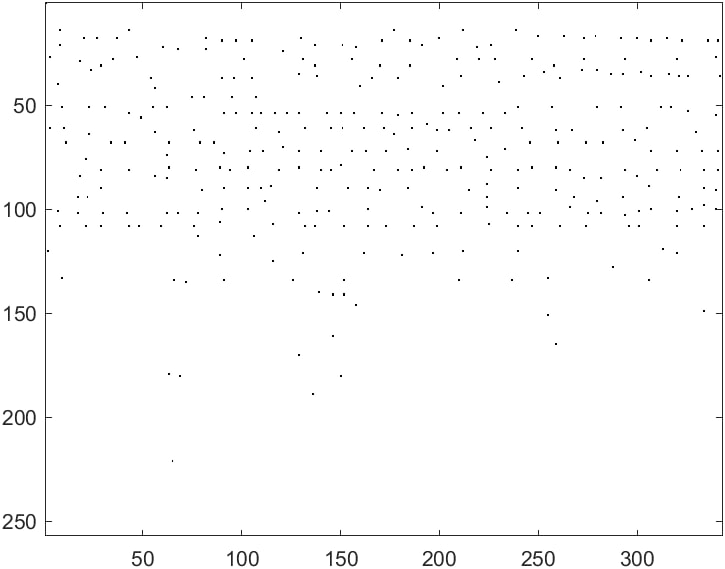
Queremos utilizar sólo los picos más grandes porque representan las características más importantes de la señal de audio que estamos analizando. Estos picos corresponden a las frecuencias más relevantes y las más fuertes de la señal, lo que significa que son las características más distintivas y únicas de la señal. Al enfocarnos en estos picos más grandes, estamos mejorando nuestra capacidad para identificar la canción con mayor precisión y reducir el riesgo de falsas identificaciones debido a características menos importantes. Además, al enfocarnos en los picos más grandes, podemos reducir el ruido y mejorar la calidad general del clip que estamos identificando.

**Q4: Vea qué tan cerca puede llegar a 30 picos por segundo ajustando el umbral. ¿Qué umbral usaste?**

**Para llegar a los 30 picos por segundo se debe usar un umbral de 0.83. Para hallar esto, se realizaron distintas pruebas ajustando el umbral, estas se evidencian a continuación:**

|  |  |  |
| --- | --- | --- |
| **Prueba 1** | **Prueba 2** | **Prueba 3** |
| threshold  10  num\_peaks  9  duracion  11.0379  picos\_por\_segundo  0.8154 | threshold  20  num\_peaks  3  duracion  11.0379  picos\_por\_segundo  0.2718 | threshold  1  num\_peaks  279  duracion  11.0379  picos\_por\_segundo  25.2766 |
| **Prueba 4** | **Prueba 5** | **Prueba 6** |
| threshold  0.5000  num\_peaks  430  duracion  11.0379  picos\_por\_segundo  38.9568 | threshold  0.8000  num\_peaks  338  duracion  11.0379  picos\_por\_segundo  30.6218 | threshold  0.8300  num\_peaks  332  duracion  11.0379  picos\_por\_segundo  30.0783 |

**M3: Usando el umbral de arriba, muestre la constelación de picos como antes. Comente la distribución de los picos. ¿Es uniforme? ¿Están muy juntos? Si es así, ¿es esto algo bueno?**



*Figura 9: Constelación de picos con el umbral de 0.83*

Si el valor del umbral se ajusta para obtener 30 picos por segundo, entonces la distribución de los picos dependerá del contenido de frecuencia y energía en la señal de audio. Si hay varias frecuencias con energía comparable, entonces los picos estarán más uniformemente distribuidos. Si hay pocas frecuencias dominantes, entonces los picos estarán más juntos alrededor de esas frecuencias.

Al revisar la figura 9, se puede identificar que los picos están uniformemente distribuidos, esto es bueno pues da evidencia de que la señal de audio es compleja y variada, lo que es positivo para la identificación de la canción. Sin embargo, en el caso contrario donde los picos estuvieran muy juntos, habría que evaluar si representan una señal de audio útil para la identificación de la canción. Siempre y cuando los picos estén bien separados en frecuencia y no haya solapamiento significativo entre ellos la señal de audio funciona. En cualquier caso, se necesitaría más análisis de la señal de audio y la estrategia de identificación de la canción para determinar la calidad de la constelación de picos y si es adecuada para la identificación de la canción.

**D3: Se ha proporcionado un código para encontrar un umbral para lograr 30 picos por segundo. Descomentar este código ahora. Quizás haya mejores formas de umbral adaptativo. Por ejemplo, un umbral fijo a través de la duración de la canción podría no ser apropiado. Además, las frecuencias altas tal vez deberían tener un umbral más bajo. Siéntase libre de jugar con esto una vez que tenga un sistema completo de Shazam funcionando.**

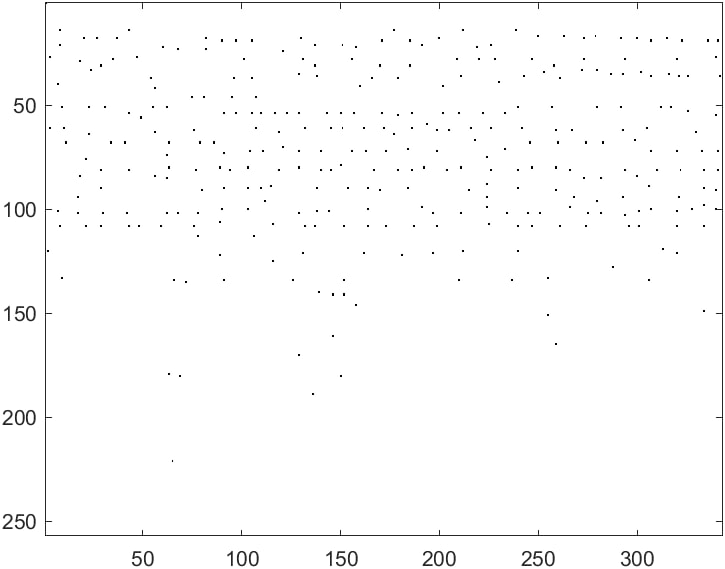
**

Figura 10: Constelación 30 picos por segundo

Realizando lo indicado en el enunciado se obtiene:

threshold

0.8346

num\_peaks

330

duracion

11.0379

picos\_por\_segundo

29.8971

2.3 Find peak pairs

**M4: Hemos proporcionado el comando convert to pairs que toma una matriz de picos y devuelve una tabla de pares que están cerca tanto en tiempo como en frecuencia, como se muestra en la introducción y se ilustra en la página siguiente. Experimente con los parámetros de esta función. Habilite el gráfico en esta función y muestre los resultados al TA.**

**Los resultados generados al experimentar con los parámetros de la función convert\_to\_pairs:**

Todas las pruebas se realizaron con la canción de ejemplo ''viva.mp3".

|  |  |
| --- | --- |
| **Resultado 1**  del\_t  35  del\_f  30  fanout  3 | Interfaz de usuario gráfica  Descripción generada automáticamente con confianza media  *Figura 11: Resultado 1 al experimentar con convert\_to\_pairs* |
| **Resultado 2**  del\_t  70  del\_f  60  fanout  6 | *Figura 12: Resultado 2 al experimentar con convert\_to\_pairs* |
| **Resultado 3**  del\_t  12  del\_f  10  fanout  1 | Diagrama  Descripción generada automáticamente con confianza baja  *Figura 13: Resultado 3 al experimentar con convert\_to\_pairs* |

## Parte 2

3.1 Match clip to song

3.1.2 Recover matches from hash table

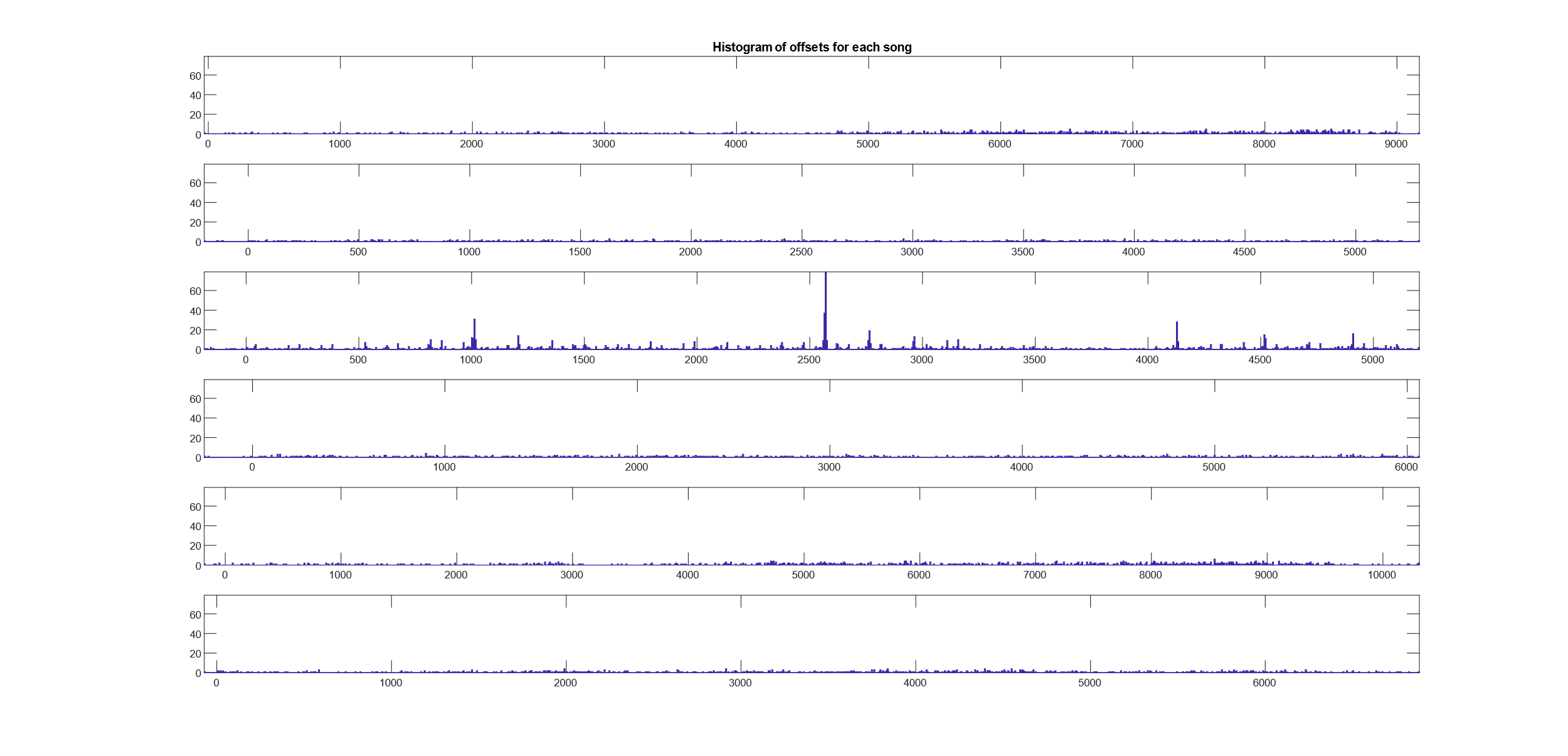
**D4: Cuando se guarda la lista de desplazamientos ¿Por qué nos preocupamos por las compensaciones en lugar del propio vector de tiempo?**

En la etapa de coincidencia de canciones, el objetivo es encontrar una canción en la base de datos que sea similar al clip que se está buscando. Dado que la canción en la base de datos y el clip pueden tener diferentes velocidades de reproducción, se espera que los pares de picos en las dos señales no estén perfectamente alineados.

Por lo tanto, lo que se hace es calcular el desplazamiento entre el clip y cada canción en la base de datos en lugar de simplemente comparar las temporizaciones de los pares de picos. De esta manera, se puede tener en cuenta cualquier diferencia en la velocidad de reproducción y aún así encontrar la canción correcta en la base de datos.

**M5: Habilite el gráfico opcional en el código para ver una visualización gráfica de los datos extraídos y muéstreselo al TA. Esto muestra una trama para cada canción en la base de datos. Este gráfico es un histograma del vector de compensación.**

Todas las pruebas posteriores se realizaron con la canción de ejemplo ''enemy.mp3".



*Figura 14: Histograma del vector de compensación.*

**D5: Un histograma cuenta el número de ocurrencias de cada valor en una lista, ignorando el orden de la lista. Las alturas de este gráfico de barras indican el número de ocurrencias de un valor particular. ¿Ves lo que esperabas? ¿Cómo se puede usar esto para identificar la canción correcta en la base de datos?**

Sí, un histograma puede ser útil para identificar la canción correcta en la base de datos. En este caso, el histograma muestra el número de coincidencias de cada tiempo de desplazamiento entre los pares de picos encontrados en el clip y los pares de picos en cada canción de la base de datos.

Si hay una canción en la base de datos que coincide con el clip, esperamos que su histograma de desplazamiento de tiempo tenga un pico prominente que indique la cantidad de veces que aparece el tiempo de desplazamiento correcto entre los pares de picos coincidentes. Podemos encontrar la canción correcta identificando la canción cuyo histograma de desplazamiento de tiempo tiene el pico más alto.

Es importante tener en cuenta que puede haber múltiples canciones en la base de datos que tengan un desplazamiento de tiempo que coincide con el del clip. Por lo tanto, debemos considerar la confianza en la canción coincidente calculada como la proporción del número de coincidencias con respecto al número total de pares de picos en el clip. Si hay múltiples canciones con el mismo recuento máximo, la canción con la confianza más alta es la candidata más probable a ser la canción correcta.

3.1.4 Confidence

**D6: El segmento de coincidencia de función devuelve una segunda variable que indica el nivel de confianza de la decisión de coincidencia de canciones. Discuta al menos una idea para medir la confianza. Implemente esto en el código si lo desea. De lo contrario, simplemente establezca la confianza de la variable en 1.**

El 'confidence' se obtiene en la última sección del código de la siguiente manera:

Se encuentra el conteo máximo de coincidencias (el máximo número de veces que se encontró una coincidencia de un par de picos en una canción determinada) utilizando la función 'max' en la matriz 'modeCounts', que representa el número de veces que se encontró un par de picos para cada canción en la base de datos. El índice correspondiente a este máximo conteo se utiliza para identificar la canción con la mayor cantidad de coincidencias, que se toma como la "mejor coincidencia".

Luego, se divide este número máximo de coincidencias por el número total de pares de picos en el clip de audio de entrada para obtener un valor de confianza que se expresa como un porcentaje.

Por lo tanto, el valor de 'confidence' es una medida de la cantidad de pares de picos en el clip de audio de entrada que coinciden con los pares de picos en la canción identificada como la mejor coincidencia en la base de datos. Un valor alto de 'confidence' indica una alta probabilidad de que la canción identificada sea la canción correcta correspondiente al clip de audio de entrada.

### 3.2 Test Shazam (Demostración de funcionamiento y conclusiones)

**M6: Inserte el código apropiado hacia el final para usar su función de segmento de coincidencia para hacer coincidir la canción. Demostrar al TA que todo está funcionando.**

**Se inserto el código apropiado para usar la función de segmento de coincidencia y con esto se logró el último detalle para que todo el programa funcionara. De esta forma, aleatoriamente se extrae y modifica el pedazo de una canción. Y luego se genera la imagen y reproduce el pedazo cortado.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

***Figura 15: Demostración funcionamiento programa***

Calendario

Descripción generada automáticamente

*Figura 16: Histograma que ayuda a evidenciar que la canción corresponde*

**Q5: ¿Cuál es la precisión de su programa usando las siguientes pruebas? Use al menos diez clips aleatorios de 10 segundos de duración. Repita durante 5 segundos. Tenga en cuenta que el código le permite agregar ruido artificial al clip.**

**Establezca la relación señal-ruido (SNR) en 0dB y repita los experimentos anteriores. El significado de SNR es el siguiente:**

Logotipo

Descripción generada automáticamente con confianza media

Primero, se establecio **SNRdB = 5** y se realizaron las pruebas para 10 segundos con **Test 10 seg**. Los resultados de las 10 pruebas fueron:

|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
|  | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |

Segundo, se establecio **SNRdB = 5** y se realizaron las pruebas para 5 segundos con **Test 5 seg**. Los resultados de las 10 pruebas fueron:

|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |

Tercero, se establecio **SNRdB = 0** y se realizaron las pruebas para 10 segundos con **Test 10 seg**. Los resultados de las 10 pruebas fueron:

|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |

Cuarto, se establecio **SNRdB = 0** y se realizaron las pruebas para 5 segundos con **Test 5 seg**. Los resultados de las 10 pruebas fueron:

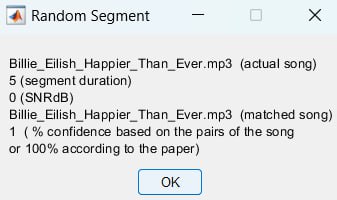
|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |

De las anteriores pruebas se puede concluir que:

* Que para *SNRdB = 5* y clips de *10 segundos* el programa tuvo **10** matchs correctos de **10** canciones presentadas. Es decir, un **accuracy del 100%.**
* Que para *SNRdB = 5* y clips de *5 segundos* el programa tuvo **9** matchs correctos de **10** canciones presentadas. Es decir, un **accuracy del 90%.**
* Que para *SNRdB = 0* y clips de *10 segundos* el programa tuvo **10** matchs correctos de **10** canciones presentadas. Es decir, un **accuracy del 100%.**
* Que para *SNRdB = 0* y clips de *5 segundos* el programa tuvo **10** matchs correctos de **10** canciones presentadas. Es decir, un **accuracy del 100%.**

Además, se evidencio que una forma más fácil de visualizar la información de la ventana Random Segment es:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente 

*Figura 17: Mejor visualización de la ventana Random Segment*