CAPÍTULO 2: LA TÉCNICA DE *AUDIO FINGERPRINTS*

En este capítulo haremos una descripción teórica muy detallada de un sistema completo de detección de objetos de audio en tiempo real basado en la técnica de *audio fingerprints* o *huellas digitales de audio*.

En un primer apartado nos centraremos en una introducción de la técnica exponiendo su origen histórico y la importancia de su uso.

En el segundo apartado definiremos en qué consiste el sistema describiendo cada uno de los bloques que lo componen y comentando el gran número de variantes con que nos encontramos a la hora de desarrollar un sistema de este tipo.

A continuación, en el tercer apartado, podremos ver los parámetros más significativos a tener en cuenta a la hora de analizar estos sistemas y que nos permiten compararlos entre ellos; así como unos breves comentarios sobre la interrelación que existe entre estos parámetros.

En el cuarto apartado veremos algunas aplicaciones comerciales que hacen uso de estos sistemas.

Por último, tras el análisis de los diferentes algoritmos con que nos encontramos, describiremos el sistema que en este proyecto es objeto de estudio, análisis y optimización. Así, veremos los dos algoritmos necesarios para trabajar con *audio fingerprints*. En una primera parte del apartado veremos la explicación teórica y la implementación en Matlab del algoritmo de extracción, es decir, del algoritmo que nos permitirá a partir de un trozo de audio obtener el *fingerprint* correspondiente. Y en una segunda parte, se describirá el algoritmo de búsqueda, tanto desde el punto de vista teórico como de implementación en Matlab, que vamos a emplear a la hora de realizar las pruebas. Podemos avanzar que este algoritmo de búsqueda no será el óptimo, pero es perfectamente válido para el análisis que es objeto de este proyecto.

2.1. INTRODUCCIÓN

Todos los seres humanos tenemos características morfológicas únicas que nos diferencian. La forma de la cara, la geometría de partes de nuestro cuerpo como las manos, nuestros ojos y tal vez la más conocida, la huella dactilar, son algunos rasgos que nos diferencian del resto de seres humanos.

La identificación por medio de huellas dactilares constituye una de las formas más representativas de la utilización de la biometría. Esta ciencia consiste en métodos de identificación y autenticación de los seres humanos a través de características fisiológicas y de comportamiento (véase referencia [55]).

No podemos pensar que la biometría es una técnica de identificación futurista, pues desde hace varios siglos los hombres se han identificado por medio de este sistema. Esta comprobado, que en la época de los faraones, en el Valle del Nilo (Egipto) se utilizaban los principios básicos de la biometría para verificar a las personas que participaban en diferentes operaciones comerciales y judiciales.

Muchas son las referencias de personas que en la antigüedad han sido identificados por diversas características físicas y morfológicas como cicatrices, medidas, color de los ojos, tamaño de la dentadura, etc. Es en el siglo XIX cuando comienzan las investigaciones científicas acerca de la biometría con el fin de buscar un sistema de identificación de personas con fines judiciales (véase referencia [55]).

Con estas investigaciones se producen importantes avances y se comienzan a utilizar los rasgos morfológicos únicos en cada persona para la identificación. Ya en el siglo XX, la mayoría de los países del mundo utilizan las huellas digitales como sistema práctico y seguro de identificación. Con el avance tecnológico nuevos instrumentos aparecen para la obtención y verificación de huellas humanas. Hoy en día se comienzan a utilizar otros rasgos morfológicos como variantes de identificación, por ejemplo el iris del ojo, el calor facial o la voz.

Los sistemas basados en huellas dactilares o huellas humanas tienen, hoy en día, más de 100 años de antigüedad. En 1893, Francis Galton fue el primero en probar que no existen dos huellas humanas iguales. Fue, aproximadamente, unos 10 años mas tarde cuando se aceptó un sistema diseñado por Edward Henry para la identificación de huellas de seres humanos que es aceptado actualmente en EEUU y Gran Bretaña. Por su parte, el sistema de identificación dactilar en España fue creado por el Dr. Federico Olóriz Aguilera, quien identificó y nombró hasta 10 características propias de las huellas dactilares que ayudarían a reducir el número de comparaciones

necesarias para la identificación de un individuo en una base de datos (véase referencia [53]).

No obstante, hace unos 2000 años, los emperadores chinos ya se habían dado cuenta de que cada huella humana era única. De hecho, utilizaban a menudo la huella de su dedo pulgar para firmar documentos importantes.

Una huella dactilar humana está formada por una serie de surcos. Las terminaciones o bifurcaciones de los mismos son llamados 'puntos de minucia'. Cada uno de estos puntos tiene una característica y una posición única, que puede ser medida. Comparando esta distribución es posible obtener la identidad de una persona que intenta acceder a un sistema en general; por tanto, conceptualmente podemos interpretarla como un resumen de una persona o una firma que es única para cada persona (véase referencia [52]).

Es importante aclarar que una huella humana se diferencia de una descripción de esa persona en que la huella humana no nos permite reconstruir aspectos o características de la persona original. Por ejemplo, una huella humana no lleva ninguna información del color del pelo o de los ojos de la persona cuya huella estemos estudiando (véase referencia [54]). Este concepto habrá que tenerlo en cuenta cuando apliquemos la técnica de *fingerprints* a audio.

Análogamente a la utilidad de identificación y al concepto que todos sabemos de huella humana, buscamos un mecanismo similar para catalogar objetos de audio sin necesidad de recurrir al objeto en su totalidad. Así una huella digital de audio buscamos que sea un código único generado a partir de una forma de onda que pueda ser usada para identificar automáticamente una muestra de audio.

Recientemente, ha incrementado mucho el interés científico e industrial en computar huellas de objetos multimedia. El crecimiento del interés industrial queda demostrado por el gran número de compañías que están empezando a desarrollar esta técnica (véase referencias [11], [12], [13], [14], [15], [16], [17]) y la reciente solicitud de información sobre las técnicas basadas en *audio fingerprinting* por parte de la Federación Internacional de la Industria de la Fonografía (IFPI).

El principal objetivo de utilizar esta nueva técnica radica en conseguir mecanismos más eficientes para establecer la igualdad entre dos objetos multimedia, no comparándolos directamente entre ellos (tamaño de los objetos multimedia típicamente largo), sino comparando las huellas digitales asociadas a dichos objetos multimedia (tamaño de los *fingerprints* cortos por diseño).

Las grandes ventajas que obtenemos con este uso, frente a utilizar el contenido multimedia directamente por si mismo, son fundamentalmente tres:

- Reducción de memoria de almacenamiento requerida, gracias a que el tamaño de los *fingerprints* son mucho menores que el tamaño de los propios objetos multimedia.

- Comparación más eficiente, incluso ante distorsiones de la señal de audio, gracias a que las irrelevancias preceptúales quedan eliminadas.

- Búsqueda más eficiente en la base de datos, gracias a que los datos guardados ahora son más pequeños.

Como podemos deducir de las ventajas citadas, dentro de estos sistemas podemos distinguir dos bloques claramente diferenciables:

- un método para calcular los *fingerprints* de los correspondientes objetos multimedia *(**algoritmo de extracción**)*,

- y un método para buscar eficientemente *fingerprints* en una base de datos conformada por muchos de ellos *(**algoritmo de búsqueda**)*.

Resumiendo y centrándonos en el objetivo de este proyecto, en la actualidad nos encontramos con la necesidad de buscar y desarrollar eficientes huellas digitales de audio que nos permitan exprimir al máximo los beneficios que se obtienen con un sistema basado en huellas, como ya la experiencia nos ha demostrado con las huellas humanas. El encontrar sistemas eficientes pasa fundamentalmente por encontrar buenos algoritmos de extracción y de búsqueda de *fingerprints*. Es por eso que este trabajo se concentrará en la optimización de un algoritmo de extracción en particular, buscando una mejora en la eficacia computacional, tanto desde el punto de vista de tiempo de cómputo como de requerimientos de almacenamiento en memoria. Obviamente todo ello manteniendo la robustez del conjunto y el correcto funcionamiento; es decir, conservando las probabilidades de error con las que hay que hacer frente por debajo de unos valores considerables.

2.2. DEFINICIÓN DE UN SISTEMA BASADO EN *AUDIO FINGERPRINT*

Audio fingerprint es una técnica que se basa en obtener una representación compacta de un objeto de audio, es decir, en obtener una huella digital que resuma un objeto de audio, y trabajar con ella a la hora de identificar el objeto de audio en una base de datos en vez de realizar la búsqueda usando el objeto de audio original. Por tanto, la función para obtener el *fingerprint* debe hacer una especie de mapeo a partir del objeto de audio, el cual contiene un gran número de bits, dando como resultado una huella cuya longitud es un número pequeño y limitado de bits.

Un sistema completo basado en *audio fingerprint* o, como también se le conoce, *Content-based Identification (CBID)*, consiste, bajo una descripción muy general, en extraer características preceptúales relevantes de objetos de audio, en nuestro caso anuncios, obteniendo así los correspondientes *fingerprints*, y almacenarlos en una base de datos junto con la información que deseemos obtener en un futuro de esos objetos de audio. Una vez que el sistema conforma la base de datos, comienza a trabajar en tiempo real con la señal de audio de entrada. Así, el sistema va recogiendo trozos de una longitud fija de esa señal, calcula la huella digital de ese trozo de audio, y busca en la base de datos el *fingerprint* más parecido. Como resultado, usando los *fingerprints* de la base de datos y el que va calculando en tiempo real junto con eficientes algoritmos de búsqueda, podemos identificar, en un periodo de tiempo relativamente corto, qué objeto de audio, o anuncio, se está emitiendo en cada momento.

A pesar de las diferencias que hay en la tarea de identificación entre las diferentes técnicas basadas en *fingerprints*, todas comparten una serie de aspectos. Como ya comentamos anteriormente, hay dos procesos principales: el proceso de extracción o generación de huellas y el proceso de búsqueda en la base de datos. El proceso de extracción debe dar como resultado un *fingerprint* con los siguientes requerimientos:

- Ser un resumen perceptual del objeto de audio. Debe retener la mayor cantidad posible de información acústica, siempre que esta sea relevante. Además, tiene que permitir discriminar entre un gran número de *fingerprints*.

- Invariante ante distorsiones. Esto deriva del requisito de robustez. Tenemos que tener en cuenta que la señal de audio que va a llegar al sistema puede sufrir múltiples distorsiones, ya sea por las condiciones del canal de comunicación como por diferentes procesados que se le apliquen por algún determinado motivo; pues bien, ante los diferentes escenarios, el *fingerprint* resultante debe ser el mismo, o muy parecido, al que resultaría caso de no producirse distorsiones en el audio.

- Compacto. Es necesario un tamaño pequeño, pues hay que guardar en memoria y comparar un gran número de *fingerprints*. Sin embargo, una representación excesivamente pequeña puede no ser suficiente para discriminar entre diferentes objetos de audio, afectando así a la precisión, la fiabilidad y la robustez del sistema.

- Sencillo desde el punto de vista computacional. Por razones de complejidad el proceso de extracción de una huella no debería consumir un tiempo excesivo. Además, este proceso debe implicar un tiempo de cómputo lo suficientemente pequeño como para permitir que el sistema global funcione correctamente en tiempo real.

Para lograr estos requerimientos expuestos, todas las soluciones en que se puedan pensar implicarán un compromiso entre pérdida de información y reducción dimensional.

Veamos una descripción más detallada de un sistema genérico basado en *fingerprints* con algunas de las muchas alternativas que se pueden plantear para implementarlo. Tal y como vemos en la figura 1, el proceso o algoritmo de extracción consiste en un primer bloque llamado "front-end" o cabecera, que divide la señal de audio original de entrada al sistema en trozos o tramas y extrae un número de características discriminatorias y robustas a partir de cada trama; y un segundo bloque de modelado de *fingerprint*, que compone la representación final del *fingerprint*.

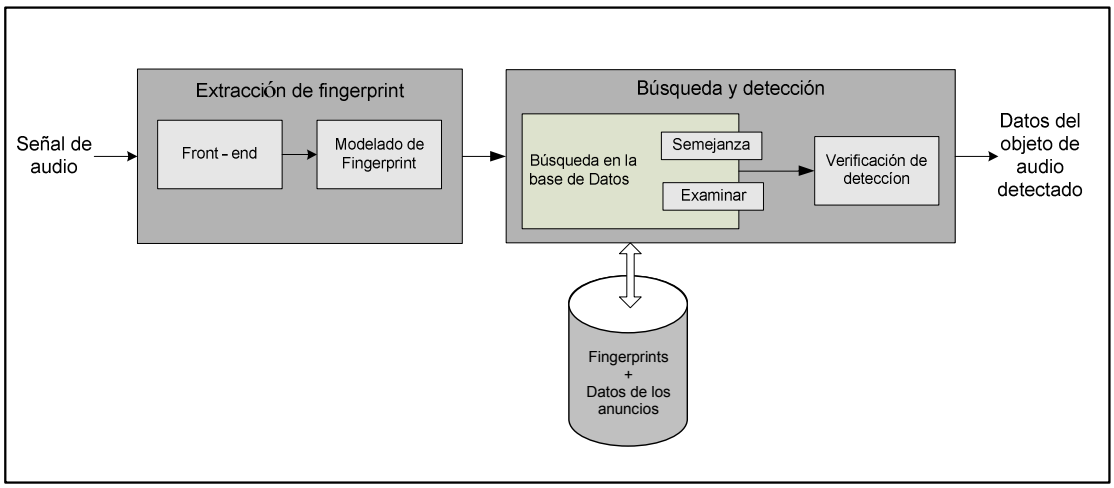

Figura 1. Estructura general de un sistema basado en fingerprints

Una vez obtenido el *fingerprint*, el proceso o algoritmo de búsqueda explora en una base de datos conformada por *fingerprints* aquel que tenga mayor semejanza con el calculado. Así pues, será necesario en este proceso un criterio de comparación. Si esta base de datos es excesivamente grande,

necesitaremos un buen método que acelere esta búsqueda. Algunos sistemas de búsqueda usan primero un método "simple" para la medida de semejanza entre *fingerprints* con el objetivo de hacer rápidamente un primer descarte de algunos de ellos y luego, un método más preciso (y por tanto, más lento) para hacer ya la elección de forma precisa entre los candidatos que quedaron tras pasar el primer filtro (véase referencia [36]). Hay también métodos que pre-calculan algunas distancias y construyen una estructura de datos que permiten reducir el número de cálculos a realizar en tiempo real (véase referencias [33], [34]). En el apartado 2.2.3 comentaremos estas variantes y algunas otras.

En general, un buen método de búsqueda de *fingerprints* debería ser (véase referencia [12]):

- **Rápido:** un escaneo secuencial de la base de datos puede ser demasiado lento para bases de datos grandes.

- **Correcto:** debe de devolver el objeto de audio y sus correspondientes campos de información guardados en la base de datos sin perder ninguno.

- **Eficiente en memoria:** la memoria temporal que se utiliza para el algoritmo debe ser relativamente pequeña.

- **Fácilmente actualizable:** la inserción, eliminación y actualización de nuevos objetos en la base de datos no debe afectar al método de búsqueda empleado.

El último bloque del sistema, "verificación de detección", verifica, basándose en unos umbrales previamente determinados, si el *fingerprint* extraído de la base de datos (por ser el de mayor semejanza con el calculado en tiempo real, que aún es desconocido) tiene que ser realmente identificado como tal.

Veremos en detalle cada uno de estos bloques y las diferentes posibilidades con que nos encontraremos en cada uno de ellos.

2.2.1 Bloque "front-end"

El bloque "front-end" o cabecera, como ya dijimos anteriormente, convierte la señal de audio de entrada al sistema en una secuencia de características relevantes que, posteriormente, servirá de entrada al bloque de modelado de *fingerprint*.

Algunos aspectos que hay que tener presente en el diseño de este bloque son:

- Conseguir una reducción considerable en la dimensión de la secuencia de salida del bloque con respecto al objeto de audio de entrada.

- Obtener unos parámetros preceptúales significativos (similares a los usados por el sistema de audición humano).

- Conseguir una secuencia que sea invariante, robusta ante distorsiones, ruidos de fondo, etc.

- Correlación temporal (sistemas que capturen espectros dinámicos).

Veamos, en la figura 2, las diferentes partes de las que se compone el bloque "front-end" y las múltiples variantes que se pueden presentar en cada uno de sus sub-bloques y en el bloque de modelado:

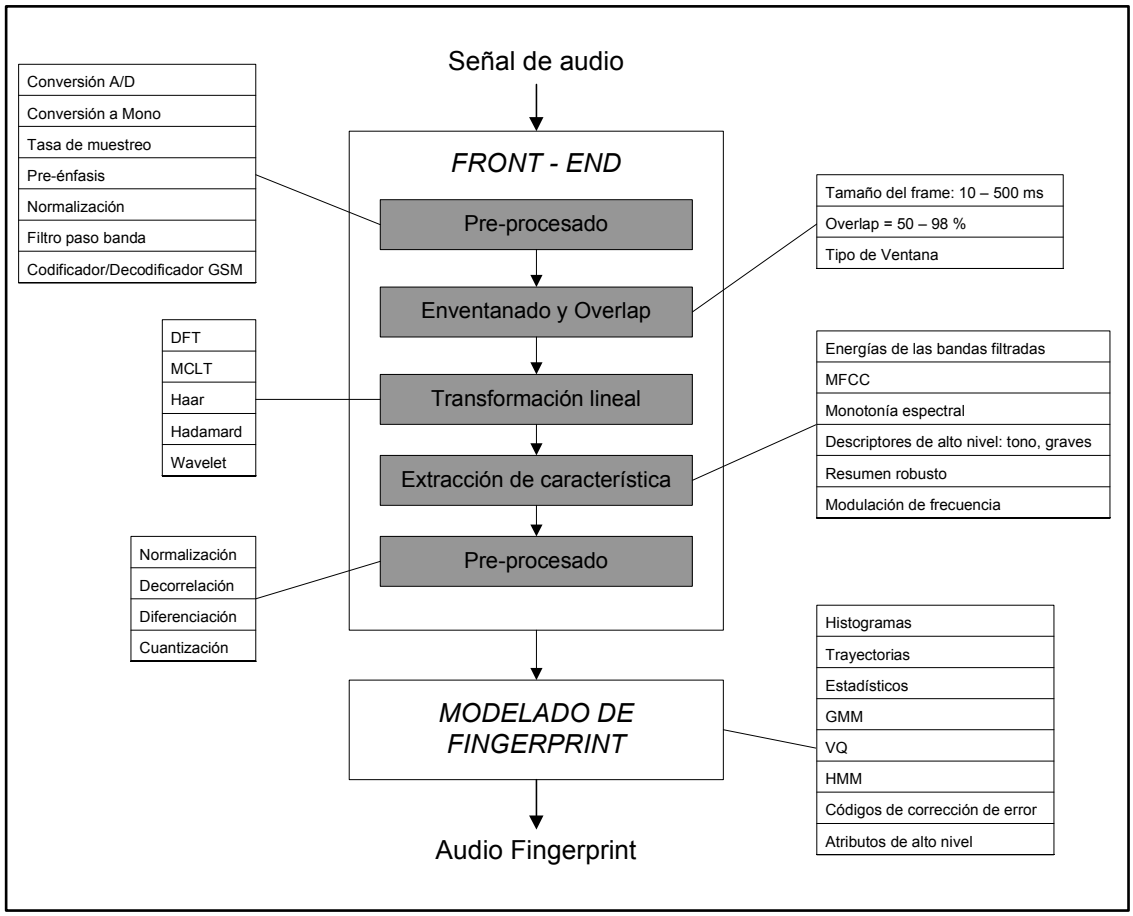

Figura 2. Estructura del bloque de extracción de *fingerprint*

En algunas aplicaciones, donde el objeto de audio a identificar está codificado, por ejemplo en *mp3*, es posible no implementar ciertas partes de este bloque y extraer las características preceptúales deseadas directamente a partir de la representación codificada del objeto de audio (véase referencia [8]).

a) Pre-procesado

En este primer paso, se digitaliza la señal de audio si fuera necesario, convirtiéndola así a un formato estándar, por ejemplo: el utilizado en este proyecto será PCM - mono (8 o 16 bits) con una frecuencia de muestreo de 8 KHz. En algunos casos la señal de audio será preprocesada con el objetivo de simular un canal, por ejemplo: un filtrado paso-banda en una aplicación de identificación en la cual la señal llega vía teléfono convencional. Otras veces se harán otros tipos de preprocesado, como por ejemplo: un codificador/decodificador GSM en un sistema de identificación de telefonía móvil, un pre-énfasis, una normalización de amplitud (limitando el rango dinámico entre (-1,1)), etc.

b) Enventanado y Overlap

Una suposición de la cual partimos a la hora de medir una característica de un objeto de audio es que la señal de audio puede ser considerada como estacionaria en un intervalo de tiempo de unos cuantos milisegundos. Por tanto, la señal es dividida en tramas o trozos de audio de tamaño comparable a la velocidad de variación de un evento acústico subyacente. Se utiliza una función de enventanado especial para cada trama con el fin de minimizar los efectos producidos por las discontinuidades al principio y al final de cada trama.

Se debe aplicar un *overlap* (es decir, una superposición entre ventanas subyacentes) tal que asegure robustez ante desplazamientos como pudiera ser por ejemplo, que el dato de entrada no este perfectamente alineado a la grabación que fue usada para generar el *fingerprint*.

Aclaremos qué significa este desalineamiento. En la figura 3 se muestran las tareas que realiza el bloque de "enventanado y overlap" de forma continuada sobre la señal de audio de entrada. Si ahora suponemos que vuelve a llegar la misma señal (es decir, se está emitiendo el mismo anuncio), es muy pequeña la probabilidad de que el inicio de la ventana coincida sobre los mismos instantes del audio que lo hizo la vez que se calcularon sus correspondientes *fingerprints* para almacenarlos en la base de datos, tal y como se muestra en la figura 3 con la ventana marcada en rojo. Por tanto es prácticamente nula la probabilidad de calcular exactamente la misma huella digital que tenemos almacenada. No obstante, esto no supone ningún tipo de impedimento para asegurar un correcto funcionamiento del sistema.

A este inconveniente de la más que probable inexactitud entre los *fingerprints* calculados y los almacenados, hay que añadirle el hecho de que por el propio algoritmo de extracción utilizado puede que sea imposible obtener dos iguales. Esto ocurre en el algoritmo utilizado en este proyecto (véase apartado 2.5.1); en nuestro caso, sólo sería posible encontrar dos huellas exactamente iguales si coincide el inicio de captura de la primera trama de la señal de audio, es decir de los primeros 0,37 segundos, con el inicio del anuncio. Esto es por ser el único caso en el que las energías de las bandas previas son nulas en ambas situaciones. Insistimos en que esto no deja de ser un mero inconveniente que no impide cumplir adecuadamente con la misión de identificación.

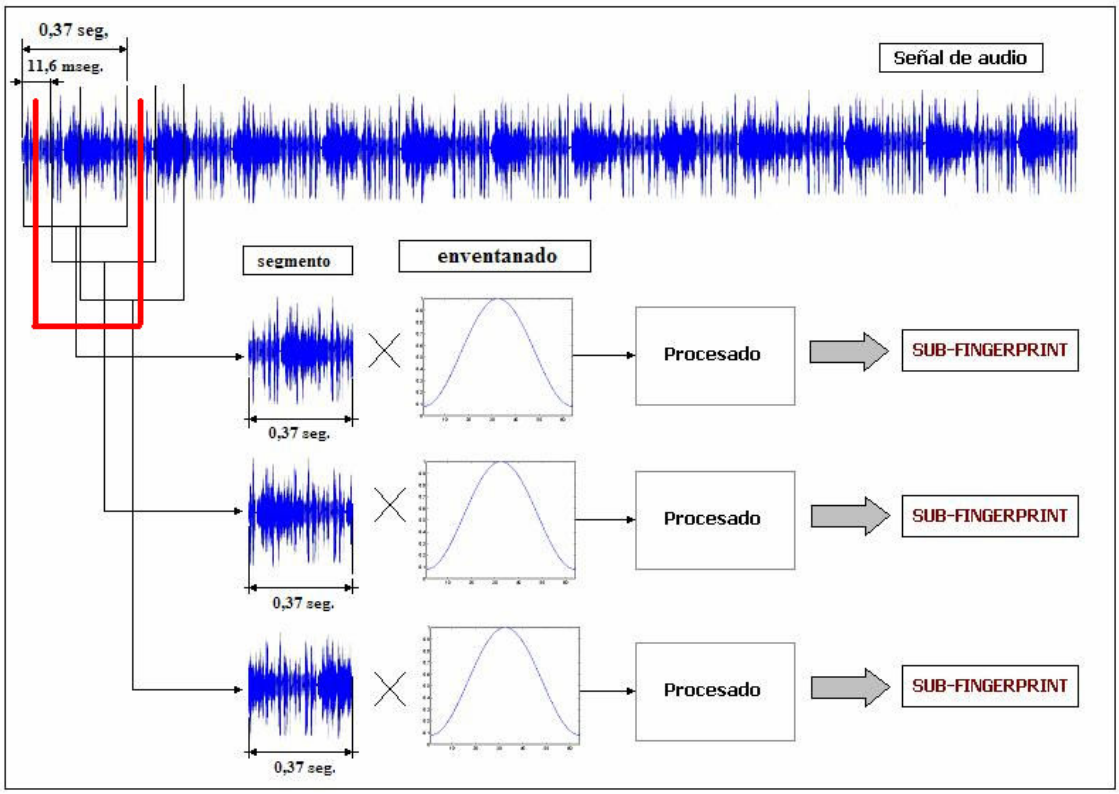

Figura 3. Tratamiento realizado por el bloque "Enventanado y Overlap"

Obviamente, tenemos que llegar a un compromiso entre esta robustez ante cambios y la complejidad computacional del sistema: cuanto más grande es el número de tramas calculadas por segundo, más robusto es nuestro sistema pero mayor carga computacional tiene.

c) Transformaciones lineales: Estimaciones espectrales

La idea que hay detrás de estas transformaciones lineales es la obtención de una secuencia de nuevas características a partir de la secuencia que teníamos anteriormente. De este modo, si la transformación es elegida

adecuadamente, podemos reducir la redundancia considerablemente y, por tanto, la dimensión de la nueva secuencia con respecto a la que teníamos.

Hay unas transformaciones que son óptimas en el sentido de compresión de la información y propiedades de decorrelación, como Karhunen-Loève (KL) o Descomposición en valores singulares (SVD) (véase referencia [19]). Sin embargo, estas transformaciones son computacionalmente complejas. Por esta razón, es más común utilizar transformaciones con menos complejidad. Por tanto, la mayoría de métodos CBID usan transformaciones estándares que pasan del dominio del tiempo al dominio de la frecuencia; estos es porque las características más importantes de una señal de audio, en nuestro caso una señal vocal, se encuentran en el dominio de la frecuencia. Estas transformaciones proporcionan una buena eficiencia en la compresión, en la eliminación de ruido y en los posteriores procesos.

La transformación más utilizada es la Transformada Discreta de Fourier (DFT); aunque se puede proponer usar otras como: la Transformada de Coseno Discreta (DCT), la Transformada de Haar o la Transformada de Walsh-Hadamard (véase referencia [20]). Podemos ver en un estudio comparativo entre la DFT y la Transformada de Walsh-Hadamard realizado por G. Richly (véase referencia [21]) que la DFT es generalmente menos sensible a cambios.

Incluso podemos encontrarnos con sistemas en los que ha sido utilizada la Transformada compleja modulada (MCLT) (véase referencias [22], [23]).

d) Extracción de características

Una vez que tenemos la representación tiempo-frecuencia, se aplican transformaciones adicionales para generar el vector final. En este paso, nos encontramos con una gran variedad de algoritmos. El objetivo de todos ellos es obtener una reducción en la dimensión del vector de salida del bloque "front-end" y, al mismo tiempo, incrementar la invarianza ante distorsiones. Es habitual basarse en el sistema auditivo humano para escoger los parámetros preceptúales más significativos. Por tanto, muchos sistemas extraen características llevando a cabo un análisis de las bandas críticas del espectro.

Como vemos en la figura 4, se pueden utilizar, por ejemplo, los Coeficientes Cepstrales en las frecuencias de Mel (MFCC) (véase referencias [24], [25]); o bien, la Medida de la Monotonía Espectral (SFM), consistente en una estimación de la calidad del tono o del ruido para una banda espectral (véase referencia [26]); o bien, "vectores representativos de bandas", consistentes en listas ordenadas de índices a aquellas bandas con tonos destacables, por ejemplo con picos de amplitud significativos (véase referencia [27]); o bien, las energías de 33 bandas espectrales distribuidas logarítmicamente para obtener un vector resumen, el cual recoge en un *bit* si la diferencia entre las energías de dos bandas consecutivas es positiva o negativa

(trabaja al mismo tiempo en el dominio del tiempo y de la frecuencia). Esta última característica es la que utilizamos en el algoritmo desarrollado en este proyecto (véase apartado 2.5.1).

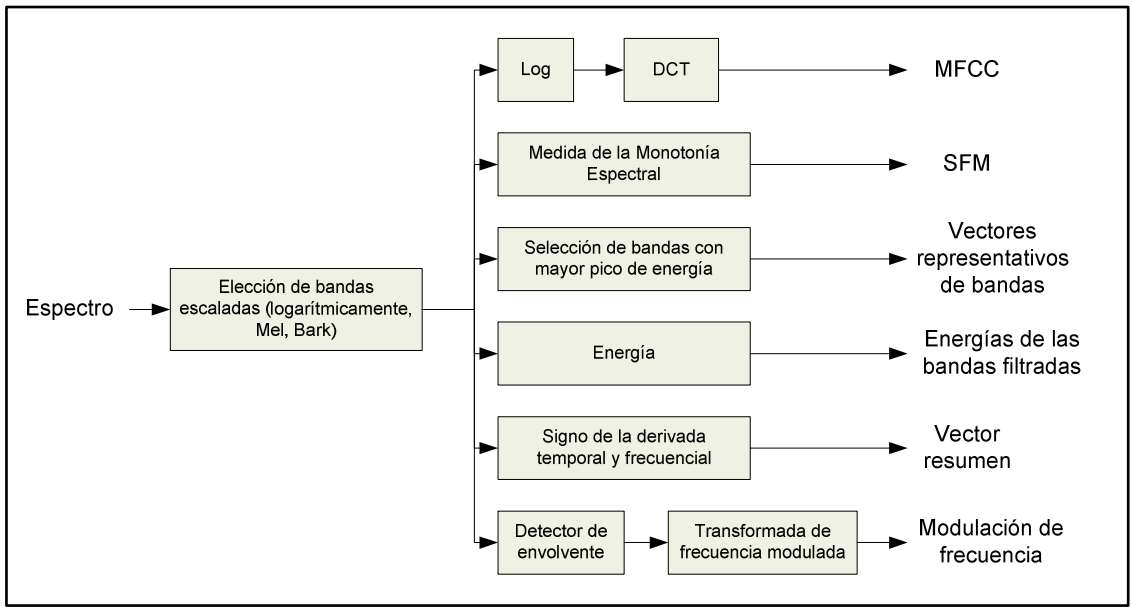

Figura 4. Ejemplos de extracción de características

En el artículo "Modulation Frequency Features for Audio Fingerprinting" (véase referencia [28]) se alega que las estimaciones espectrales y las características relacionadas son inadecuadas cuando el canal produce distorsiones. Se propone en ese caso, un análisis de modulación de frecuencia para caracterizar la variación temporal de la señal de audio. En este caso, las características se corresponden con la media geométrica de la estimación de modulación de frecuencia de la energía de 19 bandas espaciadas con filtros paso-banda.

Otras características han sido probadas como válidas para comparar sonidos, tales como: ancho de banda, intensidad, armonía.

En el artículo "Extracting Noise-Robust Features from Audio Data" (véase referencia [23]) se argumenta que las características usadas habitualmente son heurísticas y, como tal, pueden no ser óptimas. Por este motivo, propone el uso de una transformada de Karhunen-Loève modificada, el Análisis Orientado de Componentes Principales (OPCA) para encontrar las características óptimas. Si la transformada de Karhunen-Loève o Análisis de Componentes Principales (PCA) encuentra un conjunto de direcciones que maximizan la varianza de la señal, el OPCA obtiene un conjunto de posibles direcciones no ortogonales que toman en cuenta algunas distorsiones pre-definidas.

e) Pos-procesado

La mayoría de las características descritas hasta ahora son medidas absolutas. Con el fin de caracterizar mejor las variaciones temporales de la señal de audio, se añaden al modelo derivadas temporales de orden superior. Por ejemplo, podríamos pensar en un vector característico formado por varias MFCCs, su derivada y la derivada de la derivada. Algunos sistemas usan sólo la derivada de estas características y no los valores absolutos. Al usar las derivadas de las medidas tomadas sobre la señal se tiende a amplificar el ruido pero, al mismo tiempo, se filtran las distorsiones producidas en canales lineales e invariantes en el tiempo o en canales que varían muy lentamente (como, por ejemplo, una ecualización). Precisamente para reducir las distorsiones producidas por canales lineales variantes en el tiempo muy lentamente, se usa la Normalización Media Cepstral (CMN) (véase referencia [29]). Algunos sistemas representan el vector característico usando transformaciones como, por ejemplo, la PCA (véase referencias [24], [29]).

Es muy común aplicar una resolución muy baja a la cuantización de las características escogidas para crear el *fingerprint*: ternario o binario. El propósito de la cuantización es ganar robustez contra las distorsiones, normalizar, facilitar la implementación en hardware y reducir los requisitos de memoria.

2.2.2 Modelado del *fingerprint*

El bloque que modela la huella digital recibe generalmente una secuencia de los vectores de características calculados. Explotar las redundancias existentes entre vectores vecinos calculados en el mismo marco temporal es útil para reducir aún más el tamaño de los *fingerprints*. El tipo de modelo elegido condiciona la semejanza a medir entre ellos y, también, el diseño de los algoritmos de indexación de direcciones para recuperación rápida desde la base de datos (véase el apartado 2.2.3).

Una forma concisa para lograr un *fingerprint* de pequeño tamaño es resumir las secuencias multidimensionales del vector de características de un objeto de audio entero (o de un fragmento de él) en un solo vector. Por ejemplo, en el artículo de Etantrum (véase referencia [30]) se calcula el vector a partir de las medias y varianzas de las energías de 16 bandas espectrales correspondientes a 30 segundos de audio y se obtiene una huella de 512 bits. Esta, junto con la información sobre el formato del objeto de audio original, se envía a un servidor para la identificación. Otro ejemplo podría ser incluir en un vector: la tasa media de paso por cero, el ritmo estimado por minuto (BPM), el espectro medio y algunas otras características para representar un trozo de audio correspondiente a 26 segundos (véase referencia [31]).

Los dos ejemplos expuestos son eficientes computacionalmente y producen un *fingerprint* compacto. Ambos han sido diseñados para aplicaciones en las que se vinculan archivos mp3 con base de datos que almacenan información de estos archivos (título, artista, etc.) porque en estas aplicaciones se requiere una complejidad baja más que una robustez alta.

También podemos pensar en secuencias (trazas, trayectorias) de características. La representación mediante secuencias binarias (véase referencias [3], [25]) proporciona una gran eficiencia en el uso de memoria.

Otros sistemas incluyen cualidades musicales significativas de alto nivel, como ritmo o tono (véase referencia [25]).

Siguiendo el razonamiento sobre la posible solución sub-óptima dada por las características heurísticas, en el artículo "Extracting Noise-Robust Features from Audio Data" (véase referencia [23]) se propone emplear varias capas de OPCA para disminuir la redundancia estadística local de los vectores de características con respecto al tiempo. Además de reducir el tamaño de los *fingerprints*, en las transformaciones se tienen en cuenta requisitos adicionales de robustez.

También podemos trabajar con las "redundancias globales" que aparecen a lo largo de una canción (véase referencia [26]). Si asumimos que las características de una parte de un objeto de audio dado son similares entre ellas (por ejemplo, un estribillo que se repite en una canción mantendrá probablemente características similares), puede generarse una representación compacta agrupando los vectores de características. La secuencia de vectores es aproximada así por un número mucho más bajo de vectores código representativos. Deberemos tener en cuenta que la evolución temporal del audio se pierde con esta aproximación. También, en este mismo artículo, se trabaja con estadística a corto plazo sobre regiones de tiempo. Como resultado se obtiene un reconocimiento más exacto, ya que se tienen en cuenta dependencias temporales, y una búsqueda más rápida, puesto que la longitud de cada secuencia está también reducida.

P. Cano, en su artículo "Robust Sound Modeling for Song Detection in Broadcast Audio" (véase referencia [24]), utiliza un modelo basado en "redundancia global". El fundamento está muy inspirado en la "investigación del discurso". En un discurso se puede usar un alfabeto de tipos de sonidos, es decir fonemas, para dividir en segmentos una colección de datos procedentes del discurso y almacenarlos en texto, logrando una gran reducción de redundancia sin "mucha" pérdida de información. Semejantemente, podemos ver un objeto de audio como un conjunto de frases construidas concatenando clases de los sonidos procedentes de un alfabeto finito. Esta aproximación proporciona una huella consistente en secuencias de índices a una colección que contiene esas clases de sonidos. Esas clases de sonidos se estiman utilizando los Modelos Ocultos de Markov (HMMs). Modelar estadísticamente la

evolución temporal de la señal de audio permite la reducción de redundancia. Así pues, la representación de los *fingerprints* consiste en secuencias de índices a las clases de sonidos, conservándose así la información sobre la evolución temporal del objeto de audio.

2.2.3 Medidas de semejanza y métodos de búsqueda

A) Medidas de semejanza

Las medidas de la semejanza o similitud entre dos *fingerprints* están muy relacionadas con el tipo de modelo elegido. Al comparar secuencias es común usar la correlación. La distancia euclidiana, o versiones levemente modificadas de esta, se usa cuando tenemos secuencias de distintas longitudes (véase referencia [25]). Otra posibilidad es "el vecino más cercano" usando una estimación de la entropía cruzada (véase referencia [32]). En los sistemas donde las secuencias de características están cuantizadas, es común la distancia de Manhattan (véase referencias [3], [21]), o Hamming cuando la cuantización es binaria, como es el caso del sistema desarrollado en este proyecto (véase apartado 2.5.1). En el artículo "A Perceptual Audio Hashing Algorithm" (véase referencia [22]) se sugiere como más apropiado el uso de un error métrico, que llaman "Exponential Pseudo Norm" (EPN), para una mejor distinción entre valores cercanos y distantes.

Hemos presentado hasta ahora una estructura de identificación que sigue la siguiente regla de juego: tanto los *fingerprints* de los patrones de referencia – las huellas almacenados en la base de datos – como el *fingerprint* desconocido – la huella extraída del audio desconocido – están en el mismo formato y se comparan según una cierta semejanza a medir, por ejemplo: la distancia de Hamming, una correlación, etc. En algunos sistemas, solamente los extraídos de la referencia son realmente "fingerprints" - modelado compactamente como un código o una secuencia de índices a HMMs (véase referencias [26], [29]); en estos casos, las semejanzas o similitudes se computan directamente entre la secuencia de características extraída del objeto de audio desconocido y las huellas de los patrones de referencia almacenados en la base de datos. Se puede emparejar la secuencia del vector de característica a los diversos códigos usando una distancia métrica. Para cada código, se acumulan los errores y el audio desconocido se asigna a la clase que rinde el error acumulado más bajo. Otra posibilidad consiste en aplicar el algoritmo de Viterbi con la secuencia que contiene los índices que señalan a las clases de los sonidos del HMM y seleccionar la transición más probable en la base de datos.

B) Métodos de búsqueda

Una tarea fundamental para la utilidad de estos sistemas es cómo hacer eficientemente la comparación entre el *fingerprint* correspondiente al audio desconocido y, posiblemente, millones de *fingerprints* contenidos en una base de datos. Un algoritmo que compute estas semejanzas puede hacer de nuestro sistema un sistema muy poco eficiente, pues el tiempo para encontrar el *fingerprint* de la base de datos más parecido al calculado a partir del audio desconocido es proporcional a $N \cdot c(d()) + E$, donde *N* es el número de huellas que contiene la base de datos, la $c(d())$ es el tiempo necesario para el cálculo de una sola semejanza y *E* modela un cierto tiempo CPU adicional.

Veamos métodos de búsqueda que optimizan este tiempo:

- Distancias pre-computadas

Uno no puede pre-calcular las semejanzas con el *fingerprint* desconocido porque este no ha estado previamente procesado en el sistema. Sin embargo, uno puede pre-calcular distancias entre los registrados en la base de datos y construir una estructura de datos para reducir el número de evaluaciones de semejanza a realizar una vez que se presente en el sistema la huella digital desconocida. Es posible construir un conjunto de clases de equivalencia, calcular algunas semejanzas con el desconocido para descartar algunas clases y hacer una búsqueda más exhaustiva con el resto (véase referencia [33]). Si la medida de semejanza es una métrica (es decir la medida de semejanza es una función que satisface las siguientes propiedades: segura, simétrica, reflexiva y desigualdad triangular) existen métodos que, garantizando que no se produzca ningún descarte erróneo, reducen el número de evaluaciones a realizar (véase referencia [34]).

- Filtrado de candidatos no probables con una medida simple

Otra posibilidad es el uso de una medida más simple de semejanza con el objetivo de eliminar rápidamente muchos candidatos y usar una medida más exacta, pero compleja, con el resto, (véase referencia [35]). Para garantizar que no haya descartes erróneos, la medida simple (gruesa) usada para descartar *fingerprints* poco probables debe tener un límite bastante más bajo que el de la medida de semejanza más costosa (fina).

- Índices invertidos a ficheros

Consiste en índices de posibles trozos de un *fingerprint* que señalan a posiciones dentro de los objetos de audio. Siempre que un trozo de una huella desconocida esté libre de errores, puede ser recuperada eficientemente una lista de canciones candidatas y posiciones dentro de estas para hacer una búsqueda exhaustiva a través de ellas (véase referencia [3]). Se podría usar un

índice que utilizara palabras código extraídas de las secuencias binarias que representan el audio (véase referencia [19]). Este método, aunque muy rápido, algunas veces hace suposiciones sobre los errores permitidos en las palabras usadas para construir el índice que pueden dar lugar a descartes erróneos.

- Reducción de candidatos

Una optimización simple para acelerar la búsqueda es mantener el mejor resultado encontrado hasta el momento. Podemos abandonar un cálculo de medida de la semejanza si en un punto sabemos que no vamos a mejorar ese resultado (véase referencia [33]). Algunas medidas de la semejanza pueden beneficiarse de estructuras como los árboles de sufijos para evitar cálculos duplicados (véase referencia [18]). La combinación de la estructura arborescente con la técnica de "mejor resultado encontrado hasta el momento" permite evitar, no solamente cómputo a partir de "ese momento", sino también el cómputo con los otros *fingerprints* que tienen un comienzo similar.

- División de la base de datos

Podemos pensar en dividir la base de datos en dos: una primera y más pequeña contendrá los *fingerprints* con mayor probabilidad de aparecer (por ejemplo, los de las canciones más popular del momento), y la otra con el resto. Primero se busca en la base de datos pequeña y, solo en caso de no encontrar aquí la canción que se identifica con el *fingerprint* desconocido, el sistema busca en la segunda (véase referencia [36]).

Normalmente los sistemas utilizan varios de los métodos expuestos al mismo tiempo para acelerar las búsquedas. Por ejemplo, además de buscar primero en la base de datos de los *fingerprints* procedentes de las canciones más populares, se pueden aplicar índices invertidos a direcciones de archivo con el objetivo de filtrar candidatos poco probables antes de realizar una búsqueda exhaustiva con las huellas digitales no descartadas (véase referencia [36]).

2.2.4 Bloque de verificación

Este último paso pretende contestar a la pregunta de si el *fingerprint* desconocido está presente o no en la base de datos de los *fingerprints* procedentes de los objetos de audio a identificar. Durante la comparación del desconocido con los de la base de datos, se obtiene un valor numérico que refleja el grado de semejanza entre la huella desconocida y la considerada como detectada. Para decidir que hay una identificación correcta, este valor necesita estar por debajo de un cierto umbral.

No es fácil elegir este umbral puesto que depende de: el modelo de *fingerprint* empleado, la semejanza entre los *fingerprints* de la base de datos y el tamaño de la base de datos. Cuanto más grande es la base de datos, más alta será la probabilidad de considerar como detectado uno que realmente no es, es decir, tendremos una alta probabilidad de falsa alarma.

2.3. PARÁMETROS DE UN SISTEMA BASADO EN AUDIO FINGERPRINT

Los requerimientos que se deben exigir sobre cualquier sistema de este tipo dependen fuertemente de la aplicación para la que se vaya a usar. No obstante, para evaluar y comparar la gran diversidad de posibles tecnologías (véase referencia [37]) empleadas en ellos podemos enumerar los siguientes parámetros a tener en cuenta:

- ***Robustez**:* capacidad para identificar correctamente un anuncio a pesar del nivel de compresión y/o distorsión (o interferencia) que se haya producido en el anuncio. Para conseguir una gran robustez la huella debería basarse en características preceptúales que fuesen invariantes, al menos hasta cierto punto, respecto a la degradación de la señal. Deseablemente, señales fuertemente degradadas deberían dar huellas similares.

- ***Precisión**:* número de identificaciones correctas, identificaciones perdidas (tasa de probabilidad de no alarma), e identificaciones erróneas (tasa de probabilidad de falsa alarma).

- ***Fiabilidad**:* métodos para asegurar que un *fingerprint* extraído está o no entre los procedentes de los objetos de audio a identificar. En caso de que el *fingerprint* extraído no se corresponda con ninguno de los que contiene la base de datos, no se debería identificar como ninguno de los objetos de audio a identificar.

- ***Tamaño del fingerprint**:* cantidad de memoria necesaria para almacenar un *fingerprint*. Para que la búsqueda sea rápida, las huellas se suelen almacenar en memoria RAM. Por tanto, el tamaño de la huella digital, expresado en bits por segundo o en bits por objeto de audio, determinará en mayor grado la cantidad de recursos que se necesitarán para el servidor de base de datos.

- ***Granularidad**:* número de segundos de audio necesarios para identificar un anuncio. La granularidad es un parámetro que puede depender de la aplicación. En algunas aplicaciones se puede utilizar el archivo de audio completo para la identificación mientras que en otras se prefiere hacer la identificación con solo una pequeña porción. En nuestro caso, anuncios de radio, es suficiente con una pequeña porción de anuncio para identificarlo.

- ***Complejidad:*** hace referencia al coste computacional en el proceso de extracción de una huella digital, a la complejidad en el proceso de búsqueda, a la complejidad en el algoritmo de comparación de dos huellas, al coste de añadir nuevas huellas a la base de datos, etc.

- ***Escalabilidad:*** rendimiento del sistema con bases de datos muy grandes o ante un gran número de identificaciones concurrentes. Para el desarrollo de sistemas de huellas de audio la velocidad de búsqueda y la escalabilidad son parámetros clave. La velocidad de búsqueda debería ser del orden de milisegundos para una base de datos que contenga más de 100000 archivos de audio usando solo recursos limitados del computador.

- ***Versatilidad:*** capacidad de identificar objetos de audio a pesar de su formato. Si nuestro sistema es versátil nos permitirá usar la misma base de datos para diferentes aplicaciones.

- ***Seguridad:*** vulnerabilidad de la solución ante posibles manipulaciones o craqueos.

Estos parámetros básicos tienen una relación de dependencia fuerte. Por ejemplo si se quiere una granularidad más baja, se necesita extraer una huella más grande para mantener el mismo nivel de precisión. Esto es debido a que la probabilidad de falsa alarma es inversamente proporcional al tamaño de los *fingerprints.*

Otro ejemplo: el tiempo de búsqueda aumenta cuando se quiere tener una huella más robusta; esto es porque la búsqueda es una búsqueda de proximidad, es decir, tiene que encontrarse una similar (o la más similar). Si las características son más robustas, la proximidad es menor y, por tanto, el tiempo empleado en realizar la búsqueda aumenta.

Como comentario, añadir que la Federación Internacional de la Industria de la Fonografía (IFPI) y la Asociación de la Industria de la Grabación de América (RIAA), en un informe sobre las diferentes técnicas de *audio fingerprinting*, evaluaron algunos de estos sistemas de identificación (véase referencia [38]).

2.4. APLICACIONES

En este apartado comentaremos diferentes aplicaciones en las que se hace uso de los sistemas *audio fingerprinting*. La mayoría de ellas se basan en la capacidad de la huella digital para ligar audio sin etiquetar a los datos correspondientes de una base de datos, sin importar el formato de audio.

2.4.1 Supervisión o monitorización de contenido audio

- Supervisión del distribuidor final

Los distribuidores pueden necesitar saber si tienen los derechos para difundir el contenido de audio a los consumidores. *Fingerprinting* puede ayudar a identificar objetos de audio sin etiquetar en bases de datos de canales de TV y radio. Puede también identificar el contenido audio no identificado recuperado de plantas y distribuidores de CDs en investigaciones contra la piratería.

- Supervisión del canal de transmisión

En muchos países, las estaciones de radio deben pagar los derechos sobre la música que emiten. Los sostenedores de estos derechos necesitan supervisar las transmisiones de radio para verificar si los derechos se están pagando correctamente. Incluso en los países en donde las estaciones de radio pueden emitir música libremente, los sostenedores de los derechos están interesados en la supervisión de las transmisiones de radio con propósitos estadísticos. Los publicistas también necesitan supervisar las transmisiones de radio y de TV para verificar si los anuncios están siendo difundidos según lo convenido. Igual ocurre para las difusiones vía Web. Otras aplicaciones incluyen las compilaciones de gráficos para el análisis estadístico del material del programa que se está emitiendo.

La supervisión de la difusión es probablemente uno de los ámbitos en que más se hace uso de estos sistemas (véase referencias [16], [17], [39], [40], [41], [42]). Este proyecto es un ejemplo más de este tipo de aplicaciones, en el cual utilizamos este sistema para el control de la emisión de anuncios de radio.

En general, esta supervisión de la difusión consiste en un sistema que "escucha" la radio y actualiza continuamente una lista de canciones o anuncios que resume lo que se difunde por cada estación. Por supuesto, debe estar disponible para el sistema una base de datos que contenga las huellas digitales de todas las canciones y anuncios que se identificarán; además, esta base de datos se debe poner al día cuando salen nuevas canciones. Entre los

abastecedores comerciales de este servicio están, por ejemplo: Broadcast Data System (véase referencia [44]), Music Reporter (véase referencia [43]), Audible Magic (véase referencia [13]), Yacast (referencia [16]).

Napster y otras comunidades basadas en páginas Web, donde los usuarios comparten archivos de música, han sido excelentes medios para la piratería musical. En el caso de Napster, quien remonta sus comienzos a Junio de 1999, sus usuarios podían compartir y bajar gratuitamente una extensa colección de archivos de audio. Más tarde, después de una demanda por parte de la industria de la música, a Napster se le prohibió facilitar la transferencia de canciones que tuvieran copyright. La primera medida tomada por Napster (Marzo de 2001) fue la introducción de un sistema de filtrado basado en un análisis del nombre de los ficheros, según una lista de grabaciones con copyright suministrada por las compañías de grabación. Este sistema no fue nada efectivo y, por lo tanto, no solucionó el problema, porque los usuarios demostraron ser extremadamente creativos a la hora de elegir nombres para los archivos que querían intercambiar, engañando así al sistema de filtrado al mismo tiempo que todavía permitían que otros usuarios reconocieran fácilmente grabaciones específicas. El elevado número de canciones con títulos idénticos era un factor adicional en la reducción de la eficacia de tales filtros. En Mayo de 2001, Napster incluyó un sistema de supervisión basado en *fingerprints* (véase referencia [12]) y un nuevo sistema de filtrado de archivos, consiguiendo una solución muy bien adaptada al problema que debía solventar.

Además, en páginas Web ordinarias podemos encontrar una gran cantidad de contenidos de audio que no disponen de ningún tipo de control. Un sistema de *fingerprint* con otro sistema que inspeccione páginas Web podría identificar esos contenidos e informar a los correspondientes dueños de sus derechos.

- Supervisión en el extremo del consumidor

En aplicaciones que supervisan la "política de uso", el objetivo es evitar malos usos de las señales de audio por parte del consumidor. Podemos pensar en un sistema donde un pedazo de música se identifique por medio de un *fingerprint* y una base de datos sea consultada para recuperar la información sobre los derechos de ese objeto de audio. Esta información dictará el comportamiento de dispositivos "inteligentes" (por ejemplo, reproductores y grabadores de CD y DVD, reproductores de MP3 o incluso computadoras) de acuerdo con la "política de uso". Obviamente estos dispositivos "inteligentes" requerirán estar conectados a una red que le de acceso a la base de datos.

2.4.2 Servicios de valor añadido

El contenido de la información se define como información sobre un extracto de un objeto de audio el cual es relevante al usuario o necesario para la intención que persiga la aplicación. Dependiendo del uso y del perfil de usuario, pueden ser definidos varios niveles de la información contenida. Aquí se exponen algunas de las situaciones que podemos imaginarnos:

- Información describiendo características del audio, tales como ritmo, melodía o descripciones de armónicos.

- Información que describe un trabajo musical: cómo fue compuesto y cómo fue grabado. Para ejemplo: compositor, año de composición, intérprete, fecha de interpretación, realización de la grabación en directo.

- La otra información referente a un trabajo musical podría ser: la imagen de la carátula del álbum, precio del álbum, biografía del artista, información sobre los conciertos siguientes, etc.

Se pueden definir diversos perfiles de usuario. Los usuarios comunes estarían interesados en información general sobre un trabajo musical, tal como título, compositor y año de edición; por su parte, los músicos pueden desear saber qué instrumentos fueron tocados, mientras que los ingenieros de sonido podrían estar interesados en la información sobre el proceso de la grabación. La información contenida se puede estructurar por medio de un esquema de descripción de la música (MusicDS), que es una estructura de datos usada para describir y para anotar datos de audio. El estándar MPEG-7 propone un esquema de descripción para los contenidos multimedia basado en el lenguaje XML (véase referencia [45]), proporcionando un intercambio fácil de datos entre equipos.

Algunos sistemas almacenan la información en una base de datos que pueda ser accesible a través de Internet. Los *fingerprints* se pueden utilizar para identificar una grabación y recuperar su correspondiente información, sin importar el tipo de ayuda, el formato del archivo o cualquier otra particularidad de los objetos de audio. Por ejemplo, MusicBrainz, Id3man o Moodlogic (véase referencia [46], [47], [48]) etiquetan automáticamente las colecciones de archivos de audio; el usuario puede descargarse un reproductor compatible que extrae *fingerprints* y los somete a un servidor central con el objetivo de descargar todo tipo de información asociada al audio solicitado por el usuario.

La empresa Gracenote (véase referencia [49]), quien ha estado proporcionando vínculos a informaciones de música basándose en el TOC (Tabla de Contenidos) de un CD, comenzó a ofrecer tecnología *audiofingerprinting* para extender los vínculos desde los TOCs de los CDs al

nivel de la canción. Su método de identificación de audio se utiliza conjuntamente con clasificadores basados en texto para mejorar la precisión.

Otro ejemplo es la identificación de una canción a través de los dispositivos móviles, por ejemplo, un teléfono móvil. Esta es una de las situaciones más exigentes en términos de robustez, pues la señal de audio sufre distorsión de radio, conversión de D/A-A/D, ruido de fondo, codificación GSM, distorsiones propias del canal de comunicaciones móviles y, además, solamente están disponibles unos pocos segundos del audio (véase referencia [14]). Otro ejemplo, con bastante parecido a este, podría ser una radio de un coche al que se le incorpora una función de identificación de lo que se está escuchando. Otro ejemplo, una aplicación basada en *fingerprints* que esté continuamente escuchando lo que entra en la tarjeta de sonido de un PC, de tal forma que el usuario puede ser conducido a una página Web en Internet que contiene información del artista que está escuchando. Incluso se le puede dar la posibilidad de pulsar un botón para comprar el álbum de ese artista.

Como vemos, *audiofingerprinting* puede proporcionar una gran diversidad de posibilidades que permiten recuperar información sobre objetos de audio.

2.4.3 Sistemas de verificación de la integridad

En algunas aplicaciones, la integridad de las grabaciones de audio debe ser establecida antes de que la señal pueda ser utilizada realmente, es decir uno debe asegurar que la grabación no se ha modificado o que no está distorsionada. Si la señal sufre una compresión fuerte, conversión de D/A-A/D u otras transformaciones en el canal de transmisión, la integridad no se puede comprobar por medio de funciones estándares, puesto que una fluctuación en un solo bit es suficiente para que la salida de la esta función cambie. Los métodos basados en "watermarking" pueden también proporcionar una salida falsa en tal contexto. Sistemas basados en *fingerprints*, combinada a veces con "watermarking", están siendo investigados para abordar esta situación (véase referencia [2]). Entre algunos posibles usos (véase referencia [50]), podemos nombrar: comprobación de que los anuncios tienen la longitud y calidad requeridas, verificación de que una grabación de infracción sospechada es igual que una grabación cuyo propietario es conocido, etc.

2.4.4 Organización automática de una biblioteca musical

Aunque desde el punto de vista del consumidor, *audio fingerprinting* podría verse como una tecnología negativa, hay también un gran número de beneficios potenciales para el consumidor.

Muchos usuarios tienen hoy en día en su PC una biblioteca musical que contiene varios cientos, igual a veces hasta millares, de canciones. La música es almacenada generalmente en un formato comprimido (generalmente MP3) en su disco duro. Cuando estas canciones se obtienen de diferente fuentes, tales como un CD o una descarga de archivo de una red, estas bibliotecas no están a menudo bien organizadas. Los datos sobre esos archivos de audio son, a menudo, escasos, incompletos e, incluso a veces, incorrectos. Si se asume que la base de datos de *fingerprints* contiene esos datos de forma correcta, una aplicación basada en ellos puede organizar fácilmente la biblioteca bajo el criterio que el usuario desee como, por ejemplo, una organización por álbumes o artistas. Hoy en día existen tales aplicaciones; por ejemplo, ID3Man (véase referencia [47]), herramienta potenciada por Auditude (véase referencia [11]), está ya disponible para etiquetar los archivos MP3 desconocidos. Una herramienta similar de la empresa Moodlogic (véase referencia [48]) está disponible como plug-in del reproductor Winamp.

2.5. SISTEMA ANALIZADO

Una vez visto la gran variedad de algoritmos basados en *fingerprints* existentes, con este proyecto nos centraremos en el estudio de la eficacia y la eficiencia del método desarrollado por Philips, **"Audio Hash"** (véase referencia [6]), aplicándolo a anuncios de radio y centrándonos en el método de extracción con el fin de mejorarlo. Para tratar de alcanzar nuestro objetivo buscaremos los valores óptimos de los diferentes parámetros que se usan en dicha técnica para lograr que el algoritmo sea lo más eficiente posible computacionalmente y, al mismo tiempo, robusto.

2.5.1 Algoritmo de extracción

Vemos en la figura 5, un cuadro resumen del funcionamiento de nuestro sistema basado en *audio fingerprint*:

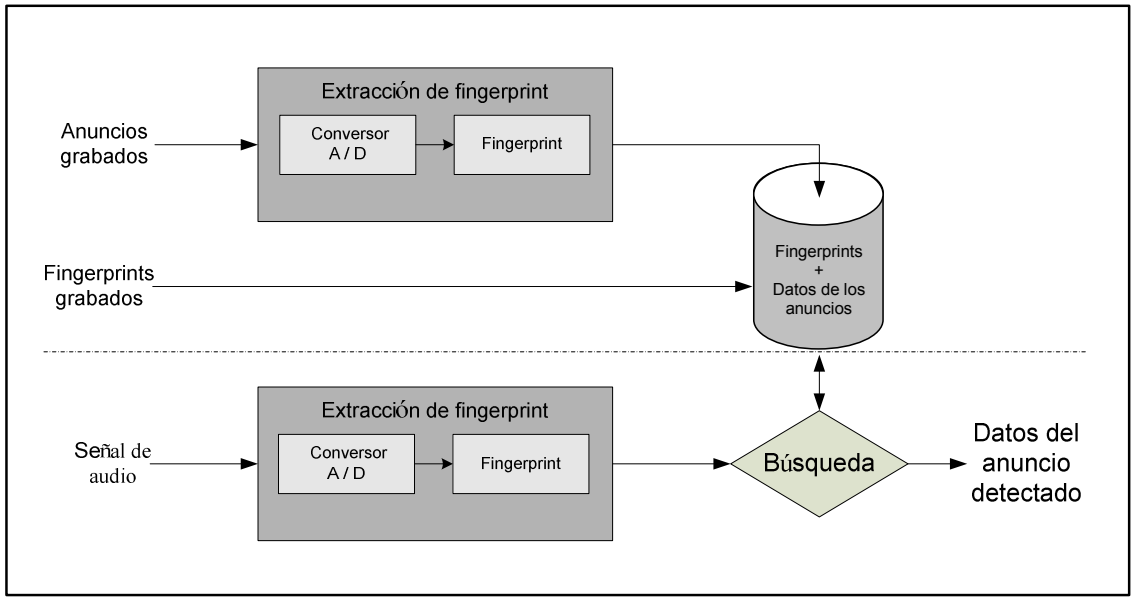

Figura 5. Esquema de funcionamiento del sistema basado en *fingerprints* analizado

Como ya comentamos en apartados anteriores, la mayoría de los algoritmos de extracción de *fingerprints* están basados en el mismo planteamiento. Primero la señal de audio es segmentada en tramas (fragmentos de dicha señal) y para cada trama se computan unas características. Estas características deben ser, preferiblemente, las menos invariantes posibles a la degradación de la señal.

A continuación mostramos un esquema que resume el algoritmo de extracción empleado en este proyecto:

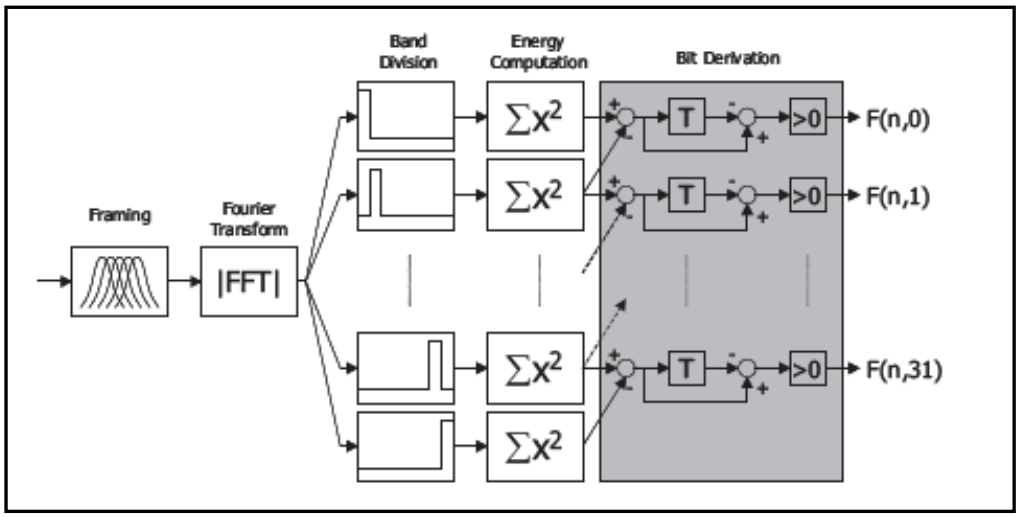

Figura 6. Esquema del algoritmo de extracción de *fingerprints*

Siguiendo este esquema de la figura 6, se realizó el código de Matlab **"fingerprint.m"** (véase Anexo 2.1) que iremos comentando al mismo tiempo que hacemos una descripción teórica de este algoritmo de extracción de *fingerprints*.

El algoritmo consiste, primeramente, en segmentar la señal de audio en tramas superpuestas de 0,37 segundos con un factor de overlap de 31/32, cargado por una ventana de Hanning.

En Matlab obtenemos la ventana de Hanning con la siguiente línea de código:

```
N=2960;
ventana=hanning(N);
```

siendo N el número de muestras por trama. Puesto que la frecuencia de muestreo de la señal de audio es 8000 Hz. y queremos un tamaño de ventana de 0.37 segundos, el número de muestras por trama es $8000 * 0.37 = 2960$.
Posteriormente, para aplicar este enventanado de Hanning, multiplicamos la trama de tamaño 2960 muestras por la variable ventana que contiene el resultado proporcionado por la función.

La representación o huella que vamos a obtener de cada una de esas tramas de audio lo llamamos *sub-fingerprints*. De este modo, el algoritmo extrae *sub-fingerprints* de 32 bits cada intervalo de 11.6 ms (370/32 ms). Debido al tamaño del factor de overlap, los *sub-fingerprints* tienen mucha similitud con los calculados antes y después; así pues, la secuencia de *sub-fingerprints* varia lentamente en el tiempo. Un anuncio consistirá pues, en una secuencia de *sub-fingerprints*, la cual es almacenada en una base de datos.

Así pues, el código **"fingerprint.m"** implementa una función que toma como variable de entrada un vector resultado de leer un archivo de audio o parte de él y devuelve como salida un conjunto de *sub-fingerprints* de 32 bits. El conjunto de estos *sub-fingerprints* conforma el *fingerprints* correspondiente al vector de entrada de la función. El *fingerprints* será de mayor o menor longitud en función del tamaño del audio.

Este tamaño de *fingerprints* también varía en función de la solución adoptada para el siguiente problema que se nos plantea. La cuestión es que el tamaño del vector de entrada a nuestra función no es un múltiplo de N; por lo tanto, para el cálculo del último *sub-fingerprints* nos planteamos dos posibles alternativas:

a) rellenar con ceros el vector de entrada hasta que tenga un tamaño múltiplo del número de muestras por trama.

b) quedarnos con el número entero de muestras del vector de entrada más alto posible que haga que el tamaño de este sea múltiplo del desplazamiento de la ventana de Hanning.

Entre estas dos opciones, nos decantamos por la segunda. Esto es porque el hecho de añadir ceros podría llevar a tener muchos *sub-fingerprints* iguales aun procediendo de fuentes de audio diferentes pudiendo provocar falsas alarmas en la detección.

Para extraer un *sub-fingerprints* de 32 bits por cada trama, se seleccionan 33 bandas de frecuencia no solapadas de la Densidad Espectral de Potencia (PSD) estimada. La PSD es estimada usando un simple estimador de período gramas.

En Matlab, para conseguir el módulo de la PSD, utilizamos las siguientes funciones de librería:

```
FRAME=abs(fft(frame));
```

Estas bandas se extienden en el rango comprendido entre los 300 Hz. y los 2000 Hz. (el rango espectral mas relevante para el sistema de audición humano (HAS, *Human Auditory System*)) y son logarítmicamente espaciadas. Elegimos este espaciado logarítmico porque se sabe que el sistema de audición humano trabaja aproximadamente sobre bandas logarítmicas. Haitsma y Kalker, experimentalmente, verificaron que el cambio en la evolución de la energía de la señal de audio (simultáneamente a lo largo del eje del tiempo y de la frecuencia) es una propiedad muy robusta ante muchos tipos de procesados.

Para realizar este espaciado logarítmico en Matlab, obtenemos un vector que contenga los índices que nos permitan seleccionar las 33 bandas con dicho espaciado. Estos índices los obtenemos con las siguientes líneas de código:

```
num_band=33;
f_inferior=300;
f_superior=2000;
indice=logspace(log10(f_inferior),log10(f_superior),num_band+1);
indice=round(indice*N/Fs);
```

siendo `num_band` el número de bandas que queremos, y `f_inferior` y `f_superior` la frecuencia inferior y superior, respectivamente, que delimitan el rango en el cual queremos la división en bandas.
Después de llamar a la función `logspace`, que devuelve los valores correspondientes a hacer una división logarítmica entre las frecuencias analógicas pasadas como parámetros, hay que pasar a frecuencia digital y redondear para tener valores que puedan servir de índices en un vector.

Denotemos la energía de la banda m-ésima de la trama n-esima por E(n,m). Las diferencias entre las energías son computadas en tiempo y en frecuencia de acuerdo a la siguiente formula:

$$ED(m,n) = E(n,m) - E(n,m+1) - (E(n-1,m) - E(n-1,m+1))$$

En nuestro código, para el cálculo de la energía de una banda llamamos a una sub-función llamada **calc_energ**, cuyo código está implementado en el archivo **"calc_energ.m"** (véase Anexo 2.2).

Los bits de cada *sub-fingerprints* son obtenidos a partir de la siguiente formula:

$$F(n,m) = \begin{cases} 1 \; si \; ED(n,m) > 0 \\ 0 \; si \; ED(n,m) \leq 0 \end{cases}$$

donde F(n,m) denota el m-ésimo bits del *sub-fingerprints* de la trama n-ésima.

En el código **"fingerprint.m"** recurrimos a otra sub-función para el cálculo de cada uno de los bits que conforman un *sub-fingerprint*. Esta función, llamada **bit_derivation** e implementada en el archivo **"bit_derivation.m"** (véase Anexo 2.3), devuelve un 1 o un 0 en función de la fórmula anterior aplicada a las siguientes variables que recibe como entrada:

- energía de la banda actual,
- energía de la banda anterior,
- energía de la banda actual en el período anterior,
- energía de la banda anterior en el período anterior.

2.5.2 Algoritmo de búsqueda

El algoritmo de búsqueda nos permite buscar entre todos los *fingerprints* que se encuentran en una base de datos, el más parecido al *fingerprint* calculado en tiempo real a partir de la señal de audio de entrada al sistema.

Para establecer la semejanza entre dos *fingerprints* comparamos ambos directamente bit a bit, y se calcula el número de bits en que se diferencian. Obviamente, consideraremos como *fingerprint* de nuestra base de datos más parecido al calculado en tiempo real, aquel que posea menor número de bits diferentes con respecto a este; es decir, aquel que tenga menor distancia de Hamming con respecto al calculado en tiempo real.

Una vez obtenido de la base de datos, el que más se parece al calculado en tiempo real, hay que establecer un número mínimo de bits que tienen que tener iguales los dos *fingerprints* para considerar que la huella calculada en tiempo real se corresponde realmente con esa; es decir, necesitamos un umbral para decidir que la señal de audio de entrada al sistema se corresponde con un anuncio de nuestra base de datos. Este umbral será necesario establecerlo correctamente para evitar en la mayor medida posible falsas alarmas o, el efecto contrario, no alarmas cuando sí deberían producirse (véase apartado 3.3).

A la hora de realizar los análisis y las pruebas expuestas en apartados posteriores, hemos hecho uso de la función **busqueda**. Esta función se encuentra implementada en el fichero de Matlab **"busqueda.m"** (véase Anexo 2.4). La función recibe como variables de entrada dos *fingerprints*; y como salida, devuelve la distancia de Hamming entre ambos *fingerprints*, es decir, el número de bits en que se diferencian.

Como ya hemos comentado anteriormente, este algoritmo de búsqueda no es el óptimo, sin embargo es perfectamente válido para el análisis que se desarrolla en este proyecto. Por otra parte, si quisiéramos trabajar con un algoritmo de búsqueda óptimo, lo tendríamos que hacer con otra herramienta que no fuera Matlab y que nos permitiera trabajar con "direccionamiento a memoria". Ya que nuestro objetivo se centra principalmente en analizar el comportamiento del algoritmo de extracción de *fingerprints* al variar los parámetros de este con el objetivo de encontrar los valores óptimos de dicho algoritmo, dejamos como futura línea de investigación el estudio de un algoritmo óptimo de búsqueda. Se pueden ver posibles algoritmos de búsqueda óptimos en el apartado 2.2.3.