3. El Sistema de Philips

El proyecto se va a centrar en el estudio del sistema propuesto por los ingenieros de Philips Jaap Haitsma y Ton Kalker en su artículo *"A Highly Robust Audio Fingerprinting System"* [10]. En primer lugar presentaremos detalladamente el algoritmo para pasar posteriormente a comentar los resultados obtenidos en la simulación del mismo (principalmente en términos de probabilidad de falso positivo y probabilidad de detección).

Antes de empezar el diseño e implementación del sistema hay que hacerse unas cuantas preguntas. La primera y más importante es qué tipo de características son las que se van a usar. Ya hemos expuesto exhaustivamente en el capítulo anterior un gran número de posibles características usadas en la literatura. Estas podían dividirse en dos grandes clases: características semánticas y características no semánticas. Las primeras serían, por ejemplo, el género o el ritmo (BPM). Este tipo suelen tener una interpretación directa y se usan de hecho para clasificar música, generar listas de reproducción, etc. El segundo tipo consiste en características que tienen una naturaleza más matemática y son más difíciles de obtener directamente a partir de la música para una mente humana. Un elemento típico de este grupo es lo que se llama "Audio Flatness", usado en MPEG-7. Para este algoritmo se decidió explícitamente trabajar con características no semánticas por una serie de razones que nombramos a continuación:

- Las características semánticas no siempre un significado claro y no ambiguo, es decir, las opiniones personales difieren sobre dichas clasificaciones. Es más, incluso pueden cambiar con el tiempo: por ejemplo, lo que se clasificaba como *hard rock* hace 25 años hoy podría considerarse bastante más suave. Esto hace que el análisis matemático sea dificilísimo.

- Las características semánticas son, en general, más difíciles de computar que las no semánticas.

- Las características semánticas no son aplicables universalmente. Por ejemplo, el ritmo (BPM) no se suele aplicar a la música clásica.

Una segunda cuestión a tener en cuenta es la representación de las huellas. Una opción evidente es representarla como un vector de números reales, donde cada componente exprese el peso de una cierta característica espectral básica. Una segunda opción es mantener el espíritu de las funciones criptográficas de "hash" y representar las huellas como cadenas de bits. Por razones de reducida complejidad en la búsqueda se decidió trabajar con la segunda opción. La primera implicaría una medida de similaridad involucrando sumas/sustracciones reales y dependiendo de la medida de similaridad, puede que incluso multiplicaciones reales. Las huellas basadas en representaciones de bits pueden ser comparadas solo contando bits. Dados los escenarios de aplicación esperados, no esperamos una alta robustez para todos y cada uno de los bits en una huella como esta. Por tanto, en contraste con los "hash" criptográficos que tienen como mucho varios centenares de bits, permitiremos huellas que tengan unos

pocos miles de bits. Las huellas que contienen un gran número de bits permiten una identificación fiable incluso si el porcentaje de bits que no coinciden es relativamente alto.

Una última cuestión se refiere a la granularidad de las huellas. En las aplicaciones que se prevén no hay ninguna garantía de que los archivos de audio estén completos. Por ejemplo, en la monitorización de emisiones, cualquier intervalo de 5 segundos es una unidad que tiene valor comercial, y por tanto debe ser identificado y reconocido. O, por ejemplo, en aplicaciones de seguridad tales como el filtrado de archivos en una red peer-to-peer, uno no desearía que el borrado de los primeros segundos de un archivo pudiera evitar la identificación. Aquí se adopta la postura de *flujos de huellas*, asignando *sub-huellas* a intervalos suficientemente pequeños (llamados tramas). Estas sub-huellas pueden no ser suficientemente grandes para identificar las tramas en sí mismas, pero un intervalo más grande, que contenga suficientes tramas permitirá una identificación robusta y fiable.

3.1 Algoritmo

3.1.1 Algoritmo de extracción

Como ya hemos comentado, la mayoría de los algoritmos de extracción están basados en el siguiente enfoque. Primero la señal de audio es segmentada en tramas. Para cada trama se computa un conjunto de características, escogidas de tal forma que sean invariantes (al menos hasta cierto punto) a las degradaciones de la señal. A la representación compacta de una trama individual se le llama *sub-huella.* El procedimiento global convierte un flujo de audio en un flujo de sub-huellas. Una sub-huella normalmente no tendrá información suficiente para identificar un archivo de audio. A la unidad básica que contiene suficiente información para identificar un archivo de audio (y, por tanto determinar la granularidad) le llamaremos *bloque de huellas*. El esquema de extracción de huellas propuesto está basado en este enfoque general. Extrae sub-huellas de 32 bits por cada intervalo de 11,6 milisegundos. Un bloque consiste en 256 sub-huellas consecutivas, correspondiendo a una granularidad de tan solo 3 segundos. Una vista general del esquema se muestra en la siguiente figura:

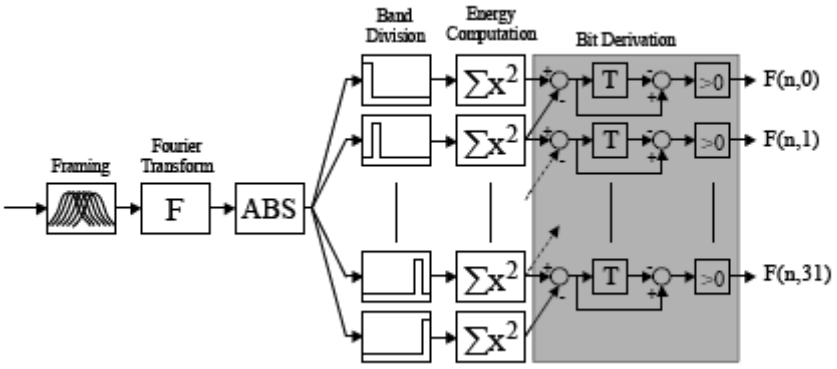

Fig.6: Vista general del esquema de extracción

La señal es primero segmentada en tramas solapadas. Dichas tramas tienen una longitud de 0,37 segundos y son ponderadas por una ventana de Hanning, con un factor de solapamiento de 31/32. Esta estrategia resulta en la extracción de una sub-huella cada 11,6 milisegundos. En el peor caso posible, los límites de la trama usados durante la identificación están 5,8 milisegundos desplazadas con respecto a los límites usados en la base de datos de huellas previamente computadas. Debido a la gran superposición, dos sub-huellas consecutivas tienen una gran similaridad y varían lentamente con el tiempo.

Ya sabemos que las características perceptuales más importantes están en el dominio de la frecuencia. Por tanto, se computa una representación espectral, realizando una transformada de Fourier a cada trama. Debido a la sensibilidad de la fase de la transformada de Fourier a diferentes límites de trama y al hecho de que el sistema auditivo humano (HAS) es relativamente insensible a la fase, solo nos quedamos con la información de módulo.

Para poder extraer un valor de sub-huella de 32 bits para cada trama, se seleccionan 33 bandas de frecuencias no solapadas. Estas bandas están entre los 300 y los 2000 Hz (el rango espectral más relevante para el HAS) y están logarítmicamente espaciadas. El espaciamiento logarítmico se escoge porque es bien sabido que el HAS opera en bandas aproximadamente logarítmicas (la llamada Escala Bark). Está verificado experimentalmente que el signo de la diferencia de energías (simultáneamente en el eje del tiempo y en el de la frecuencia) es una propiedad que es muy robusta a distintos tipos de procesado. Si denotamos la energía de la banda n de la trama n por E(n,m) y el m-ésimo bit de la sub-huella de la trama n por F(n,m), los bits de la sub-huella son formalmente definidos como:

$$F(n,m) = \begin{cases} 1, E(n,m) - E(n,m+1) - \left(E(n-1,m) - E(n-1,m+1)\right) > 0 \\ 0, E(n,m) - E(n,m+1) - \left(E(n-1,m) - E(n-1,m+1)\right) \leq 0 \end{cases} \quad (1)$$

En la siguiente figura se muestra un ejemplo de un bloque de 256 sub-huellas de 32 bits, extraídos con este esquema a partir de un trozo de "O Fortuna" de Carl Orff. Un bit a '1' corresponde a un píxel blanco y uno a '0', a un píxel negro. La fig. 7a corresponde al bloque obtenido a partir de la versión en calidad CD, mientras que el 7b ha sido obtenido a partir de la versión comprimida en MP3 (32kbps). Aunque idealmente ambas figuras deberían ser idénticas, no lo son, algunos de los bits cambian. Estos bits erróneos, que aparecen en negro en la figura 7c, se usan como medida de similaridad en el esquema.

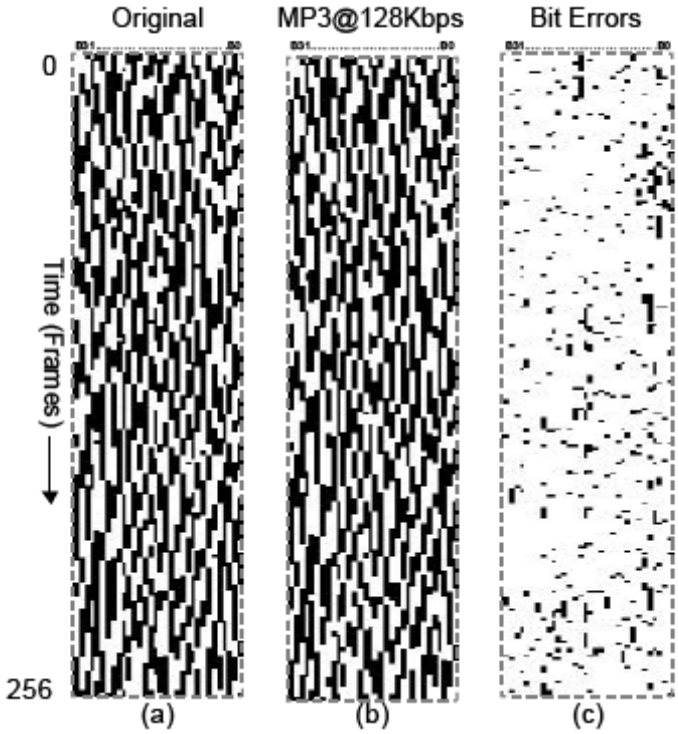

Fig.7: (a)Bloque de huellas de la versión original, (b)de la versión comprimida en mp3, (c) diferencia de ambas (BER= 0.078)

Los recursos de computación requeridos para el algoritmo propuesto son limitados. Puesto que el algoritmo solo tiene en cuenta frecuencias por debajo de 2KHz, el archivo de audio es primero submuestreado a un flujo de audio mono con una tasa de muestreo de 5 KHz (en realidad, 44100/8=5512,5 Hz). Las sub-huellas han sido diseñadas de tal forma que sean robustas contra degradaciones de la señal. Por tanto se pueden usar filtros de submuestreo muy sencillos sin introducir ninguna degradación en el comportamiento del sistema. Se usan 16 filtros FIR. La operación que más requiere computacionalmente es la transformada de Fourier de todas las tramas de audio. En la señal de audio submuestreada, la trama tiene una longitud de 2048 muestras. Si la transformada se implementa con una FFT, el algoritmo ha demostrado poder ejecutarse eficientemente en aparatos portátiles tales como una PDA o un teléfono móvil.

3.1.2 Algoritmo de Búsqueda

Aunque no será objeto de este proyecto (nos centraremos en analizar y proponer mejoras al algoritmo de extracción), vamos a presentar también, más brevemente, el algoritmo de búsqueda en la base de datos, una vez que ya se ha extraído la huella.

La tarea no es trivial. En vez de buscar por una huella con todos los bits exactamente iguales, lo que hay que encontrar es la huella *más similar.* Pongamos por ejemplo una base de datos de tamaño moderado, con 10.000 canciones de una duración media de 5 minutos. Esto corresponde a unos 250 millones de sub-huellas. Para identificar un bloque originado a partir de un trozo de audio desconocido hay que encontrar cual entre 250 millones da

la tasa de error mínima. Esto es realizable por fuerza bruta, en un ordenador con una capacidad de análisis de 200.000 bloques por segundo tardaría unos 20 minutos, lo cual no es viable en la práctica.

Se propone un algoritmo más eficiente. Se calcula la BER sólo para una serie de "candidatos", que contienen una alta probabilidad de ser la mejor posición en la base de datos. En una versión simple del algoritmo se hace la suposición de que es muy probable que, por lo menos una sub-huella (32 bits, recordemos) tenga una coincidencia exacta en la posición óptima de la base de datos. Si esto es válido, sólo hay que comprobar las posiciones en las que una de las 256 sub-huellas del bloque a identificar coincida. Como ejemplo, en la fig.8 se muestra el número de errores por sub-huella para la huella mostrada en la fig.7. Se observa que de hecho hay 17 sub-huellas de las 256 que no tienen errores.

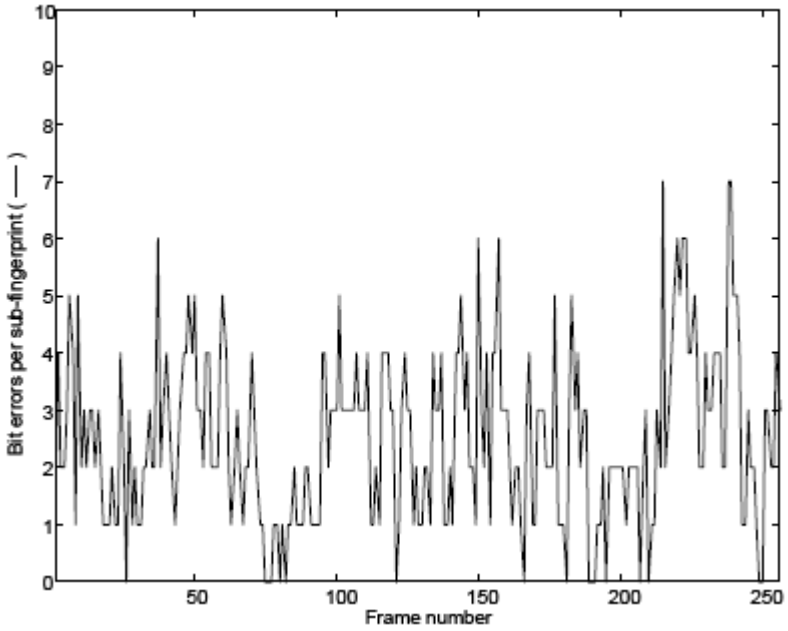

Fig.8: Bits erróneos por sub-huella para el caso de O Fortuna de Orff

Si asumimos que la huella original de la fig. 7a está en la base de datos, su posición estará entre las candidatas seleccionadas cuando el audio a identificar sea la versión mp3 de la fig. 7b.

Las posiciones de una base de datos donde se encuentra una sub-huella de 32 bits específica se obtienen usando la arquitectura de base de datos de la fig.8.

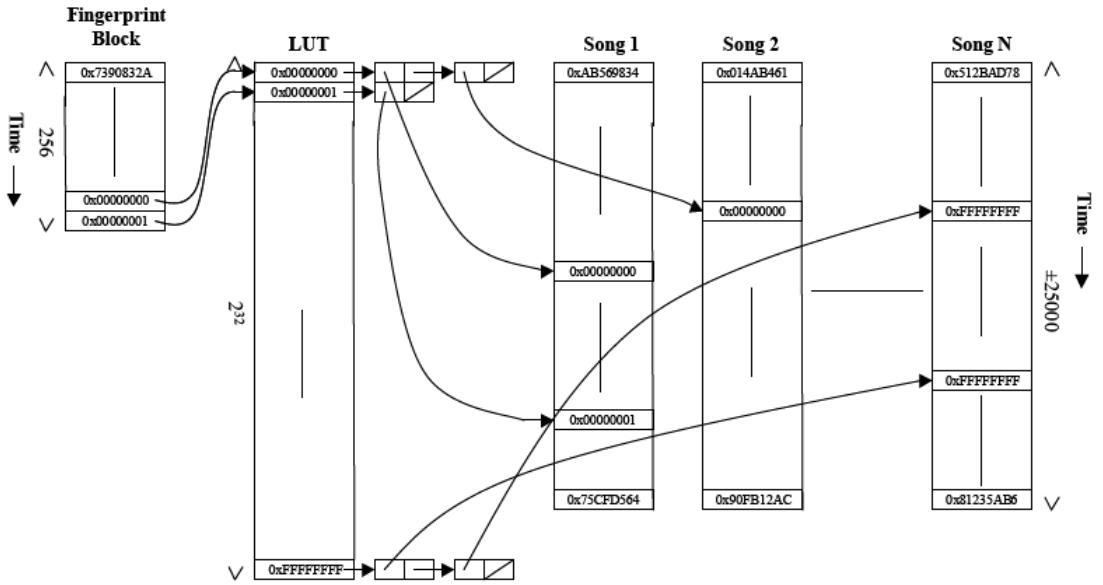

Fig.9: Esquema de la base de datos

La base de datos contiene una "Lookup Table" (LUT) con todas las posibles sub-huellas de 32 bits como entrada. Cada entrada apunta a una lista con punteros a las posiciones en la base de datos real donde se encuentran las respectivas huellas de 32 bits. En sistemas prácticos con memorias limitadas una tabla que contenga 2^{32} entradas no es factible o no es práctica. Además, la LUT estará rellena de una forma dispersa, porque solo un número limitado de canciones pueden almacenarse en memoria. Por tanto, en la práctica, se usa una tabla de "hash" en vez de una LUT.

Si volvemos a hacer cuentas, para una base de datos de 10.000 elementos, tendremos unos 250 millones de sub-huellas. Por tanto, el número medio de posiciones en una lista será 0,058 ($=256*10^6/2^{32}$). Si suponemos que todas las sub-huellas son igualmente probables, el número medio de comparaciones por identificación sería 15 (=0,058*256). En la práctica se observa que la distribución de las sub-huellas es no uniforme, lo que hace aumentar el número de comparaciones en un factor de 20 más o menos. Aun así, en un buen ordenador, 300 comparaciones se pueden hacer en 1,5 milisegundos. Por lo tanto este algoritmo es unas 800.000 veces más rápido que la búsqueda simple.

Se había supuesto que una de las sub-huellas estaba libre de errores, lo cual es casi siempre cierto para señales con degradación más o menos suave. Para señales fuertemente degradadas la suposición no es siempre válida. Si esto no ocurre, se mira si hay sub-huellas con un solo error y, en vez de buscar en la base de datos posiciones donde ocurra una de las 256 huellas buscamos posiciones con una distancia de Hamming de 1 (es decir, un bit cambiado) con respecto a todas las 256 sub-huellas. Esto incrementará el número de búsquedas en un factor de 33, lo que todavía es asumible. Pero, si el número mínimo de bits erróneos en una sub-huella es 3, el tiempo de búsqueda se incrementa en un factor de 5489, lo que ya no es asumible. Es interesante observar que el factor de no-uniformidad de 20 va disminuyendo al ir aumentando el número de bits a conmutar.

Puesto que este método lleva muy rápido a tiempos de búsqueda inaceptables se propone otro enfoque que usa decodificación suave. Es decir, se propone estimar y usar la probabilidad de que un bit de una huella sea recibido correctamente. Las sub-huellas se obtienen comparando y umbralizando diferencias de energía. Si la diferencia de energía está muy cerca del umbral es razonablemente probable que el bit sea erróneo y viceversa. Derivando la información de fiabilidad de cada bit de la sub-huella a es posible expandir una huella dada en una lista de probables sub-huellas. Asumiendo que una de las más probables tiene una coincidencia exacta en la posición óptima de la base de datos, la identificación se hace como antes. Lo que se hace es ordenar los 32 bits de menos a más fiables y se va haciendo la lista de sub-huellas probables conmutando solo los menos fiables. Más concretamente, la lista consiste en todas las sub-huellas con los N bits más fiables fijados y el resto variables. Por ejemplo, si la información de fiabilidad es perfecta y el número mínimo de errores por sub-huella es 3, puede identificarse el bloque con sólo 8 ($=2^3$) búsquedas, que, comparado con 5489 es una gran mejora. Por supuesto, la información de fiabilidad no es perfecta y la mejora es menos, pero aun así sigue siendo sustancial. Lo podemos ver en la siguiente figura:

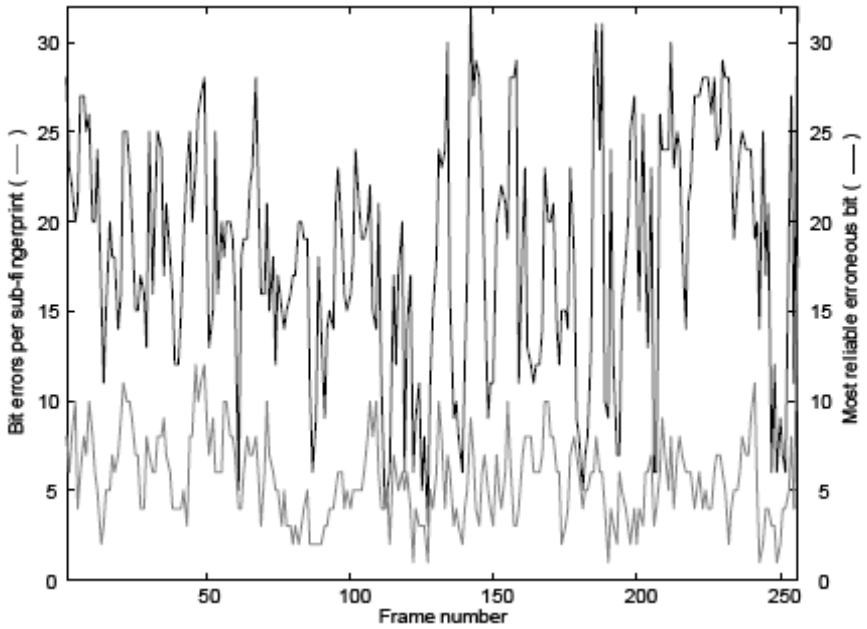

Fig. 10: Bits erróneos por sub-huella (en gris) y fiabilidad del bit erróneo más fiable (en negro) de la versión mp3@32kbps de O Fortuna de Orff.

Por ejemplo, la primera sub-huella tiene 8 errores. Estos 8 bits erróneos no son los 8 más débiles, porque uno de ellos tiene una fiabilidad de 27. Sin embargo, la sub-huella tiene sólo un error, que resulta ser el tercero menos fiable. Por tanto, este bloque habría sido apuntado al lugar correcto cuando se conmutaran los 3 bits más débiles. La canción se habría identificado bien.

Acabamos dando un ejemplo de como funciona el esquema de la fig.8. La última sub-huella extraída del bloque de la figura es 0x00000001. Primero el bloque es comparado con las posiciones en la base de datos

donde se encuentra dicha sub-huella. La LUT sólo apunta a una posición para dicha huella, una cierta posición p de la canción 1. Ahora se calcula la BER entre el bloque y los valores entre la posición p-255 y p de la canción almacenada. Si la BER está por debajo de 0,35, la probabilidad de que corresponda a esa canción es alta, si no, o bien la canción no está en la base de datos o la sub-huella tiene un error. Asumimos que el menos fiable es el '1' del final y la sub-huella más probables es 0x00000000. Ésta tiene dos candidatos, en la canción 1 en la 2. Si el bloque tiene una BER por debajo del umbral al comparar con alguna de las dos, se dirá que hay coincidencia. Si no, bien se usan otras sub-huellas probables o se coge alguna de las otras 254 sub-huellas, donde se repite el proceso. Si todas las 256 sub-huellas y sus sub-huellas más probables han generado posiciones candidatas y ninguna ha dado una BER por debajo del umbral, el algoritmo decide que no puede identificar la canción.

Como conclusión, podemos resumir el sistema según los parámetros de diseño de este tipo de sistemas:

- Tamaño de huella: Se extrae una huella de 32 bits cada 11,8 milisegundos, lo que da una tasa de 2,6 kbps.

- Granularidad: Un bloque consistente en 256 sub-huellas y correspondiente a 3 segundos de audio es la unidad básica de identificación.

- Velocidad de búsqueda y escalabilidad: Usando un algoritmo en dos fases una base de datos de huellas que contenga 20.000 canciones y maneje docenas de peticiones por segundo puede ser ejecutada en un ordenador moderno.

3.2 Análisis prácticos a realizar

Una vez expuesto el algoritmo usado por Philips, el primer objetivo del presente proyecto es realizar una serie de análisis prácticos del mismo, sobre todo de dos parámetros fundamentales para describir la actuación de todo sistema: La Probabilidad de Falso Positivo y la Robustez ante degradaciones de la señal.

Para ello, se ha programado en Matlab el algoritmo de extracción de huellas, junto con una serie de funciones de comprobación de la BER entre dos huellas dadas. Todos estos códigos se incluyen en el Anexo 1 del proyecto.

Una vez demostrada la idoneidad o no del sistema, se propondrán una serie de opciones de preprocesado de la señal de audio antes de ser introducida en el sistema y se realizarán los mismos análisis para comprobar si dichas opciones suponen una mejora en las prestaciones o no. Para esto se usarán las funciones anteriormente mencionadas junto con otras que serán necesarias para programar los distintos algoritmos de preprocesado.

3.2.1 Probabilidad de falso positivo

Diremos que dos señales de audio de 3 segundos son iguales si la distancia Hamming (es decir, el número de bits erróneos) entre las huellas derivadas de ambos bloques está por debajo de un cierto umbral T. El valor T de dicho umbral condiciona directamente la probabilidad de falso positivo, P, es decir, la probabilidad de que dos señales de audio distintas sean consideradas iguales. Mientras más pequeño sea dicho umbral, evidentemente menor será la probabilidad de que esto pase. Pero, por otro lado, mientras más pequeño sea T, mayor será la probabilidad de que dos señales iguales sean consideradas diferentes (Probabilidad de Falso Negativo).

Si empezamos con un análisis teórico del problema, para analizar la elección de dicho umbral T, asumimos que el proceso de extracción de huellas da como resultado unos bits aleatorios e i.i.d. (independientes e idénticamente distribuidos). El número de bits erróneos tendrá una distribución binomial (n,p), donde n es el número de bits extraídos y p (=0,5) es la probabilidad de extraer un '1' o un '0'. Puesto que n (=32*256) es grande en nuestra aplicación, la distribución binomial puede aproximarse por una distribución normal de media m=n*p y desviación estándar $\sigma=\sqrt{np(1-p)}$. Dado un bloque de huellas F_1, la probabilidad de que un bloque seleccionado aleatoriamente F_2 tenga menos de T=a*n errores con respecto a F_1 viene dada por:

$$P_f(\alpha) = \frac{1}{\sqrt{2\Pi}} \int_{(1-2\alpha)\sqrt{n}}^{\infty} e^{-x^2/2}\,dx = \frac{1}{2}\,erfc\left(\frac{(1-2\alpha)}{\sqrt{2}}\sqrt{n}\right) \qquad (2)$$

Donde α denota la probabilidad de error, la BER.

Sin embargo, en la práctica las sub-huellas tienen una alta correlación a lo largo del eje temporal. Esta correlación es debida no sólo a la inherente correlación temporal existente en el audio, sino también al gran solapamiento de las tramas usadas en la extracción de huellas. Una correlación más alta implica una desviación estándar más grande. Esto se demuestra en la siguiente discusión.

Asumamos una fuente binaria simétrica con alfabeto {-1,1}, tal que la probabilidad del símbolo x_i y del símbolo x_{i+1} sean iguales es q. Entonces se podría demostrar que

$$E[x_i x_{i+k}] = a^{|k|} \qquad (3)$$

donde a=2*q-1. Si la fuente Z es el XOR de dos secuencias como esa X e Y, entonces Z es simétrica y

$$E[z_i z_{i+k}] = a^{2|k|} \qquad (4)$$

Para N grande, la desviación estándar de la media $\overline{Z}_N$ sobre N muestras consecutivas de Z puede ser descrita aproximadamente por una distribución normal de media 0 y de desviación estándar igual a

$$\sqrt{\frac{1+a^2}{N\left(1-a^2\right)}} \qquad (5)$$

Trasladando este razonamiento al caso de bits de las huellas, un factor de correlación a entre bits de huellas consecutivas implica un incremento en la desviación estándar en un factor

$$\sqrt{\frac{1+a^2}{\left(1-a^2\right)}} \qquad (6)$$

En [10], para determinar la distribución de la BER con bloques reales lo que se hizo fue generar una base de datos de 10.000 canciones. Después se determinó la BER de 100.000 pares de bloques seleccionados aleatoriamente. La desviación estándar de la distribución resultante se midió en 0,0148, aproximadamente 3 veces mayor de los 0,0055 que se esperarían de bits aleatorios i.i.d.

En la figura 10 se muestra la función densidad de probabilidad de la distribución de la BER medida y de una distribución normal de media 0,5 y desviación estándar 0,0148. La PDF de la BER es una aproximación bastante cercana a la distribución normal. Para BERs por debajo de 0,45 se observan algunos valores atípicos debido a un número insuficiente de valores.

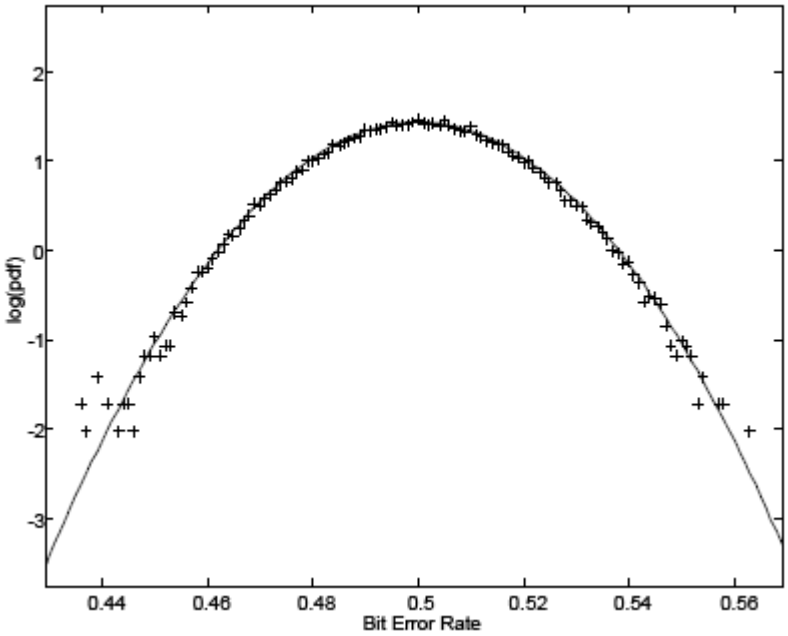

Fig. 11: Comparación de la PDF de la BER dibujada con + y la distribución normal

Para incorporar la mayor desviación estándar de la distribución de la BER la fórmula () es modificada incluyendo un factor 3.

$$P_f(\alpha) = \frac{1}{2} erfc\left(\frac{(1-2\alpha)}{3\sqrt{2}} \sqrt{n} \right) \qquad (7)$$

Usando esta fórmula con un umbral de $\alpha = 0{,}35$ y sustituyendo en la fórmula anterior se llega a un resultado teórico de FPR=3,6*10^{-20}.

En la práctica, lo primero que se nos podría ocurrir hacer es coger un trozo cualquiera de 3 segundos y compararlo con un archivo de audio de varias horas. Si se compara con un archivo de una hora, el número de comparaciones es del orden de las 300.000. Como hemos visto en el análisis teórico, la BER es del orden de 10^{-20}, así que este enfoque directo es inviable. Lo que habrá que hacer es obtener el histograma de dichas medidas de error, su media, su desviación estándar y su función densidad de probabilidad y, a partir de ahí, obtener una FPR "teórica", tal y como se hace en [10]. Para obtener la f.d.p. a partir de una serie de medidas se usa la herramienta de Matlab, *Distribution Fitting Tool (Herramienta de Ajuste de Distribución),* que nos permite representar la f.d.p. de un conjunto de datos, que en nuestro caso será el vector de probabilidades de error y encontrar una distribución que se ajuste bien a la distribución empírica. Vamos a realizar el ajuste con una distribución normal, ya que previamente se ha usado otra instrucción de Matlab, **normplot**, que nos permite saber, observando una simple gráfica si los datos en un determinado vector tienen una distribución normal. Si la tienen, en la gráfica se observará una recta, mientras que si la distribución que siguen es otra aparecerán curvaturas. El resultado de la orden normplot se muestra en la siguiente figura:

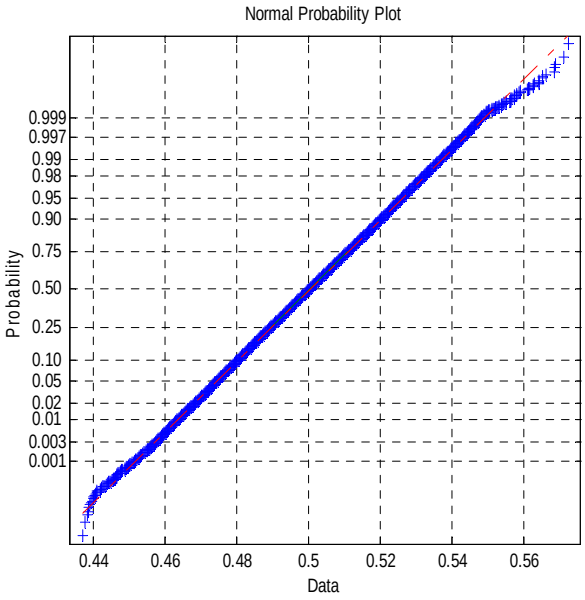

Fig 12a: Resultado de la orden normplot

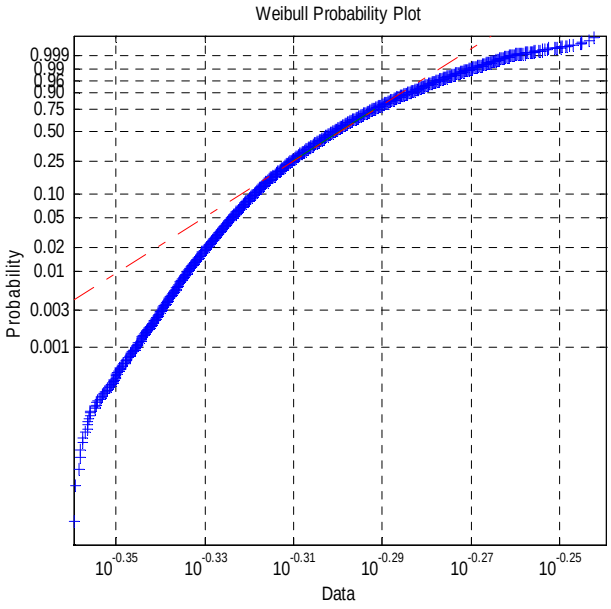

Fig 12b: Resultado de weibplot

El dibujo tiene tres elementos gráficos. Los signos '+' muestran la probabilidad empírica contra el valor de los datos para cada punto en la muestra. La línea continua roja conecta los percentiles 25 y 75 de los datos

y representa un ajuste lineal robusto (es decir, insensible a los extremos de la muestra). La línea punteada extiende la línea continua hasta los extremos de la muestra.

Si todos los datos caen cerca de la línea la asunción de normalidad es razonable, y en este caso podemos observar que solo en los casos muy extremos, es decir, para probabilidades por debajo de 0.001 o por encima de 0.999 se aleja de la recta. Como comparación, si usamos **weibplot**, que hace lo mismo pero suponiendo que la distribución es una distribución de Weibull el resultado se observa en la fig. 11b. Comparando ambas, se puede observar fácilmente por qué vamos a trabajar con la suposición de que la distribución del error es normal. Sin embargo, por definición, la BER no puede ser negativa, así que trabajaremos con una f.d.p. lognormal

Una vez hecho esto ejecutamos **dfittool**, para cargar la herramienta de ajuste y definimos un ajuste lognormal. El resultado del ajuste se presenta en la siguiente figura y la siguiente tabla:

```
Distribution:    Lognormal
Log likelihood:  281939
Domain:          0 < y < Inf
Mean:            0.500153
Variance:        0.000247017

Parameter  Estimate   Std. Err.
mu         -0.693335  9.78455e-005
sigma      0.0314162  6.91877e-005

Estimated covariance of parameter estimates:
       mu            sigma
mu     9.57374e-009  1.48481e-020
sigma  1.48481e-020  4.78694e-009
```

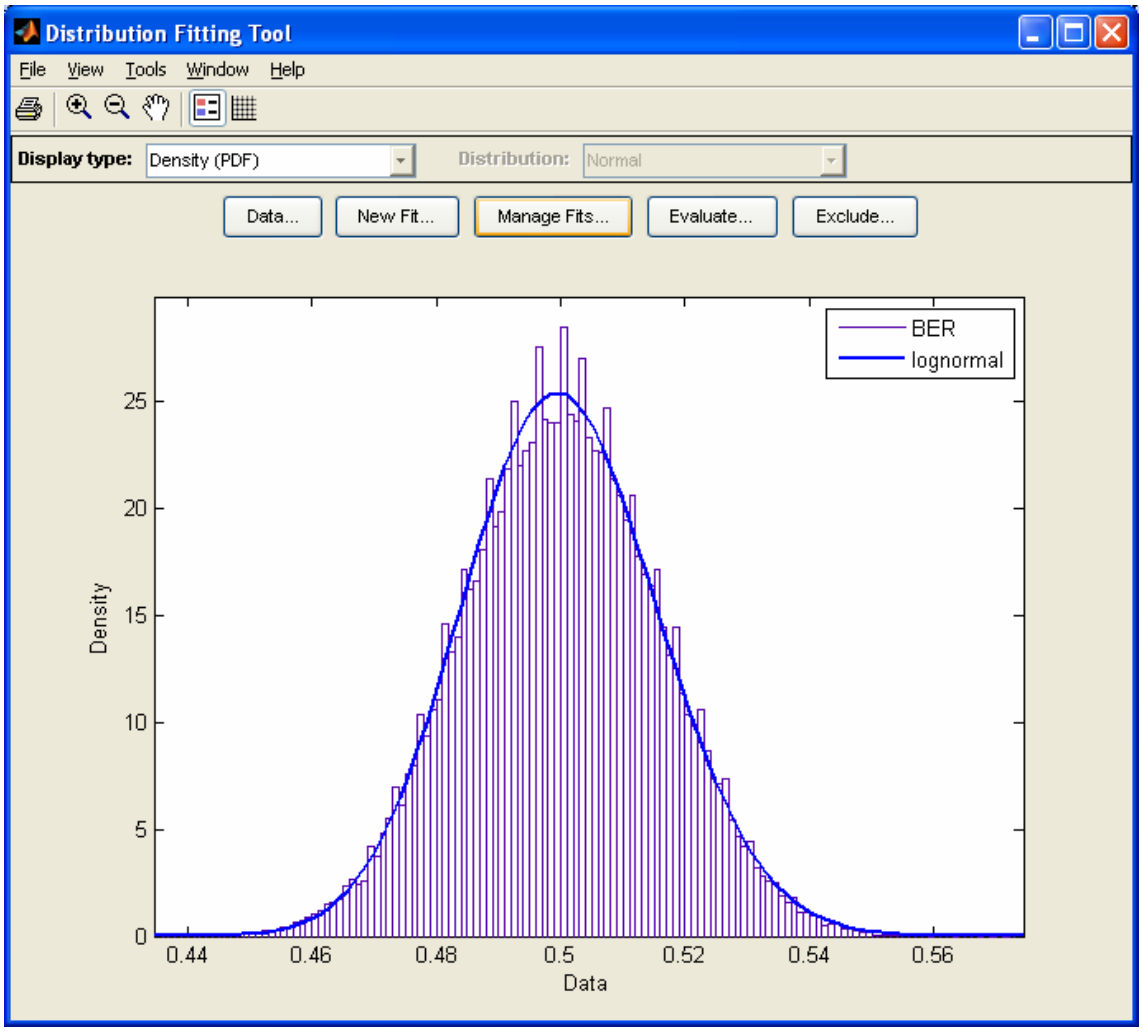

Fig 13: Comparación de la f.d.p. de la BER y de una f.d.p lognormal

El último paso que nos queda es obtener la FPR en sí misma, que será simplemente la probabilidad de que una distribución lognormal de media -0.693335 y desviación típica 0.0314162 esté por debajo de 0.35. Una posible manera de hacerlo es integrando la f.d.p de una lognormal entre 0 y 0.35, pero dicha integración resulta complicada, por lo que finalmente se ha optado por hacerlo en matlab. Si observamos la fig. 12, arriba de la gráfica tenemos la opción **evaluate,** que evalúa varios parámetros de la f.d.p estimada. La probabilidad de falso positivo será simplemente el valor de la función de cuantía (c.d.f., cumulative density function) para x=0.35. Lo podemos ver en la siguiente figura:

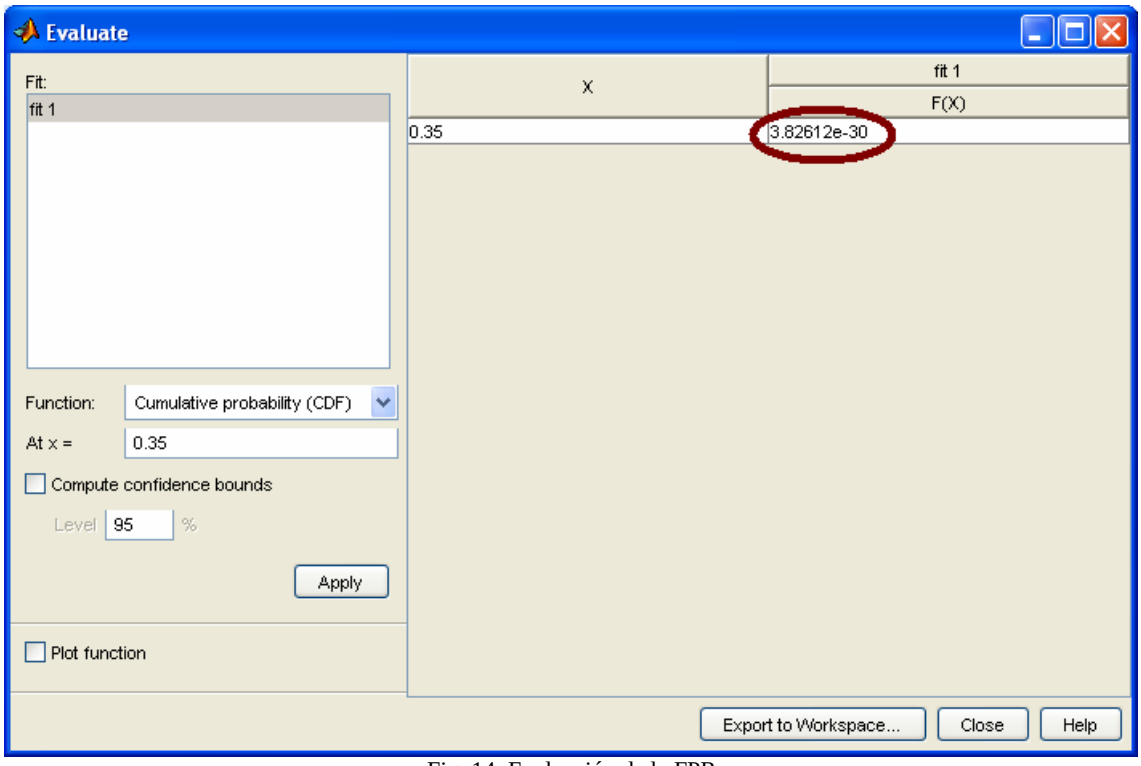

Fig. 14: Evaluación de la FPR

FPR= 3.82612e-30

Comparado con el resultado teórico de Haitsma [10], se ve que hay una diferencia apreciable en los órdenes de magnitud. Esto se debe a que nosotros hemos usado una distribución lognormal, que calcula la probabilidad acumulada entre 0 y 0.35 y en el de Haitsma se usa una normal, que lo hace entre -∞ y 0.35. De hecho, si nosotros realizamos este mismo análisis con una distribución normal el resultado que se obtiene es del orden de 10^{-22}. En realidad, si se piensa bien, esta diferencia de los órdenes de magnitud no tiene una gran importancia, el hecho importante era demostrar que la probabilidad de que se produzca un falso positivo en una aplicación normal es prácticamente despreciable, lo cual, a la vista del resultado, se puede afirmar sin miedo a equivocarse mucho.

3.2.2 Análisis experimental de robustez

Con este análisis pretendemos demostrar la robustez del esquema propuesto. Es decir, tratamos de responder la cuestión de si la BER entre la versión original y la versión degradada de un clip de audio está por debajo o no del umbral α, que sigue siendo igual a 0.35.

Para ello vamos a coger cuatro extractos cortos (estéreo, 44100 Hz, 16 bps) de diferentes archivos de audio de diferentes géneros musicales: el ya mencionado anteriormente *"The Dream Of Unreality"* de Pictures Of Shorelines, un fragmento hablado de *"A Day in the Life"* de The Beatles,

"Drugs or Me" de Jimmy Eat World y un fragmento de *"Chariots Of Fire"* de Vangelis.

A cada uno de los extractos se les ha sometido a una serie de degradaciones de la señal, usando para ello el programa *Adobe Audition*. En la siguiente figura se muestra la apariencia general del programa:

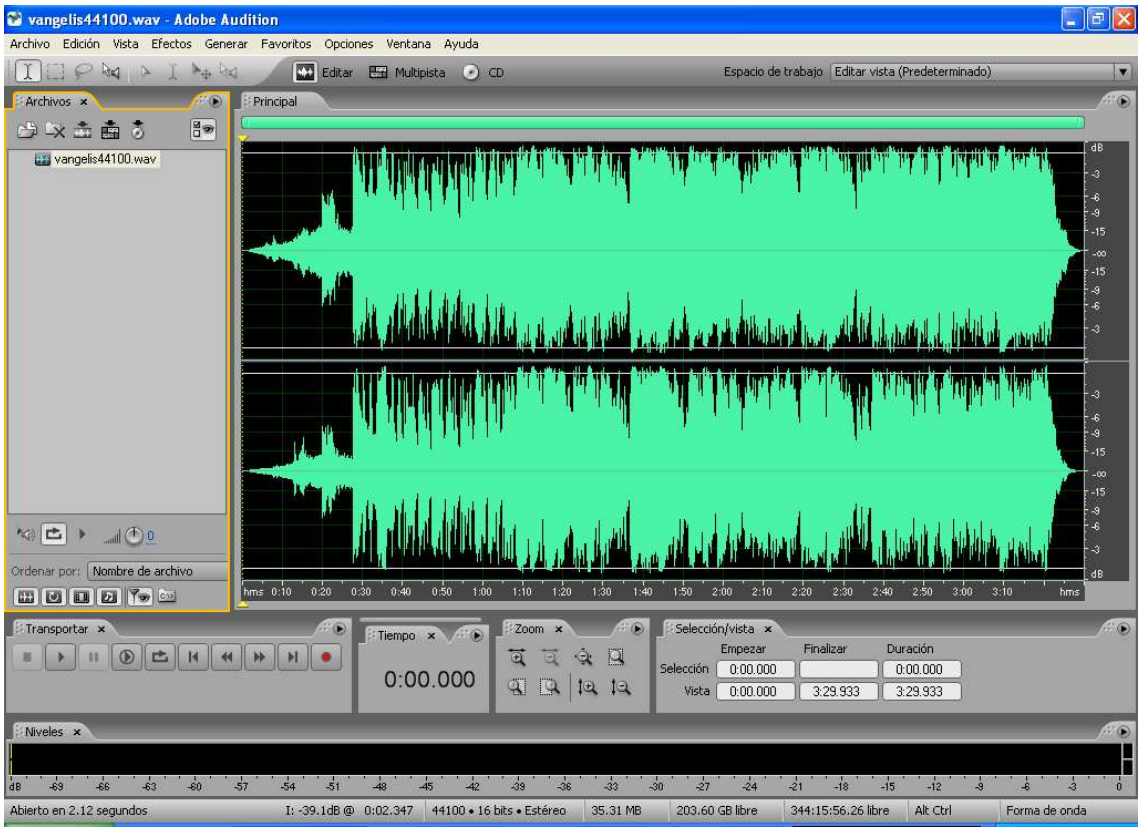

Fig. 15: Apariencia general de Adobe Audition

Los procesos a los que han sido sometidos los fragmentos de audio son los siguientes:

- **Codificación/decodificación MP3** a 128 kbps y 32 kbps.
- **Compresión de Amplitud** con las siguientes tasas de compresión: 8.94:1 para $|A| \geq -28.6$ dB; 1.73:1 para $-46.4 \text{ dB} < |A| < -28.6$ dB; 1:1.61 para $|A| \leq -46.4$ dB.

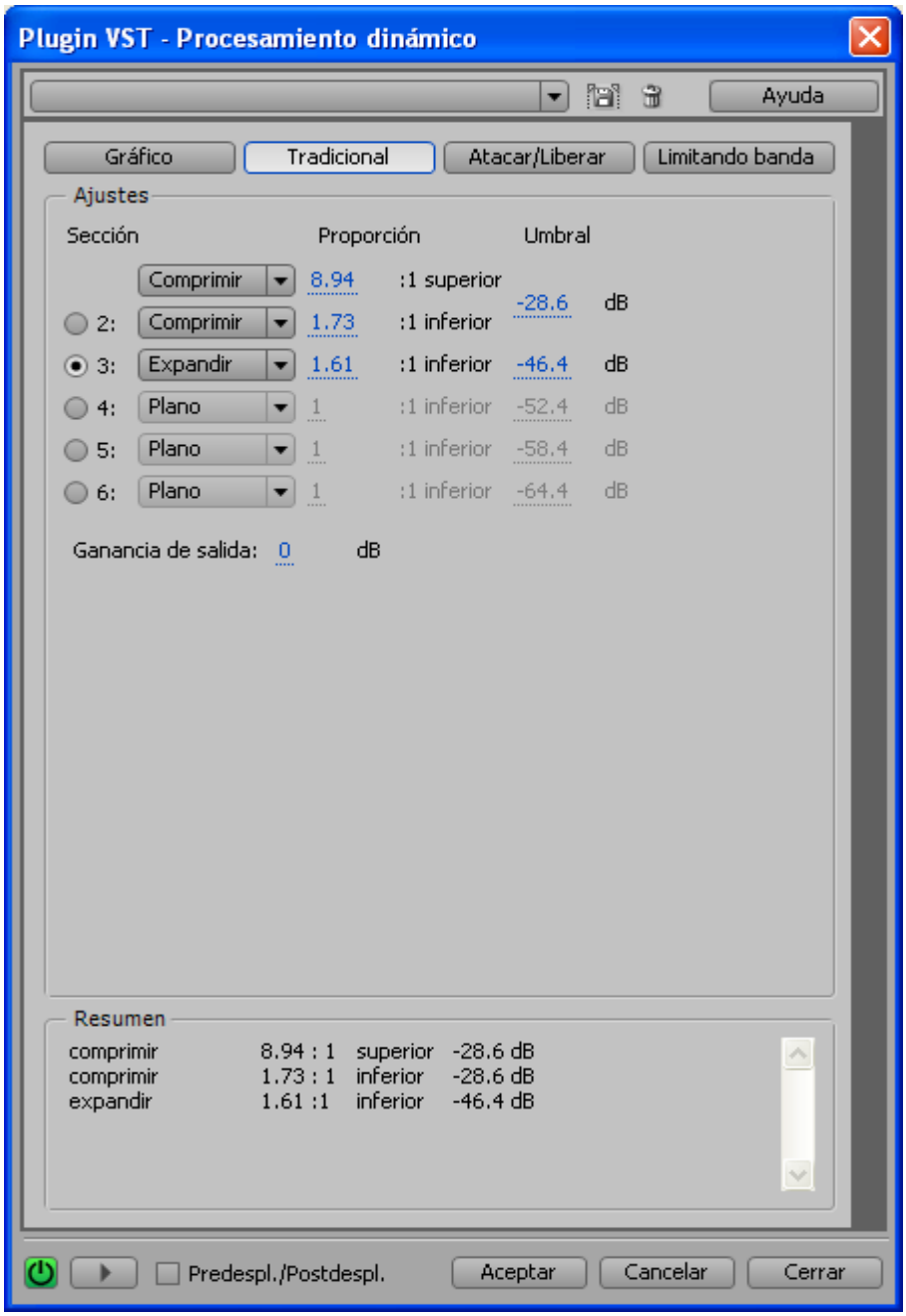

Fig 16: Compresión en Audition

- **Ecualización** Un ecualizador típico de 10 bandas con los siguientes parámetros:

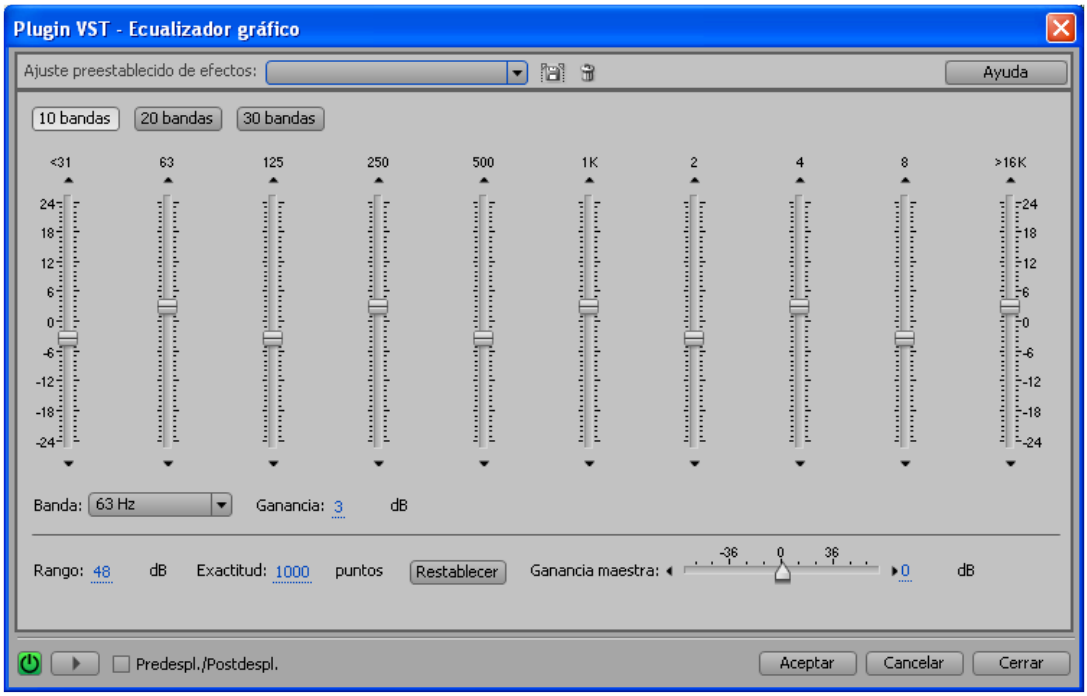

Fig.17: Ecualizador utilizado

- **Adición de eco**
- **Filtrado paso de banda** usando un filtro Butterworth de segundo orden con frecuencias de corte de 100 Hz y 6000 Hz.

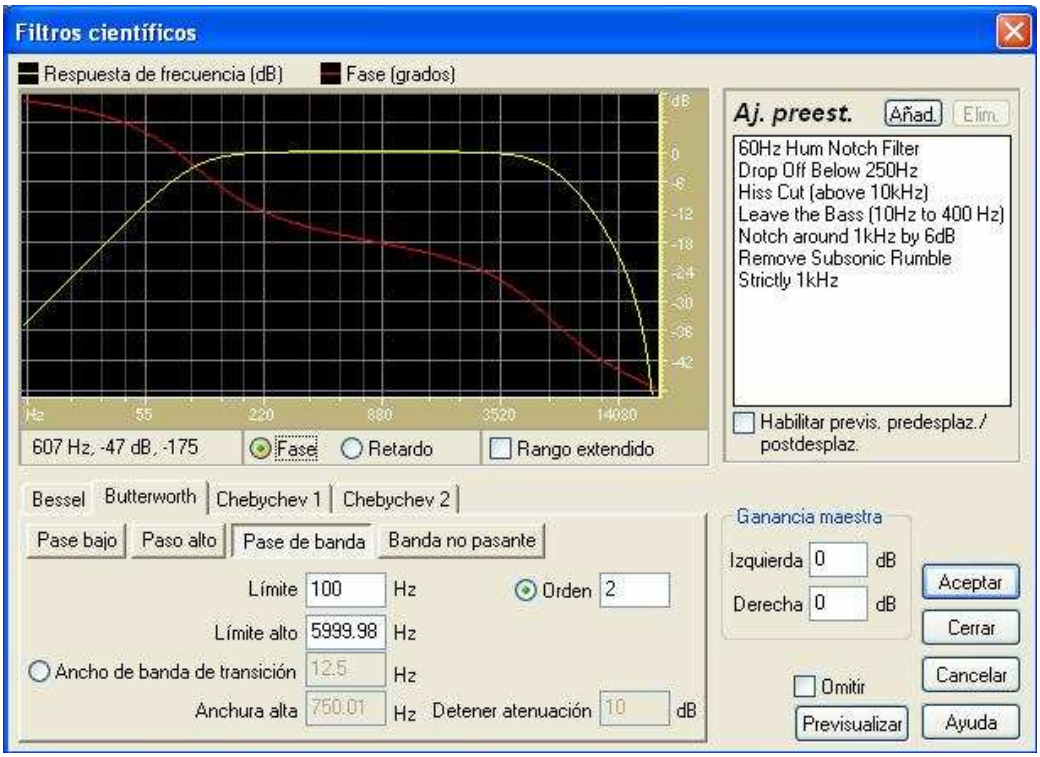

Fig.18: Filtro Butterworth en Audition

- **Modificación de la escala temporal** en un +4% y -4%. Solo cambia el tempo, el tono permanece inalterado.

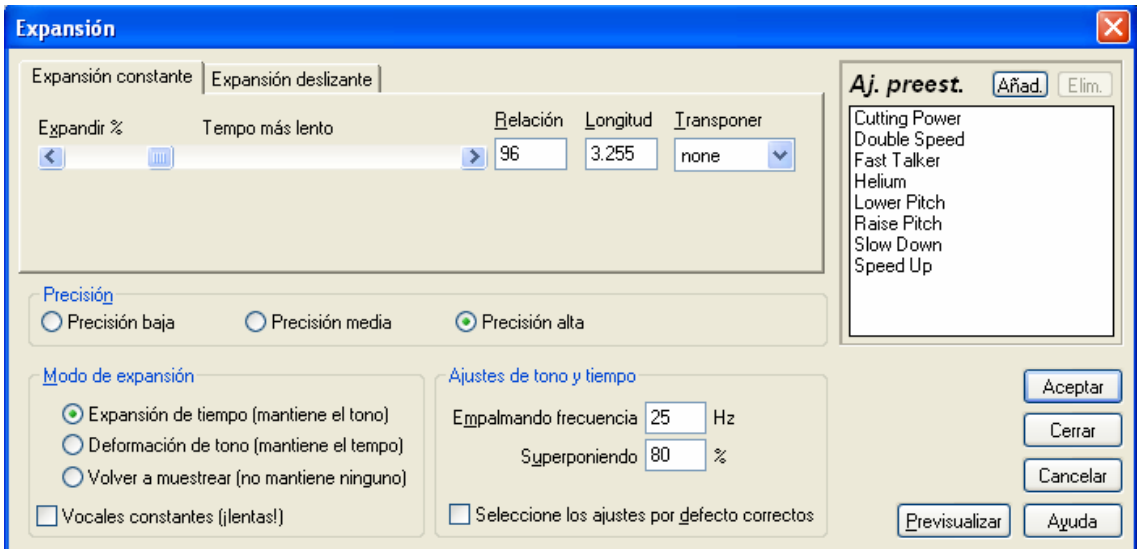

Fig. 19: Expansión de un -4% en Audition

- **Cambio lineal de la velocidad** del +1%, -1%, +4%, -4%. Cambian tanto el tono como el tempo. No incluimos imagen ya que el menú es el mismo que en la anterior transformación, señalando en modo de expansión la opción "Volver a muestrear".
- **Adición de Ruido** blanco uniforme.

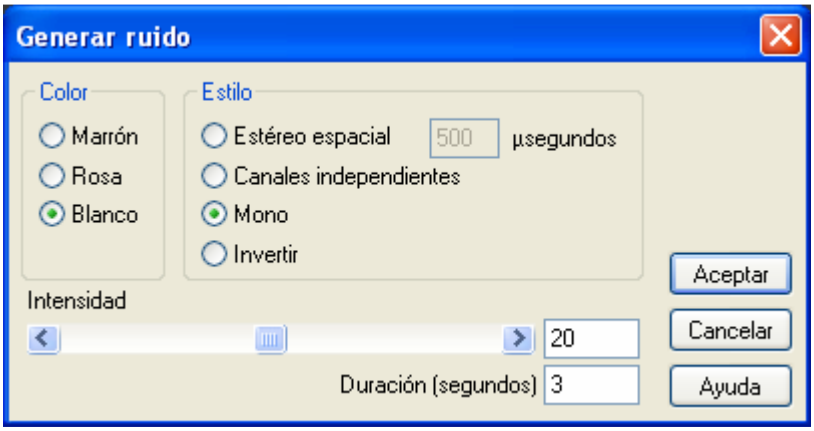

Fig.20: Generación de ruido blanco

- **Resampling** consistente en submuestreo y posterior sobremuestreo a 22050 Hz y otra vez a 44100 Hz.

Después de realizar dichas transformaciones (por separado, a ningún clip de audio se le hacen dos transformaciones distintas), se mide la BER entre el fragmento original y el fragmento degradado para cada una de las cuatro canciones originales. Los resultados se muestran en la siguiente tabla:

Procesado	Pictures	Beatles	Jimmy Eat World	Vangelis
MP3@128 kbps	0.0851	0.0379	0.0666	0.0796
MP3@32 kbps	0.2313	0.0961	0.0984	0.1406
Comp. Amplitud	0.1430	0.0522	0.0722	0.1308

Ecualización	0.0343	0.0161	0.0142	0.0134
Adición de Eco	0.1707	0.0904	0.1216	0.1423
Filtrado paso banda	0.0357	0.0078	0.0106	0.0099
Esc. Tiempo +4%	0.1571	0.2208	0.1488	0.1819
Esc. Tiempo -4%	0.1980	0.1925	0.2259	0.2260
Veloc. Lineal +1%	0.1590	0.1622	0.1254	0.1057
Veloc. Lineal -1%	0.1795	0.1403	0.2695	0.2303
Veloc. Lineal +4%	0.4312	0.4437	0.3571	0.3260
Veloc. Lineal -4%	0.5288	0.4343	0.5435	0.5592
Adición Ruido	0.0809	0.0611	0.0487	0.0541
Resampling	0.000	0.000	0.000	0.000

Tabla 1: BER para los distintos tipos de degradación de señal

Vemos que sólo en los casos señalados en rojo se supera el umbral de error de 0.35. Corresponden con el caso de los cambios lineales de velocidad del 4%, tanto en un sentido como en el contrario. Esto es debido a la desviación del tramado (desalineamiento temporal) y al escalado espectral (desalineamiento frecuencial).

.