UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Máster Oficial en Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería

Trabajo Final de Máster

Prototipo funcional de aplicación móvil para la mejora de la accesibilidad de contenidos audiovisuales para personas sordas o ciegas

Marcos Jesús Santana Quintana

Tutor: Cayetano Guerra Artal

Diciembre del 2014

Agradecimientos

Ante todo agradecer el apoyo recibido por mi familia. Nadie mejor que ellos saben la dedicación y el esfuerzo que me ha supuesto este trabajo final de carrera. Su apoyo ha sido vital para que hoy pueda presentar este trabajo fin de máster y con él, terminar mis estudios universitarios.

Quiero expresar mi más sincero agradecimiento al Dr. Don Cayetano Guerra Artal por su ayuda incondicional, su colaboración y su estimado apoyo.

Por último, no puedo olvidarme de mis compañeros: Diego Nieto, Daniel Tejera, Imanol Pérez y Mahy Quintana. Sin ellos no habría sido lo mismo este camino recorrido. Va por todos ellos.

Resumen

En la actualidad las personas con deficiencias auditivas o visuales no tienen acceso a las adaptaciones (subtitulado y audiodescripción) de contenidos audiovisuales de forma habitual, salvo por los cauces tradicionales como DVD Home, canales de TV con esa característica, etc., y aun así todavía hay mucho por hacer. Incluso hoy día, mediante DVD Home, Blu-ray Home o la visualización de contenidos por Internet de las adaptaciones de contenidos, son difíciles de encontrar.

Las posibilidades para acceder al cine, la televisión, teatro o cualquier evento en directo para personas con discapacidad son reducidas, ya que no se suele facilitar ese contenido adicional y además cada colectivo tiene unas necesidades concretas:

- Las personas sordas pueden acceder por medio del subtitulado para sordos y/o la lengua de signos.

- Para las personas ciegas las posibilidades con las que se cuentan son las audiodescripciones para ciegos y el sistema Braile para la lectura de algún contenido.

Este trabajo fin de máster trata de modelar un producto que significará una gran libertad de acceso de las personas con discapacidades sensoriales. La idea consiste en desarrollar una aplicación para Smartphone o Tablet para que este colectivo pueda acceder al subtitulado o audiodescripción de cualquier película, pudiendo disfrutar así de dicho contenido que se proyecte en el cine, en su casa, o donde sea, siendo totalmente sincronizado con lo que se proyecte. Como cabe esperar, estos recursos se mostrarán de forma totalmente automática y sincronizada. Gracias a esta aplicación, este colectivo será capaz de acceder a vídeos, documentales o a cualquier película que se reproduzca en el lugar que sea, simplemente con la ayuda de un teléfono inteligente.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Ecuaciones

1. Introducción

A continuación se comentarán cuáles son los objetivos de este trabajo fin de máster, cuáles son los productos que actualmente se encuentran en el mercado que cubren necesidades parecidas a las que trata de abordar en este documento y por último, se hará un breve repaso sobre conceptos básicos sobre los que se hará mención en capítulos posteriores.

1.1 Objetivos

El objetivo principal de este trabajo fin de máster es la de desarrollar un prototipo funcional de aplicación móvil para que personas sordas o ciegas puedan disfrutar de contenido audiovisual en cualquier lugar con la ayuda de su teléfono móvil.

Por otro lado, el objetivo a medio-largo plazo es la de desarrollar la aplicación móvil para distintas plataformas, siempre y cuando el prototipo que se pretende implementar en este trabajo funcione de forma adecuada.

1.2 Estado del arte

Dentro de las opciones que podemos encontrar en el mercado de aplicaciones móviles que ofrecen facilidades a personas con deficiencias auditivas o visuales para hacer accesible cualquier contenido e información con ellas, hay diferentes tipos.

Por un lado, para personas con deficiencias visuales podemos encontrar aplicaciones que trasladan los contenidos desde el teléfono móvil al usuario por medio del sonido o del tacto, mediante un dispositivo de braile conectado por Bluetooth al Smartphone. En este sentido se pueden encontrar aplicaciones como:

- BrailleBack
- TalkBack
- Mobile Accesibility

Por otro lado, para las personas con deficiencias auditivas se puede encontrar gran cantidad de aplicaciones y de distinta finalidad. Para aquellas personas con dificultades auditivas pero no llegan a ser sordas completamente, se puede encontrar aplicaciones destinadas a amplificar la señal de audio que captan los dispositivos desde los micrófonos y reproducirlo por los auriculares. De este tipo de aplicaciones podemos encontrar:

- LouderTV
- Play It Down
- Tap Tap

También existen aplicaciones móviles que tratan de hacer de traductores a partir del sonido hablado al lenguaje escrito (para personas con problemas de audición) o lenguaje de signos (para personas sordas). En este sentido podemos encontrar:

- ASL Dictionary
- Dragon Dictation

- iASL

Sin embargo, pocas son las aplicaciones que tratan de acercar el contenido audiovisual, como son películas, documentales, etc. a estos colectivos. En este nicho de mercado es donde se pretende indagar un poco con este trabajo fin de máster.

Sin embargo, este mercado no está completamente inexplorado, ya que hay productos que ofrecen este tipo de servicios, tales como **AYNA**, **AUDESCMOBILE**, **AppCine**.

AYNA y **AUDESCMOBILE** son aplicaciones de emisión de adaptaciones de contenidos audiovisuales para deficientes sensoriales y personas con déficit de atención, en soportes: Smartphone y Tablet (subtitulado para sordos y audiodescripciones para ciegos) aplicable en salas de cine, video home, contenidos de televisión, etc.

Ambas aplicaciones móviles sincronizan la emisión de contenidos en cualquier momento de la reproducción de contenidos audiovisuales con contenidos adaptados (subtítulos y/o audiodescripciones) con sólo captar unos segundos de audio. Los contenidos disponibles son totalmente accesibles, y se pueden consulados desde la propia aplicación.

Tanto **AYNA** como **AUDESCMOBILE** son proyectos de aplicaciones ya publicados en los principales servicios de distribución de aplicaciones de las grandes plataformas móviles. Actualmente ambas aplicaciones publicadas parecen ser prototipos funcionales con las características comentadas anteriormente. Parecen ser prototipos ya que tras probar sendas aplicaciones y no recibir confirmación alguna por parte de los desarrolladores de cuál es el estado actual en el que se encuentran dichas aplicaciones, se ha llegado esa conclusión.

Por otro lado se encuentra **AppCine**, que es una aplicación móvil similar a las aplicaciones comentadas anteriormente, con la diferencia de que trata de acercar la experiencia del cine a personas con discapacidad. A parte de la característica de que es una aplicación para usar solamente en cines adaptados, ofrece contenido adaptado al lenguaje de signos.

En el caso de **AYNA** y **AUDESCMOBILE,** el contenido audiovisual es totalmente accesible desde la app desde cualquier lugar. A diferencia de estas aplicaciones, **AppCine** permite el acceso a los contenidos solamente desde la red WiFi del propio cine.

Figura 1. Gafas Epson Moverio BT-100.

Otro detalle importante de este servicio ofrecido por **AppCine** es que pueden usarse gafas de realidad aumentada como las que se muestran en la Figura 1, para hacer más cómodo a los usuarios la visualización del contenido adaptado; es decir, que pueden visualizar en dichas gafas los subtítulos, el lenguaje de signos y oír las audiodescripciones; tal y como se muestra en la Figura 2, sin necesidad de apartar la vista de la proyección para visualizar los subtítulos en el teléfono móvil, como ocurre en el caso de las aplicaciones **AYNA** y **AUDESMOBILE**.

Figura 2. Ilustración del uso de gafas con AppCine.

El uso de este tipo de gafas resulta bastante cómodo, atractivo y mejora la experiencia de usuario. Teniendo esto en cuenta y el reciente auge de los gadgets visuales como 'Google Glass' o dispositivos de realidad aumentada como 'Oculus Rift', no resulta descabellada la idea del desarrollo de una aplicación como la que se plantea en el

presente documento, ya que el traslado de dicho algoritmo a una plataforma similar a las comentadas abriría un amplio abanico de casos de uso en la vida real.

1.3 Preámbulo

A continuación se comentará brevemente detalles y conceptos propios relacionados con el tratamiento de señales, concretamente con la señal de audio, para facilitar la comprensión del documento en capítulos posteriores.

Antes de comenzar a hablar de muestreo, frecuencias y demás jerga de la rama de tratamiento de señales, se comenzará hablando de lo que es el sonido y cómo este se representa de forma digital en un ordenador.

El sonido es la percepción de nuestro cerebro de las vibraciones que produce un cuerpo y que llegan hasta nuestros oídos a través de un medio, como lo es el aire. Las vibraciones que un cuerpo puede generar y son transmitidas directamente al aire, producen un desplazamiento en las moléculas del propio aire, empujándose unas a otras en forma de ondas. La onda que puede generar esas vibraciones tiene propiedades tales como *amplitud* de onda, que tendría relación con la intensidad o fuerza con la que se produce la vibración, *período*, es el tiempo transcurrido entre dos puntos equivalentes dentro de la misma onda, y el tono es una característica perceptiva que solo captamos en los sonidos periódicos (agudo o grave); es decir, los sonidos que tienen una frecuencia más o menos constante.

En la Figura 3 se representa una onda, indicándose el período y amplitud de la misma. A su vez se puede obtener su frecuencia, sabiendo que ésta es la inversa del período.

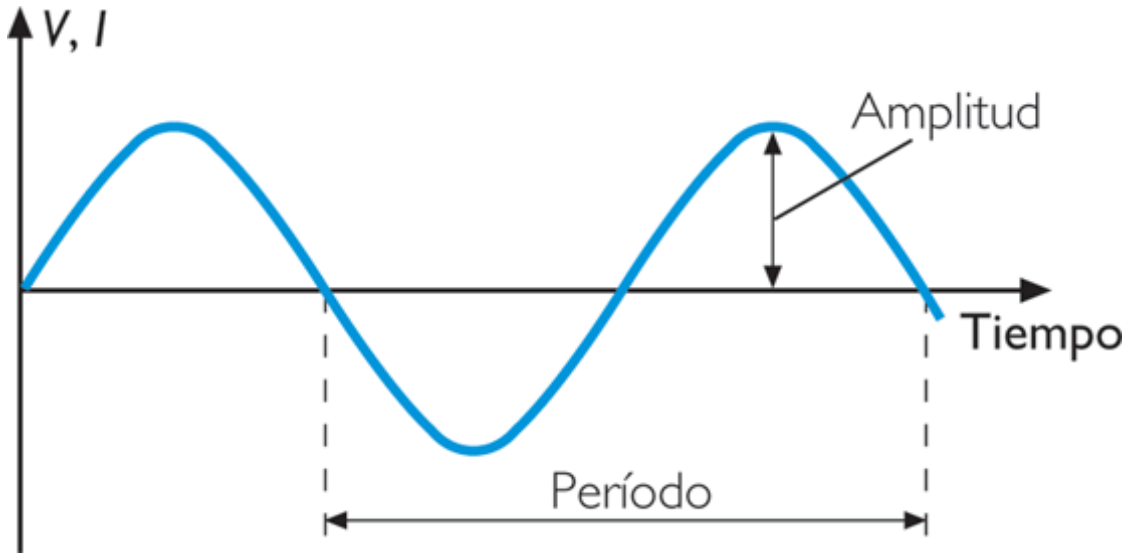


Figura 3. Representación de una onda.

Hay diferentes formas de representar una onda. Una de esas representaciones puede ser la representación en el espacio temporal tal y como se muestra en la Figura 3, donde se puede ver el valor de la amplitud de la onda a medida que pasa el tiempo. Por otro lado también existe la representación frecuencial de una onda, donde se muestran las componentes de la señal según la frecuencia en la que oscila dentro de un rango determinado.

Supongamos ahora que queremos estudiar el comportamiento de una onda **periódica**; es decir, una onda que repite su movimiento a lo largo del tiempo siguiendo un cierto patrón. Para ello necesitamos modelar su movimiento mediante la función que la representa. Esta función no la conocemos a priori, por este motivo es necesario hacer uso de uno de los conceptos fundamentales dentro del tratamiento de señales, las series de Fourier. Esta serie infinita converge puntualmente a una función periódica y continua a trozos.

Las series de Fourier constituyen la herramienta matemática básica para analizar funciones periódicas a través de la descomposición de dicha función en una suma infinita de funciones sinusoidales mucho más simples (como combinación de senos y cosenos con frecuencias enteras). Las series de Fourier tienen la forma que se muestra en la Ecuación 1:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos \frac{2n\pi}{T} t + a_n \sin \frac{2n\pi}{T} t \right] \quad (1)$$

Ecuación 1. Series de Fourier.

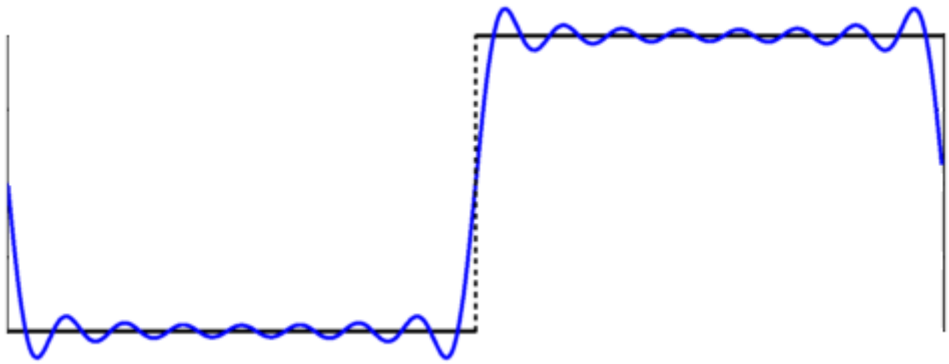
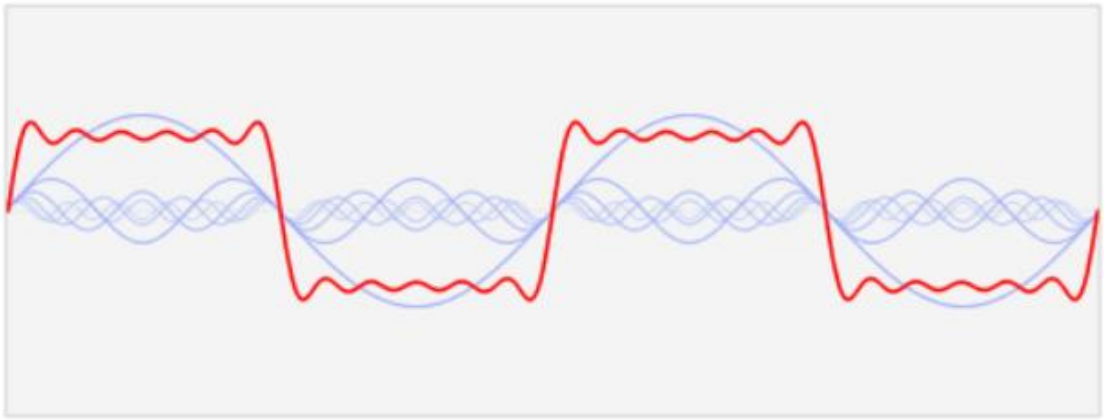


Figura 4. Señal cuadrada generada a partir de senos y cosenos.

La Figura 4 se muestra una señal cuadrada periódica original (color **negro**) y una señal generada a partir de la suma de senos y cosenos (color **azul**) que forma una señal muy parecida a la cuadrada periódica original. A medida que se usa más cantidad de senos y cosenos la señal se ajustará cada vez más a la señal original. A su vez en la Figura 5 se muestra una serie de señales de distinta frecuencia (color **azul**), que sumadas todas ellas dan como resultado la señal (cuasi cuadrada) señalada con el color **rojo**.



$$a_n \cos(nx) + b_n \sin(nx)$$

Figura 5. Señal descompuesta por señales de diferentes frecuencias.

En la Figura 6 se muestra a un lado la representación temporal de la señal cuadrada a partir de la suma de senos y cosenos (color **rojo**), mientras que por otro lado se muestra la representación frecuencial de la señal (color **azul**). En esta última representación, el eje de abscisas representa las distintas frecuencias; o lo que es lo mismo, los *nx* valores a los que se le aplica el seno y el coseno en la ecuación que se muestra en la Figura 5. Por otro lado, en el eje de ordenadas se muestra el valor del coeficiente o peso que tiene esa frecuencia en la señal final (señal **roja**). Este valor corresponde con a_n ó b_n de la ecuación que se muestra en la Figura 5. Estos valores son importantes ya que nos indican la amplitud de cada frecuencia por las que está constituida la señal que estamos estudiando.

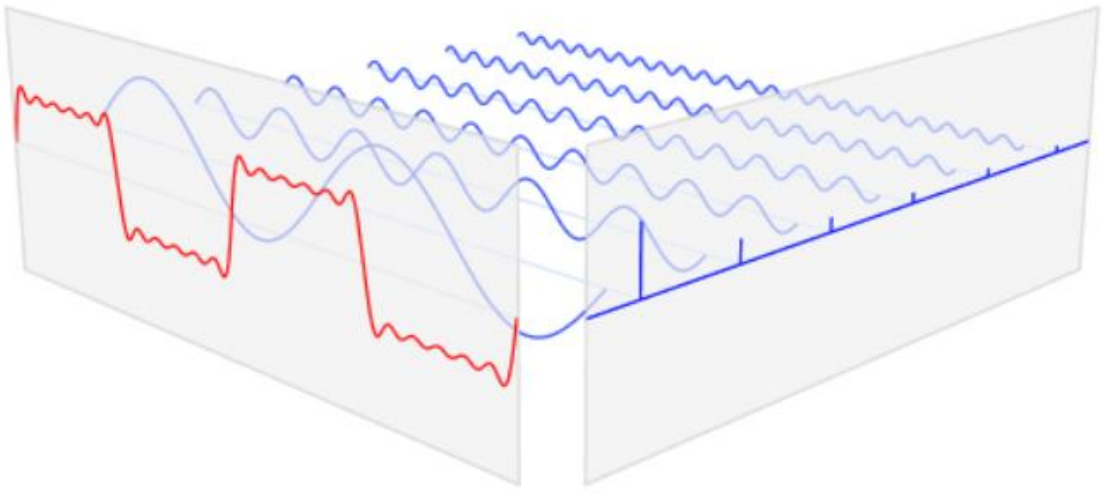


Figura 6. Representación tridimensional de la descomposición frecuencial de una señal.

En la naturaleza, pocos son los sonidos que son periódicos; es decir, predominan las señales no periódicas ante las periódicas. Para este tipo de señales existe una herramienta matemática denominada como Transformada de Fourier. Básicamente la

Transformada de Fourier es una transformación matemática empleada para transformar funciones de una representación a otra más simplificada. En nuestro caso nos permite transformar señales entre el dominio del tiempo (o espacial) y el dominio de la frecuencia. Las series de Fourier, comentadas anteriormente, son un caso simplificado de la Transformada de Fourier para las funciones periódicas. Dicha transformada se rige por la Ecuación 2:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t} dt \qquad (2)$$

Ecuación 2. Transformada de Fourier.

Donde:

$$f \in L^1(\mathbb{R})$$

Es decir, que *f* tiene que ser una función integrable en el sentido de la integral de Lebesgue, por lo tanto, el resultado de la Transformada de Fourier es una función acotada. Ideal para obtener una representación reducida de la función original; tal y como se hace al transformar de la representación espacial a la representación frecuencial.

Una vez repasado el concepto de sonido, y cómo puede ser descompuesta una señal sonora en un conjunto de frecuencias, se pasará a comentar cómo se lleva a cabo todo esto en un ordenador.

El sonido puede ser capturado o generado por un ordenador por medio de dispositivos como los micrófonos o altavoces, respectivamente. Ambos dispositivos tienen un comportamiento similar aunque inverso. Por un lado, el micrófono está formado por una membrana sensible a las vibraciones del aire, la cual se conecta a un transductor (generalmente una bobina) que se encarga de convertir las vibraciones captadas por la membrana en pulsos eléctricos. Por otro lado, el altavoz no es más que una bobina alrededor de un imán que, al recibir una corriente eléctrica, mueve una membrana que genera ondas sonoras.

Cuando se transforma la señal de sonido a una señal eléctrica, mediante un micrófono, por ejemplo; ésta sigue siendo una señal analógica. Dicha señal debe ser discretizada para poder ser almacenada en un dispositivo digital, como una memoria flash o un disco duro. Esta discretización de la señal analógica se lleva a cabo tomando una serie de muestras por unidad de tiempo sobre la señal original, tal y como se muestra en la Figura 7.

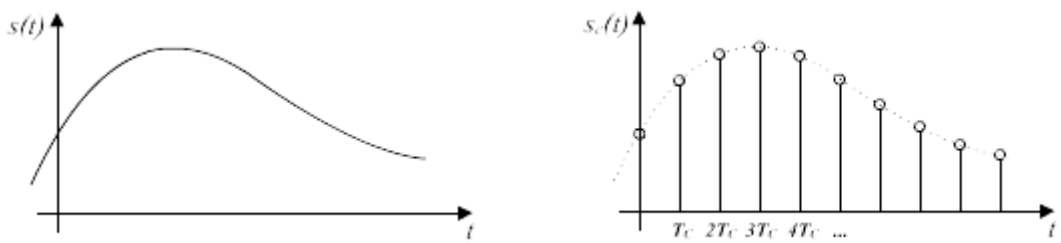


Figura 7. Discretización de una señal continúa.

Este número de muestras por unidad de tiempo se conoce como *tasa o frecuencia de muestreo*. Como todas las frecuencias, generalmente se expresa en hercios (Hz, ciclos por segundo).

El *Teorema del Muestreo de Nyquist-Shannon* dice que se puede reproducir de manera exacta una onda si la frecuencia de muestreo es, como mínimo, el doble de la más alta que se pueda escuchar. En el caso del oído humano esta frecuencia corresponde a 20.000 Hz por lo tanto la frecuencia de muestreo más adecuada será de 40.000 Hz. Algunos estudios aumentan esta cifra hasta los 44.100 Hz, que es la que se suele usar como estándar.

Como se ha comentado anteriormente, la transformada de Fourier requiere que la función a transformar en una descomposición en distintas frecuencias sea una secuencia continua. Como la señal de la que se dispone en un computador es una señal discreta (después de digitalizar la señal) no se le puede aplicar la transformada de Fourier, sino otro método específico.

Este método se conoce como la transformada de Fourier discreta (DFT). Esta transformación es equivalente a la Transformada de Fourier con la particularidad de que la realiza para secuencias discretas (no continuas) y de duración finita. Este proceso se lleva a cabo siguiendo la Ecuación 3:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} ; k = 0, \dots, N-1 \quad (3)$$

Ecuación 3. Transformada discreta de Fourier.

El único inconveniente que presenta la DFT es su alto coste computacional. Teniendo una señal de N muestras, el coste computacional de la DFT es de orden O(N^2). Sin embargo, existe la FFT o Transformada Rápida de Fourier que es un método más eficiente y optimizado en el cálculo de las frecuencias basado en la DFT, reduciendo el coste computacional a O($N \cdot log(N)$).

La idea que permite esta optimización es la descomposición de la transformada a tratar en otras más simples y éstas a su vez hasta llegar a transformadas de 2 elementos donde *k* puede tomar los valores *0* y *1*. Una vez resueltas las transformadas más simples hay que agruparlas en otras de nivel superior que deben resolverse de nuevo y así sucesivamente hasta llegar al nivel más alto. Al final de este proceso, los resultados obtenidos deben reordenarse.

2. Prototipo de aplicación

El propósito final de este trabajo es desarrollar una aplicación móvil que facilite el acceso de contenido multimedia en formato tanto subtitulado como audiodescriptivo. De este modo los colectivos de personas sordas o ciegas podrán hacer uso del mismo desde cualquier lugar y en cualquier momento para disfrutar de una película, documental, etc. mediante su teléfono móvil.

Tal y como se muestra en la Figura 8, la intención es la de facilitar la descarga de este tipo de contenido, indexado y publicado en una lista accesible mediante dicha aplicación. La descarga de este contenido no supone la descarga del contenido visual sino de los subtítulos o audiodescripción e información relativa al audio de la propia película, documental, o cualquier otro contenido.

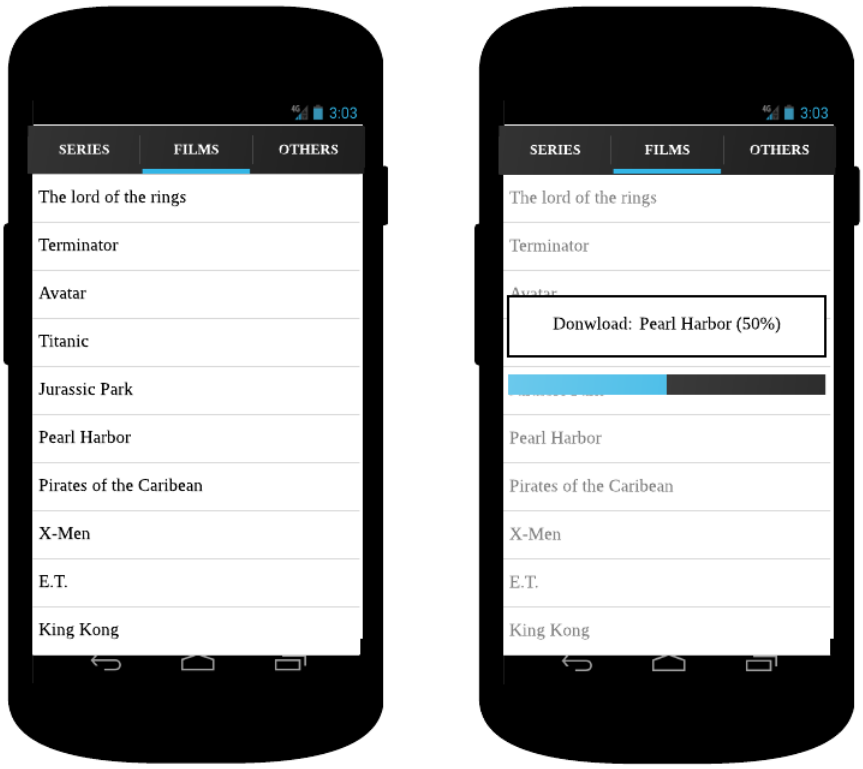


Figura 8. Prototipo de aplicación.

De este modo, una vez descargado el contenido podemos hacer uso de él mediante el lanzamiento de la propia aplicación, que se encargará de sincronizar los sonidos obtenidos desde el micrófono del dispositivo con el audio que se encuentra contenido en la información descargada. De este modo, la aplicación detectará en qué momento exacto nos encontramos de la película y así ésta mostrará los subtítulos o reproducirá la audiodescripción correspondiente a esa escena.

Figura 9. Ejemplo del funcionamiento de la aplicación.

En la Figura 9 se muestra un ejemplo ilustrativo de cómo funcionaría la aplicación final, donde una vez detectada la escena actual de la película que se está visualizando (Pearl Harbor en el ejemplo de la Figura 9), se muestran los subtítulos (en el caso de personas sordas) acorde a ese instante concreto.

Una vez explicado el funcionamiento de la aplicación, se ha de comentar que lo más complejo que plantea dicho funcionamiento es la detección del instante en el que se encuentra la película en el momento en el que el usuario lanza la aplicación. Por tal motivo, este trabajo fin de máster gira en torno al desarrollo de un prototipo funcional de la aplicación en el que se demuestre que es posible este comportamiento mediante el estudio frecuencial de las señales de audio.

En esencia se pretende estudiar la señal de audio de la película original, detectando características frecuenciales en cada instante de tiempo y almacenando esta información junto con la información relativa a los subtítulos y/o audiodescripciones en un fichero con un formato determinado.

Este fichero, ya preprocesado, será el que se use de referencia para buscar las correspondencias entre la información que almacena y la información captada desde un micrófono. Como se comentará en la siguiente sección, la información captada desde micrófono está muy condicionada a ruido y alteraciones. Este hecho hace inviable

realizar un algoritmo de búsqueda por correspondencia sin previo paso de ambas señales por filtrados y procesamiento de la propia señal.

2.1 Análisis del proyecto

La aplicación que se pretende modelar en este trabajo fin de máster trata de determinar en qué minuto y segundo exacto se encuentra la película que el usuario está visualizando para mostrar los subtítulos correspondientes a la escena actual (subtítulos en el caso de personas sordas). Para ello es necesario discriminar de algún modo el sonido correspondiente a cada instante de tiempo de la película.

Este discriminante será las frecuencias, ya que resulta más rápida, fiable y robusta la respuesta en una búsqueda por frecuencias de un determinado instante de tiempo. Cuando se dispone de una señal de audio a una determinada amplitud y se le aplica la FFT (Transformada Rápida de Fourier) para obtener las frecuencias dominantes, observamos que para la misma señal de audio con una amplitud distinta (con una intensidad de sonido mayor o menor), al aplicarle la FFT, el resultado es muy parecido.

Por otro lado, en muchas ocasiones, un micrófono al capturar un audio, actúa como un ecualizador. Un ecualizador es un dispositivo que modifica la intensidad del contenido frecuencial de la señal que procesa; es decir, que el micrófono ya sea por la propia tecnología o por la realimentación del dispositivo disminuye la amplitud de las frecuencias altas. Por lo tanto la señal que se obtiene a partir de la señal original vendrá con más o menos ruido en función de la calidad del micrófono con el que se capta el sonido.

Por ejemplo, supongamos que generamos con la ayuda de un editor de audio una señal de sonido de 440 Hz de frecuencia y otra señal con una frecuencia 6 veces superior (6.600 Hz). En la Figura 10 se muestran ambas señales con una frecuencia de muestreo de 44.1 kHz.

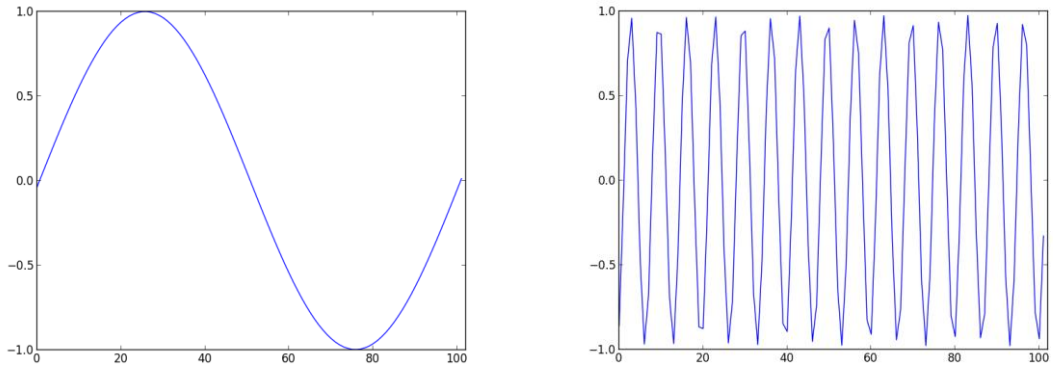


Figura 10. Representación de las muestras de dos audios.

En la Figura 10 se muestran 102 muestras de cada señal. Si se ha muestreado la señal a 44.1 kHz, esto quiere decir que se están visualizando las muestras de un intervalo de 2.3 milisegundos de ambas señales.

$$\frac{102}{44.100} = 0.0023\ secs = 2.3\ msecs$$

Un detalle a tener en cuenta en la Figura 10 es que las señales están normalizadas entre los valores 1 y -1, y la amplitud de las señales tiene el máximo valor en cada período.

Por otro lado se muestra en la Figura 11 ambas señales representadas en el espacio frecuencial. Nótese que solamente se muestran las componentes positivas de las frecuencias, para facilitar cualquier tratamiento y procesado.

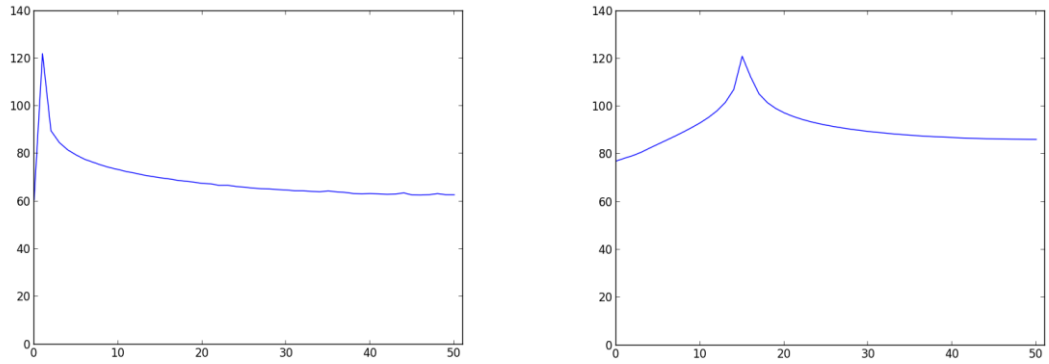


Figura 11. Representación frecuencial de dos audios.

Vemos en la Figura 11 como hay sendos picos en las representaciones frecuenciales de las señales, donde cada pico corresponde con la frecuencia con la que se han generado ambas señales (440 y 6.600 Hz, respectivamente).

Si decidiéramos generar una señal de sonido sumando las dos señales anteriores obtendríamos una señal como la que se muestra en la Figura 12.

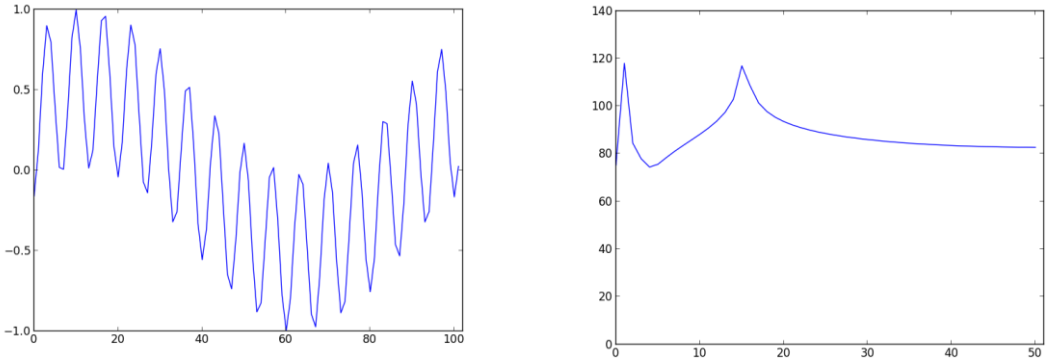


Figura 12. Representación temporal y frecuencial de la suma de dos audios.

Se puede observar en la Figura 12 la representación temporal y la representación frecuencial de una señal de sonido generada a partir de la suma de las dos señales anteriores. Se puede observar en la representación frecuencial como ahora tenemos dos picos de frecuencias situadas en 440 y 6.600 Hz. Por otro lado vemos que la representación temporal de la señal, en el lateral izquierdo de la Figura 12, no tiene parecer alguno con las representaciones temporales de la Figura 10.

Si en lugar de representar las señales originales, se muestran las mismas señales obtenidas a través de un micrófono mientras se reproducen por un altavoz, obtendríamos algo como lo que se muestran en la Figura 13 y Figura 14.

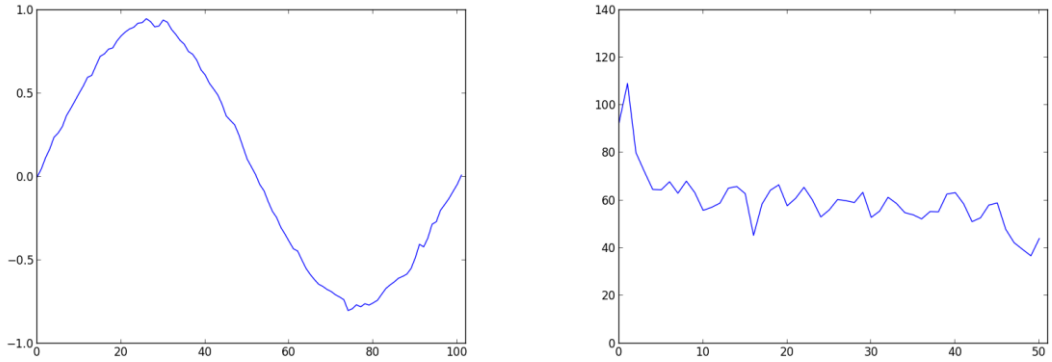


Figura 13. Representación temporal y frecuencial de un sonido grabado.

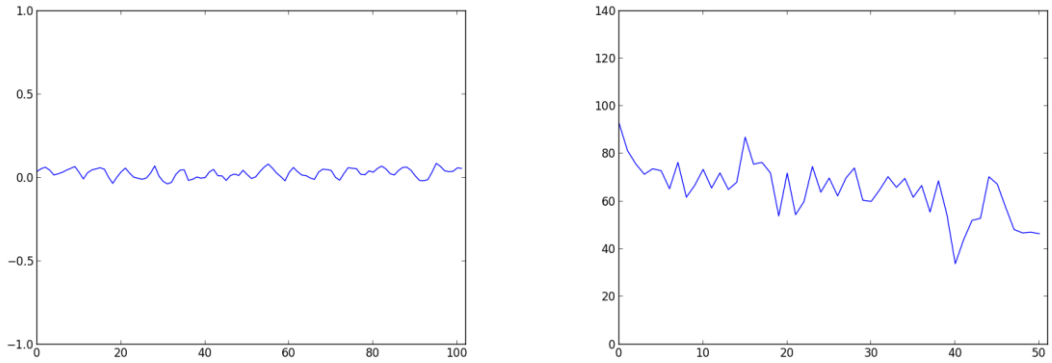


Figura 14. Representación temporal y frecuencial de otro sonido grabado.

Se puede observar en la Figura 13 como la señal temporal se ve ligeramente afectada en comparación con la señal temporal original. Sin embargo, esas pequeñas deformaciones generan una serie de perturbaciones en la representación frecuencial de la misma, añadiendo ruido a la representación frecuencial.

Por otro lado, en la Figura 14 se observa como la señal temporal no parece tan deformada (ya que la frecuencia de muestreo no nos permite obtener más detalles de la señal), pero sí se puede observar como la amplitud de la señal es menos de la mitad de la amplitud de la señal original. Esto es debido a lo que se comentó en anteriormente, y es que en ocasiones el micrófono actúa como ecualizador (dependiendo de la calidad del dispositivo con el que se grabe). En este caso, observamos que para frecuencias altas (6.600 Hz) la amplitud de la onda temporal se ve afectada. Este problema se traslada a la representación frecuencial de la propia señal, pudiéndose observar que, a pesar de que se ha añadido ruido hay un pico dominante en la señal, pero la amplitud de este pico es menor que el del pico de la representación frecuencial original.

Otro ejemplo en el que se puede visualizar perfectamente el comportamiento del micrófono ante sonidos de frecuencias altas lo encontramos en la Figura 15, que corresponde con la señal que se mostró en la Figura 12 pero grabada desde micrófono.

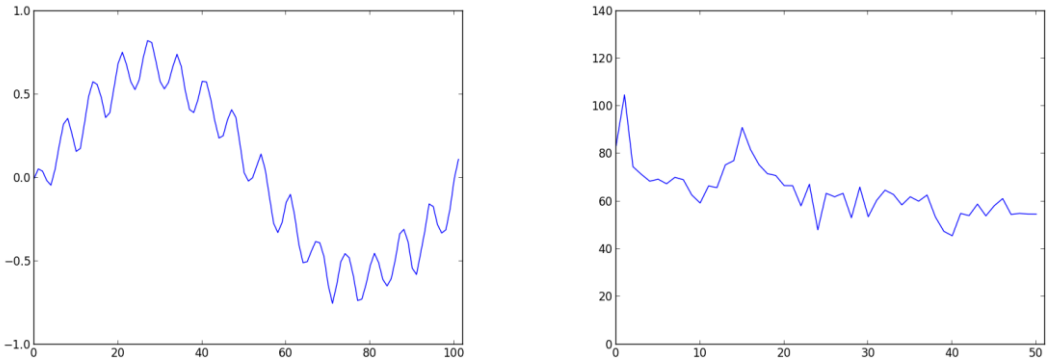


Figura 15. Representación muestra y frecuencial de la suma de ambos sonidos.

En la representación temporal de la señal de la Figura 15 no se ven deformaciones destacables en comparación con la representación temporal de la señal original (Figura 12). Lo más destacable es la amplitud de las pequeñas oscilaciones dentro de la propia onda. Sin embargo, en la representación frecuencial de la señal grabada observamos que aparte del ruido introducido por el micrófono vemos que la amplitud de los picos de frecuencia de 440 Hz y 6.600 Hz no son iguales. Teniendo en cuenta que la señal de audio original es una señal generada a partir de la suma de dos señales (suma ponderada de señales con ambos pesos iguales a 1) con frecuencias diferentes pero de amplitud similar, al ver ahora que la amplitud de las frecuencias del audio capturado por el micrófono no son iguales podemos intuir que este gazapo es debido al dispositivo de captura actúa como un ecualizador.

Finalmente, un detalle a comentar es que para una misma señal de audio podemos obtener una representación frecuencial diferente de la señal, dependiendo del intervalo temporal de la señal que se seleccione para aplicar la FFT. Por ejemplo, en la Figura 10 se mostraba en el lado izquierdo un fragmento de 2.3 milisegundos de una señal temporal y su representación frecuencial en el lado izquierdo de la Figura 11. Se puede observar, en dicha Figura 10, como el fragmento de la señal temporal comienza y finaliza con una amplitud igual a 0 y ésta varía a lo largo del eje temporal. Por otro lado, se puede observar como la representación frecuencial, en dicha Figura 11, es bastante armónica sin otros picos que no sea la frecuencia propia de la señal.

Sin embargo, en la Figura 16 se puede observar un fragmento de la misma señal con la diferencia de que en lugar de comenzar con amplitud 0 comienza con otro valor.

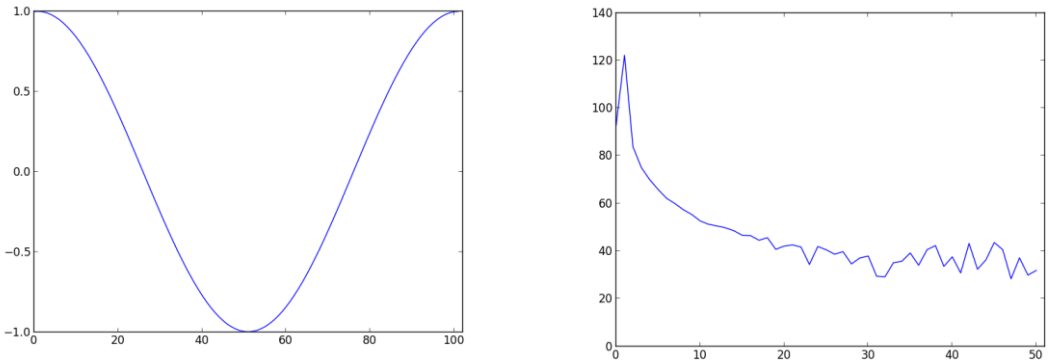


Figura 16. Representación muestra y frecuencial de un audio en otro instante de tiempo.

A su vez, se observa en la Figura 16 como la representación frecuencial del fragmento de señal temporal que se muestra en la parte derecha deja de ser armónico como el que se mostró en la Figura 11. Se observa como la FFT ha necesitado añadir nuevas señales de frecuencias altas para descomponer la señal temporal de la parte izquierda de la Figura 16. Esto se debe a que, como el fragmento de señal temporal que se pretende descomponer, no comienza con una amplitud nula sino con una amplitud considerable, es necesario añadir señales con otro tipo de frecuencias (generalmente frecuencias altas) para corregir ese desfase que se tiene de la señal.

Por lo tanto se puede concluir que al trabajar con representaciones frecuenciales de las señales de audio, nos podemos encontrar con algunos problemas como:

- La aparición de frecuencias a partir del ruido que nos añade el micrófono.
- La modificación de las amplitudes de frecuencias altas por causa del propio micrófono.
- La aparición de ruido frecuencial cuando el fragmento de audio a transformar no parte del reposo ni regresa a él; es decir, no comienza ni termina en valores cercanos 0.

2.2 Desarrollo

En apartados anteriores se comentó que la aplicación internamente trata de realizar una búsqueda por correspondencia entre la señal de audio que se tiene almacenada de una película (por ejemplo) y una señal de audio que se captura desde el micrófono, y conocer así el momento exacto en el que se encuentra dicho filme.

También se comentó que lo que se pretende estudiar son las señales de audio, buscando características frecuenciales y así, hacer posible la búsqueda por correspondencia de forma efectiva. Por este motivo, se parte de la idea de que en la parte ‘back-end’ o parte del administrador de la aplicación se dispone tanto del contenido en vídeo como del contenido adaptado (subtitulado o audiodescrito) de una película (por ejemplo).

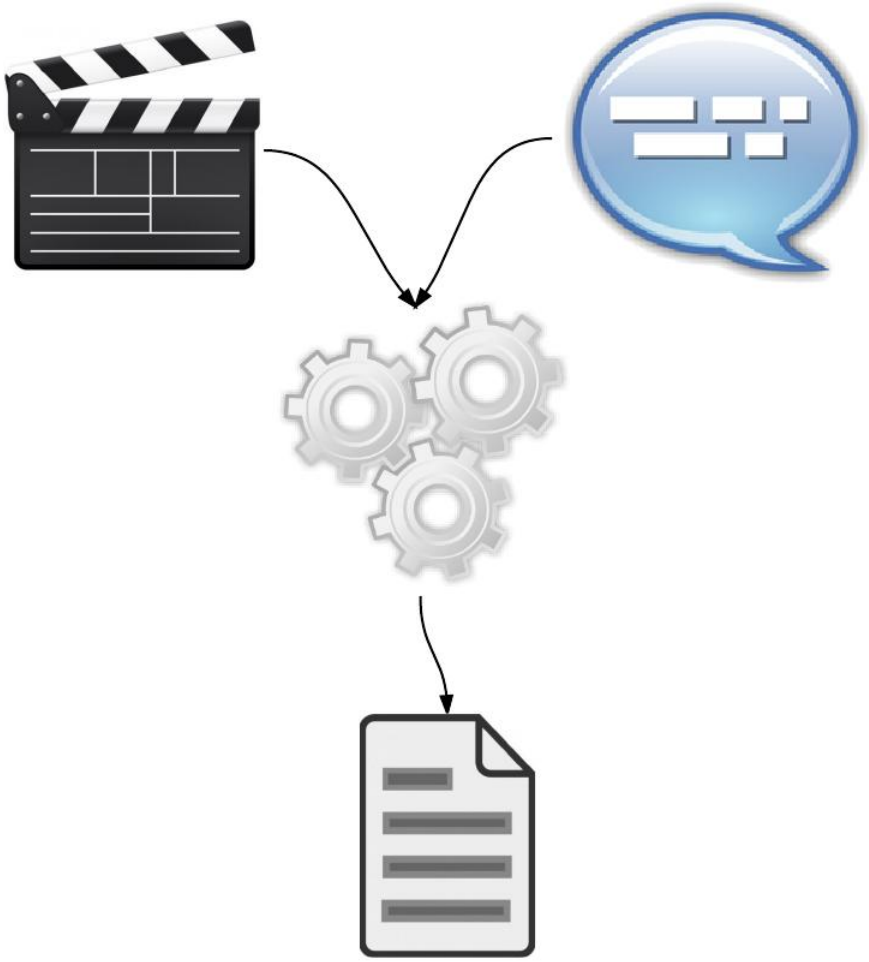


Figura 17. Representación del "Preprocesado"

Partiendo de ahí, el administrador de la aplicación puede generar un fichero con un formato determinado, tal y como se muestra en la Figura 17, mediante un algoritmo desarrollado en este trabajo fin de máster. Este algoritmo lleva a cabo un procesamiento de la señal de audio extrayendo características frecuenciales, almacenando dicha información en un fichero con un formato concreto e incrustando a su vez la información relativa al subtitulado o audiodescripción. A este paso se le conoce como '**Preprocesado**'.

Por otro lado, en la parte 'front-end' o parte del usuario de la aplicación, para hacer uso del algoritmo de búsqueda por correspondencia que ofrece la aplicación, se debe disponer del fichero generado por el administrador de la aplicación. Como se comentó en secciones anteriores, esta información será accesible a través de la propia aplicación.

Por su parte, aplicación deberá obtener la señal de audio, procesarla del mismo modo que se hizo con la señal de audio original de la película para obtener así las características frecuenciales del sonido y por último, realizar la búsqueda por correspondencia con la información generada por el administrador de la aplicación.

Para garantizar la mejor respuesta en el menor tiempo posible, la aplicación se ha diseñado e implementado con procedimientos que trabajan de forma paralela en todo momento.

Para ello se ha planteado un escenario como el que se muestra en la Figura 18, donde se puede observar que la aplicación mantiene en ejecución en todo momento cuatro procesos: grabación, procesamiento, búsqueda por correspondencia y control de la información de salida.

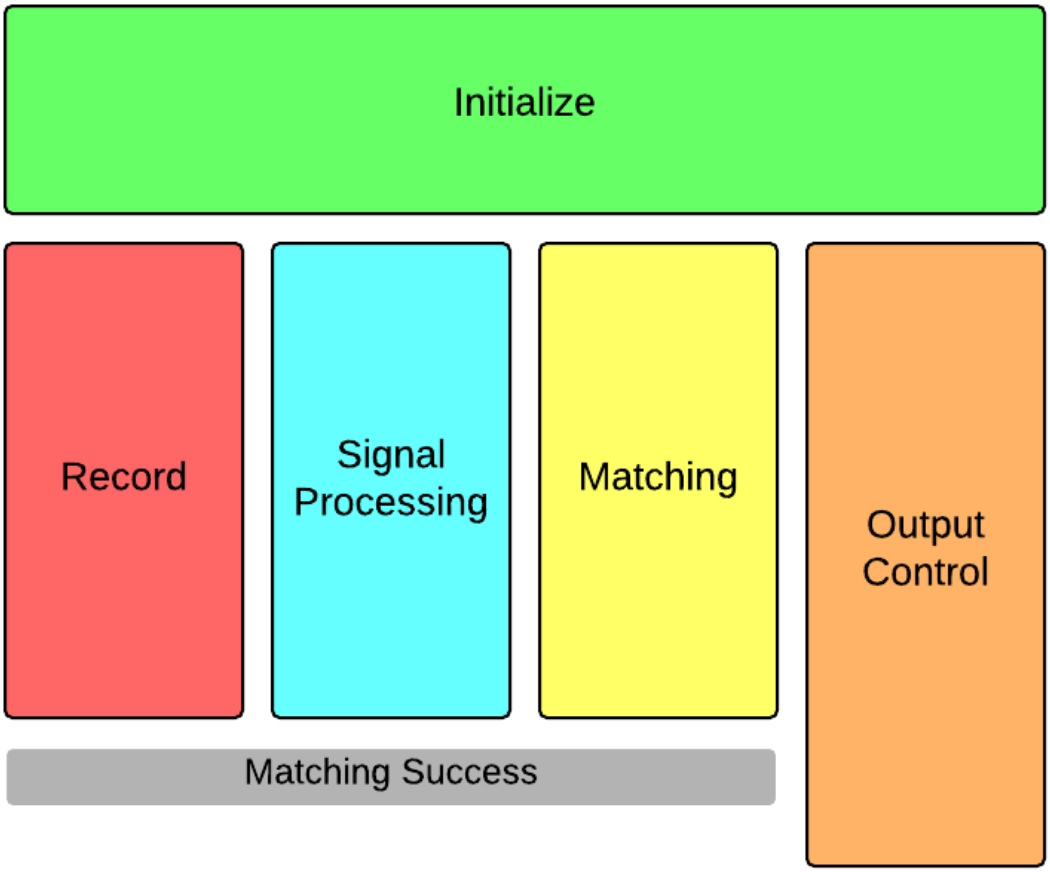


Figura 18. Representación de los diferentes procesos que conviven en la aplicación.

En la Figura 18 se puede observar cómo hay un proceso inicial que se encarga de inicializar las variables de la aplicación y de preparar el escenario para que funcione correctamente el algoritmo. Una de las tareas que realiza este proceso de inicialización es la de leer el fichero generado en el 'Preprocesado', y obtenido a través la aplicación, ya que sin este fichero no se podría llevar a cabo la búsqueda por correspondencia. Una vez inicializado el algoritmo queda a la espera que el usuario dé la orden de comenzar con la búsqueda del instante en el que se encuentra el contenido audiovisual en ese momento. Cuando el usuario lanza la orden, se lanzan en paralelo cuatro proceso que son explicados a continuación.

Fase de grabación

La fase de grabación se encarga en todo momento de grabar, desde el micrófono del dispositivo, fragmentos de sonido de una determinada duración e ir encolándolos en un contenedor de tipo FIFO (First In Fist Out) compartido con el proceso denominado 'Signal Processing'.

Fase de procesamiento de la señal

Al mismo tiempo, el proceso de 'procesamiento de la señal' recoge la información capturada por el proceso de 'grabación' y procede a aplicar diferentes filtros a la señal para obtener una información lo más representativa posible del sonido ambiente.

Esta fase del procesamiento de la señal, a su vez, pasa por distintas etapas que se llevan a cabo de forma secuencial. Estas etapas son las siguientes:

- División
- Solape
- Filtro de Hamming
- FFT (Fast Fourier Transform)
- Reducción
- Detección de picos

A continuación se explicarán detalladamente cada una de las etapas, como se llevan a cabo y el porqué de dicho paso. Para ayudar a comprender la evolución de la señal de audio a lo largo de la fase de procesamiento se mostrarán diferentes Figuras, que representará el estado de un fragmento de audio grabado a lo largo de la presente fase.

- División

Como se había comentado en el capítulo anterior, la Transformada Rápida de Fourier es la operación matemática; basada en la Transformada Discreta de Fourier, que transforma una señal discreta del dominio temporal al dominio frecuencial. Para ello es necesario tomar segmentos temporales de audio pequeños para aplicar la FFT y obtener las frecuencias características de dicho intervalo.

Para obtener estos segmentos temporales, se divide la señal de audio en segmentos de milisegundos iguales. El tamaño de estos milisegundos puede configurarse en el propio algoritmo.

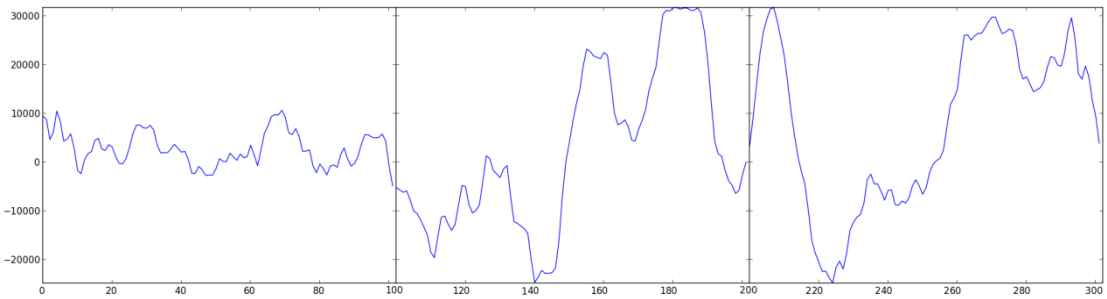


Figura 19. Representación del espacio temporal de tres fragmentos de un audio.

En la Figura 19 se puede observar las muestras de una señal de audio agrupadas en tres segmentos de 100 muestras cada uno. Como se trata de una señal de audio muestreada a 44.100 Hz, podemos calcular el tamaño temporal de los fragmentos:

$$44.100\ muestras \longrightarrow 1\ seg$$
$$100\ muestras \longrightarrow X\ seg$$

$$X = \frac{100 \cdot 1}{44.100} = 0.002267\ seg = 2.267\ mseg$$

Con el sencillo cálculo anterior observamos que cada fragmento es de 2.267 milisegundos.

- Solape

Una vez troceada la señal de audio obtenemos una serie de fragmentos totalmente independientes, con características propias del fragmento de la señal que contienen. El hecho de tener fragmentos totalmente independientes en ocasiones no es una buena idea, ya que el fragmento de audio pierde información de la situación en la que se encuentra. Para ayudar a conservar información del entorno más cercano a la ubicación de cada fragmento se opta por adjuntar a cada fragmento parte de la información de los fragmentos adyacentes.

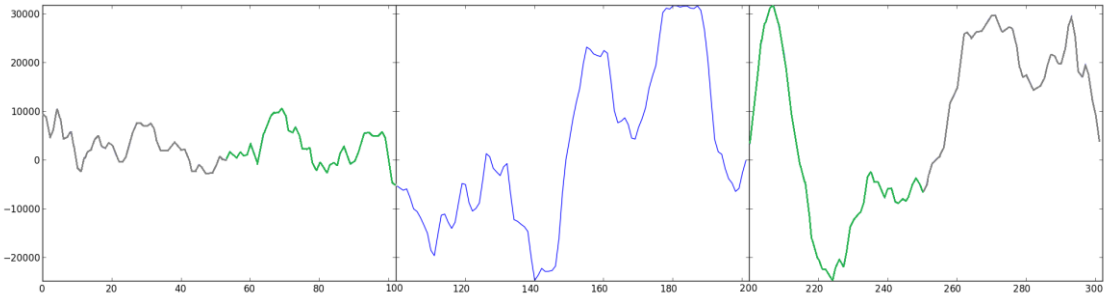


Figura 20. Representación temporal del solape de un fragmento con los adyacentes.

En la Figura 20 se muestra como al fragmento central (**azul**) se le añade información solapada de los fragmentos adyacentes (**verde**). Esto permite almacenar no solo información concreta del fragmento, sino del entorno en el que se encuentra dicho fragmento. Cabe destacar que en la Figura 20 el solape que se lleva a cabo es del 50 % de los fragmentos adyacentes (este valor de porcentaje de solape es otro parámetro configurable dentro del algoritmo), quedando finalmente fragmentos de 200 muestras; es decir, del doble del tamaño de los fragmentos iniciales.

- Filtro de Hanning

En la sección anterior se comentó la importancia de la selección del comienzo y fin de los fragmentos y cómo puede afectar al resultado obtenido por la FFT de dichos fragmentos. Se vio como la representación frecuencial de fragmentos que comenzaban en valores altos tenía gran cantidad de ruido añadido por el método matemático, en el intento de descomponer la señal. Este hecho no ocurría cuando el fragmento comenzaba y terminaba por valores cercanos a 0.

Para evitar que esto suceda y que se añada ruido a la representación frecuencial de los fragmentos, y con él información frecuencial inexistente, se aplicará a cada uno de los fragmentos un filtro conocido como la ventana de Hanning. La ventana Hanning fuerza a las extremidades de una señal hacia cero, pero también agrega distorsión a la forma de la onda. La ventana de Hanning es representada por la Ecuación 4:

$$v(n) = a_0 - a_1 \cdot \cos\left(\frac{2\pi n}{N-1}\right) \quad (4)$$

Ecuación 4. Ecuación de la ventana de Hanning.

Donde los valores $a_0 = 0.5; a_1 = 0.5$. Si representamos gráficamente esta función sería algo como la función que se muestra en la Figura 21.

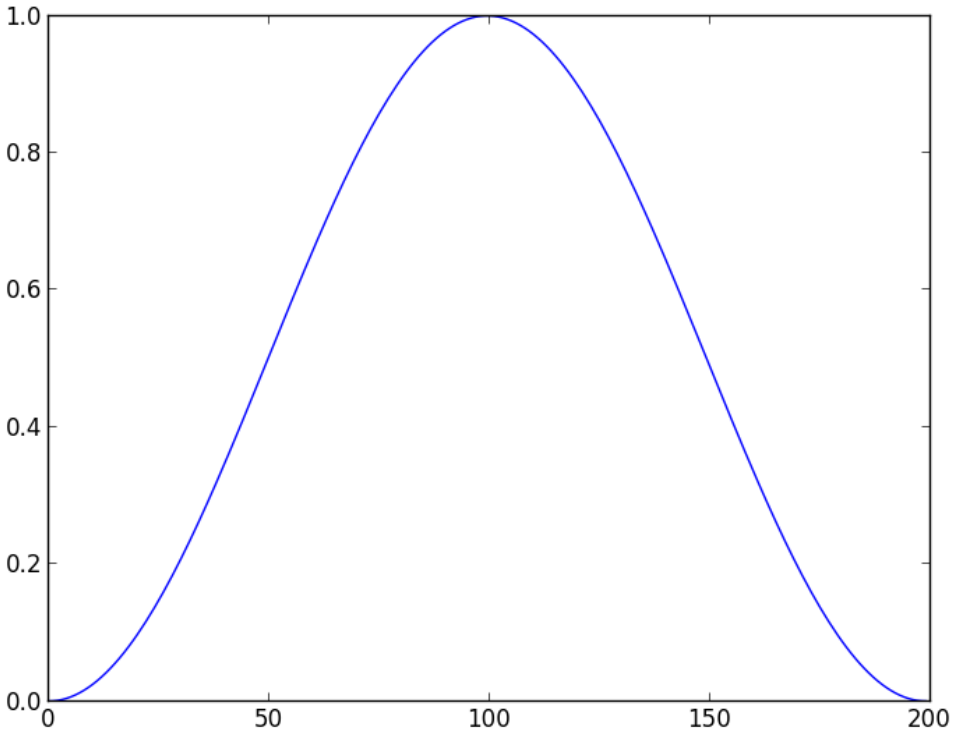


Figura 21. Representación de la función de Hanning.

Se puede observar que la ventana de Hanning que se muestra en la Figura 21 es de tamaño 200 (el mismo número de muestras que un fragmento de señal con solapamiento). Una vez se tenga esta ventana el objetivo es multiplicar cada valor de la señal por su homólogo en la función de Hanning. Haciendo esto con el caso ejemplo obtenemos lo que se muestra en la Figura 22.

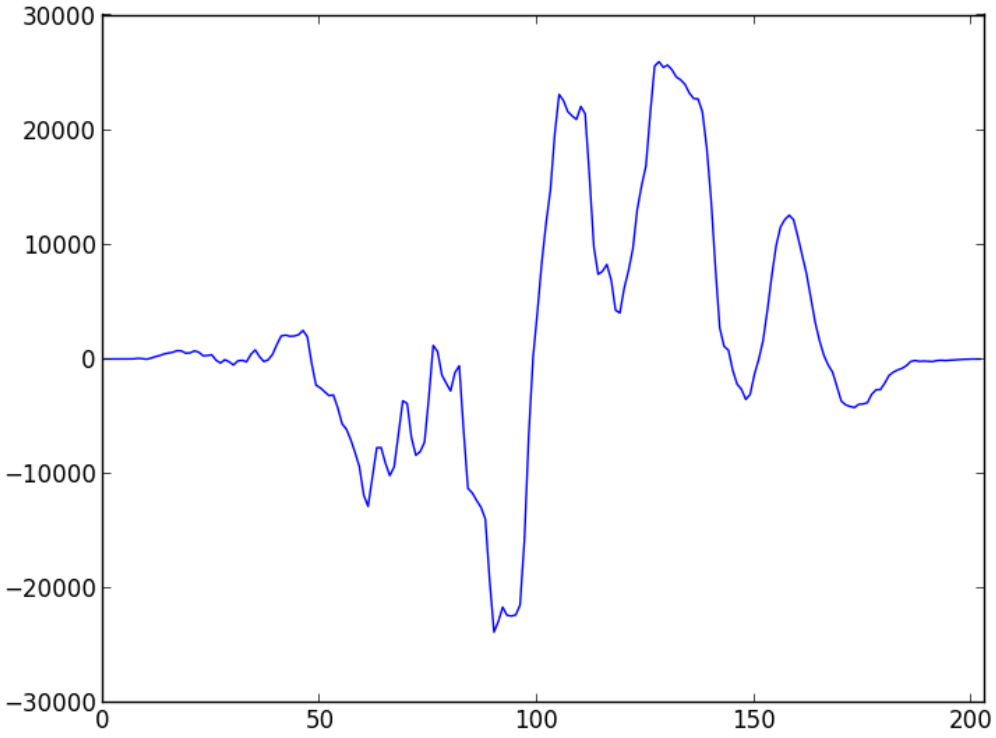


Figura 22. Representación de un fragmento de audio tras varios pasos aplicados.

En la Figura 22, en comparación con la Figura 21, podemos observar que toma los mayores valores en el medio de la señal, disminuyendo los valores de los extremos cada vez más hasta tal punto que la señal comienza y termina en 0. De esta manera se realza el fragmento intermedio quedando los fragmentos de solape adyacentes carentes de cierta importancia en la señal final.

- FFT (Fast Fourier Transform)

En este punto de la fase de procesado de la señal se le aplica a la señal tratada hasta el momento la FFT para obtener la representación frecuencial de la misma. Para observar la evolución por la que pasa la señal y comprender mejor el por qué de los diferentes pasos, se muestra a continuación la representación frecuencial de la señal a medida que se le aplican los distintos pasos comentados hasta el momento.

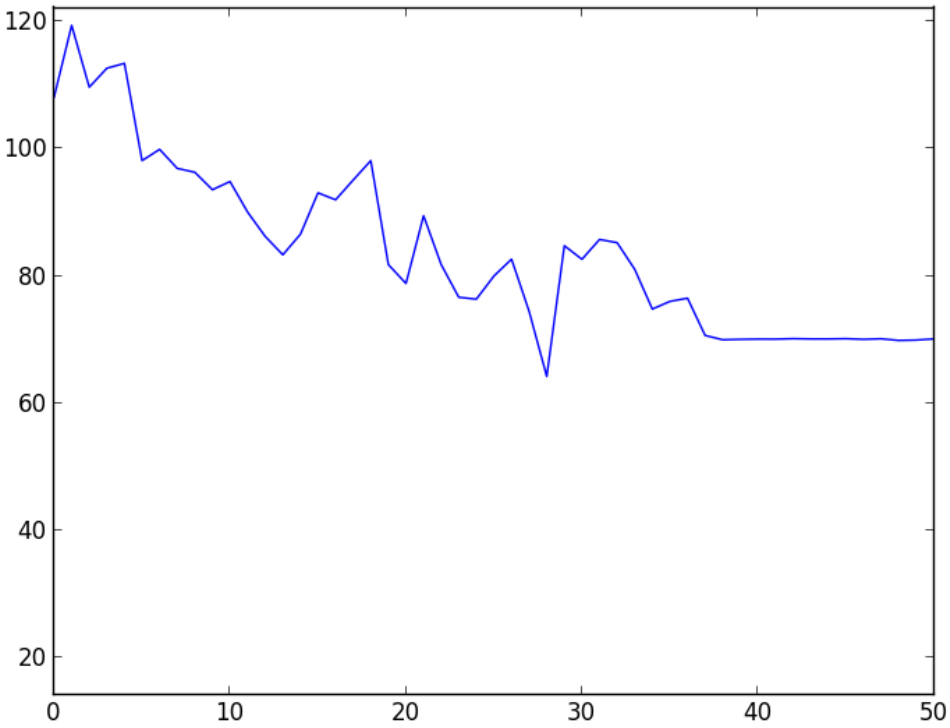


Figura 23. Representación frecuencial del fragmento inicial.

En la Figura 23 se puede observar las frecuencias del fragmento original mostrado anteriormente en la Figura 19 como el fragmento central de la señal.

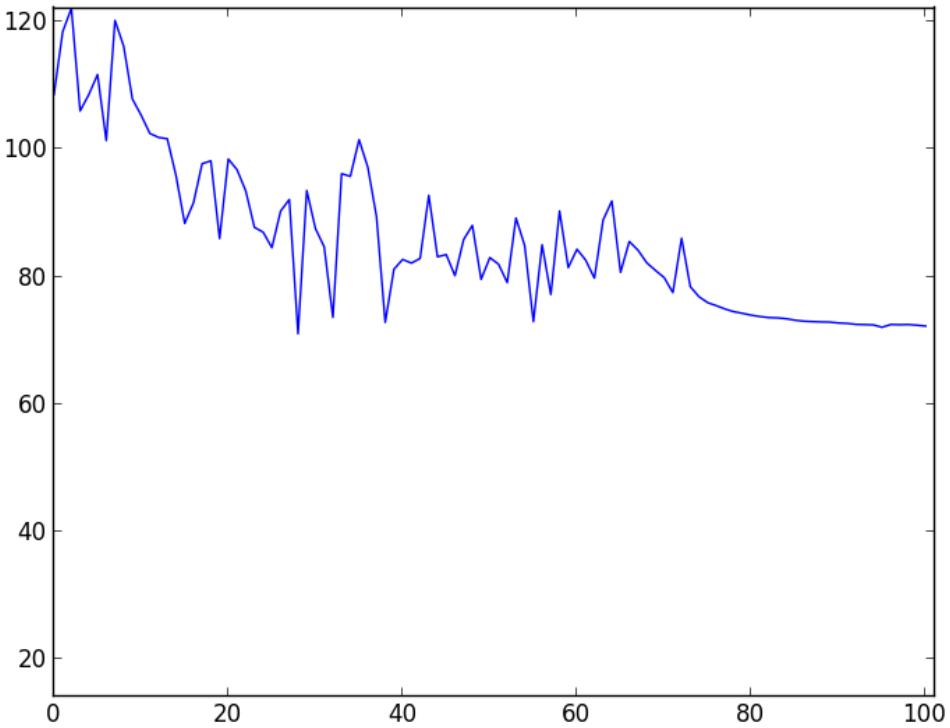


Figura 24. Representación frecuencial de un fragmento con solape.

Por otro lado, en la Figura 24 se puede observar la representación frecuencial del fragmento original con un porcentaje de solapamiento con los fragmentos adyacentes (Figura 20). Si comparamos la Figura 24 con la Figura 23, se puede observar como

aparece una gran cantidad de picos adicionales. Estas frecuencias son añadidas por la nueva información adjuntada al fragmento base.

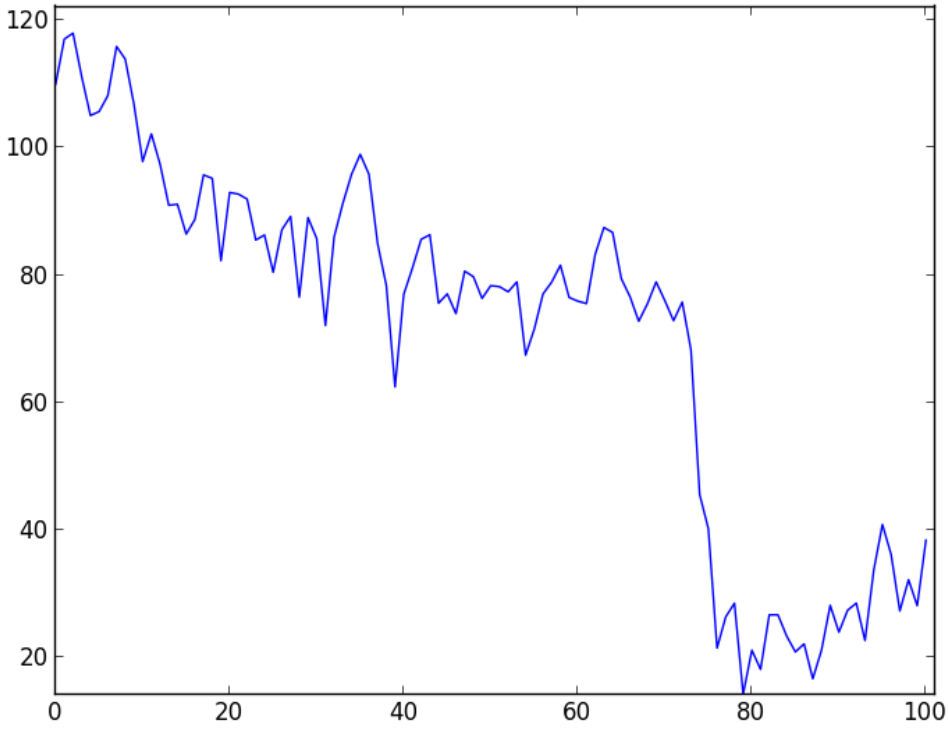


Figura 25. Representación frecuencial de un fragmento tras el filtro de Hanning.

Por último se muestra en la Figura 25 la representación frecuencial del fragmento mostrado en la Figura 22. En la Figura 22 se muestra el fragmento tras haberse aplicado un filtro de Hanning, y si comparamos la Figura 25 con la Figura 24 observamos que el hecho de aplicar este filtro realza algunos picos de frecuencias relevantes y disminuye en gran medida las amplitudes de las frecuencias altas.

En definitiva podemos observar que las frecuencias que se muestran en la Figura 23, que son las frecuencias del fragmento original de audio, y las comparamos con las frecuencias de la Figura 25, que es la representación frecuencial del fragmento tratado y filtrado, vemos que existen diferencias destacables. En esencia ambas representaciones frecuenciales son similares, con la diferencia que en la Figura 25 los picos de frecuencia son más pronunciados y destacables que los que podemos encontrar en la Figura 23.

- Reducción

Volviendo al principio, se ha de recordar que cada fragmento inicial está formado por una serie de valores que corresponde con el voltaje que se le comandan a los altavoces en cada instante de tiempo para reproducir dicho audio. El tamaño de cada fragmento en milisegundos viene determinado por una macro definida en el algoritmo, la cuál es totalmente configurable. En función del tamaño de cada fragmento y de la frecuencia de muestreo con la que se encuentra formateado el audio habrá un mayor o menor número de valores por fragmento.

Cuando se le aplica la FFT a un fragmento de milisegundos obtenemos el mismo número de valores que contiene dicho fragmento. En ocasiones el número de valores

que obtenemos de la FFT puede ser demasiado grande como para trabajar con él, por este motivo se ha definido una macro que podemos configurar en el algoritmo para pasar de tener *n* valores de frecuencias a un valor *m* menor. A este paso se le ha denominado como *reducción* del espacio frecuencial.

- Detección de picos

A continuación, teniendo un espacio frecuencial fijo de *m* frecuencias, procedemos a identificar picos de frecuencia en ese intervalo. Estos picos se usarán para identificar cada intervalo de tiempo.

Volviendo al ejemplo de intervalo de audio mostrado en Figuras anteriores, donde por último habíamos obtenido las frecuencias de dicho intervalo (Figura 25), procedemos a obtener sus picos de frecuencia tal y como se muestra en la Figura 26.

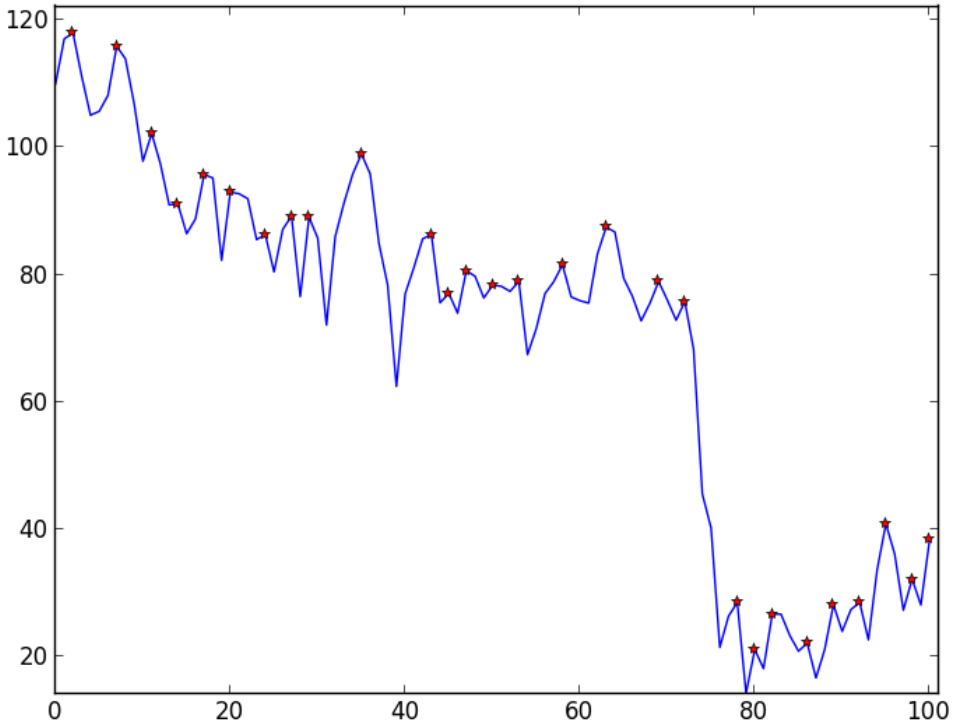


Figura 26. Picos detectados en la representación frecuencial anterior.

Se considera un pico aquel punto dentro de la función cuya amplitud de los puntos adyacentes se encuentran a menor posición que la del propio punto. En la Figura 26 podemos observar cómo se marcan los picos frecuenciales con un asterisco rojo (*). Estos puntos, tanto su posición en el eje de abscisas (frecuencia) como en el eje de ordenadas (amplitud), serán los identificadores característicos del intervalo central mostrado anteriormente en la Figura 19.

Fase de búsqueda por correspondencia (Matching)

Esta fase de búsqueda por correspondencia, es un algoritmo diseñado para buscar, dada las características frecuenciales de un audio original y las características frecuenciales de una parte del audio original grabado desde un micrófono, realizar una búsqueda de

similaridad entre ambas y determinar así en qué momento exacto pertenece el trozo de audio grabado del audio original.

Básicamente lo que se hace es, teniendo la información de las características frecuenciales de la película original (que las obtenemos desde el fichero generado por la administración de la aplicación) y las características frecuenciales obtenidas de la fase de procesamiento de la señal, se realiza una comparación.

En este caso la comparación se lleva a cabo contabilizando el número de picos que coinciden entre los dos fragmentos de frecuencia. Si los dos fragmentos tienen un pico frecuencial en una misma frecuencia, esta frecuencia se considera como un pico de frecuencia común.

Por ejemplo, supongamos que tenemos dos fragmentos frecuenciales que queremos comparar tales como los que se muestran en la Figura 27.

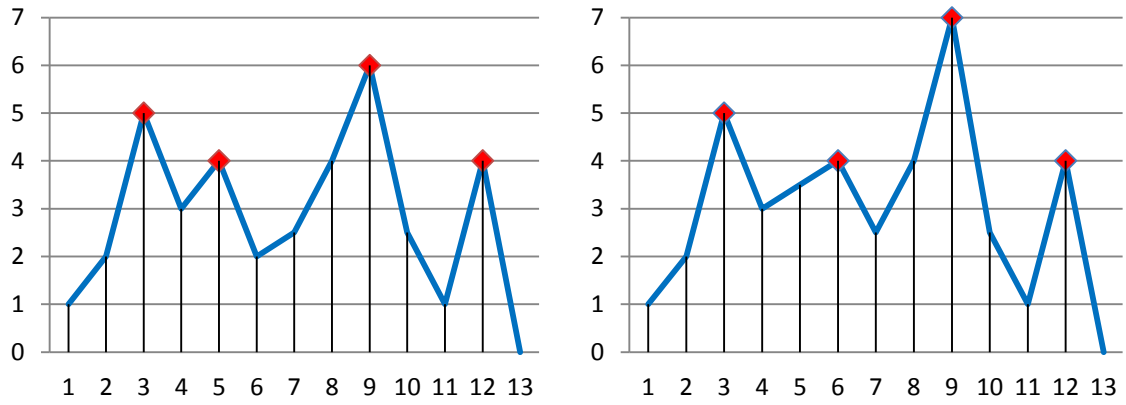


Figura 27. Fragmentos frecuenciales de prueba.

En la Figura 27 podemos observar dos fragmentos frecuenciales los cuales poseen 4 picos cada uno. En el primer caso (izquierda) los picos son localizados en las siguientes posiciones [3, 5, 9, 12], mientras que en el segundo caso (derecha) los picos se encuentran en [3, 6, 9, 12]. En este caso particular, ambos fragmentos tienen en común la posición de tres de sus cuatro picos [3, 9, 12]. Por lo tanto si aplicamos la Ecuación 5, obtenemos que ambos fragmentos se parecen un *75 %*. $[(3/4) \cdot 100 = 75]$

$$P = \frac{\sum_{i=1}^{n} 1 \mid \forall i \in \{of_x(j) = rf_x(i)\} \wedge \{j \leq m\}}{m} \cdot 100 \qquad (5)$$

Ecuación 5. Expresión del cálculo de similitud entre fragmentos.

Donde *P* es el porcentaje de similitud, *n* es el número de picos en la grabación, *m* el número de picos en el fragmento original, *rf* las frecuencias de la grabación y *of* las frecuencias del fragmento original, siendo *x* la componente del eje de abscisas de ambas frecuencias (valor de la frecuencia).

Anteriormente se comentó que en la *fase de grabación* se graban varios segundos de audio, a los cuales posteriormente se aplica un procesado, en *la fase de procesamiento de la señal*, cuyo resultado del mismo es el que se usa para hacer la búsqueda por correspondencia.

Al grabarse varios segundos de audio, el resultado obtenido de *la fase de procesamiento de la señal* consta de gran número de fragmentos frecuenciales, por lo que la búsqueda por correspondencia no se basa sólo en el resultado de la comparación de dos fragmentos entre sí, sino de una búsqueda conjunta de todos estos fragmentos. Al algoritmo que realiza esta comparativa conjunta se le denominará a partir de este punto, algoritmo de *matching*.

Para comprender mejor el funcionamiento de este algoritmo, haremos uso de otro ejemplo. Supongamos que tenemos una señal de audio de una película original y una señal de audio grabada de un fragmento de la propia película. Supongamos que la señal de audio de la película corresponde con la imagen superior de la Figura 28, mientras que la señal de audio de la grabación corresponde con la imagen inferior de la Figura 28.

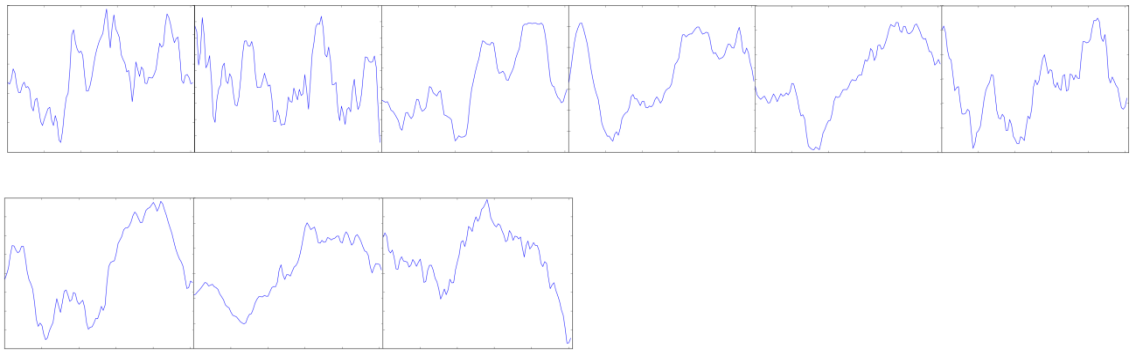


Figura 28. Muestras de un audio de prueba y una grabación de prueba.

Si le aplicamos *la fase de procesamiento de la señal* a ambas señales obtenemos las representaciones frecuenciales con sus respectivos picos, tal y como se muestra en la Figura 29.

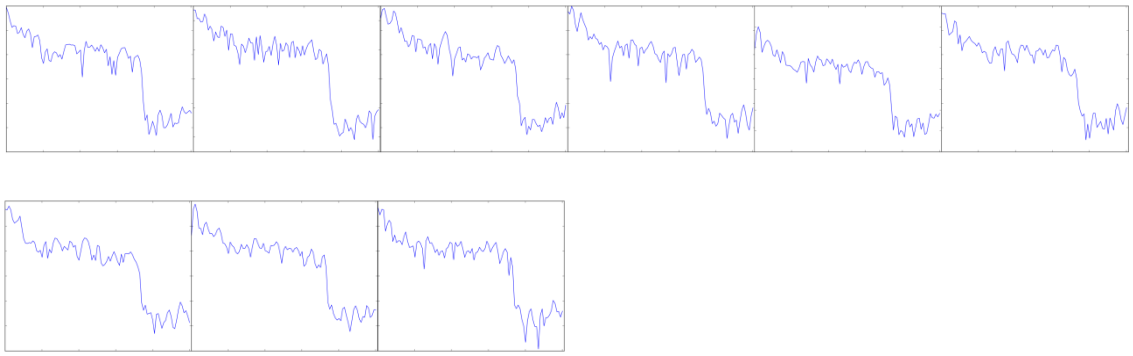


Figura 29. Frecuencias de un audio de prueba y una grabación de prueba.

Una vez tenemos esta información, el matching se lleva a cabo de forma iterativa de modo que se respete la secuencialidad de los fragmentos en ambos audios. Esto quiere decir, que para encontrar el mejor encaje del fragmento grabado dentro del audio original se calcula el valor de similitud de forma conjunta entre ambas señales.

Para el caso del ejemplo, en primer lugar se calcula el valor de similitud entre los primeros fragmentos de ambas señales tal y como se muestra en la Figura 30. De este modo obtendremos tres valores de similitud (uno por cada fragmento). Para tener una visión general de cuán similar son ambos audios en esa posición concreta, se calcula el valor medio de los tres valores de similitud y nos quedamos con él.

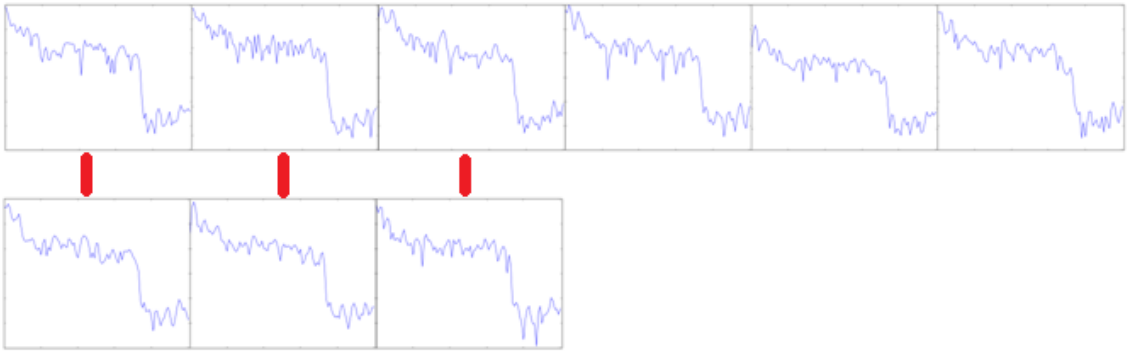


Figura 30. Correspondencia en la primera iteración del algoritmo de matching.

De forma similar se realiza el cálculo de la media de los valores de similitud entre los fragmentos que se señalan en la Figura 31; es decir, de los fragmentos adyacentes y consecutivos al primero, analizado en la iteración anterior. Este cálculo nos dará otro valor de similitud, el cuál compararemos con el valor de similitud calculado en la iteración anterior, quedándonos con el mayor de los valores.

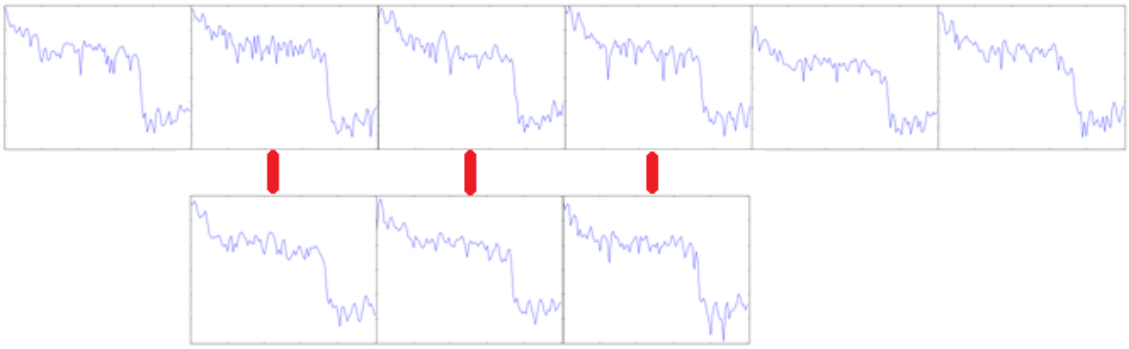


Figura 31. Correspondencia en la segunda iteración del algoritmo de matching.

Este proceso ejecutado de forma iterativa a lo largo de la señal de audio de la película original, nos devolverá finalmente una posición dentro de dicha señal, que será donde mejor encaja la señal de audio grabada. A parte de la posición de "mejor encaje" de la señal grabada, el algoritmo también devuelve ese valor de similitud correspondiente a dicha posición. A partir de este punto pasará a llamarse *valor de calidad* al *valor de similitud* nombrado hasta el momento.

Por otro lado, hacer una aplicación en la que base su resultado solamente en un fragmento de audio grabado hace bastante dependiente a la aplicación de ese fragmento en concreto. Para evitar que esto suceda y hacer que el algoritmo sea lo más robusto posible ante ruido y otros factores que pueden hacer fallar al algoritmo de matching se ha diseñado y desarrollado una unidad de control de resultados.

Básicamente esta unidad de control de resultados, gestionará los resultados obtenidos por parte del algoritmo de matching y será la que decida finalmente si la aplicación muestra los subtítulos o no de una determinada posición.

Para llevar a cabo esta gestión se tiene, por un lado, una variable de *referencia* que apuntará a la posición de la película donde se cree, en un momento determinado, que se encuentra la película; y por otro lado, una variable que controlará la *tasa de acierto* de la variable *referencia*, anteriormente nombrada.

Para ayudar a explicar el funcionamiento de estas variables se muestra a continuación el pseudocódigo del cálculo de estas variables:

```
1.  correlation = abs(reference – positionTime(index)) / 100
2.  if correlation > 0.1:
3.      if successRate > quality:
4.          successRate -= quality / 2
5.      else:
6.          reference = positionTime(index)
7.          successRate = quality
8.  else:
9.      reference += positionTime(index)
10.     reference /= 2
11.     successRate += quality / 2
```

En primer lugar se busca la correlación existente entre el fragmento que se tiene de referencia y la posición del fragmento devuelto por el algoritmo de matching. Esta correlación se lleva a cabo siguiendo la Ecuación 6:

$$c = \frac{|r - pTime|}{100} \tag{6}$$

Ecuación 6. Correlación en el algoritmo de búsqueda por correspondencia.

Donde *c* es la correlación, *r* la posición de referencia y *pTime* la posición devuelta por el algoritmo de matching. Una vez calculado este valor comprobamos si se trata de un valor pequeño; en nuestro caso particular hemos decidido compararlo con 0.1. En otras palabras, si el resultado devuelto por el algoritmo de matching se encuentra 10 segundos por delante o 10 segundos por detrás de la posición que se tiene de referencia, se considera como un resultado favorable y por lo tanto se refuerza la confianza que se tiene de esa referencia. Este refuerzo consiste en sumar a la tasa de acierto total la mitad de la tasa de acierto del fragmento devuelto por el matching. A su vez se desplaza la posición de referencia hasta la posición intermedia entre la propia posición de referencia y la posición devuelta por el matching.

Por otro lado, si la correlación es demasiado grande entre el resultado del matching y la posición de referencia, se consideran resultados ambiguos y como tal se lleva a cabo una corrección. La corrección pasa por comparar la tasa de acierto total que se tiene de la posición de referencia y la tasa de acierto de la posición devuelta por el matching. En el caso en el que la tasa de acierto total sea superior, su valor se verá afectado, restándosele la mitad de la tasa de acierto del fragmento devuelto por el matching. En caso contrario se pasa a tomar como nuevo punto de referencia la posición devuelta por el matching.

De esta forma, conseguimos que gradualmente el algoritmo vaya convergiendo a la posición en la que realmente se encuentra la película. Pero, ¿cómo saber si algoritmo ha convergido correctamente?, en este caso se ha hecho mediante la superación de umbrales.

El hecho de usar umbrales en ocasiones resulta "peligroso" ya que si imponemos un umbral muy bajo podemos obtener una respuesta rápida por parte del algoritmo pero menos fiable que si imponemos un umbral un tanto mayor. Para solucionar este problema, se ha diseñado una forma de obtener la ventaja de obtener rápidamente un resultado por parte del algoritmo al mismo tiempo que se asegure que se trata de una respuesta fiable.

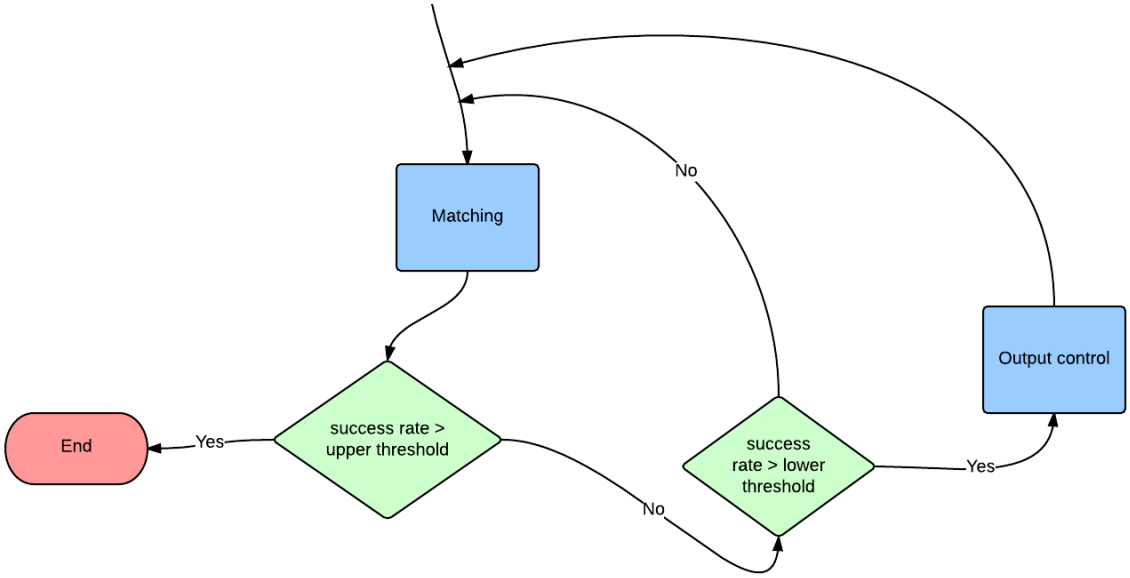


Figura 32. Flujograma de algoritmo de control de la salida de información.

En la Figura 32 podemos observar el flujo del proceso de control de salida de información. Se puede observar que una vez realizado el matching y recalculado el valor de la tasa de acierto total (*success rate*), se comprueba si es mayor que un cierto umbral superior. Si es así el proceso de búsqueda por correspondencia finaliza dando por bueno el resultado. Si no es así, se comprueba que la tasa de acierto total sea mayor que un umbral inferior para lanzar *la fase de control de la información de salida*. Si no se cumple ninguna de las dos condiciones el algoritmo seguirá iterando una y otra vez hasta que la tasa de acierto total sea lo suficientemente elevada para superar alguno de los dos umbrales.

En nuestro caso, hemos establecido el umbral superior (upper threshold) en 95% mientras que el umbral inferior (lower threshold) sea el 50%. Con esto conseguimos que el algoritmo muestre los subtítulos de manera rápida y con una respuesta medianamente fiable, mientras que al mismo tiempo se sigue comprobando, en un segundo plano, la posición en la que se encuentra la película hasta llegar a un umbral lo suficientemente fiable para así dar por finalizada la búsqueda.

Fase de control de la información de salida

Una vez la *fase de búsqueda por correspondencia* da la orden de mostrar los subtítulos (cuando la tasa de acierto total haya superado el umbral inferior), la *fase de control de la información de salida* calcula cuál es la posición exacta en la que se encuentra la película (ya que la fase de *búsqueda por correspondencia* determina en qué posición se encuentran los audios grabados desde micrófono). Esto se hace colocando una marca de tiempo cuando se comienza a grabar, para que, una vez sabido el momento concreto de la película donde corresponden los fragmentos de audio grabados, se le sume el tiempo que ha necesitado la aplicación para determinar esa información y así apuntar a la escena exacta en la que se encuentra la película.

Una vez conocida la posición por la que se encuentra la película, se muestran los subtítulos correspondientes a cada escena, tal y como se muestra en la Figura 33.

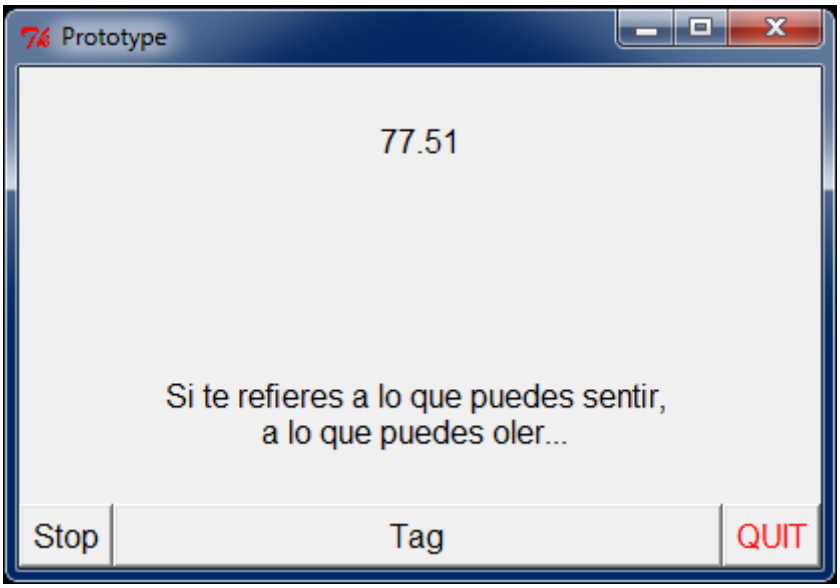


Figura 33. Ejemplo de funcionamiento de la interfaz de la aplicación.

2.3 Resultados obtenidos

A continuación se procede a mostrar los resultados obtenidos de una serie de pruebas a las que se le ha sometido al algoritmo, para poner a prueba su robustez, fiabilidad y rapidez.

Para realizar estas pruebas se ha hecho uso de 5 fragmentos de películas de 5 minutos cada uno aproximadamente y de una canción de una duración similar. En este caso, tanto las películas como la canción son:

1. A todo gas (2001)
2. Matrix (1999)
3. Resacón en Las Vegas (2009)
4. Skyfall (2012)

5. Un ciudadano ejemplar (2009)

6. Ahora quien (Mark Anthony, versión salsa)

A partir de este punto, las gráficas que se muestren en las figuras posteriores, corresponderá cada una a los resultados obtenidos del algoritmo en cada una de las películas y la canción, colocadas tal y como se muestra en la Tabla 1:

1.	2.
3.	4.
5.	6.

Tabla 1. Distribución de las gráficas de resultados en relación a los casos de prueba.

Los números de la tabla anterior corresponden con la enumeración anterior en la que se indicaban el nombre de las películas y la canción que se han usado en las pruebas.

En la Figura 34 se pueden observar los resultados obtenidos de la aplicación original.

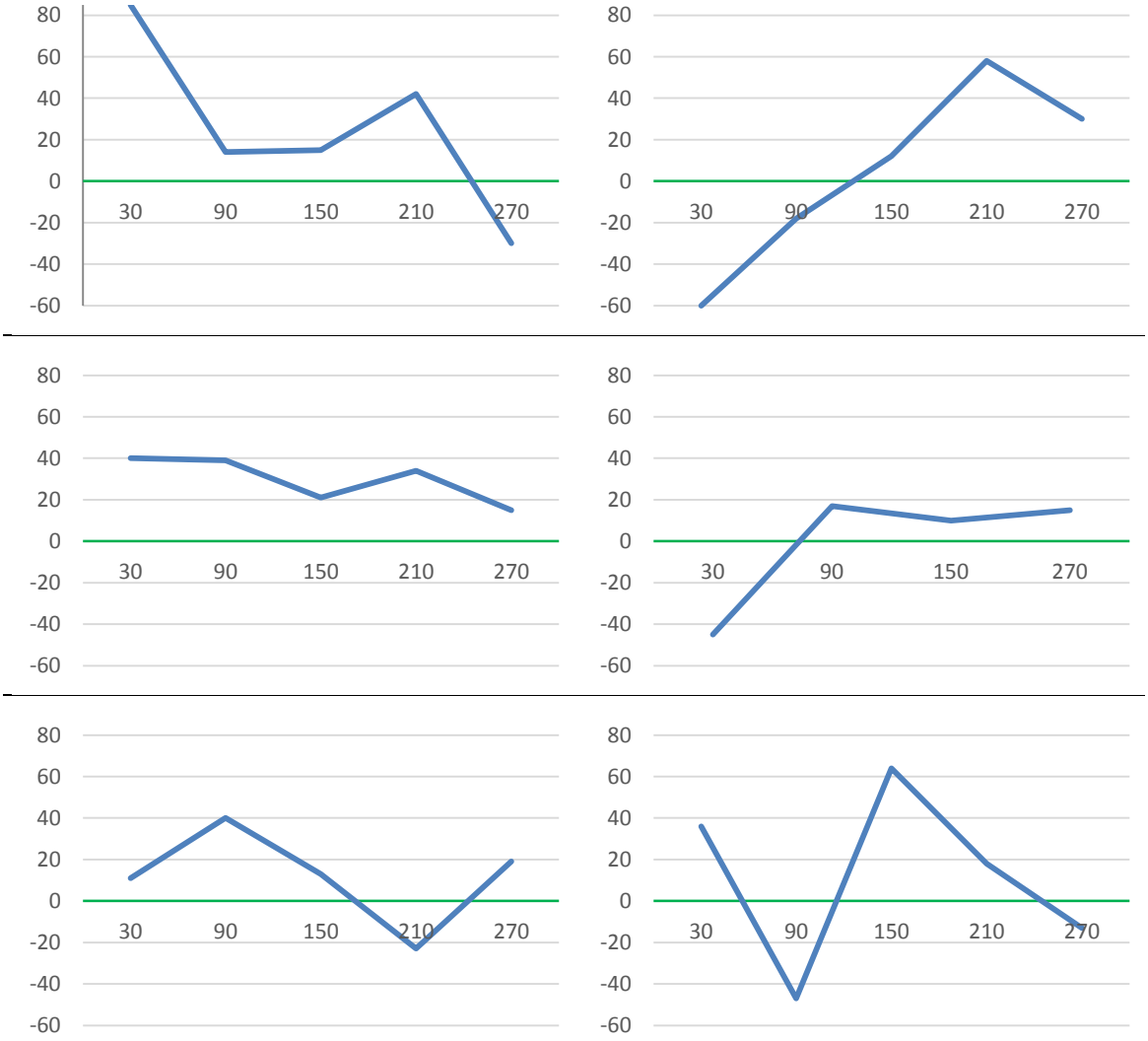


Figura 34. Resultados obtenidos con la versión original del programa.

En primer lugar se procederá a explicar los resultados mostrados en la Figura 34. En la Figura 34 se intenta plasmar cuál es el funcionamiento del algoritmo tanto en la validez de los resultados como en el tiempo de respuesta de los mismos. Dado que la validez del resultado se basa en un valor binario; es decir, muestra los subtítulos de la escena actual (resultado favorable) o por el contrario muestra los subtítulos de otra escena (resultado desfavorable), se considerarán en las gráficas, los valores positivos aquellos donde el algoritmo se ha ubicado correctamente y los valores negativos todo lo contrario, siendo cada uno de los valores los segundos que ha tardado el algoritmo en devolver dicho resultado. En este caso, el algoritmo devuelve un resultado cuando la tasa de acierto total alcanza el umbral inferior que se había establecido en 50 %.

Por otro lado en el eje de abscisas podemos observar la siguiente secuencia [30, 90, 150, 210, 270]. Esta secuencia de valores corresponde con ubicaciones dentro de la película en las que se ha realizado la prueba. Es decir, en el segundo 30 de la película se ha lanzado el algoritmo devolviendo un resultado en un tiempo determinado. Esto se ha repetido en distintos puntos de la película para observar el comportamiento del prototipo.

Esto se hace ya que, dependiendo de lo característico del momento en el que se realiza la grabación, el algoritmo tendrá mayor o menor facilidad para enganchar el momento en el que se encuentra la película. Por ejemplo, en el caso de la película 'Skyfall' (gráfica de la segunda fila y segunda columna) se puede observar que hay un tramo en el que no se tiene valor. Este hecho ocurre cuando se lanza el algoritmo de búsqueda por correspondencia a partir del segundo 150 del fragmento de la película, y lo que sucede es que el fragmento de la película llega a su final sin que el algoritmo haya devuelto un resultado aún.

Esto sucede porque el algoritmo resulta bastante costoso computacionalmente, y el tiempo que necesita para procesar un segundo de grabación normalmente es mayor a ese segundo de grabación. Este hecho hace que se acumule gran cantidad de información cuando el algoritmo no logra enganchar la escena rápidamente, traduciéndose en retardos importantes del algoritmo a la hora de devolver un resultado.

Volviendo a la Figura 34, cuando la gráfica toma valores por debajo de la línea verde que señala el 0, significa que el algoritmo ha devuelvo un resultado erróneo; es decir, que muestra los subtítulos de otra escena. En el caso de la película 'Resacón en Las Vegas' (gráfica de la segunda fila y primera columna) observamos que los resultados son todos positivos, lo cual nos indica que se trata de una película con características frecuenciales singulares y por lo tanto el algoritmo logra discernir fácilmente en minuto y segundo exacto en el que se encuentra la película.

En contrapartida podemos observar los resultados devueltos por el algoritmo para el fragmento de la película Matrix (gráfica de la primera fila y segunda columna) o para la canción (gráfica de la tercera fila y segunda columna) donde no le resulta tan sencillo a la aplicación encontrar el punto de enganche entre la película/canción y la grabación.

En definitiva, lo que se intenta indicar en las gráficas que se muestran en la Figura 34 es la capacidad que tiene el algoritmo de detectar correctamente un instante determinado a lo largo de cada una de las películas y de la canción de prueba.

Sin embargo, el algoritmo desarrollado ha sido implementado de forma que sea lo más parametrizable posible, de tal modo que las variables principales que afectan en mayor medida al funcionamiento del mismo puedan ser modificadas.

Básicamente los principales parámetros de los que depende el algoritmo son:

- El porcentaje de solapamiento con los intervalos adyacentes.
- El tamaño de los intervalos de tiempo en los que se divide la señal, en segundos.
- El tamaño de la ventana de frecuencias a la que se reduce el resultado de la FFT.
- El tamaño del fragmento grabado, en segundos.

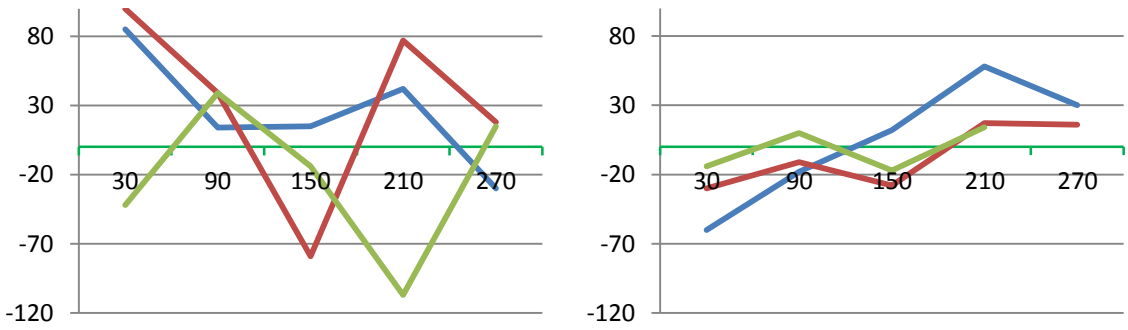
En un principio, cuando se desarrolló la aplicación original, se tomaron una serie de valores iniciales con los que la aplicación llegó a funcionar. Los valores iniciales son los que se muestran en la Tabla 2.

Variable	Valor
Porcentaje de solapamiento	50
Tamaño de los intervalos	0.1
Tamaño de la ventana de frecuencias	256
Tamaño del fragmento grabado	1

Tabla 2. Tabla de parámetros por defecto del algoritmo original.

Para estudiar el comportamiento del algoritmo en función de estos parámetros y analizar así en cuánto afectan los mismos al algoritmo para devolver un buen resultado, se ha llevado a cabo un pequeño estudio donde se muestra el tiempo de respuesta del algoritmo en función de los parámetros en los que se basa.

En primer lugar se observará el impacto que tiene el porcentaje de solapamiento en el algoritmo. En la Figura 35 se muestra la comparación, entre los diferentes casos de prueba, de los resultados obtenidos por la aplicación con los parámetros originales (**azul**), con el solapamiento al 20% y resto de parámetros originales (**rojo**) y con el solapamiento al 80% y resto de parámetros originales (**verde**).



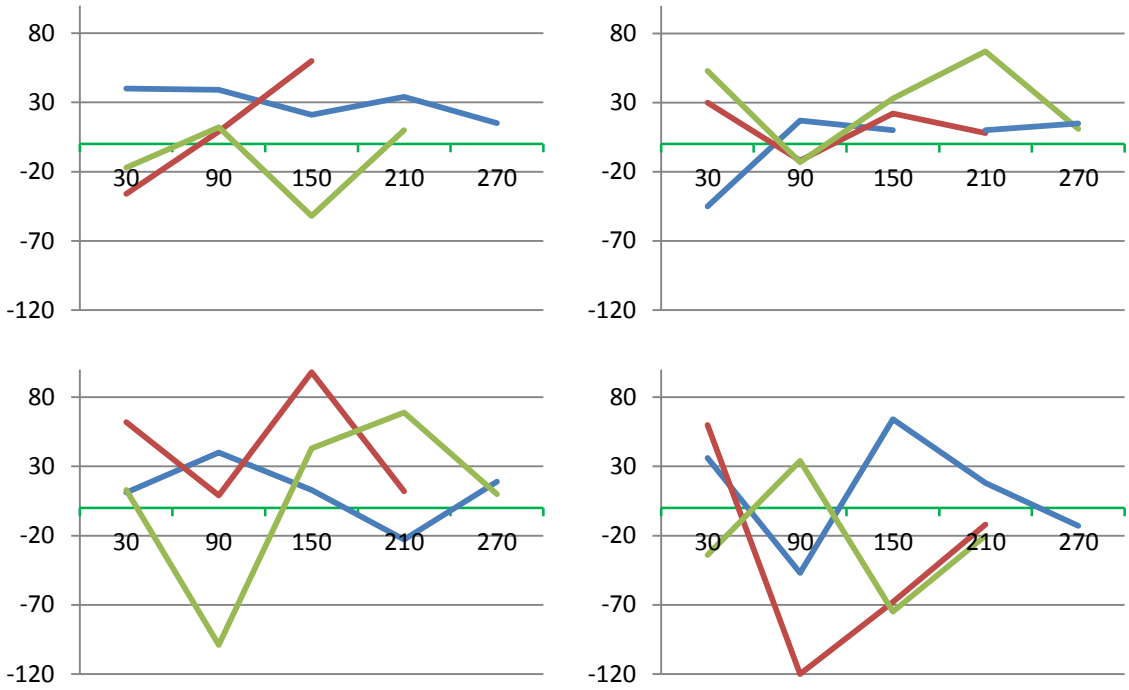
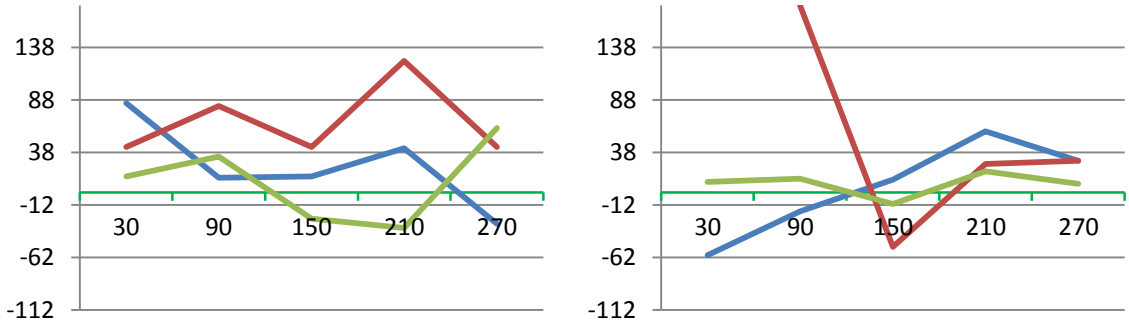


Figura 35. Comparativa de resultados con configuraciones del porcentaje de solape.

En la Figura 35 podemos observar en sus gráficas, como el comportamiento del algoritmo para las diferentes configuraciones del porcentaje de solapamiento es bastante irregular. No se observa ninguna mejoría generalizada cuando el valor de porcentaje de solapamiento tiene un valor u otro. Por lo tanto, el resultado obtenido en cada caso es altamente dependiente del instante y de la película o canción que analice.

Por otro lado se comprueba cómo afecta el tamaño del intervalo en los resultados del algoritmo. En la Figura 36 se muestra la comparación, entre los diferentes casos de prueba, de los resultados obtenidos por la aplicación con los parámetros originales (**azul**), con el intervalo a 0.05 segundos y resto de parámetros originales (**rojo**) y con el intervalo a 0.2 segundos y resto de parámetros originales (**verde**).



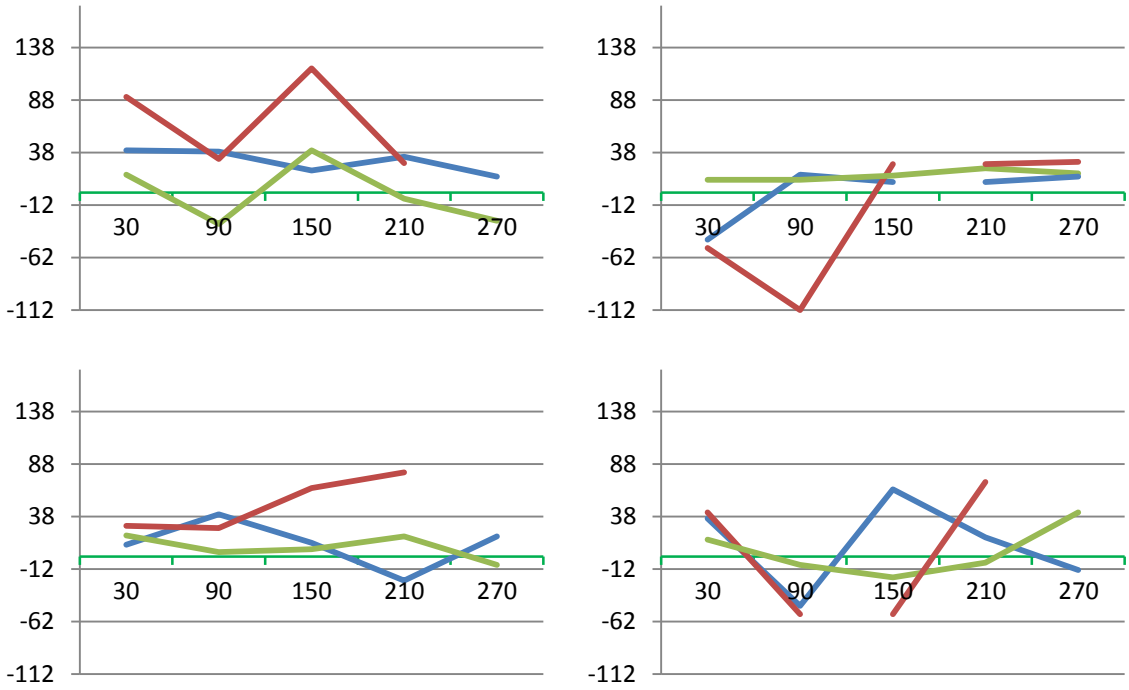


Figura 36. Comparativa de resultados con configuraciones del tamaño de los intervalos.

En la Figura 36 podemos observar que la línea **verde** (cuando el tamaño de los intervalos es de 0.2 segundos) los resultados son más estables; es decir, no se observan tantos altibajos como se pueden encontrar en los resultados resaltados con la línea **roja** (cuando el tamaño de los intervalos es de 0.05 segundos).

En el caso de la configuración del algoritmo con el tamaño de los intervalos a 0.05, los intervalos al ser tan pequeños y siendo la longitud de la grabación fija (1 segundo en este caso) tenemos un mayor número de intervalos. Esto supone un coste computacional mayor ya que hay que ejecutar el algoritmo de matching para un mayor número de intervalos. Esto sumado a que los intervalos, al ser más pequeños y haber menos distancia de tiempo entre ellos, hace que hayan muchos intervalos con características frecuenciales similares, lo que supone un problema añadido al algoritmo para localizar de forma exacta la posición en la que se encuentra. Por este motivo se puede observar en la Figura 36 altibajos en los resultados obtenidos con esta configuración.

A su vez se comprueba el factor de impacto del tamaño de la ventana de frecuencias en los resultados del algoritmo. En la Figura 37 se muestra la comparación, entre los diferentes casos de prueba, de los resultados obtenidos por la aplicación con los parámetros originales (**azul**), con la ventana de frecuencias a 128 valores y resto de parámetros originales (**rojo**) y con la ventana de frecuencias a 512 valores y resto de parámetros originales (**verde**).

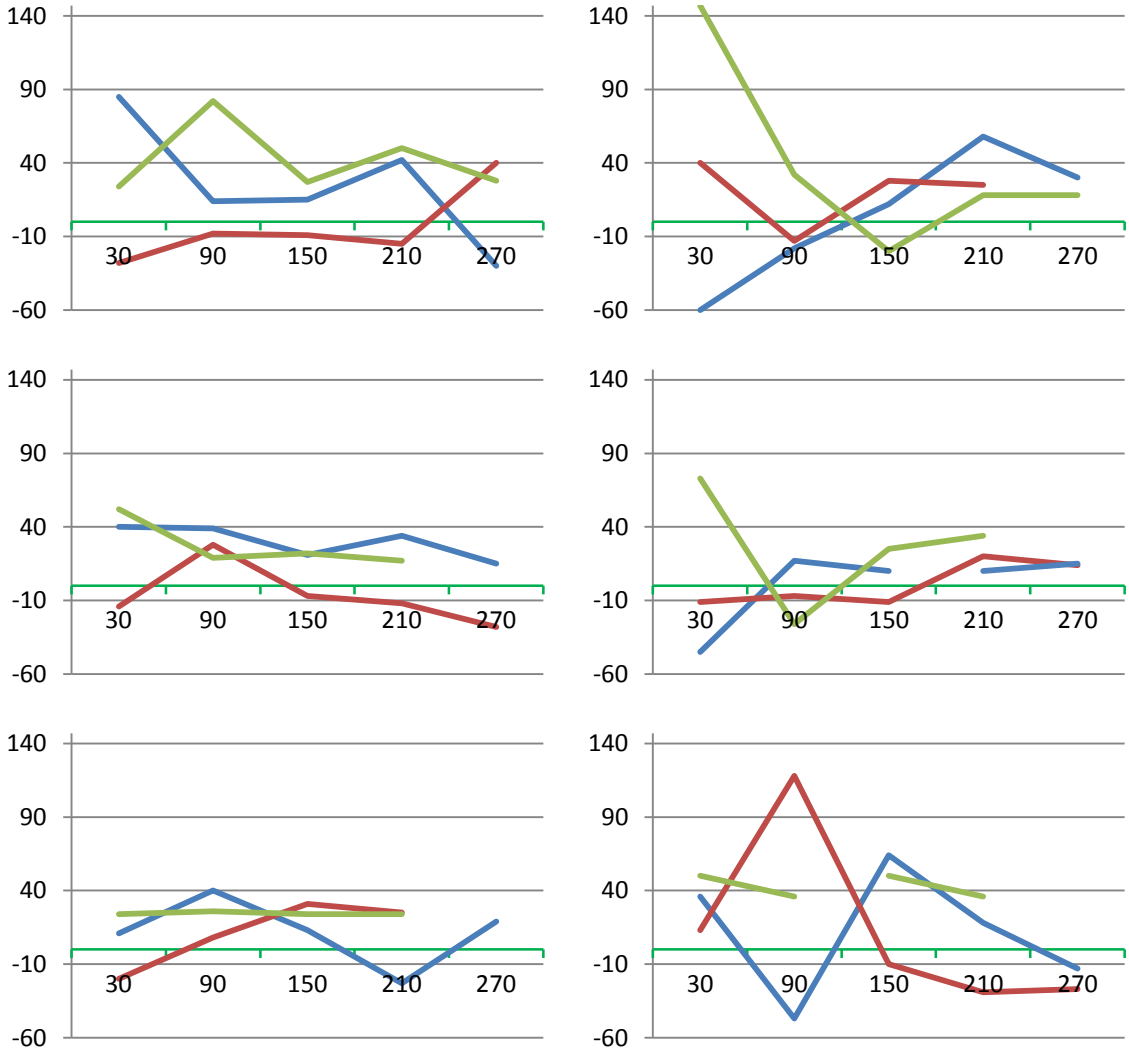


Figura 37. Comparativa de resultados con configuraciones de la ventana de frecuencial.

En cuanto a los resultados obtenidos del algoritmo con diferentes configuraciones del tamaño de la ventana de frecuencias, podemos observar en la Figura 37 como, para la configuración con la ventana de frecuencias de tamaño 512 (línea **verde**) la mayoría de los resultados son positivos. Sin embargo para la otra configuración, (línea **roja**) con la ventana de frecuencias de tamaño 128, se puede observar que la mayoría de los resultados son negativos, lo que quiere decir que el algoritmo ha devuelto una posición errónea de la película.

Esto se debe a que, como el tamaño de la ventana de frecuencias es menor en el caso de la primera configuración (línea **roja**), se tiene menor cantidad de información relativa a las frecuencias de cada fragmento. Esto se traduce en un incremento en la dificultad para que el algoritmo pueda discernir entre un fragmento u otro. En contrapartida observamos que ocurre totalmente lo contrario cuando se tiene más información frecuencial, donde el algoritmo devuelve un resultado correcto en gran parte de las pruebas realizadas, dado que se tiene mayor información de cada fragmento.

Por último se observará el impacto que tiene el tamaño de la grabación en los resultados del algoritmo. En la Figura 38 se muestra la comparación, entre los diferentes casos de prueba, de los resultados obtenidos por la aplicación, con los parámetros originales

(**azul**), con 0.5 segundos de grabación y resto de parámetros originales (**rojo**) y con 2 segundos de grabación y resto de parámetros originales (**verde**).

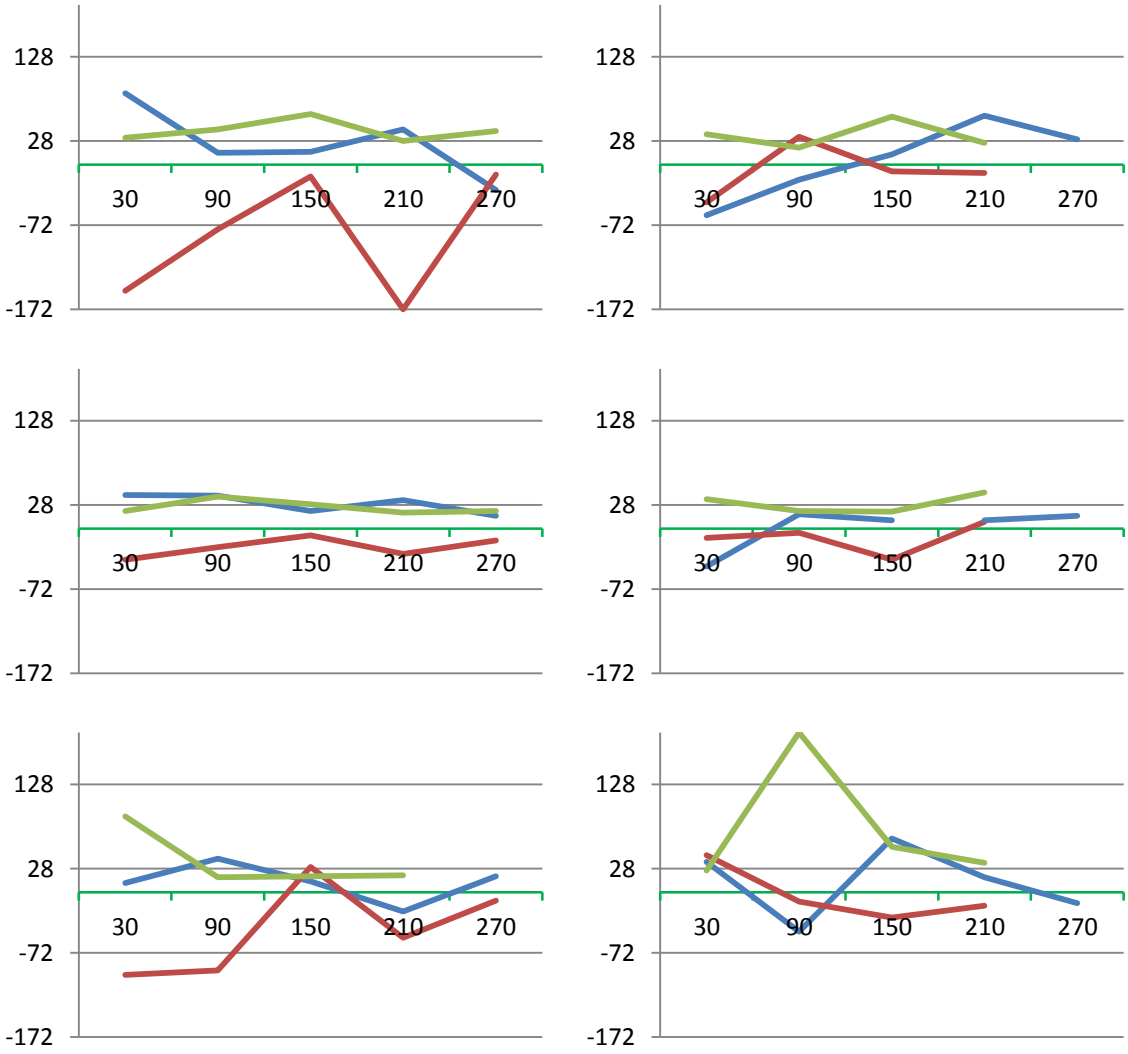


Figura 38. Comparativa de resultados con configuraciones del tiempo de grabación.

Por último, en la Figura 38 se muestran los resultados del algoritmo cuando se realiza la búsqueda por correspondencia con grabaciones de 0.5 segundos (línea **roja**), grabaciones de 1 segundo (línea **azul**, configuración original) y grabaciones de 2 segundos (línea **verde**). Para los diferentes casos podemos observar en las gráficas como a medida que el intervalo de la grabación va aumentando, los resultados se estabilizan, hasta tal punto que para la configuración del algoritmo en la que se toman grabaciones de 2 segundos (línea **verde**), todos los resultados devueltos son positivos.

Esto se debe a que al tener mayor número de fragmentos en la grabación, cuando se realiza el matching se tiene en cuenta más información y por lo tanto el resultado devuelto por el algoritmo es más fiable. Por otro lado, en comparación con las otras dos versiones se puede observar que para muchos casos, a pesar que la línea **verde** siempre está por encima del 0, también está por encima de las otras líneas; es decir, que esta configuración a pesar de que siempre nos devuelve un resultado favorable, la principal desventaja que presenta es su elevado tiempo de respuesta, ya que al tener mayor

tamaño cada fragmento, computacionalmente se traduce en un mayor número de operaciones.

Después de haber realizado diferentes pruebas con el algoritmo y con distintas configuraciones del mismo, procedemos a compactar éstas en la Tabla 3, donde se muestra una visión más generalizada de los resultados.

Versión	Tasa de resultados	Tasa de acierto	Segundos
Original	96.67 %	75.86 %	29.45 segs
Con porcentaje de solapamiento: 20 %	83.33 %	64 %	39.81 segs
Con porcentaje de solapamiento: 80 %	90 %	55.56 %	28.86 segs
Tamaño de los intervalos: 0.05 segundos	80 %	83.33 %	60.5 segs
Tamaño de los intervalos: 0.2 segundos	100 %	66.67 %	20.35 segs
Tamaño de la ventana de frecuencias: 128	93.33 %	42.86 %	32.5 segs
Tamaño de la ventana de frecuencias: 512	80 %	91.67 %	38.72 segs
Tamaño de la grabación: 0.5 segundos	90 %	14.81 %	28.75 segs
Tamaño de la grabación: 2 segundos	86.67 %	100 %	40 segs

Tabla 3. Tabla final de resultados obtenidos con diferentes configuraciones.

En la Tabla 3, se puede ver de forma más resumida lo que se ha comentado anteriormente. Donde la *tasa de resultados* es el porcentaje en el que algoritmo ha devuelto un resultado antes de que el fragmento de película o canción llegase a su fin. De estos resultados devueltos, la *tasa de acierto* es el porcentaje de resultados positivos en relación al conjunto total de resultados. Y por último, los segundos, muestran el valor medio de segundos que tarda el algoritmo en devolver un resultado positivo.

En primer lugar, vemos que a pesar de las diferentes configuraciones probadas con diferentes valores para el porcentaje de solapamiento no obtenemos mejoras en comparación a la versión del algoritmo original. Esto lo observamos tanto en la tasa de aciertos como en la tasa de resultados.

Por otro lado, observamos que para diferentes configuraciones del algoritmo en cuanto al tamaño de los intervalos se refiere, el hecho de tener valores pequeños (como 0.05 segundos) hace que el tiempo de respuesta del algoritmo se dispare, ya que resulta más complicado discernir entre un fragmento u otro. Este hecho hace que también el algoritmo se comporte de una manera más irregular (tal y como se mostró en la Figura 36).

En cuanto a las configuraciones del algoritmo con diferentes tamaños de ventana de frecuencias observamos que, para tamaños muy pequeños la tasa de acierto desciende. Todo lo contrario ocurre cuando aumentamos el tamaño de la ventana de frecuencias,

aunque el tiempo de respuesta se vea afectado al tener que procesar mayor cantidad de información.

Por último, observamos algo muy similar a lo comentado anteriormente para las configuraciones del algoritmo en las que se graba intervalos de tiempo con diferentes longitudes. Donde a media que las grabaciones son mayores las tasas de acierto se ven claramente beneficiadas a pesar de que los tiempos de respuesta se vean afectados.

2.4 Otras pruebas realizadas

A pesar de haber obtenido unos resultados aceptables por parte de la aplicación, se llevaron a cabo otra serie de pruebas para intentar obtener mejores resultados; es decir, mayor tasa de acierto y mejores tiempos de respuesta.

En principio se planteó usar otra medida de comparación entre fragmentos, donde no sólo se tuviera en cuenta la frecuencia de los picos sino también su amplitud. En este caso se planteó una expresión para obtener un nuevo valor de similitud entre intervalos. Este valor viene dado por la Ecuación 7.

$$P = \frac{\sum_{i=1}^{n} \frac{of_y(j) - \left|of_y(j) - rf_y(i)\right|}{of_y(j)} \mid \forall i \in \{of_x(j) = rf_x(i)\} \wedge \{j \leq m\}}{m} \cdot 100 \qquad (7)$$

Ecuación 7. Expresión del nuevo cálculo de similitud entre fragmentos.

Donde *P* es el porcentaje de similitud, *n* es el número de picos en la grabación, *m* el número de picos en el fragmento original, *rf* las frecuencias de la grabación, *of* las frecuencias del fragmento original siendo *x* e *y* sus componentes; es decir, *x* su frecuencia e *y* su amplitud.

En la forma de comparación entre dos fragmentos que se había implementado hasta el momento en el algoritmo, y que se explicó anteriormente, lo que hacía era contar el número de picos coincidentes entre ambas señales. Esto quiere decir que, cuando ambas señales tenían un pico frecuencial en una misma frecuencia se sumaba un pico adicional (+1) a una variable contadora. Sin embargo, lo que se plantea a continuación es algo similar, con la diferencia que cuando haya un pico en común, se sume parte de ese pico, en función de la diferencia que exista entre las amplitudes de ese pico en ambas señales.

Para comprenderlo mejor haremos uso de un ejemplo. Supongamos un escenario como el que se planteó en la Figura 27. Para hallar el porcentaje de similitud solo se tiene en cuenta los picos que coinciden en ambas señales. En este caso coinciden en los puntos [3, 9, 12]. A partir de aquí se calcula la diferencia de amplitud entre los picos de ambas señales, teniendo como referencia la amplitud de los picos frecuenciales del audio original, de modo que obtenemos algo tal que así (en este caso, las señal de audio original será la gráfica izquierda de la Figura 27):

$$P = \frac{\frac{5 - |5 - 5|}{5} + \frac{6 - |6 - 7|}{6} + \frac{4 - |4 - 4|}{4}}{4} = \frac{1 + 0.83 + 1}{4} = \frac{2.83}{4} = 0.7083$$

Podemos observar que, al coincidir 3 de los 4 picos de la señal, en el mejor de los casos, el porcentaje de similitud será un 75 %. Como en los picos ubicados en $x = 3$ y $x = 12$, ambas señales tienen la misma amplitud, se cuenta como puntos exactos (+1). Sin embargo, para los picos ubicados en $x = 9$, en ambas señales su amplitud es distinta. Como tal, se calcula la diferencia existente entre ambas amplitudes y se divide entre el valor de la amplitud del pico que tenemos como referencia, que es la amplitud del pico de la señal original (gráfica izquierda de la Figura 27). De este modo observamos que ambos picos se parecen un 83%. Finalmente si obtenemos el valor similitud total entre ambos fragmentos vemos que corresponde con un 70.83%.

Para comprobar el funcionamiento de esta nueva medida de similitud se implementó el algoritmo con esta medida de similitud entre fragmentos y se realizaron las mismas pruebas que se mostraron en la sección anterior.

Los resultados obtenidos se pueden observar en la Figura 39, donde se muestran los valores por la aplicación, con la medida de similitud original (**azul**) y los valores por la aplicación con la nueva medida de similitud (**rojo**).

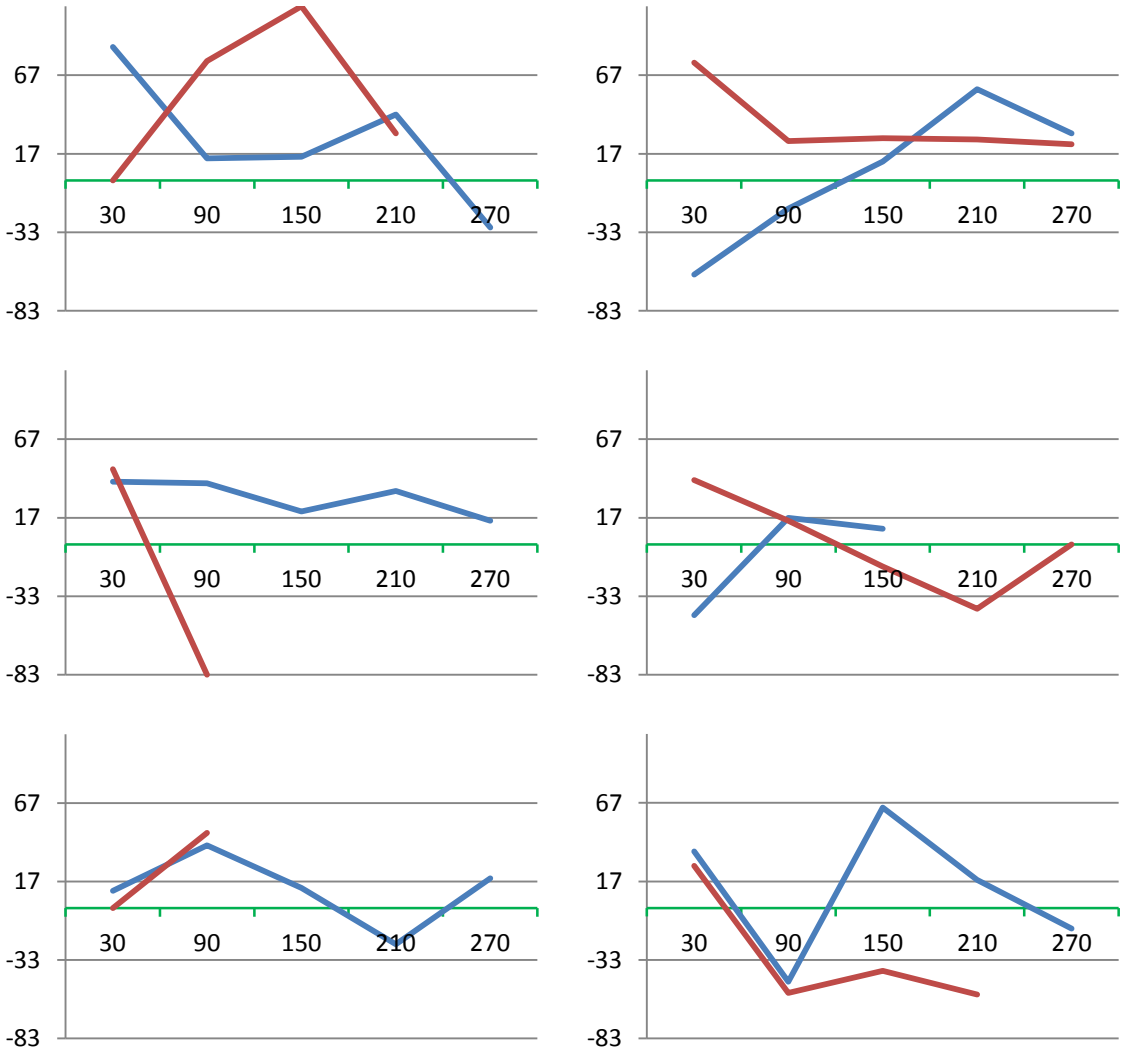


Figura 39. Comparativa de resultados con algoritmos con comparaciones diferentes.

En la Figura 39 se puede observar como los resultados obtenidos por la aplicación con la nueva medida de similitud (línea **roja**) tienen un comportamiento más irregular e incluso se puede observar que, la línea no tiene valores en muchos puntos. Esto se debe a que la película ha finalizado sin que el algoritmo haya podido detectar el momento en el que se encuentra. Esto en gran parte se debe al tiempo que necesita el algoritmo para calcular la nueva medida de similitud, ya que en este caso, se realiza un mayor número de operaciones que en la medida original.

Por otro lado, podemos visualizar la Tabla 4, donde se recaba toda la información mostrada en la Figura 39. En ella se puede observar que la tasa de resultados con la nueva versión es mucho menor que con la versión original. Esto se debe a la gran cantidad de tiempo que necesita esta versión para obtener un resultado, dato que también podemos observar en la columna *segundos*. Este hecho junto con la comparación de tasas de acierto, se puede concluir que no supone una mejora el hacer uso de esta nueva medida de similitud que tiene en cuenta la amplitud de los picos de frecuencia.

Versión	Tasa de resultados	Tasa de acierto	Segundos
Medida de similitud original	96.67 %	75.86 %	29.45 segs
Nueva medida de similitud	63.33 %	68.42 %	44 segs

Tabla 4. Tabla final de resultados con algoritmos con comparaciones diferentes.

Por otro lado, se intento implementar una versión del algoritmo en la que se etiquetara de algún modo cada intervalo de forma que indicara al algoritmo cuán característico era y de esta forma ayudar a discernir entre posiciones dentro de la película.

Para ello se realizó un algoritmo de 'Preprocesamiento' que se le aplica al fichero obtenido en la fase de 'Preprocesado' para: comparar cada fragmento con el resto, obtener un valor de confianza del mismo y generar un nuevo fichero, similar al anterior pero añadiendo la información de confianza entre intervalos.

Este nuevo proceso trata de comparar o buscar similitudes de cada fragmento con el resto. El resultado de esta comparación, que será un valor entre 0 y 1, nos indicará qué fragmento es más característico (cercano a 1) y cuál es más común a lo largo de la película (cercano a 0). Por lo tanto, en una señal de audio de *n* fragmentos, tendremos a su vez *n* valores de similitud de cada fragmento con el resto. Con estos *n* valores podemos obtener la desviación típica de dicho conjunto y de este modo obtener cuánto se distancia este fragmento del resto. Conviene recordar que la desviación típica viene dada por la Ecuación 8:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (8)$$

Ecuación 8. Ecuación de la desviación típica.

Donde $\bar{x}$ es la media de los valores del conjunto. Al valor devuelto por la expresión anterior, se le denomina 'confianza' del fragmento.

Por lo tanto, lo que añadimos en el nuevo fichero como intervalo de confianza es la desviación típica normalizada entre 0 y 1, siendo 1 el mayor valor de intervalo de confianza obtenido a lo largo del proceso.

De esta forma, el algoritmo cuando realice el matching, multiplica el valor de confianza de dicho intervalo con el valor de la tasa de acierto del mismo (devuelto por el algoritmo de matching) y obtiene así un valor de similitud diferente, el cuál tiene en cuenta la singularidad de cada fragmento.

Luego de haber implementado esta nueva versión del algoritmo, se realizaron varias pruebas obteniendo malos resultados. Es decir, que el intento de facilitar al algoritmo un valor de confianza de cada fragmento para obtener resultados más robustos resultó inútil, ya que si la película se encontraba en una posición donde predominaban silencios, los valores de confianza de estos intervalos eran muy cercanos a 0 y por lo tanto era casi imposible que el algoritmo encontrase un ‘enganche’ en esa posición.

3. Conclusiones y línea de trabajos futuros

A continuación se comentarán las principales conclusiones a las que se ha llegado a lo largo del desarrollo y documentación de este trabajo fin de máster.

3.1 Conclusiones

Ante todo se ha de comentar que el objetivo principal planteado, de desarrollar un prototipo funcional de aplicación móvil para que personas con discapacidad sensorial pudieran disfrutar de contenido audiovisual, se ha conseguido. Se ha podido desarrollar un prototipo de aplicación en el que se ha demostrado que el funcionamiento planteado es posible.

Este último hecho hace que el objetivo a medio-largo plazo que se planteó es posible, ya que el algoritmo desarrollado a lo largo de este trabajo fin de máster es totalmente trasladable a otras plataformas, como pueden ser las plataformas móviles en forma de aplicación.

Se ha demostrado a lo largo del documento las altas tasas de acierto obtenidas por el algoritmo, así como sus tiempos de respuesta. Se ha documentado y mostrado a través de gráficas y tablas de resultados, cómo existe una relación directa entre tasas de acierto y tiempos de respuesta. Es decir, que hay un estrecho vínculo entre ambos valores, donde si queremos una aplicación totalmente robusta y confiable, debemos sacrificar en tiempo de respuesta de la aplicación. Sin embargo, si queremos un algoritmo que devuelva rápidamente un resultado, debemos correr el riesgo que se devuelva un resultado erróneo.

3.2 Línea de trabajos futuros

A pesar de que este algoritmo ha sido desarrollado en un lenguaje de alto nivel, como es Python, sus tiempos de respuesta se consideran aún elevados. Estos registros podrían mejorarse al programar el algoritmo en un lenguaje de más bajo nivel, donde aproveche al máximo las características del hardware.

También hay que tener en cuenta que las pruebas realizadas se han llevado a cabo en un ordenador de sobremesa y en un ordenador portátil, cuyos recursos hardware son claramente superiores a los que dispone un dispositivo móvil. Por este motivo, se plantea la optimización de este algoritmo, intentando tomar algún tipo de atajo para evitar tanta carga computacional, ya que el objetivo final es que el algoritmo se ejecute en un dispositivo con recursos hardware limitados.

Este atajo que se plantea podría ser el uso de tablas hash, tal y como hace la famosa aplicación Shazam, donde se almacene en una tabla similar características frecuenciales correspondientes a cada instante de una película, agilizando el proceso.

A parte de las mejoras planteadas en los párrafos previos, si se consiguiese implementar un algoritmo robusto y fiable, como el que se presenta en este trabajo fin de máster, con mejores tiempos de respuesta y menos carga computacional, sería fácilmente trasladable dicha aplicación a dispositivos móviles como SmartPhones o Tablets, así como a

gadgets electrónicos como las Google Glass o SmartWatchs, tan populares en los últimos tiempos.

Abrir el abanico de dispositivos que puedan ejecutar un algoritmo como este, amplía a su vez los diferentes escenarios en los que podría usarse. Podría así usarse en la propia casa del usuario, así como en lugares públicos como pueden ser teatros, espectáculos, etc.

4. Bibliografía

- **Autores**: Avery Li-Chun Wang
  Título: An Industrial-Strength Audio Search Algorithm
  Lugar: Proceedings of the 4th International Conference on Music Information Retrieval
  Páginas: 0-23
  Año: 2003

- **Autores**: Dupraz, E.; Richard, G.
  Título: Robust frequency-based Audio Fingerprinting
  Revista: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference
  Páginas: 281-284
  Año: 2010

- **Autores**: Yang, C.
  Título: MACS: music audio characteristics sequence indexing for similarity retrieval
  Revista: Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop
  Páginas: 123-126
  Año: 2001

Anexo

En este último capítulo se comentará cómo instalar la herramienta software usada en este trabajo fin de máster así cómo ejecutar dicha aplicación.

Para el desarrollo del prototipo de aplicación se ha usado Python así como diferentes librerías adicionales para distintas funcionalidades. El desarrollo y pruebas del trabajo fin de máster se han llevado a cabo bajo Windows 7 y Windows 8.

El entorno de Python puede ser descargado de la siguiente URL:

https://www.python.org/download/

En nuestro caso, hemos instalado la versión de Python 2.7.6.

Una vez instalado el software de Python en el equipo Windows, procedemos a añadir al PATH de las variables de entorno la ruta donde se encuentra el fichero ejecutable de dicho software. Esto facilita el hecho de lanzar algoritmos desde la consola MS-DOS de Windows.

Para hacer esto debemos seguir los siguientes pasos:

1. Nos ubicarnos en 'Equipo' con el explorador de carpetas.
2. Hacemos el click con el botón derecho sobre la zona vacía del explorador y luego clicamos en 'Propiedades'.
3. Una vez aquí, clicamos en 'Configuración avanzada del sistema' ubicado en el menú de la izquierda de la pantalla que nos aparece.
4. Posteriormente se nos abre una nueva ventana. Ubicados en la pestaña de 'Opciones avanzadas', clicamos sobre 'Variables de entorno…'.
5. Se abrirá una nueva ventana, aquí seleccionamos la variable 'Path' de la parte superior y clicamos en 'Editar…'.
6. Por último, en 'Valor de la variable' nos ubicamos al final de la línea, insertamos un punto y coma ';', y seguidamente añadimos la ruta donde se encuentra el fichero ejecutable de Python (*python.exe*). En nuestro caso la ruta es la siguiente:

C:\Python27

Una vez se tiene el intérprete del lenguaje Python, descargamos e instalamos las siguientes librerías de Python de [aquí] (importante el orden):

- Pymedia
- Numpy
- Dateutil
- Pytz
- Pyparsing
- Six
- Setuptools
- Matplotlib
- p2exe
- Reportlab

- wxPython

En Windows se hace uso de estas librerías ya que no nos sirven los instaladores oficiales de cada modulo. Esto se debe a que dichos módulos están pensados para ser instalados en Linux.

Una vez instalado Python y todas las librerías de las que hace uso el algoritmo desarrollado en este trabajo fin de máster, procedemos a indicar cómo se debe lanzar dicho algoritmo.

En primer lugar, se presupone que hay un directorio en el equipo con los ficheros con el código fuente (*process_prototype.py* y *app_prototype.py*) del algoritmo con extensión '.py'. También se presupone que se tiene en un directorio del equipo un **fragmento** de una película con formato AVI ó WMV, así como un fichero de subtítulos en formato SRT para ese fragmento de película.

Es importante destacar que se debe tratar de fragmentos de una película de una duración recomendada de 5 minutos, ya que los algoritmos desarrollados son demasiado costosos computacionalmente para trabajar con películas enteras de larga duración.

Antes de lanzar la aplicación prototipo es necesario procesar el audio de la película y hacer correspondencia entre esa señal de audio y los subtítulos de cada fragmento de película. Para ello se lanzará en primer lugar el 'algoritmo de preprocesamiento'. Para ello abrimos una consola de comandos MS-DOS de Windows.

Posteriormente mediante el comando 'cd', nos ubicamos donde tengamos los ficheros del código fuente del algoritmo de procesamiento. Una vez allí lanzamos la ejecución del algoritmo pasándole por parámetros: en primer lugar la ruta hasta la ubicación de la película a procesar y posteriormente la ubicación hasta el archivo de subtitulado.

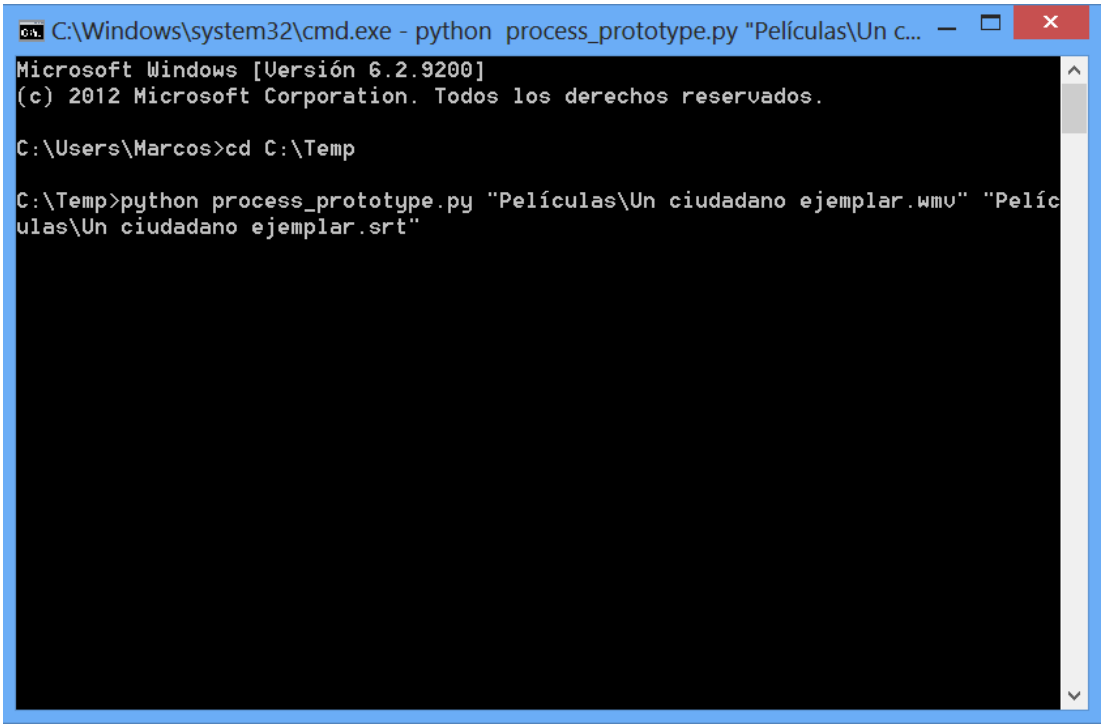


Figura 40. Ejemplo de cómo lanzar el algoritmo de procesado.

En la Figura 40 se muestra como se ha de lanzar el algoritmo de procesado de la película. Este algoritmo puede tardar varios minutos en terminar de generar un archivo binario con extensión '.proto' donde contendrá la información indispensable para el correcto funcionamiento del algoritmo prototipo.

Por último para lanzar el algoritmo prototipo basta con ubicarnos en el directorio donde se encuentra ubicado el código fuente de dicho algoritmo y lo lanzamos tal y como se hizo con la aplicación de preprocesamiento pero pasando por parámetro el fichero generado por el algoritmo anterior.

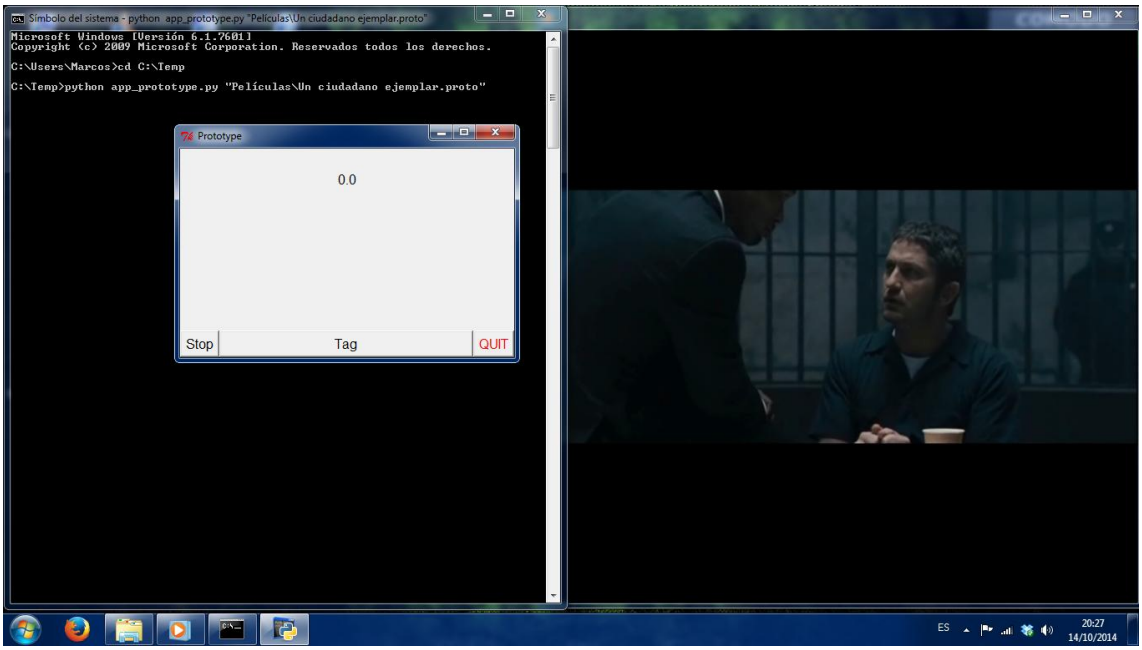


Figura 41. Ejemplo de la interfaz del algoritmo una vez lanzado.

En la Figura 41 se puede observar una instantánea de la pantalla una vez se haya lanzado el algoritmo prototipo. A un lado de la pantalla se puede ver el reproductor de Windows Media, el cual está reproduciendo una película. El audio de esta película está siendo reproducido por los altavoces, cuyo sonido es capturado por un micrófono conectado al ordenador al mismo equipo.

En cuanto se pulse el botón 'Tag' de la interfaz, el algoritmo prototipo procederá a analizar el audio capturado por el micrófono y a buscar la correspondencia con el audio del fragmento de la película para mostrar los subtítulos correspondientes a la escena actual.