INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

MPX
FORMATO DE COMPRESIÓN DE AUDIO

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTAN:

LEDEZMA TREJO JUAN MANUEL
PALLARES GONZÁLEZ JORGE ALBERTO

DIRECTORES:

DR. OSCAR HERRERA ALCÁNTARA
M. en C. JOSÉ SÁNCHEZ JUAREZ

México D.F. 2002

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA

Serie: Amarilla
Documentos Técnicos

Mayo del 2002

"MPX" Formato de compresión de audio

Ledezma Trejo Juan Manuel
manuel_led@yahoo.com

Pallares González Jorge Alberto
jorge_pallo@yahoo.com

Resumen

Este trabajo presenta un formato de compresión de audio basado en la Transformada Wavelet que promete ser un sustituto de la Transformada de Fourier en muchas aplicaciones.

Para esto se toma como referencia el estándar MP3 de compresión de audio con todas sus características para crear finalmente un formato de compresión propio.

Advertencia

Este informe contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional a partir de datos y documentos con derecho de propiedad y por lo tanto su uso queda restringido a las aplicaciones que explícitamente se convengan.

La aplicación no convenida exime a la escuela de su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Agradecimientos

Manuel:

Al eterno creador y dueño del conocimiento.

A mi padres que siempre me han apoyado e impulsado a seguir adelante.

A mi hermano por su ayuda y noches de desvelo junto a mi.

A Diana quien me ha impulsado a terminar este proyecto.

Jorge:

Al eterno creador y eterno dueño del conocimiento.

A mis padres por todo el apoyo que me han dado desde siempre.

A mi hermana por su apoyo incondicional.

A todo aquellos que me apoyaron en el desarrollo de este proyecto.

A nuestro director y amigo Oscar Herrera.

Introducción

El advenimiento de la computación multimedia ha llevado a un creciente uso de audio digital. La reproducción de audio digital representa un costo computacional elevado debido a que esta se realiza en tiempo real, además de que este tipo de archivos son de un tamaño excesivamente grande.

La compresión de audio es un proceso destructivo. Los algoritmos empleados truncan el archivo original por lo que el sonido obtenido no es el mismo. Los métodos de codificación de audio que existen en la actualidad se basan en algoritmos de compresión y en codificación multicanal. Toda compresión de datos de audio se basa en la comprensión del mecanismo auditivo, por lo que constituye una forma de codificación perceptual.

El oído es sólo capaz de extraer una cierta proporción de la información contenida en un determinado sonido, siendo redundante el sonido adicional. Un sistema ideal debe eliminar toda redundancia, dejando únicamente el sonido audible al oído humano. Básicamente son dos los fenómenos que son objeto de estudio y que han originado los métodos de compresión: la curva de sensibilidad del oído y el fenómeno de enmascaramiento.

Digitalización.

El sonido es una onda continua que se propaga a través del aire u otros medios. Al tratarse de una onda continua, se requiere un proceso de digitalización para representarla como una serie de números. El almacenamiento como el procesado y transmisión de la señal en forma digital ofrece ventajas muy significativas sobre los métodos analógicos. La tecnología digital es más avanzada y ofrece mayores posibilidades, menor sensibilidad al ruido en la transmisión y capacidad de incluir códigos de protección frente a errores, así como encriptación. La desventaja principal de la señal digital es que requiere un ancho de banda mucho mayor que el de la señal analógica, de ahí que se realice un exhaustivo estudio en lo referente a la compresión de datos.

El proceso de digitalización se compone de dos fases: muestreo y cuantización. En el muestreo se divide el eje del tiempo en segmentos discretos. En estos momentos se realiza la cuantización, que, en su forma más sencilla, consiste simplemente en medir el valor de la señal en amplitud y guardarlo. Por tanto, siendo el rango superior de la audición humana en torno a los 20 Khz., la frecuencia que garantiza un muestreo adecuado para cualquier sonido audible será de unos 40 Khz. Concretamente, para obtener sonido de alta calidad se utilizan frecuencias de 44'1 Khz., en el caso del CD, por ejemplo, y hasta 48 Khz., en el caso del DAT. Otros valores típicos son submúltiplos de la primera, 22 y 11 Khz. Según la naturaleza de la aplicación, por supuesto, las frecuencias adecuadas pueden ser muy inferiores, de tal manera que el proceso de la voz acostumbra a realizarse a una frecuencia de entre 6 y 20 Khz.

En lo referente a la cuantización, es evidente que cuantos más bits se utilicen para la división del eje de la amplitud, más "fina" será la partición y por tanto menor el error al atribuir una amplitud concreta al sonido en cada instante. El margen dinámico de la audición humana es de unos 100 dB. El proceso completo se denomina habitualmente PCM (Pulse Code Modulation). Todo esto es tratado a fondo en el capítulo 3.

Para hacer práctico el uso del audio digital, es necesario utilizar alguna técnica de compresión. En el presente trabajo se desarrolla un formato de compresión de audio que utiliza el análisis de wavelets.

En el capitulo 2 se presenta la obtención de los datos; esto se refiere al rippeo (destripador) en este proceso se pasan los datos del CDDA (CD de audio) a un archivo a disco duro.

En el capitulo 4 se muestran algunas transformaciones lineales para llevar el archivo de un dominio del tiempo a otro espacio mas comprimible.

En el capítulo 6 se presentan algunas de las técnicas de compresión de datos más utilizadas, i.e., RLE, Huffman, Codificación Aritmética y LZW.

En el capítulo 7 se describe el formato de compresión de audio desarrollado.

Los capítulos 8, 9 y 10 discuten los resultados obtenidos, la revalorización de objetivos y las limitantes y recomendaciones del formato de compresión wavelet, que pueden servir de base a trabajos futuros.

Índice Temático

Lista de figuras

Figura 8.13 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido diez veces de un MP3 a 192 Kbps

Figura 8.14 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido diez veces de un MP3 a 128 Kbps.

Lista de tablas

Glosario

API. Application Programming Interface

RIPPER. Por ripper entendemos el software que es capaz de extraer de un CD de Audio archivos en formato WAV. Un "CD Ripper" copia los datos de audio desde un CD de audio al disco duro.

CODIFICACIÓN DE TRANSFORMADA. Técnica de codificación basada en una transformación lineal ortogonal.

CODIFICADOR ENTRÓPICO. Codificador que aprovecha la redundancia de información existente en una señal digital para crear secuencias de bits. Se utilizan menos bits para aquellas secuencias que aparecen con mayor frecuencia logrando así la compresión de datos. Puesto que es posible recuperar la señal original se conoce también como compresión sin pérdida.

CODIFICACIÓN DE SUBBANDA. Técnica de codificación donde la señal de entrada es filtrada y separada en bandas de frecuencia.

CUANTIZACIÓN. Proceso de conversión de una señal discreta en tiempo y continua en valores, a una señal discreta en tiempo y discreta en valores. El valor de cada muestra de la señal es representada a partir de un conjunto finito de valores.

SOBREMUESTREO (UPSAMPLING). Proceso de insertar una muestra por cada m de un conjunto existente, generalmente con valor de cero.

SUBMUESTREO (DOWNSAMPLING). Proceso que consiste en despreciar una de cada m muestras de una señal.

WAVELET. Función matemática que puede representarse como una combinación lineal de sí misma aplicando traslaciones y dilataciones y cuya función de escalamiento asociada da lugar a un análisis de multiresolución.

1. El Problema

En este capítulo se presenta el problema a tratar, su descripción, justificación y los objetivos perseguidos.

1.1 Descripción

El mayor obstáculo en la revolución multimedia es la obesidad digital, i.e, el congestionamiento que ocurre en los medios de comunicación cuando las imágenes, sonido y video son convertidos de su estado natural analógico al lenguaje de las computadoras para su manipulación o transmisión.

La compresión de datos ha sido una herramienta matemática que ha permitido el desarrollo de muchas aplicaciones en Internet.

En este trabajo terminal se desarrolla un nuevo formato de compresión que utiliza análisis de wavelets, y se propone la creación de un formato de compresión de audio basado en transformaciones lineales sin perdida de calidad.

1.2 Formulación del problema

Desarrollar un formato de compresión de audio que utilice el análisis de wavelets y ponerlo a disposición de los múltiples usuarios del Web de manera análoga al formato estándar MP3 o como sustituto de formato CDA, ya que nuestro formato no tiene perdida de calidad.

1.3 Objetivos

- Desarrollar un formato de compresión de audio basado en el estándar MP3.
- Desarrollar el player (software para reproducir el archivo de música)

1.4 Justificación

En este trabajo de Tesis se presenta un formato de compresión que utiliza las técnicas matemáticas del análisis de wavelets, que representan un marco de trabajo alterno a lo que ha sido por los últimos dos siglos el estado del arte en el análisis de señales: el Análisis de Fourier.

1.5 Alcances

Una vez finalizado el nuevo formato de compresión de audio se pretende obtener:

- Nuevo formato MPX basado en el estándar MP3.
- Software para convertir de un track de CD a formato WAV y MPX.
- Software (player) para tocar el formato MPX.
- Manual de usuario para los software, tanto el convertidor como el player.

1.6. Diagramas

1.6.1 Diagrama Nivel CERO

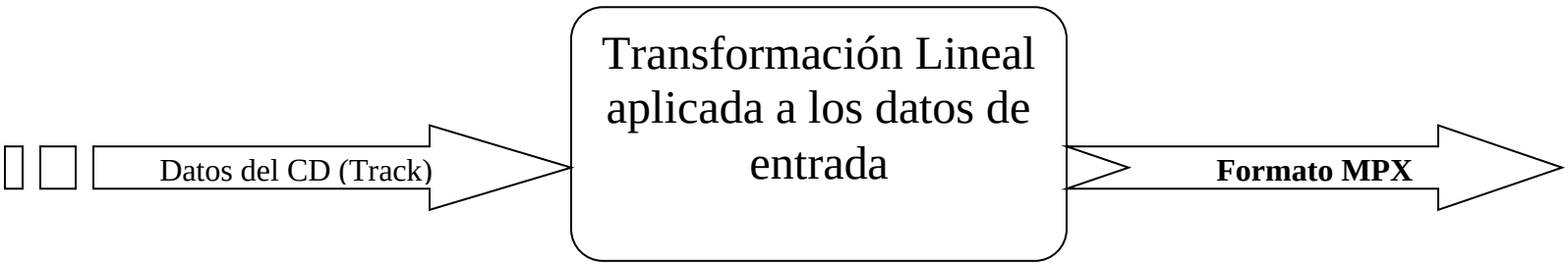


Figura 1.1 Diagrama Nivel Cero

1.6.2 Diagrama a Bloques del Formato MPX

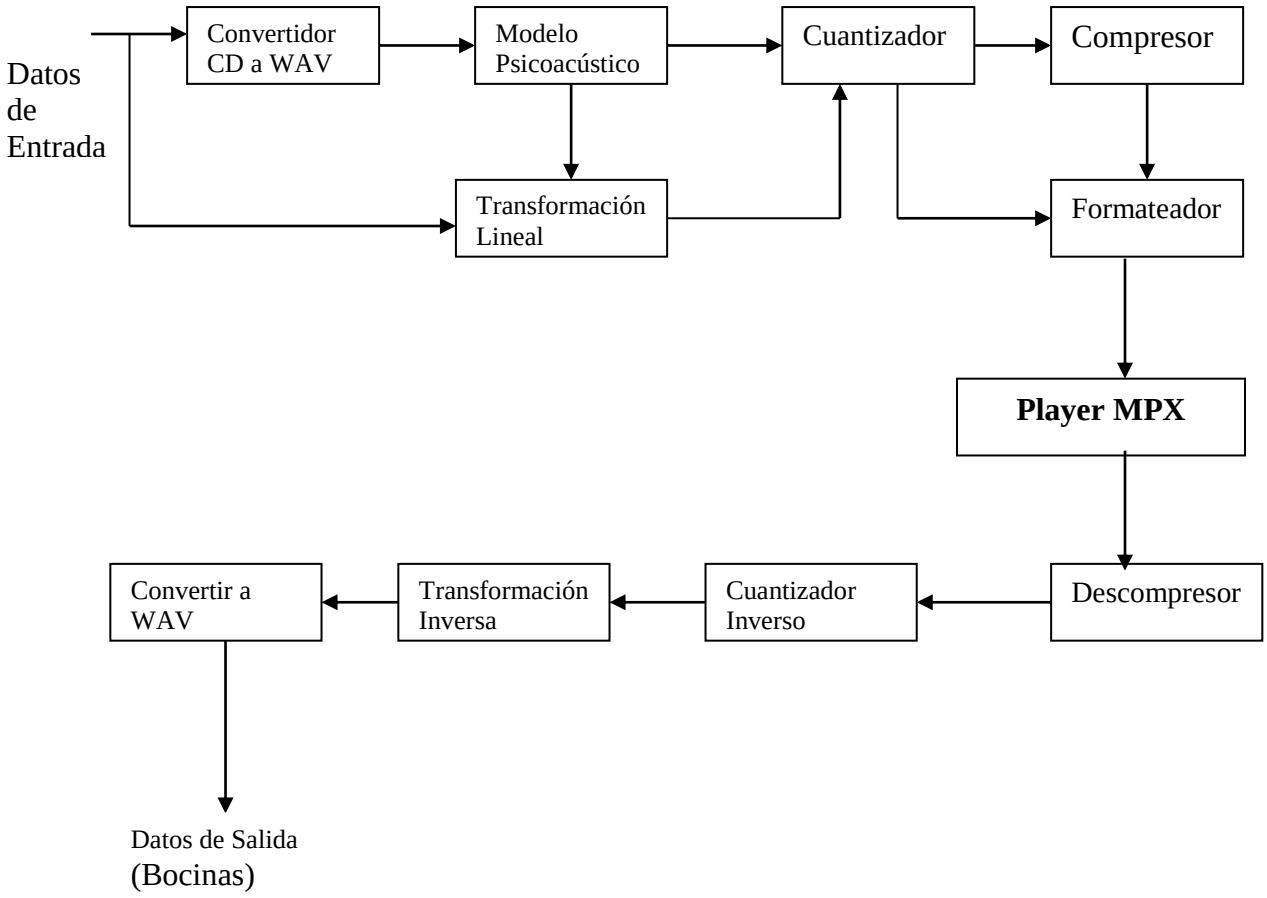


Figura 1.2 Diagrama a Bloques

2. Obtención de Datos (Ripper)

2.1 Convertidor de CD a WAV (Ripper)

Por ripper entendemos el software que es capaz de extraer de un CD de Audio archivos en formato WAV. Un "CD Ripper" copia los datos de audio desde un CD de audio al disco duro en un archivo. La razón por la cual se hace esto es que no se pueden manipular o analizar los datos de un CDDA y teniéndolos en un archivo en el disco duro ya es posible hacer un análisis y procesamiento de datos.

2.2 Problemas que pueden ocurrir al rippear

1. El CD-ROM tal vez no aguante el DAE (digital audio extraction). Generalmente esto es así por motivos de "copyright", algunos CD-ROM viejos no lo aguantan porque la tecnología no existía en ese momento.

2. EL archivo WAV tiene ruido esto es porque tu unidad de CD no es tan bueno en el DAE o tienes problemas de hardware como poco memoria ram o no tienes suficiente espacio en disco duro.

3. Algunos CD no se pueden "rippear", verifica que tu CD no este rayado.

Para llegar a obtener una buena copia hay que tener en cuenta muchos factores: funcionalidad del CD-Audio, etc. Al copiar digitalmente una pista de CD-Audio a nuestro disco duro en formato 44.1 Khz. 16 Bit Estéreo, representa que se ocupan 10 Mb de disco duro por cada minuto de canción.

Los rippers o programas extractores de audio son programas que convierten una pista de audio de un CD a un archivo en formato WAV. La velocidad de estos programas es variable, y depende, en primer lugar, de la velocidad del lector CD-ROM usado, aunque no la alcanza. Por ejemplo, con un lector CD-ROM de 32x, he conseguido velocidades típicas de 9x a 10x, es decir, 9 ó 10 veces más rápido que la velocidad de reproducción normal de la música.

2.3 Como funciona

Pasos que sigue el ripper para convertir un track en un archivo WAV

1. Verificar que la unidad de CD-ROM soporte CD de Audio y tarjeta de sonido de 16 bits.
2. Leer el contenido del CD y listar las pistas de CD – Audio.
3. Seleccionar la pista a rippear.
4. Posicionarse en el inicio de la pista seleccionada.
5. Crear la cabecera del archivo WAV.
6. Leer los datos de la pista seleccionada y copiarlos al archivo WAV.
7. Editar la cabecera del archivo WAV.

2.4 Requerimientos del Sistema

Windows 95, Windows 98, Windows 2000 o Windows NT
ASPI manager (wnaspi32.dll) instalado
ATA (E)IDE CD-ROM
16 MB RAM

2.5 Obtención del Archivo WAV

Finalmente obtenemos el archivo WAV que es una copia idéntica al track de CD que rippeamos es decir es un archivo de sonido que fue copiado bit a bit del track original del CD sin perdida de calidad. Los datos quedan como 16 bit estereo 44,000 Khz (calidad CD) en un archivo WAV.

3.- El Estándar MPEG de Audio

El estándar MPEG Audio contempla tres niveles diferentes de codificación-decodificación de la señal de audio, de los cuales sólo el primero está totalmente terminado. Los otros dos son aplicables, y de hecho se utilizan habitualmente, pero siguen abiertos a ampliaciones. Estos tres niveles son:

- MPEG-1: "Codificación de imágenes en movimiento y audio asociado para medios de almacenamiento digital hasta 1’5 Mbit/s"
- MPEG-2: "Codificación genérica de imágenes en movimiento e información de audio asociada"
- MPEG-3: la planificación original contemplaba su aplicación a sistemas HDTV; finalmente fue incluido dentro de MPEG-2.
- MPEG-4: "Codificación de objetos audiovisuales"

MPEG-3 (Capa 3) Es de desarrollo más reciente y utiliza un modelo psicoacustico, una codificación Huffman y un análisis de la señal basado en la DCT en vez de en la codificación en sub-bandas de las capas 2 y 3. Están permitidos los dos tipos de codificación joint-stereo. Permite un flujo variable y una tasa de compresión aproximadamente dos veces más elevada que la capa 2, a costa de una complejidad claramente mayor del codificador y del decodificador, así como de un tiempo de codificación / decodificación más largo. La calidad Hi-Fi se obtiene a partir de los 64 Kbits/s por canal (128 Kbits/s en estéreo). Está destinada principalmente a aplicaciones de redes de baja velocidad (por ejemplo INTERNET).

La siguiente tabla da un visión acerca de la calidad que puede ser alcanzada usando ISO/MPEG capa 3:

Compresión	Kbytes/seg.	Modo / calidad	Esquema de codificación	Ancho de Banda
1:11	15.6	Stereo/Calidad CD	MPEG-1 Capa 3	>= 16 Khz.
1:15	11.7	Stereo/Cerca al CD	MPEG-1 Capa 3	15 Khz.
1:22	7.8	Stereo/Calidad FM	MPEG-2 Capa 3	11 Khz.
1:22	3.9	Mono/Mejor que AM	MPEG-2 Capa 3	7.5 Khz.
1:44	2.0	Mono/Mejor que SW	MPEG-2 Capa 3	4.5 Khz.

Figura 3.1 Tabla de calidad usando ISO/MPEG capa 3

3.1 Modelo Psicoacustico

Los modelos psicoacusticos se componen a partir de las percepciones de un grupo de personas entrenadas para rendir al máximo en este campo. Por medio de una serie de experimentos se puede determinar la sensibilidad del oído humano a una serie de fenómenos, de forma que aparezcan resultados útiles para el tratamiento del sonido.

Para ello se llevan a cabo experimentos para medir la sensibilidad del oído humano, el enmascaramiento en frecuencia, el enmascaramiento temporal, y el enmascaramiento pretemporal.

Del víbrate que utilicemos para producir la codificación depende la eliminación de menor o mayor cantidad de datos siguiendo el modelo psicoacustico hasta lograr la compresión necesaria. Luego se cuantifican y codifican las sub bandas restantes y el resultado es finalmente comprimido mediante un algoritmo estándar Huffman o LZW.

Cuando se percibe una señal de un volumen alto (agudo o de mayor energía) en una frecuencia y otra de volumen mas bajo (grave o de menor energía) en una frecuencia cercana esta ultima no será audible.

Este efecto se llama enmascaramiento simultaneo o en frecuencia (también existe el enmascaramiento asimultáneo o en el tiempo).

Y a cierta distancia de la frecuencia enmascaradora, el efecto se reduce tanto que resulta despreciable.

El rango de frecuencias en las que se produce el fenómeno se denomina banda critica (critical band). La amplitud de la banda critica varia según la frecuencia en la que nos situemos y viene dada por determinados datos que demuestran que es mayor con la frecuencia.
De esta manera se trata de desechar todo aquello que el oído humano no podría captar.

Para aprovechar estas características se utiliza un sistema denominado Codificación de Sub bandas o CBS (sub-band coding). Donde el MP3 es el ejemplo más popular. En este proceso la señal original es descompuesta en sub bandas mediante una base de filtros.

Las sub bandas son comparadas con el original mediante el modelo psicoacustico que determina que bandas son importantes, cuales no y por ultimo cuales pueden ser eliminadas. En las siguientes figuras se muestran gráficas del modelo psicoacustico.

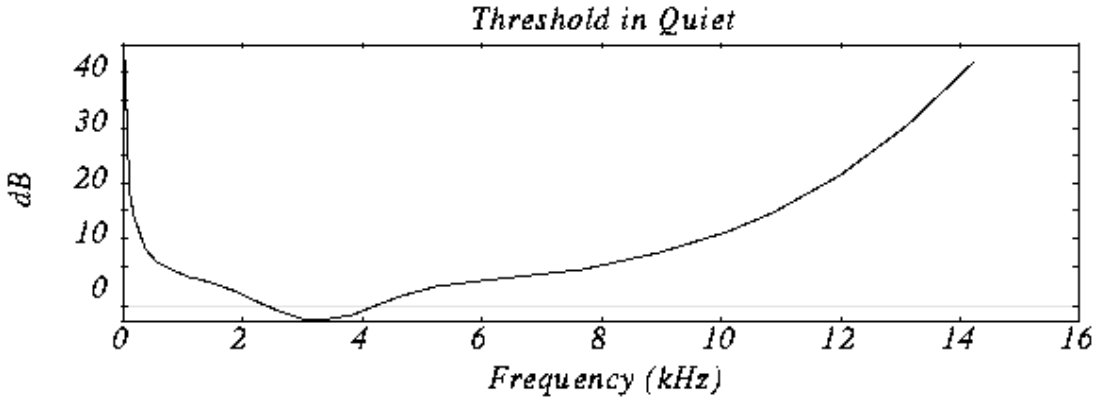


Figura 3.2 sensibilidad del oído humano en función de la frecuencia

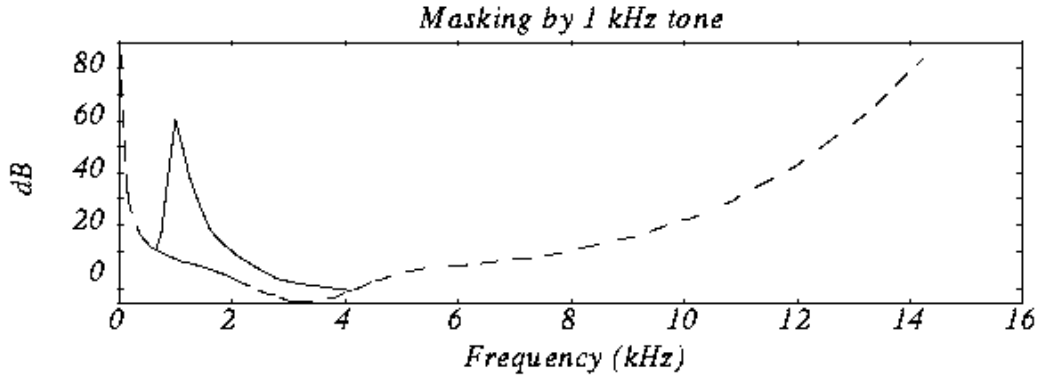


Figura 3.3 : enmascaramiento en frecuencia del tono de 1 Khz

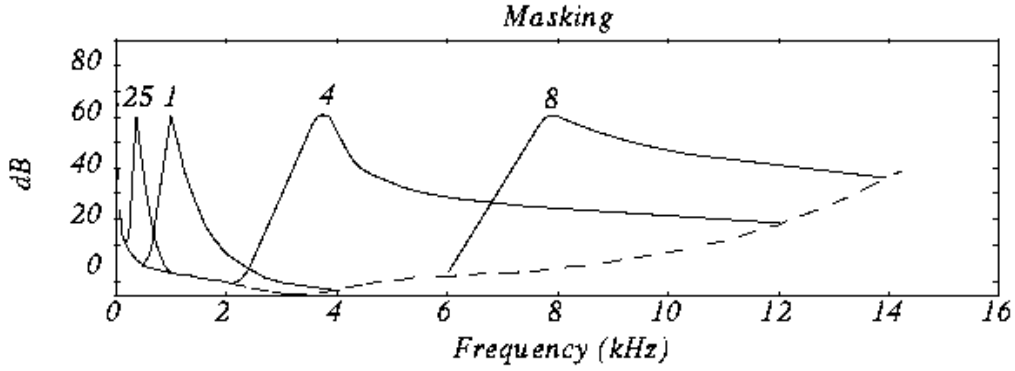


Figura 3.4 : enmascaramiento de diversos tonos de prueba

Bitrate en diferentes Sistemas

sound quality	bandwidth	mode	bitrate	reduction ratio
telephone sound	2.5 kHz	mono	8 kbps *	96:1
better than short-wave	4.5 kHz	mono	16 kbps	48:1
better than AM radio	7.5 kHz	mono	32 kbps	24:1
similar to FM radio	11 kHz	stereo	56...64 kbps	26...24:1
near-CD	15 kHz	stereo	96 kbps	16:1
Sound like CD	>15 kHz	stereo	112..128kbps	14..12:1

4. Transformaciones Lineales

El análisis de Fourier permite representar cualquier forma de onda mediante un conjunto de componentes armónicamente relacionados de amplitud y fase adecuadas. La transformada de una forma de onda de audio típica varía de manera relativamente lenta. La lenta señal sonora procedente del tubo de un órgano o de la cuerda de un violín, o el lento decrecimiento de la mayoría de los sonidos musicales, permite la reducción de la frecuencia a la que la transformada es muestreada, obteniéndose una ganancia de codificación. Es posible obtener una ganancia de codificación adicional si las componentes que experimentarán el enmascaramiento se cuantifican de manera más rudimentaria.

Las transformadas prácticas requieren bloques de muestras en lugar de cadenas interminables. La solución está en cortar la forma de onda en cortos segmentos solapados y, seguidamente, transformar cada uno de ellos individualmente, de este modo, cada muestra de entrada aparece en sólo dos transformadas, pero con una ponderación variable dependiendo de su posición en el eje temporal.

4.1 Transformada Discreta de Fourier

La DFT (Discrete Frequency Transform o Transformada de Frecuencia Discreta) requiere gran número de cálculos, debido al requisito de tener que utilizar una aritmética compleja para obtener la fase de las componentes, así como la amplitud. Una alternativa consiste en emplear la Transformada Discreta del Coseno (DCT). Esta presenta una ventaja cuando se utiliza con ventanas solapadas. En la Transformada Discreta del Coseno Modificada (MDCT), se usan ventanas con un solapamiento del 50%. De este modo, se obtiene el doble de coeficientes necesarios, que se submuestrean por un factor de dos para obtener una transformada muestreada críticamente, lo cual tiene como resultado un efecto un aliasing potencial en el dominio de la frecuencia. Sin embrago, variando levemente la transformada, los productos de aliasing en la segunda mitad de una determinada ventana son iguales en tamaño, pero de polaridad opuesta a los productos de aliasing de la primera mitad de la siguiente ventana, por lo que se eliminarán en su reconstrucción. Éste es el principio de la eliminación del aliasing en el dominio temporal (TDAC, Time Domain Aliasing Cancellation).

La recuantificación realizada en el codificador eleva el ruido de cuantificación en el bin de la frecuencia, pero lo hace durante todo el tiempo que dura el bloque. Si se produce un transitorio hacia el extremo final de un bloque, el decodificador reproduce la forma de onda correctamente, pero el ruido de cuantificación comenzará al principio del bloque y puede dar lugar a un pre-eco en el que el ruido se oye antes que el transitorio.

La solución es utilizar una ventana de tiempo variable de acuerdo con el contenido del transitorio de la forma de onda de audio. Cuando se producen transitorios musicales, se necesitan bloques cortos, por lo que la resolución de la frecuencia y, por tanto, la ganancia de codificación serán bajas. En otras ocasiones, los bloques pueden hacerse más grandes, mejorando así la resolución de la frecuencia de la transformada y obteniéndose una mayor ganancia de codificación.

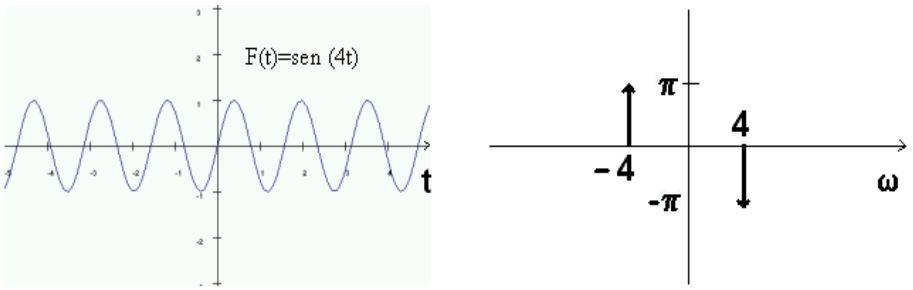
El estándar MPEG-3 (MP3) utiliza la transformación de tiempo-frecuencia y añade un nuevo banco de filtros, el DCT (Discrete Cosine Transform), que con el polifase forman el denominado filtro híbrido. Proporciona una resolución en frecuencia variable, 6x32 o 18x32 subbandas, ajustándose mucho mejor a las bandas críticas de las diferentes frecuencias.

La señal de audio es pasada primero por un banco de filtros, los cuales transforman la señal en bandas de frecuencia, donde es críticamente muestreada, lo cual significa que hay muchas muestras en el dominio del tiempo y de la frecuencia. Entre tanto un modelo acústico analiza las muestras de audio para encontrar la mejor colocación de los niveles de cuantización para usar en la codificación en cada banda de frecuencia. Las diferentes bandas de frecuencia son cuantificadas y definidas en términos de bits, se hace uso del análisis acústico para hacer una cuantización de las bandas de energía y una secuencia de bits se forma de acuerdo a la capa de codificación.

El banco de filtros opera sobre 64 muestras a la vez, con 50% de superposición, creando 32 bandas de frecuencias. Las bandas de frecuencia son congregadas en bloques de cuantización de una manera fija y un modelo acústico simple asigna bits al cuantizador para que él pueda estructurar y cuantizar las bandas de energía, asigna resolución de frecuencias dobles gracias al banco de filtros híbridos con 576 bandas, a la cuantización no uniforme y a la codificación Huffman.

4.2 Los wavelets

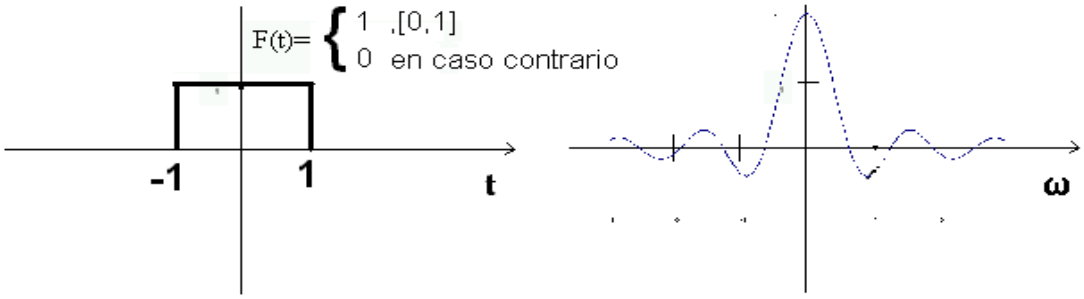
Los wavelets -onditas, ondelettes- son funciones con buena localización en tiempo y frecuencia. Considérese por ejemplo la función sen(4t) que tiene una excelente localización en el dominio de la frecuencia pero en el dominio del tiempo se extiende a lo largo de todo el eje horizontal. Véase la figura 4.1



$$F(\omega)= \int_{-\infty}^{+\infty} sen(4t)\, e^{-j\omega t}\, dt$$

$$F(\omega)= \pi\, [\, \delta(\omega+4) - \delta(\omega-4)\,]$$

Figura 4.1 La Transformada de Fourier de la función F(t) = sen (4t)



$$F(\omega)= \int_{-\infty}^{+\infty} F(t)\ e^{-j\omega t}\ dt$$

Figura 4.2 La Transformada de Fourier de la función escalón

El caso contrario se encuentra en la función escalón en el intervalo [-1,1] que tiene una excelente localización en tiempo pero no en frecuencia. Véase la figura 4.2

4.3 Transformada Haar

Partamos de un banco de filtros de análisis *H* que transforma una señal de entrada x en una salida y=*H*(x). Los filtros separan las altas y bajas frecuencias de una señal discreta

$x \in \mathbf{R}^n$

que consiste de n muestras.

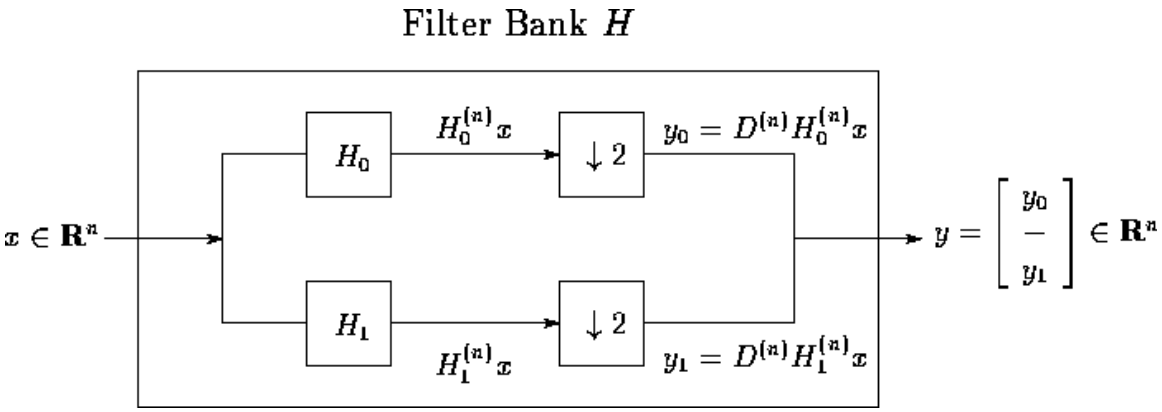


Figura 4.3 Banco de filtros de análisis

Si el banco de filtros es lineal, la transformación correspondiente se puede representar como un producto matricial y=Hx.

Para el caso de la Transformada Haar el filtro pasabajas esta determinado por

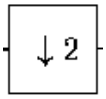
$$H_0^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & 1 & & \\ & & & \ddots & & \\ & & & & 1 & 1 \\ & & & & & 1 \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

y simplemente promedia las muestras adyacentes de la señal de entrada x.

El filtro pasaaltas esta determinado por

$$H_1^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & 1 & -1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & -1 \\ & & & & & & 1 \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

y obtiene las diferencias de las muestras adyacentes de la señal de entrada x.

El bloque con el símbolo  simboliza la operación de downsampling -despreciar una de cada dos muestras- está representada por la matriz

$$D^{(n)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2}\times n}$$

Las operaciones de filtrado en frecuencias bajas y downsampling se pueden expresar mediante

$$L^{(n)} = D^{(n)}H_0^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & & & \cdots & & \\ & & & & & 1 & 1 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2}\times n}.$$

Similarmente el filtrado en frecuencias altas y downsampling se pueden expresar mediante

$$B^{(n)} = D^{(n)} H_1^{(n)} = \frac{1}{2} \begin{bmatrix} 1 & -1 & & & & & & \\ & & 1 & -1 & & & & \\ & & & & \cdots & & & \\ & & & & & & 1 & -1 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2} \times n}.$$

4.4 Wavelets y sus aplicaciones

Los wavelets han sido aplicados en una amplia variedad de áreas. Conjuntos de datos sin componentes periódicos no pueden ser bien procesados usando técnicas de Fourier.

Los wavelets permiten construir filtros complejos para este tipo de datos. Hay un creciente campo de la literatura de técnicas de wavelets para la reducción de ruido.

4.5 Reconstrucción perfecta en un mundo finito

Una de las características de los wavelets en el procesamiento de señales y en compresión es que en la literatura es referido como una perfecta reconstrucción.

Un algoritmo wavelet tiene una perfecta reconstrucción cuando la transformada inversa del resultado de la transformada wavelet da el conjunto original de datos.

IWT(**WT**(*D*)) = *D*

(Aquí **IWT** es la transformada wavelet inversa y **WT** es la transformada wavelet).

4.6 Wavelets vistos como compresión

La versión mas simple de la transformada wavelet expresada en el esquema de elevación es mostrada abajo en la figura 5.4. El paso predictivo calcula la función wavelet en la transformada wavelet.

Este es un filtro pasa altas. El paso actualizado calcula la función de escalamiento, la cual resulta en una versión mas suave de los datos.

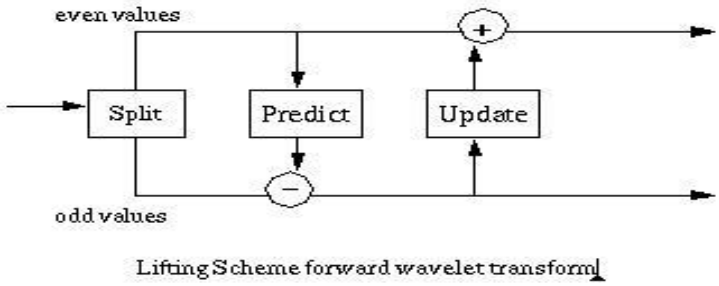


Figura 4.4 Esquema de Elevación de Transformada Wavelet

El Esquema de Elevación de la transformada, consiste en dos pasos:

1. Paso de división: divide los datos de entrada en elementos pares e impares. En un conjunto de datos finitos los elementos impares son movidos a la segunda mitad del arreglo, dejando los elementos pares en la primera mitad.
2. Paso predictivo: predice los elementos impares de los elementos pares.

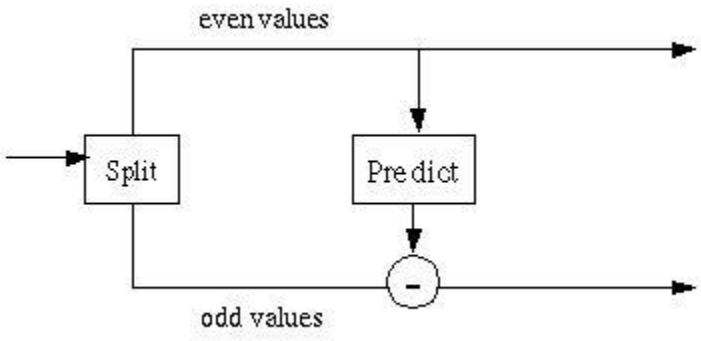


Figura 4.5 Esquema de Elevación de 2 Pasos

Una forma de ver el paso predictivo es a través de la compresión de datos. Si nuestro objetivo es comprimir un conjunto de datos y los elementos impares pueden ser completamente predecidos de los elementos pares usando la ecuación

impares = pares * 2;

los elementos impares pueden ser reemplazados por cero. Si se aplica un algoritmo de compresión como run length encoding los elementos impares serán reducidos a cero, comprimiendo el conjunto de datos original por lo menos 50%. Si el conjunto de datos consiste de puntos en una línea, entones este puede ser reducido algo cerca de un solo elemento y de la longitud del conjunto de datos. En la mayoría de los casos el conjunto de datos es mas complejo y no puede ser completamente representado por la condición inicial, longitud y ecuación. Sin embargo, una representación mas compacta puede ser llevada acabo aproximando los datos en una región local usando una función. La etapa de predicción remplaza el elemento impar con la diferencia entre el elemento impar una función calculada de los elementos pares. El ejemplo mas sencillo de cómo la etapa de predicción toma un solo elemento impar como su argumento para calcular el valor predictivo del elemento impar:

$impar_{j+1,i} = impar_{j,i} - \mathbf{P}(par_{j,k})$

Aquí la función **p**( ) es la función predictiva. Los algoritmos wavelets son recursivos, así que el paso recursivo *j* genera datos para el siguiente paso recursivo *j+1*. El subíndice i pone en un índice la parte impar del arreglo. El subíndice k pone en un índice la parte par del arreglo.

4.7 La Transformada Wavelet Daubechies D4

La transformada wavelet Daubechies fue inventada por la matemática Ingrid Daubechies. La transformada Daubechies D4 tiene cuatro funciones wavelet de escalamiento.
Los coeficientes de función de escalamiento son:

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$$

$$h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$$

$$h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$$

$$h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

Cada paso de la transformada wavelet aplica la función de escalamiento al dato de entrada. Si el conjunto de datos original tiene N valores, la función de escalamiento será aplicada en el paso de la transformada wavelet para calcular N/2 valores suaves. En el orden de la transformada wavelet los valores mas suaves son almacenados en la parte baja de los N elementos del vector de entrada.

Los valores de los coeficientes de la función wavelet son:

$g_0 = h_3$
$g_1 = -h_2$
$g_2 = h_1$
$g_3 = -h_0$

Cada paso de la transformada wavelet aplica la función de escalamiento al dato de entrada. Si el conjunto de datos original tiene N valores, la función wavelet será aplicada para calcular N/2 diferencias. En el orden de la transformada wavelet los valores wavelet son almacenados en la parte alta de los N elementos del vector de entrada.

Las funciones de escalamiento y wavelet son calculadas tomando el producto interno de los coeficientes y cuatro valores de datos. Las ecuaciones se muestran a continuación:

Daubechies D4 scaling function:

$$a_i = h_0 s_{2i} + h_1 s_{2i+1} + h_2 s_{2i+2} + h_3 s_{2i+3}$$

$$a[i] = h_0 s[2i] + h_1 s[2i+1] + h_2 s[2i+2] + h_3 s[2i+3];$$

Daubechies D4 wavelet function:

$$c_i = g_0 s_{2i} + g_1 s_{2i+1} + g_2 s_{2i+2} + g_3 s_{2i+3}$$

$$c[i] = g_0 s[2i] + g_1 s[2i+1] + g_2 s[2i+2] + g_3 s[2i+3];$$

Figura 4.6 Función de Escalamiento y Función Wavelet

Cada iteración en el paso de la transformada wavelet calcula un valor de la funcion de escalamiento y un valor de la función wavelet. El índice *i* es incrementado por dos con cada iteración, y nuevos valores de función de escalamiento y función wavelet son calculados.

En el caso de la transformada hacia adelante, con un conjunto de datos finitos, *i* será incrementado hasta que este igual a N-2. En la última iteración el producto interno será calculado del calculo de s[N-2], s[N-1], s[N] y s[N+1]. Desde s[N] y s[N+1] no existen (estos estan mas alla del final del arreglo), esto presenta un problema. Esto se muestra en la matriz de transformación abajo.

$$\begin{bmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 & & \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 & & \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 & & \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 & & \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 & & \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 & & \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \end{bmatrix} \bullet \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Figura 4.7 Daubechies D4 matriz de transformación para 8 elementos

Note que este problema no existe para el wavelet Haar, desde que este es calculado en solo dos elementos, s[i] y s[i+1].

Un problema similar existe en el caso de la transformada inversa. Aquí los coeficientes de la transformada inversa se extienden mas alla del principio de los datos, donde los dos primeros valores inversos son calculados de s[-2], s[-1], s[0] y s[1]. Esto se muestra en la matriz de transformación inversa abajo.

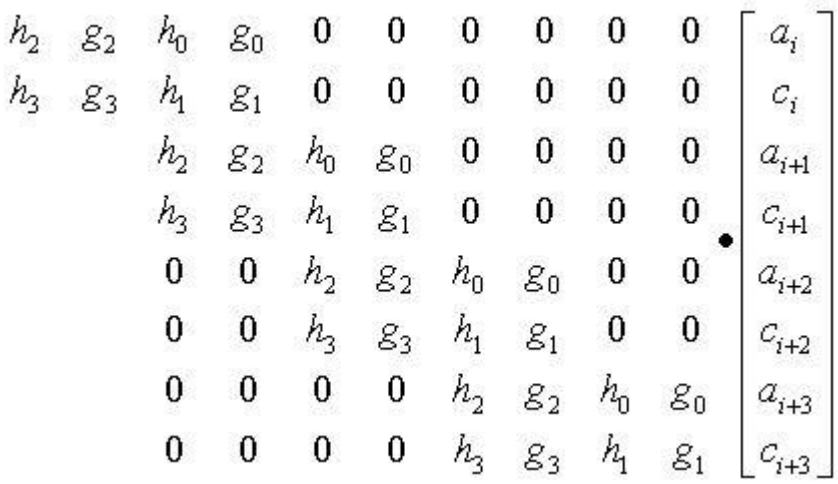


Figura 4.8 Daubechies D4 matriz de transformación inversa para 8 elementos del resultado de transformación

Tres métodos para manejar este problema:

1. Tratar el conjunto de datos como si este fuera periódico. El comienzo de la secuencia de datos se repite siguiendo el final de la secuencia (en el caso de la transformada hacia delante) y el final de los datos alrededor del principio (en el caso de la transformada inversa).
2. Tratar el conjunto de datos como si este fuera un espejo al final. Esto significa que los datos son reflejados de cada final, como si un espejo fuera soportado para cada final de secuencia de datos.
3. Ortogonalización Gram-Schmidt. La ortigonalización Gram-Schmidt calcula escalas especiales y funciones wavelet que son aplicadas en el principio y fin del conjunto de datos.

Los ceros pueden ser también ser usados para llenar en cuanto hagan falta elementos, pero esto puede introducir un error significativo.

El algoritmo Daubechies D4 presentado en este trabajo trata a los datos como si estos fueran periódicos. El código para un paso de la transformada es mostrado abajo. Note que en el cálculo de los dos últimos valores, el inicio de los datos alrededor del final y de los elementos a[0] y a[1] son usados en el producto interno.

```
protected void transform( double a[], int n )
   {
      if (n >= 4) {
         int i, j;
         int half = n >> 1;

         double tmp[] = new double[n];

         i = 0;
         for (j = 0; j < n-3; j = j + 2) {
            tmp[i]      = a[j]*h0 + a[j+1]*h1 + a[j+2]*h2 + a[j+3]*h3;
            tmp[i+half] = a[j]*g0 + a[j+1]*g1 + a[j+2]*g2 + a[j+3]*g3;
            i++;
         }

         tmp[i]      = a[n-2]*h0 + a[n-1]*h1 + a[0]*h2 + a[1]*h3;
         tmp[i+half] = a[n-2]*g0 + a[n-1]*g1 + a[0]*g2 + a[1]*g3;

         for (i = 0; i < n; i++) {
            a[i] = tmp[i];
         }
      }
   } // transform
```

La transformada inversa trabaja sobre N datos, donde los primeros N/2 elementos son los valores suaves y los segundos N/2 elementos son valores de función wavelet. El producto interno que es calculado para reconstruir el valor de la señal es calculado de dos valores suaves y dos valores wavelet. Lógicamente, los datos del final son envueltos alrededor desde el fin al inicio. En los comentarios el "coef. val" se refiere a un valor de función wavelet y un valor suave se refiere a un valor de la función de escalamiento.

```
 protected void invTransform( double a[], int n )
   {
      if (n >= 4) {
        int i, j;
        int half = n >> 1;
        int halfPls1 = half + 1;

        double tmp[] = new double[n];

        tmp[0] = a[half-1]*Ih0 + a[n-1]*Ih1 + a[0]*Ih2 + a[half]*Ih3;
        tmp[1] = a[half-1]*Ig0 + a[n-1]*Ig1 + a[0]*Ig2 + a[half]*Ig3;
        j = 2;
        for (i = 0; i < half-1; i++)
{
          tmp[j++] = a[i]*Ih0 + a[i+half]*Ih1 + a[i+1]*Ih2 +

a[i+halfPls1]*Ih3;

          tmp[j++] = a[i]*Ig0 + a[i+half]*Ig1 + a[i+1]*Ig2 +

a[i+halfPls1]*Ig3;
        }
        for (i = 0; i < n; i++) {
          a[i] = tmp[i];
        }
      }
   }
```

Los algoritmos wavelet de elevación tienen muchas ventajas. Existe memoria eficiente y no requiere de arreglos temporales como en la versión de la Transformada Daubechies D4 mostrada arriba. Como el diagrama mostrado abajo, la transformada inversa es el espejo de la transformada hacia delante, cuando las sumas son cambiadas por restas.

4.7.1 Transformada hacia delante

Rizos en matemáticas describen la versión del esquema de elevación de la transformada Daubechies D4. El esquema de elevación de las transformadas wavelet son compuestos de actualizaciones y pasos predictivos. En este caso un paso de normalización ha sidoañadido también. Un paso de la transformada es mostrado en el siguiente diagrama.

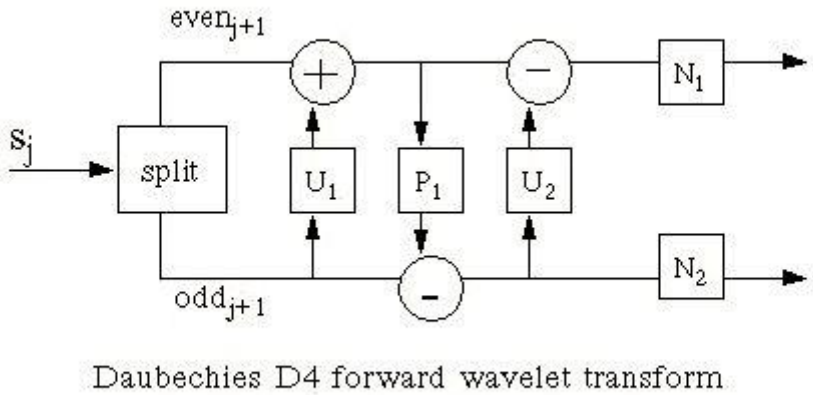


Figura 4.9 Transformada Wavelet Daubechies D4

El paso a la mitad divide los datos de entrada en elementos pares los cuales son almacenados en la primera mitad de un arreglo de N elementos (S_0 a S_{half-1}) y los elementos impares los cuales son almacenados en la segunda mitad del arreglo de N elementos (S_{half} a S_{N-1}). En las ecuaciones de la transformada abajo en la expresión S[half+n] se refiere a un elemento impar y S[n] se refiere a un elemento par. Aunque el diagrama arriba muestra dos pasos de normalización, en la práctica estos son envueltos en una única función.

4.7.2 Transformada Inversa

Una de las características elegantes del esquema de elevación de la transformada wavelet es el hecho de que la transformada inversa es un espejo de la transformada, qué adición y operaciones de la substracción intercambiaron.

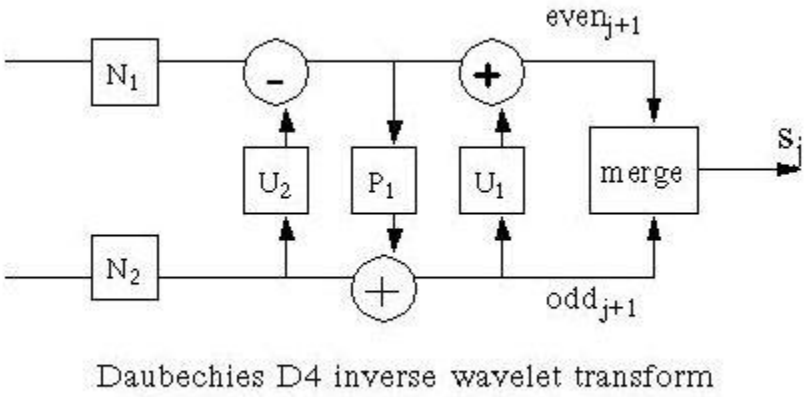


Figura 4.10 Transformada Wavelet Inversa Daubechies D4

El paso de progresión de la fusión interpola elementos de las mitades uniformes e impares del vector (e.g., even0, odd0, even1, odd1...).

Como el diagrama mostrado arriba, las ecuaciones de la transformada inversa tienen operaciones de sumas y restas intercambiadas.

$$\frac{\sqrt{3}-1}{\sqrt{2}} \cdot \frac{\sqrt{3}+1}{\sqrt{2}} = 1$$

Update 1′:

for $n = 0$ to $half$ -1

$$S[n] = S[n] - \sqrt{3}S[half + n]$$

Predict′:

$$S[half] = s[half] + \frac{\sqrt{3}}{4}S[0] + \frac{\sqrt{3}-2}{4}S[half - 1]$$

for $n = 1$ to $half$ -1

$$S[half + n] = s[half + n] + \frac{\sqrt{3}}{4}S[n] + \frac{\sqrt{3}-2}{4}S[n - 1]$$

Update 2′:

for $n = 0$ to $half$ - 2

$$S[n] = S[n] + S[half + n + 1]$$

$$S[half - 1] = S[half - 1] + S[half]$$

Normalize′:

for $n = 0$ to $half$ -1

$$S[n] = \frac{\sqrt{3}+1}{\sqrt{2}}S[n]$$

$$S[n + half] = \frac{\sqrt{3}-1}{\sqrt{2}}S[n + half]$$

Figura 4.11 Ecuaciones de paso de transformación

5. Cuantización

La cuantización es la conversión de una señal discreta en el tiempo evaluada de forma continua a una señal discreta en el tiempo discretamente evaluada. El valor de cada muestra de la señal se representa como un valor elegido de entre un conjunto finito de posibles valores.

Para incrementar el número de ceros y reducir la magnitud de los coeficientes de transformación se utiliza un cuantizador.

Existen varias técnicas de cuantización: lineales y no lineales. En la práctica conforme se incrementa la eficiencia del cuantizador también lo hace su complejidad y costo computacional.

Un cuantizador lineal está representado por:

$$Valorcuantizado = \left\lfloor \frac{ValorDelCoeficiente \pm \left\lfloor \frac{Q}{2} \right\rfloor}{Q} \right\rfloor$$

donde:

- Q (Factor Q) Es el tamaño del paso de cuantización
- $\pm$ El signo negativo es para un coeficiente menor que cero y positivo en caso contrario.

El paso inverso a la cuantización es conocido como descuantización y se representa por:

$$\text{ValorRecuperadoDelCoeficiente} = \text{Q} \bullet \text{Valorcuantizado}$$

Nótese que no es una función inversa, puesto que el valor original del coeficiente no es en la mayoría de los casos igual al valor recuperado. El factor Q controla la cuantización. Un factor Q elevado proporciona mayor compresión. Un factor Q pequeño mejora la calidad de la imagen. La diferencia entre una entrada sin cuantizar y una salida cuantizada se denomina error de cuantización o ruido.

Se conoce como error de cuantificación (o *ruido*), a la diferencia entre la señal de entrada (sin cuantificar) y la señal de salida (ya cuantificada), interesa que el ruido sea lo más bajo posible. Para conseguir esto, se pueden usar distintas técnicas de cuantificación:

5.1 Cuantificación uniforme

En los cuantificadores uniformes (o lineales) la distancia entre los niveles de reconstrucción es siempre la misma, como se observa en la siguiente figura:

No hacen ninguna suposición acerca de la naturaleza de la señal a cuantificar, de ahí que no proporcionen los mejores resultados. Sin embargo, tienen como ventaja que son los más fáciles y menos costosos de implementar.

En la siguiente figura se ve un ejemplo de cuantificación uniforme:

Figura 5.1 Cuantizador uniforme

5.2 Cuantificación logarítmica

Las señales de voz pueden tener un rango dinámico superior a los 60 dB, por lo que para conseguir una alta calidad de voz se deben usar un elevado número de niveles de reconstrucción. Sin embargo, interesa que la resolución del cuantificador sea mayor en las partes de la señal de menor amplitud que en las de mayor amplitud. Por tanto, en la cuantificación lineal se desperdician niveles de reconstrucción y, consecuentemente, ancho de banda. Esto se puede mejorar incrementando la distancia entre los niveles de reconstrucción conforme aumenta la amplitud de la señal.

Un método sencillo para conseguir esto es haciendo pasar la señal por un compresor logarítmico antes de la cuantificación. Esta señal comprimida puede ser cuantificada uniformemente. A la salida del sistema, la señal pasa por un expansor, que realiza la función inversa al compresor. A esta técnica se le llama *compresión*. Su principal ventaja es que es muy fácil de implementar y funciona razonablemente bien con señales distintas a la de la voz.

Para llevar a cabo la compresión existen dos funciones muy utilizadas: Ley-A (utilizada principalmente en Europa) y ley-μ(utilizada en EEUU).

$$c(x) = \begin{cases} \dfrac{A|x|}{1 + \log_e A}\,\mathrm{sgn}(x) & \text{si } 0 \le \dfrac{|x|}{x_{max}} \le \dfrac{1}{A} \\ x_{max}\,\dfrac{1 + \log_e (A|x| / x_{max})}{1 + \log_e A}\,\mathrm{sgn}(x) & \text{si } \dfrac{1}{A} \le \dfrac{|x|}{x_{max}} \le 1 \end{cases}$$

Figura 5.2 *Ley-A :*

$$c(x) = x_{max}\,\frac{\log_e (1 + \mu |x| / x_{max})}{\log_e (1 + \mu)}\,\mathrm{sgn}(x)$$

Figura 5.3 *Ley-μ :*

La siguiente figura muestra la gráfica de la ley-μ para distintos valores de μ:

Figura 5.4 Ley -μ para distintos valores de μ

5.3 Cuantificación no uniforme

El problema de la cuantificación uniforme es que conforme aumenta la amplitud de la señal, también aumenta el error. Este problema lo resuelve el cuantificador logarítmico de forma parcial. Sin embargo, si conocemos la función de la distribución de probabilidad, podemos ajustar los niveles de reconstrucción a la distribución de forma que se minimice el error cuadrático medio. Esto significa que la mayoría de los niveles de reconstrucción se den en la vecindad de las entradas más frecuentes y, consecuentemente, se minimice el error (ruido).

La siguiente figura representa la cuantificación no uniforme:

Figura 5.5 Cuantización no Uniforme

En la práctica, se puede usar una estimación de la distribución para diseñar los cuantificadores. Esta estimación se puede obtener a partir de los datos a cuantificar de forma iterativa.

5.4 Cuantificación vectorial

En los métodos anteriores, cada muestra se cuantificaba independientemente a las muestras vecinas. Sin embargo, la teoría demuestra que ésta no es la mejor forma de cuantificar los datos de entrada. Resulta más eficiente cuantificar los datos en bloques de N muestras. El proceso es sencillamente una extensión de los anteriores métodos escalares descritos anteriormente. En este tipo de cuantificación, el bloque de N muestras se trata como un vector N-dimensional.

En la siguiente figura vemos un ejemplo de cuantificación vectorial (VQ) en dos dimensiones:

Figura 5.6 Cuantización Vectorial

El plano XY está dividido en seis regiones distintas. El vector de entrada (con dos componentes) se reemplaza por el centroide i (representa todos los vectores de una determinada región i) de la región a la que pertenece.

La cuantificación vectorial ofrece mejores resultados que la cuantificación escalar, sin embargo, es más sensible a los errores de transmisión y lleva consigo una mayor complejidad computacional.

5.5 Cuantización en MPEG

El modelo psicoacústico: calcula el nivel a partir del cual el ruido comienza a ser perceptible, para cada banda. Este nivel se utiliza en el bloque de asignación de bit/ruido para determinar la cuantización y sus niveles. De nuevo, se utilizan dos diferentes. En ambos, los datos de salida forman el SMR (signal-to-mask ratio) para cada banda o grupo de bandas.

Asignación de bit/ruido: examina tanto las muestras de salida del banco de filtros como el SMR proporcionado por el modelo psicoacústico, y ajusta la asignación de bit o ruido, según el esquema utilizado, para satisfacer simultáneamente los requisitos de tasa de bits y de enmascaramiento.

6. Compresión de Datos

En este capítulo se presentan cuatro técnicas de codificación entrópica, *Run Length Encoding* basada en datos repetitivos, la *codificación de Huffman*, la *codificación Aritmética*, basada en un modelo probabilístico y finalmente la técnica *de Lempel Ziv Welch* basada en un esquema de diccionario.

En la figura 9.1 se muestra un sistema típico de compresión de datos que muestra claramente en que parte se realiza la codificación y decodificación entrópica.

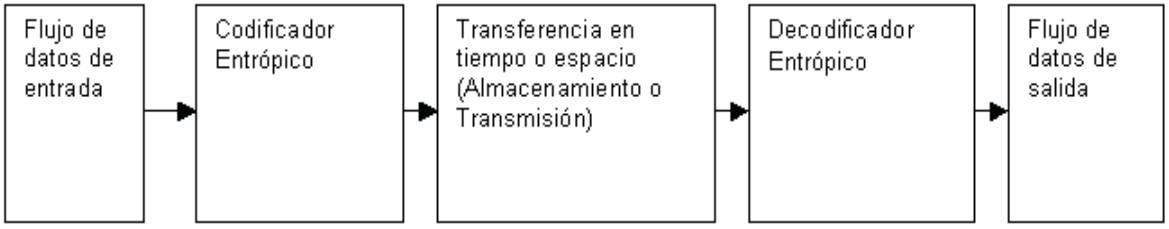


FIGURA 6.1 UN SISTEMA TÍPICO DE COMPRESIÓN DE DATOS

6.1 Run Length Encoding

Run Length Encoding es una de las técnicas más simples de compresión de datos que toma ventaja de los datos repetitivos. Algunas imágenes tienen áreas de color constante, a estos caracteres repetidos se les denominan corridas o "runs". La técnica de codificación es simple, cada corrida se representa con una cuenta de los octetos de datos originales[Nelson, 1995]. Por ejemplo la cadena origen dada por:

AAAABBBBBCCCCCCCCDEEEE

Puede representarse por:

4A5B8C1D4E.

6.2 Codificación Huffman

Generalidades:

Se trata de un algoritmo que puede ser usado para compresión o encriptación de datos. Este algoritmo se basa en asignar códigos de distinta longitud de bits a cada uno de los caracteres de un fichero. Si se asignan códigos más cortos a los caracteres que aparecen más a menudo se consigue una compresión del fichero.

Esta compresión es mayor cuando la variedad de caracteres diferentes que aparecen es menor. Por ejemplo: si el texto se compone únicamente de números o mayúsculas, se conseguirá una compresión mayor.

Para recuperar el fichero original es necesario conocer el código asignado a cada carácter, así como su longitud en bits, si ésta información se omite, y el receptor del fichero la conoce, podrá recuperar la información original. De este modo es posible utilizar el algoritmo para encriptar ficheros.

Mecanismo del algoritmo:

- Contar cuantas veces aparece cada carácter en el fichero a comprimir. Y crear una lista enlazada con la información de caracteres y frecuencias.
- Ordenar la lista de menor a mayor en función de la frecuencia.
- Convertir cada elemento de la lista en un árbol.
- Fusionar todos estos árboles en uno único, para hacerlo se sigue el siguiente proceso, mientras la lista de árboles contenga más de un elemento:
    - Con los dos primeros árboles formar un nuevo árbol, cada uno de los árboles originales en una rama.
    - Sumar las frecuencias de cada rama en el nuevo elemento árbol.
    - Insertar el nuevo árbol en el lugar adecuado de la lista según la suma de frecuencias obtenida.
- Para asignar el nuevo código binario de cada carácter sólo hay que seguir el camino adecuado a través del árbol. Si se toma una rama cero, se añade un cero al código, si se toma una rama uno, se añade un uno.
- Se remodifica el fichero según los nuevos códigos.

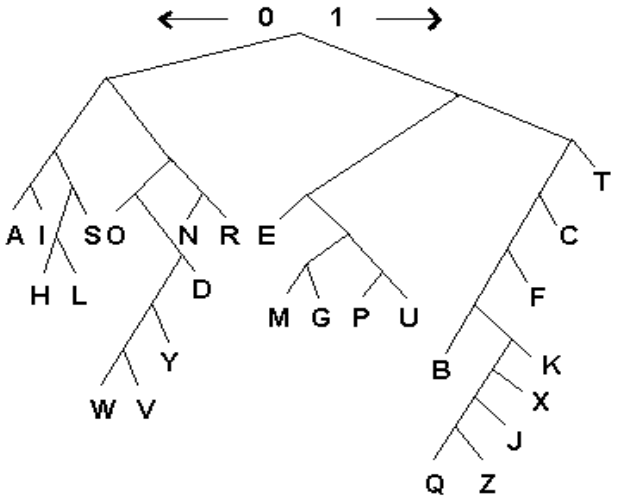


FIGURA 6.2 Árbol binario del alfabeto

El MPEG – III también utiliza la clásica técnica del algoritmo de Huffman. Funciona al final de la compresión para codificar información, por lo tanto, esto no es un algoritmo de compresión en si sino, más bien, un método de codificación. Esta codificación crea códigos de longitud variable en gran cantidad de "bits". Símbolos de alta probabilidad tienen códigos más cortos.
Los códigos de Huffman tienen la propiedad de tener un prefijo único, y por lo tanto, pueden ser decodificados correctamente a pesar de su longitud variable. La decodificación es muy rápida (vía una tabla de correspondencia). Este tipo de codificación permite ahorrar, en promedio, un 20% del espacio.

Es un complemento perfecto a la codificación perceptual: Durante las polifonías grandes, la codificación perceptual es muy eficiente porque muchos sonidos están enmascarados o mermados, pero siendo escasa la información idéntica, el algoritmo de Huffmann es poco eficiente. Durante los sonidos "puros" hay algunos efectos de enmascaramiento, pero Huffman es entonces muy eficiente porque el sonido digitalizado contiene muchos bytes redundantes ó innecesarios, que serán reemplazados por códigos más cortos.

6.3 Codificación Aritmética

La técnica de codificación aritmética ha emergido en la última década como una herramienta importante de compresión usada ampliamente para la codificación adaptativa en alfabetos multi-símbolos, gracias a su velocidad, bajos requerimientos de almacenamiento y eficiencia en la compresión.

Ideada por Rissanen y Langdon en la década del 70 [Rissanen, 1979], esta técnica fue tomada por varios años más como una curiosidad que como una técnica aplicable a la codificación y compresión de datos. Comparada con la técnica de Huffman, resultaba menos eficiente para el tratamiento de alfabetos compuestos por muchos símbolos. En el año 1987 Witten propone una implementación del codificador aritmético multi-símbolo que hace que la técnica emerja nuevamente como una solución a los problemas de compresión.

La codificación aritmética toma la idea de reemplazar un símbolo de entrada con un código específico. Reemplaza un flujo de símbolos de entrada en un solo número de punto flotante. Se requieren más bits conforme el mensaje es más complejo.

La salida de un proceso de codificación aritmética es un solo número entre 0 y 1. Este número se utiliza en la decodificación para recuperar de manera única el flujo original de símbolos de entrada.

La codificación aritmética, al igual que Huffman, es un algoritmo de dos pasos. El primero calcula la frecuencia de caracteres y genera una tabla de probabilidad. El segundo paso hace la compresión.

La tabla de probabilidad asigna un rango entre 0 y 1 a cada carácter de entrada. El tamaño de cada rango es directamente proporcional a la frecuencia de

caracteres. El orden de asignación de esos rangos no es importante pero debe ser utilizado en el codificador y el decodificador.

El rango consiste de un valor alto y un valor bajo. Estos parámetros son muy importantes para el proceso de codificación/decodificación. A los caracteres de mayor ocurrencia se les asigna un rango mayor en el intervalo requiriendo. A los de menor ocurrencia se les asigna rangos menores, requiriendo más bits.

En la codificación aritmética, se comienza con un rango de 0.0 a 1.0 El primer carácter de entrada generará el numero de salida con su rango correspondiente. El rango del siguiente carácter fija el siguiente numero de salida. Entre más sean los caracteres de entrada más preciso será el número de salida.

La codificación aritmética requiere un elevado número de operaciones aritméticas de punto flotante por lo que es significativamente más lenta que la codificación de Huffman tanto en la codificación como en la decodificación. Conforme los CPU están siendo cada vez mas rápidos, la codificación aritmética parece ser un sucesor de la codificación de Huffman.

6.4 Codificación Lempel Ziv Welch (LZW)

En 1977, Abraham Lempel y Jacob Ziv propusieron un algoritmo de compresión de datos denominado LZ77.

Mientras que los Códigos de Huffman proporcionaron buenos resultados, estos se limitaron a codificar un carácter a la vez. Lempel y Ziv propusieron un esquema de diccionario (adaptativo) para codificar cadenas de datos. Este algoritmo tomó ventaja de la secuencia de caracteres que ocurren frecuentemente como la palabra “the” del idioma inglés.

En 1984 Terry Welch publicó para la *IEEE* el algoritmo LZW (Lempel Ziv Welch), una variante del algoritmo LZ77, que llegó a ser el algoritmo de compresión de datos para las computadoras personales.

En la codificación LZW se busca reemplazar cadenas de caracteres con códigos simples que son almacenados en un diccionario o tabla de cadenas. El algoritmo no analiza para nada el texto de entrada; únicamente añade cada nueva cadena a una tabla de cadenas de caracteres.

El código que genera el algoritmo puede ser de cualquier longitud arbitraria, pero debe tener más bits que un carácter simple. Los primeros 256 códigos (cuando se usan caracteres de 8 bits) se asignan por omisión al conjunto estándar de caracteres. Los restantes son asignados a cadenas a medida que el algoritmo realiza su trabajo.

La porción de pseudo código siguiente ilustra el algoritmo en su forma más simple.

```
STRING = get(caracter de entrada)
    WHILE "existan caracteres de entrada" DO
        CHARACTER = get(caracter de entrada)
        IF STRING+CHARACTER "está en la tabla" then
            STRING = STRING+character
        ELSE
            "Enviar como salida el código de STRING"
            add STRING+CHARACTER " a la tabla
            STRING = CHARACTER
        END of IF
    END of WHILE
    "Enviar como salida el código de STRING"
```

En la descompresión LZW se crea la misma tabla de cadenas y se actualiza por cada carácter en el flujo de entrada excepto para el primero. Después de que el carácter ha sido expandido a su cadena correspondiente vía la tabla de cadenas, el carácter final de la cadena es agregado a la cadena previa. Esta nueva cadena se agrega a la tabla en la misma localidad que en la tabla de cadenas del compresor.

El principal obstáculo en el uso de LZW es la patente sostenida por *Unisys Corporation* por la que todo desarrollador que lea o escriba archivos con este algoritmo debe adquirir la licencia correspondiente.

En este capítulo se han presentado cuatro técnicas de compresión de datos:

1. Run Length Encoding, es una técnica elemental de compresión de datos que proporciona buenos resultados cuando existen secuencias de caracteres repetidos.

2. La codificación de Huffman que fue hasta 1977 el estado de arte en compresión de datos, es relativamente fácil de implementar y de poco costo computacional en la codificación y decodificación , sin embargo tiene la desventaja de que utiliza números enteros de bits por código, que no lo hace óptimo como es el caso de la codificación aritmética.

3. La codificación aritmética tiene la principal desventaja de ser un algoritmo de costo computacional elevado tanto en la codificación como en la decodificación.

4. Un esquema basado en diccionario como LZW usa un concepto diferente a la codificación de Huffman que asigna un código por cada carácter.

La implementación de LZW busca reemplazar cadenas de caracteres con códigos simples que son almacenados en un diccionario o tabla de cadenas, motivo por el cual no es un algoritmo de costo computacional elevado.

6.5 Formateador

El formateador de trama: la definición de trama para este esquema según ISO varía respecto de la de los niveles MPEG I y II: "mínima parte del bitstream decodificable mediante el uso de información principal adquirida previamente". Las tramas contienen información de 1152 muestras y empiezan con la misma cabecera de sincronización y diferenciación, pero la información perteneciente a una misma trama no se encuentra generalmente entre dos cabeceras. La longitud de la trama puede variarse en caso de necesidad. Además de tratar con esta información, el esquema III incluye codificación Huffman de longitud variable, un método de codificación entrópico que sin pérdida de información elimina redundancia. Los métodos de longitud variable se caracterizan, en general, por asignar palabras cortas a los eventos más frecuentes, dejando las largas para los más infrecuentes.

El audio codificado en MPEG 1 se configura en archivos (*.mp1, *.mp2, *.mp3) con una o más tramas. Cada trama tiene un campo de encabezamiento y uno de datos. La longitud de la trama depende de la capa y puede variar entre tramas.

El encabezamiento de trama consiste en cuatro bytes, los cuales están al comienzo de esta, así:

Byte	**0**	**1**	**2**	**3**
Binario	11111111	1111abbc	ddddeefg	hhjjkmpp

Bit	**Concepto**
1	Sincronización de encabezamiento
a	Versión (MPEG 1, MPEG 2)
b	Capa (1, 2, 3)
c	Protección de error (Si, No)>
d>	Tasa de bits (32 - 384 Kbits)
e	Frecuencia (16, 22, 24, 38, 44.1, 48Khz)
f	Bits de relleno
g	Extensión (Ninguno, Privado)
h	Modo de canal (stereo, joint stereo, dual y mono)>
j	Extensión de modo (capa 1-2, capa 3)
k	Derechos de copia (Si, No)
m>	Original (Si, No)
P	Enfasis (Ninguno, 50/15 ms, CITT j.17)

Figura 6.3 El encabezamiento de trama

6.6 Decodificación (Descompresor)

Consecuentemente el primer paso en el proceso de decodificación es encontrar el comienzo de una trama, buscando los bits de sincronización. Luego de esto el encabezamiento y la información pueden ser leídos. Los factores de escala son los primeros datos transmitidos después del encabezamiento, ellos controlan la ganancia para cada banda de frecuencia, detrás de esto viene la energía de las frecuencias que ha sido cuantizadas y codificadas en algún codificador entrópico.

6.6.1 Cuantizador Inverso

Los factores de escala son los primeros datos transmitidos después del encabezamiento, ellos controlan la ganancia para cada banda de frecuencia, detrás de esto viene la energía de las frecuencias que ha sido cuantizadas, debemos volver a cuantizar dependiendo de la técnica de cuantización utiliza para el formato.

6.6.2 Transformación Lineal Inversa

Transformar así las energías en el dominio del tiempo. Cuando se tienen los valores decodificados se ubican usando un factor dentro del valor real del espectro de energía.

Si la secuencia es codificada en señal estéreo, cada canal puede ser transmitido separadamente en cada trama. Pero a menudo el codificador busca la utilización de redundancias entre los dos canales para transmitir su suma y diferencia. Si este es el caso, el decodificador tiene que utilizar un procesamiento estéreo para recobrar la señal original de los dos canales. Después de recobrar el estéreo, el valor de cada banda de frecuencia es simétricamente sumado para disminuir las distorsiones que aparecen debido a la cuantización. Hasta ahora todas las señales tienen que estar en el dominio de la frecuencia, y para sintetizar la salida, una transformada es aplicada en forma inversa a la del tiempo-frecuencia usada en el codificador. En la capa 3 dos transformaciones son hechas para conseguir una resolución de frecuencias mejor que en las otras capas. Para evitar discontinuidades entre bloques transformados, que producirían ruidos muy perceptibles, las transformadas usan un 50% de superposición, el decodificador va hacia atrás en cada re-transformación de bloque y se sobrepone en la mitad del anterior.

Y finalmente, antes de la salida las muestras son llevadas a un filtro pasa-bajos para reconstruir la señal original, se forma el archivo WAV con la cabecera correspondiente a ese archivo y ya se obtiene el archivo WAV.

7. MPX

7.1 Archivo WAV

El formato de archivo WAV es un subconjunto de la especificación de RIFF de Microsoft para el almacenaje de los archivos de multimedia. Un archivo RIFF comienza hacia fuera con una cabecera del fichero seguida por una secuencia de los datos.

Estos datos se encuentran en un formato de dos canales izquierdo y derecho 16 bit por muestras estereo a una frecuencia de 44100 hz con 176400 Bytes por segundo, las muestras se encuentran en formato PCM, todo esto para obtener un archivo MPX ya que MPX solo trabaja con archivos de calidad CD.

Una vez obtenidos los datos del CD (track), estos son tomados del archivo WAV, para lo cual se lee la cabecera de dicho formato; véase la figura 7.1.

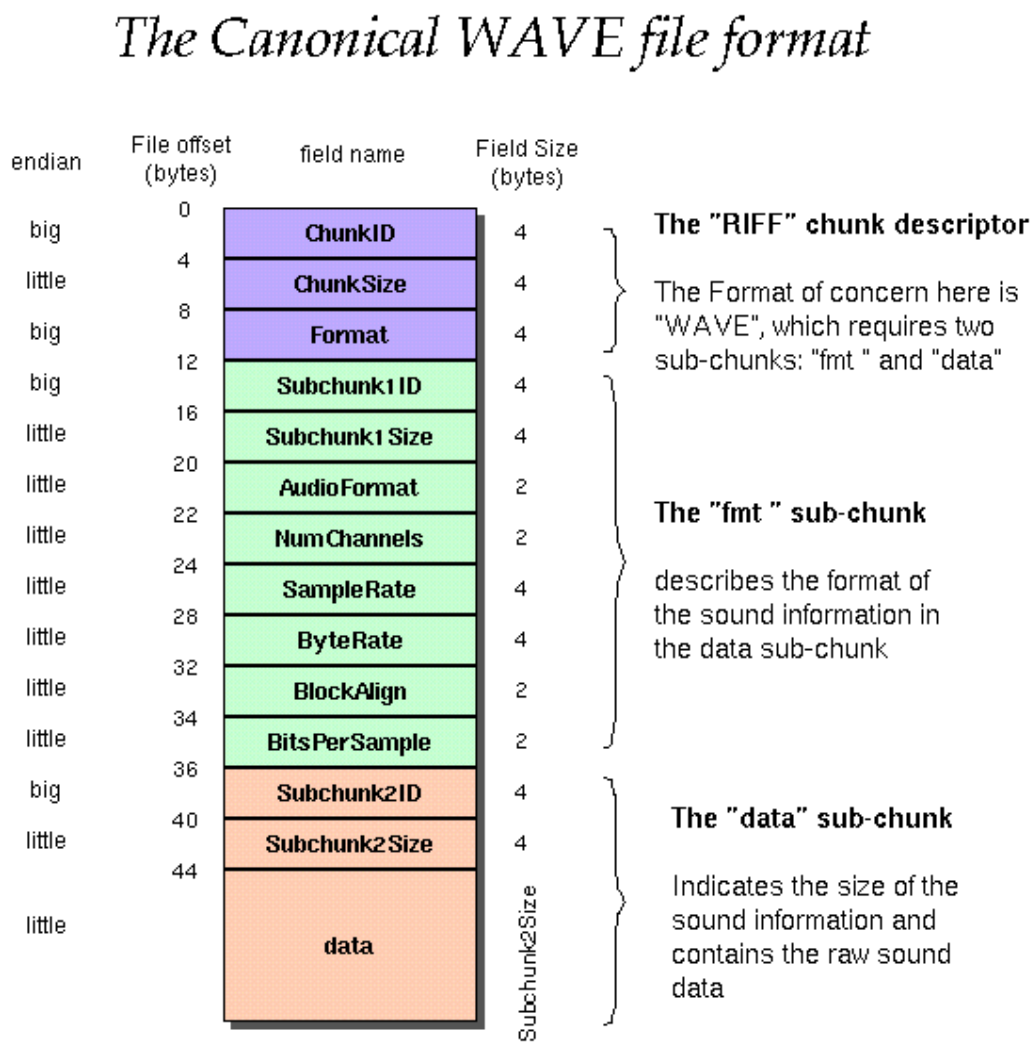


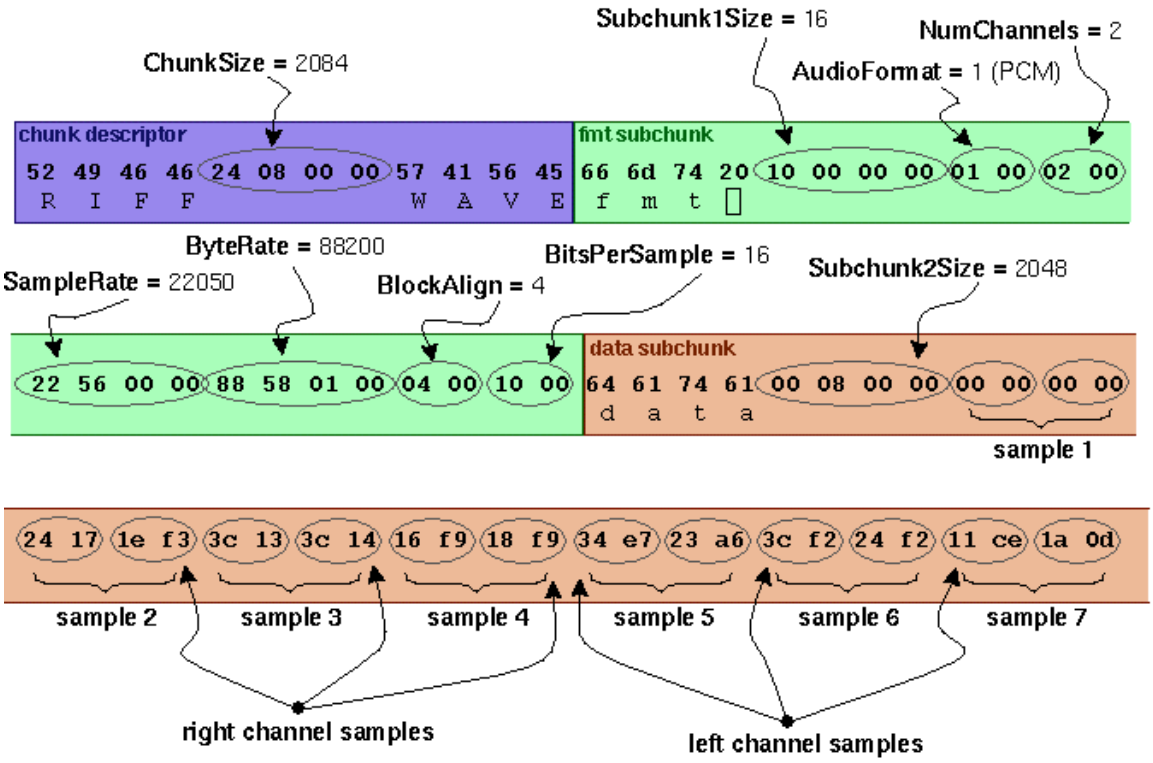
Figura 7.1 Cabecera del archivo WAV

7.2 Separación de Canales

Al realizar las verificaciones correspondientes del archivo, tales como verificar si es realmente un archivo WAV si tiene dos canales (Estereo o Mono) si es de 16 bit por muestras, etc, y se procede a obtener los datos.

Por que separar los canales?

Como es sabido el tener dos canales (Estereo) es tener dos señales independientes una señal izquierda y una señal derecha esto nos lleva a transfórmalas por separado para obtener una mayor razón de compresión al aplicarle los algoritmos de compresión.



Una vez obtenido el tamaño del archivo comienza la lectura de los datos, como se ve en la figura 7.2 cuatro bytes son una muestra estéreo L-R (Left-Right) con 2 bytes cada canal.

La obtención de las muestras se realiza en bloques de potencias de dos debido a que la transformada Daubechies solo trabaja de esta forma; nuestro método para solucionar esto fue tomas 8MB exactos de datos para que sea una potencia de 2 si al tomar los 8 MB aun no se agotan los datos seguimos con este procedimiento hasta agotarlos y los datos restantes no son 8MB se toma la potencia de 2 mas grande dentro de los datos restante y se continua con este proceso hasta agotar los datos.

Cabe señalar que la transformada acepta como mínimo cuatro muestras (2 por canal); así que si el archivo después de tomar los bloques de potencias de dos no termina con cuatro muestras exactas se tendrá una pérdida de 3 muestras(12 bytes) a lo más; lo cual es despreciable ya que como mencionamos anteriormente para tener un segundo de audio necesitamos 176400 bytes por segundo.

7.2 Transformación Lineal

Una vez que tenemos nuestros datos en potencias de dos antes de agotar nuestro primer bloque de datos en potencia de dos aplicamos la transformación lineal que en este caso es la Wavelet Daubechies 4 D4 que fue descrita anteriormente en el capitulo 4.7 y tomamos de nuevo un bloque de datos en potencia de dos y aplicamos de nuevo la transformada se sigue este procedimiento hasta agotar nuestros datos.

Los datos transformados se van guardando en bloques de memoria para después ser cuantizados y posteriormente transformados.

7.3 Cuantización por Nivel

De cada bloque de datos transformados tanto izquierdos como derechos se calcula el valor máximo de cada bloque de transformación es decir de niveles que se obtiene de la transformada ya que puede ser diferente por el tamaño dinámico de las potencias de dos calculadas, este valor máximo nos sirve para poder cuantizar los datos transformados de cada bloque, y obtenemos un factor *q* el cual es determinada al inicio por el usuario. La cuantización que realizamos es la cuantización lineal o uniforme descrita en el capitulo 5.1.

7.4 Compresión de Datos

Ya teniendo datos cuantizados después del proceso de transformación y ya escritos en disco en un archivo temporal es posible comprimirlos con un codificador entropico en este caso el algoritmo de compresión de Huffman descrito en el capitulo 6.2, después de esto ya podemos borrar el archivo temporal y poder darle el formato al archivo MPX.

7.5 Formato de Archivo

Ya teniendo datos comprimidos el paso final es darle el formato al archivo es decir nuestra cabecera y esto no es mas que escribir datos de suma importancia para nuestro formato ya que sin estos datos seria imposible reconstruir el archivo, partimos de escribir el factor *q* que es el factor o paso de cuantización que fue puesto por el usuario escribir los valores de cuantización por nivel de cada bloque de datos y escribir la cabecera de nuestro archivo WAV original ya que vamos a reconstruir ese mismo archivo sin perdida de datos.

7.6 Descompresión de datos

En este bloque de la reconstrucción de archivo es mas sencillo ya que en la mayoría de los pasos es solo aplicar los pasos inversos, ya que tenemos el archivo MPX es necesario descomprimirlo para poder escucharlo y este es el primer paso aplicar el algoritmo de Huffman para descomprimirlo y obtener los datos cuantizados ya teniendo los datos cuantizados es necesarios llevarlos obtener los coeficientes de la transformada y poder aplicar la transformada inversa este se lleva acabo descuantizandolos por medio de un algoritmo que hace el paso inverso a la cuantización lineal descrita anteriormente.

7.8 Transformación Inversa

Ahora que tenemos los coeficientes de transformación es necesario volver a tomar bloques que sean potencias de 2 como se describió el procedimiento anteriormente hasta agotar los datos transformados conforme se van tomando los bloques de datos se va aplicando la transformada lineal inversa que es la Wavelet Daubechies 4 D4 y obtener los datos del Archivo WAV.

7.9 Reconstrucción del Archivo WAV

Ya que tenemos los datos de la transformada inversa estos son los datos del archivo WAV original lo que nos queda por hacer es reconstruirlo es decir reacomodar las muestras ye que se separaron en los 2 canales (izquierdo y derecho) y escribir la cabecera del archivo WAV original y ya tenemos nuestro archivo WAV reconstruido del original sin perdida de calidad.

8. Resultados

Se cumplieron los objetivos los cuales eran el desarrollo de un nuevo formato de compresión de audio basado en el estándar MP3 y el software que hace esto es decir el software de transforma un archivo WAV en un archivo MPX y viceversa un archivo MPX a un archivo WAV para poder oírlo así como el software de extrae un track de CD a un archivo WAV “RIPPER”.

El desempeño del formato de compresión desarrollado en el presente trabajo es satisfactorio.

Los parámetros de comparación son razón de compresión, el error cuadrático medio correspondiente y así como el análisis de espectros de frecuencia en un tiempo *t,* entre el formato MPX y el formato MP3 que es el mas popular y en el formato en que nos basamos.

A mayor razón de compresión mayor es el error cuadrático medio del formato MPX a comparación del formato MP3 su error cuadrático medio es casi constante a pesar de esto el error cuadrático medio de MPX sigue siendo menor que el de MP3 lo cual quiere decir que al momento de reconstruir el archivo (MP3 y MPX) a formato WAV (archivo original) siempre el formato MPX es mas similar al archivo original WAV que el formato MP3.

A continuación se presentan tablas y gráficas comparativas entre MP3 y el formato de compresión MPX, por medio del error cuadrático medio y el análisis de espectros de frecuencia realizado a canciones reconstruidas a partir de un MP3 a WAV y de un MPX a WAV.

Las canciones que se tomaran para este análisis fueron aleatorias.

Tabla 8.1 Comparación entre MP3 y MPX para Corrs.wav

Razón Compresión	Error MP3	Error MPX
4:1	10236	949
5:1	10228	1023
6:1	10222	1381
7:1	10216	1586
8:1	10210	1722
11:1	10206	2169
12:1	10202	2291
14:1	10249	2528
17:1	8105	2813
22:1	8103	3212
25:1	8101	3438
176:1	8101	6002

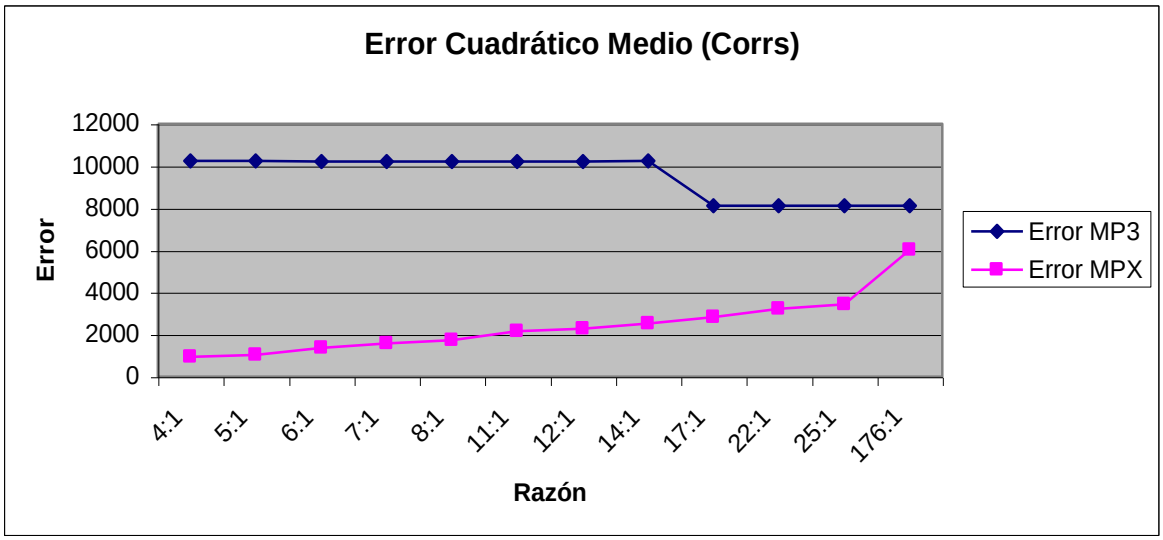


Figura 8.1 Gráfica de Comparación entre MP3 y MPX para Corrs.wav

Tabla 8.2 Comparación entre MP3 y MPX para U2.wav

Razón Compresión	Error MP3	Error MPX
4:1	5168	401
5:1	5165	592
6:1	5161	667
7:1	5157	776
8:1	5150	920
11:1	5153	1193
12:1	5151	1291
14:1	5270	1386
17:1	4955	1574
22:1	4954	1836
25:1	4950	1972
176:1	5101	3374

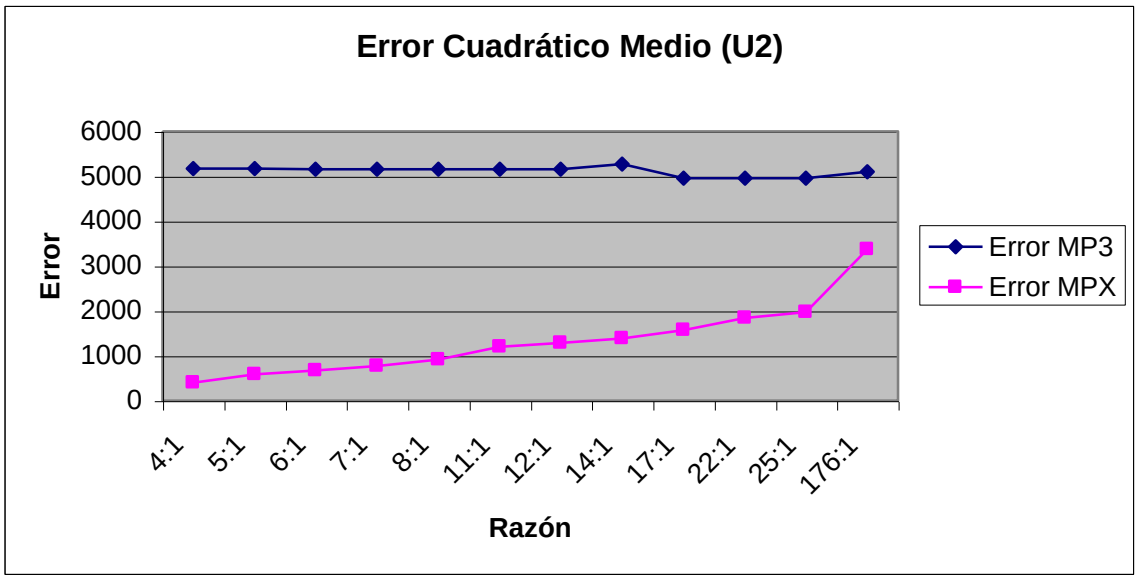


Figura 8.2 Gráfica de Comparación entre MP3 y MPX para U2.wav

Tabla 8.3 Comparación entre MP3 y MPX para Maná.wav

Razón Compresión	Error MP3	Error MPX
4:1	15056	1824
5:1	15044	2149
6:1	15033	2388
7:1	15022	2761
8:1	15012	2986
11:1	15007	3698
12:1	14995	4006
14:1	15038	4306
17:1	12833	4606
22:1	12832	4906

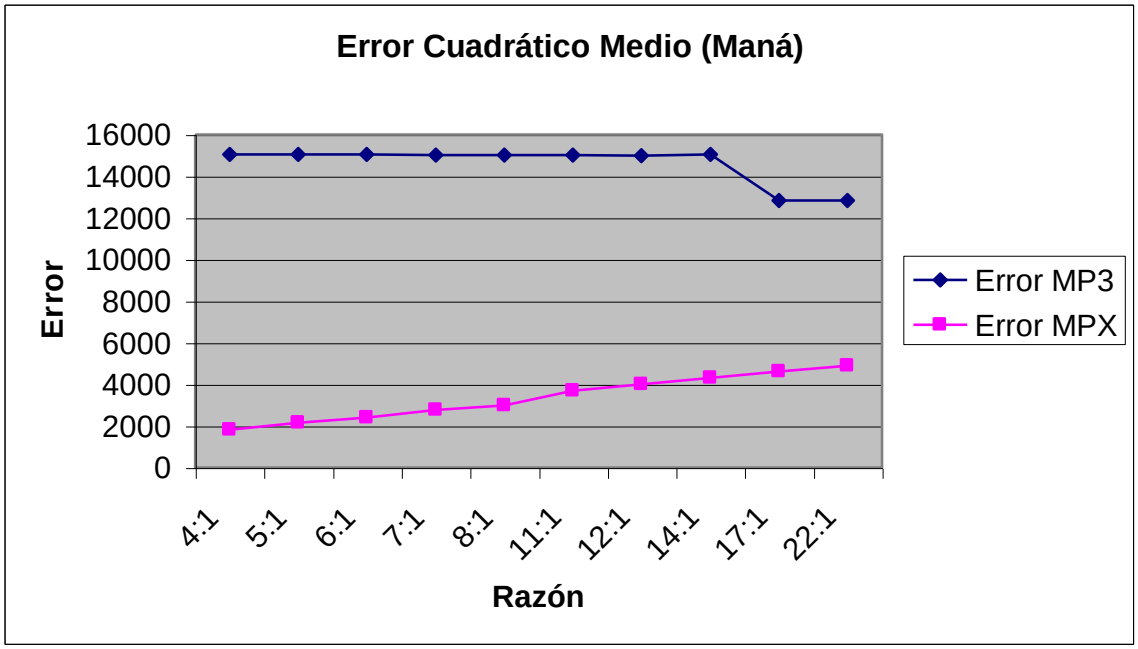


Figura 8.3 Gráfica de Comparación entre MP3 y MPX para Maná.wav

Tabla 8.4 Comparación entre MP3 y MPX para Ricky.wav

Razón Compresión	Error MP3	Error MPX
4:1	11751	859
5:1	11741	1176
6:1	11732	1353
7:1	11722	1517
8:1	11714	1600
11:1	11709	2200
12:1	11701	2400
14:1	11723	2800
17:1	9572	3200
22:1	9572	3850

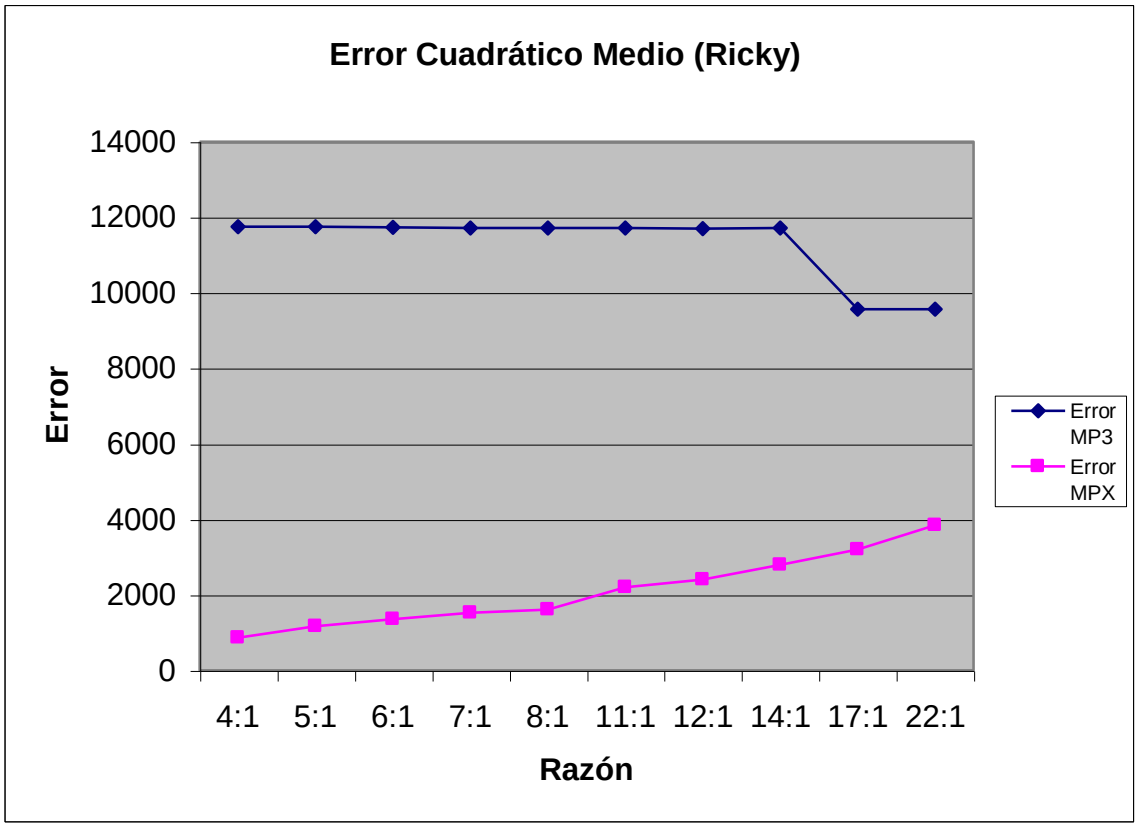


Figura 8.4 Gráfica de Comparación entre MP3 y MPX para Ricky.wav

Figura 8.5 Gráfica del espectro de frecuencias de un archivo WAV(1) original directamente “rippeado”

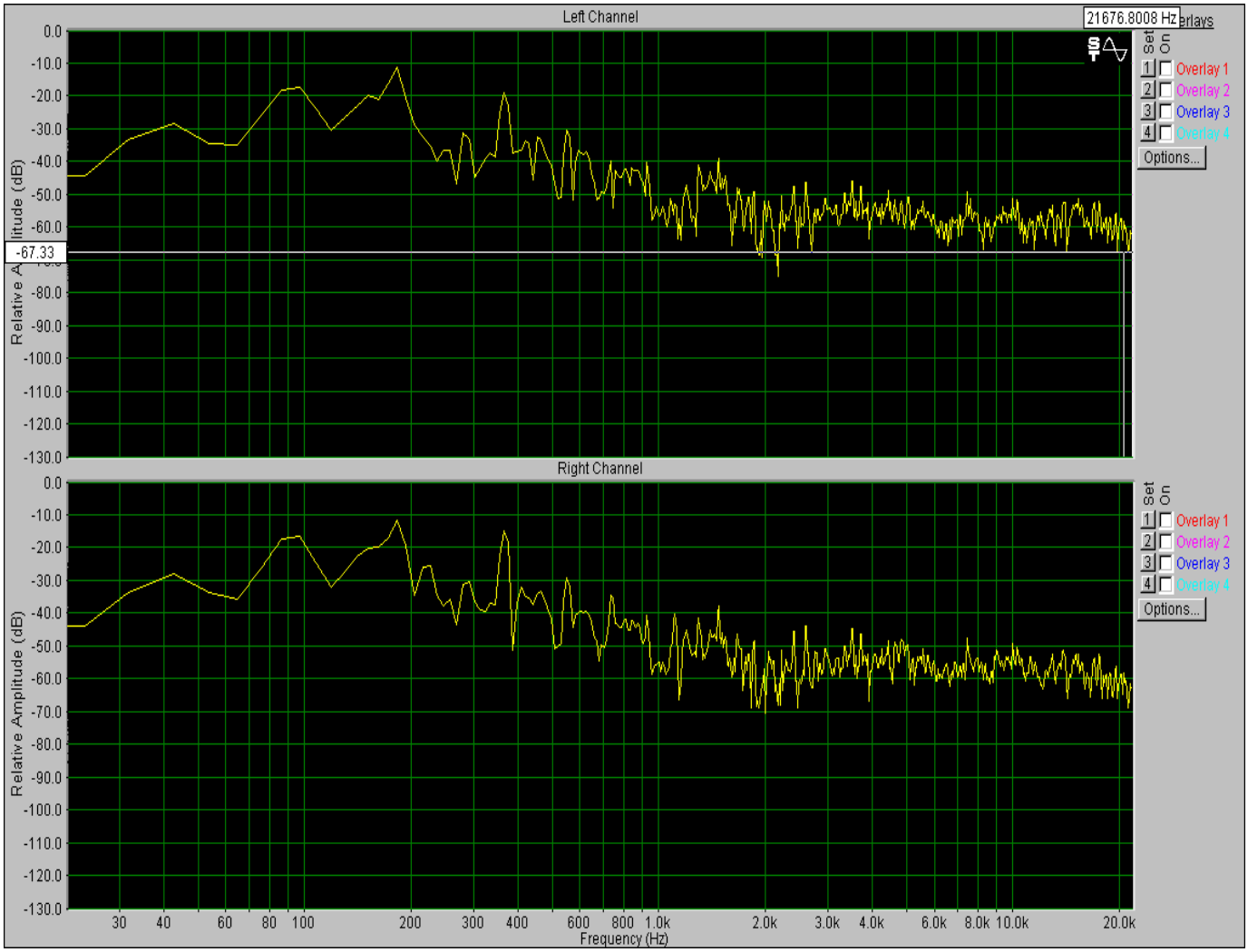


Figura 8.6 Gráfica del espectro de frecuencias de un archivo WAV(1) reconstruido de un MPX

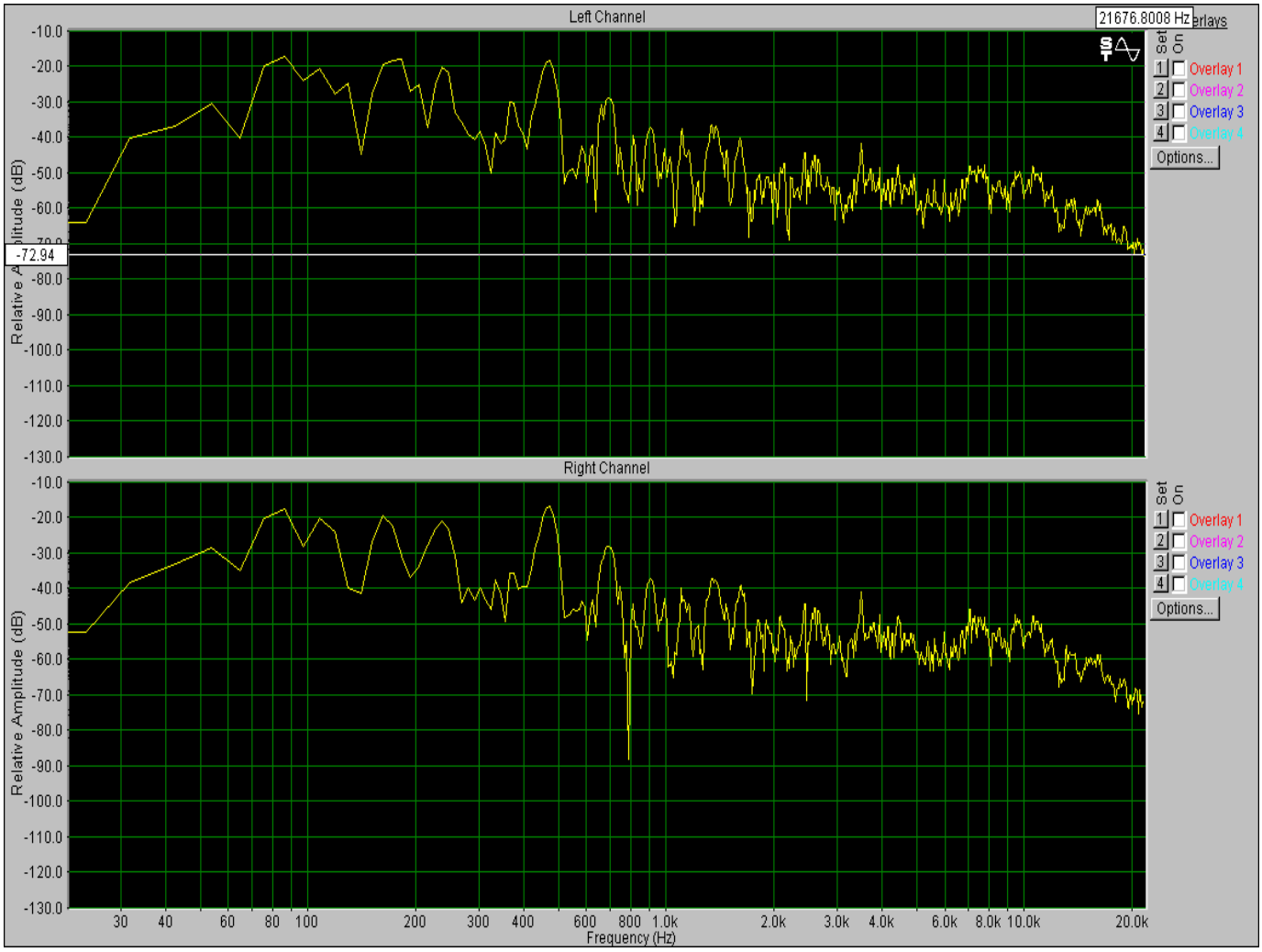


Figura 8.7 Gráfica del espectro de frecuencias de un archivo WAV(1) reconstruido de un MP3 a 128 Kbps

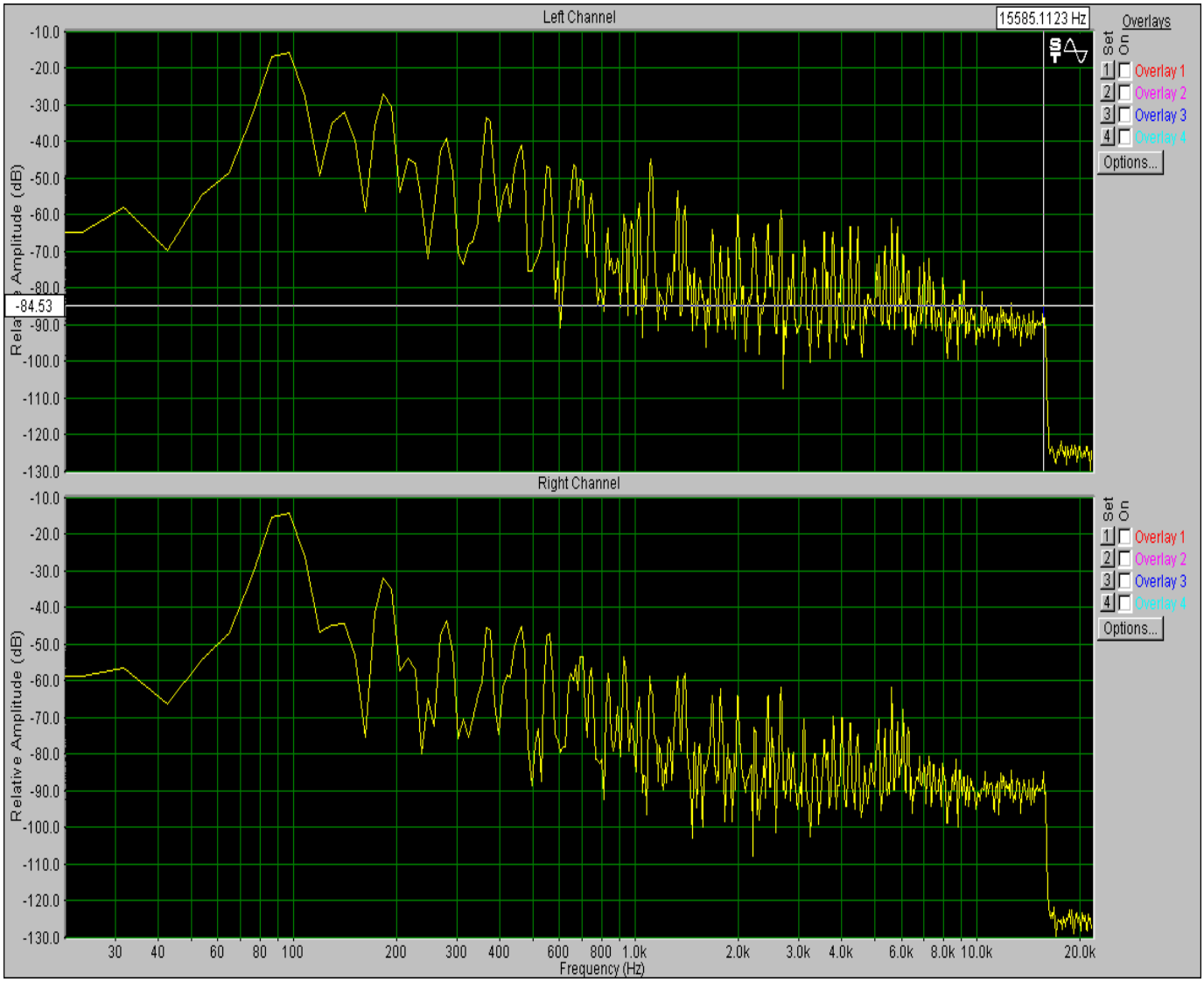


Figura 8.8 Gráfica del espectro de frecuencias de un archivo WAV(1) reconstruido diez veces de un MP3 a 192 Kbps

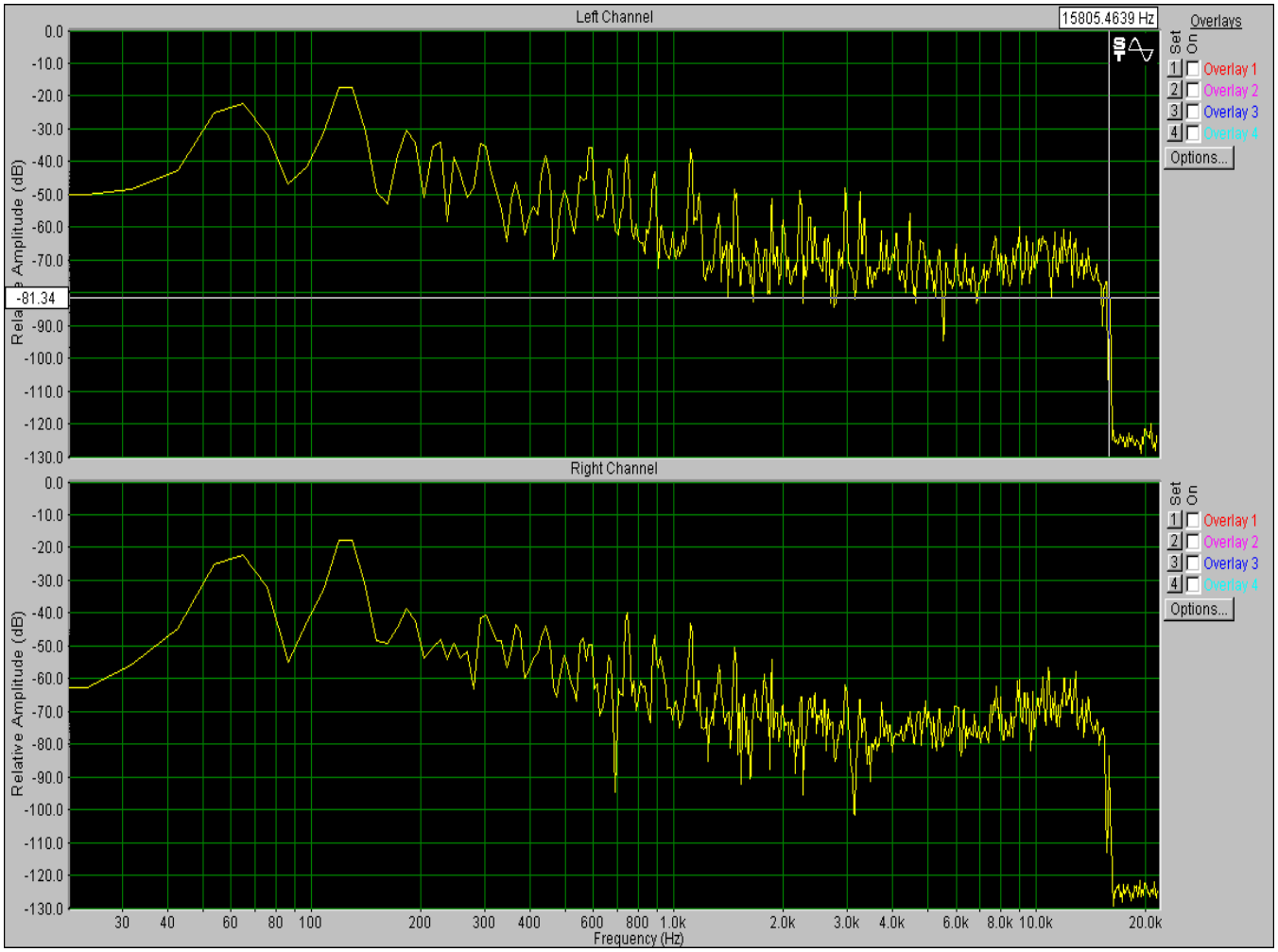


Figura 8.9 Gráfica del espectro de frecuencias de un archivo WAV(1) reconstruido diez veces de un MP3 a 128 Kbps

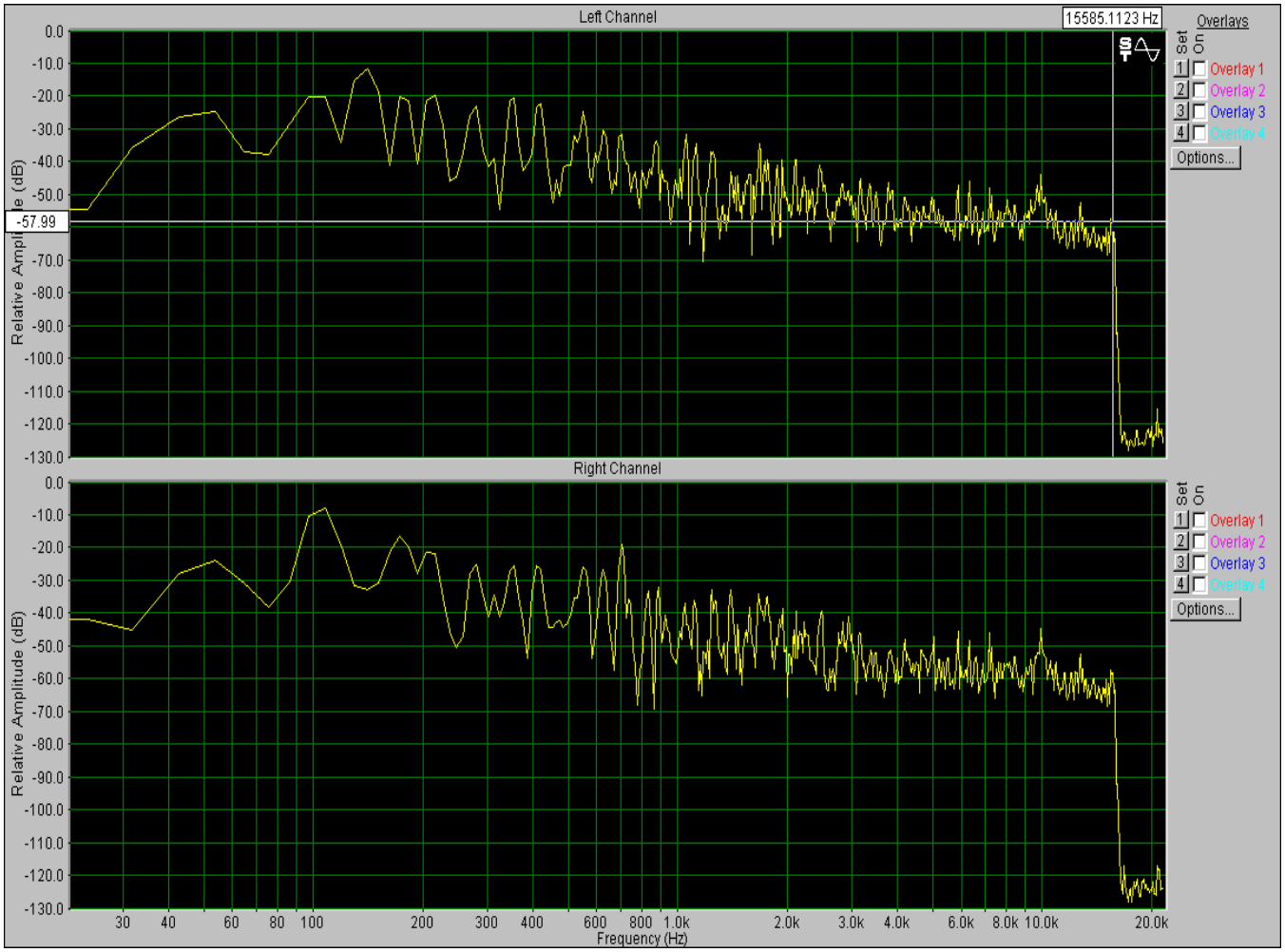


Figura 8.10 Gráfica del espectro de frecuencias de un archivo WAV(2) original directamente “rippeado”

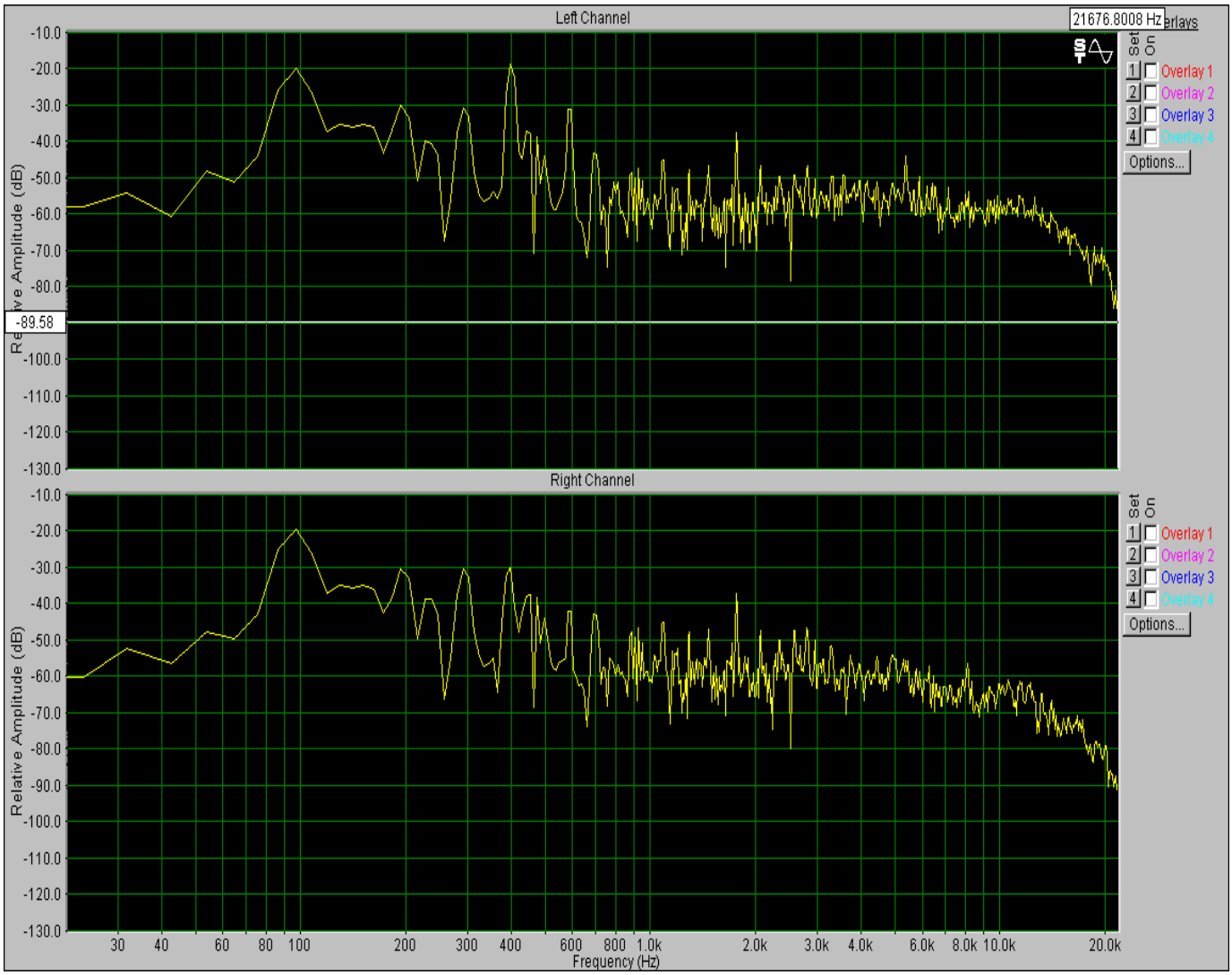


Figura 8.11 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido de un MPX

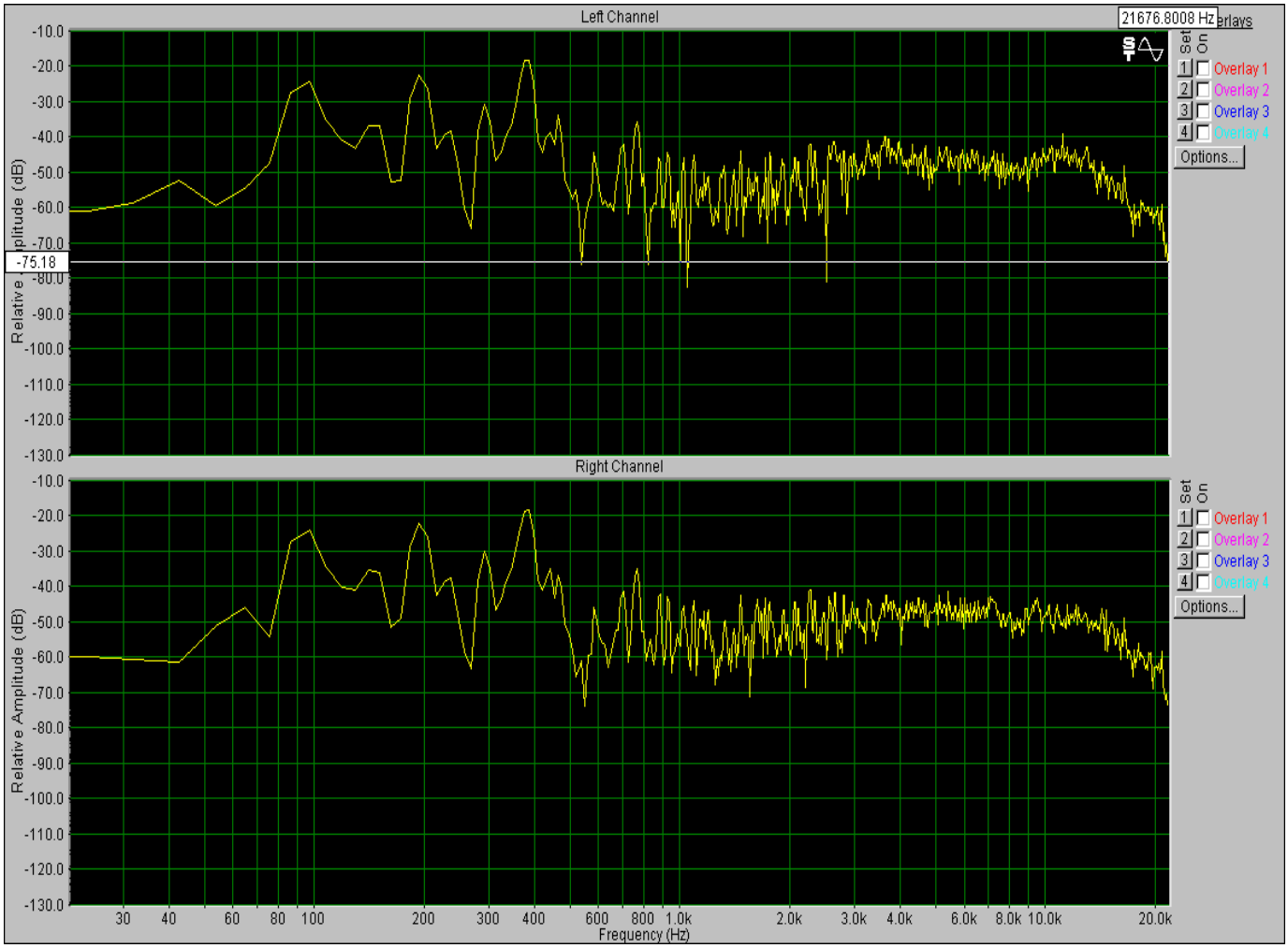


Figura 8.12 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido de un MP3 a 128 Kbps

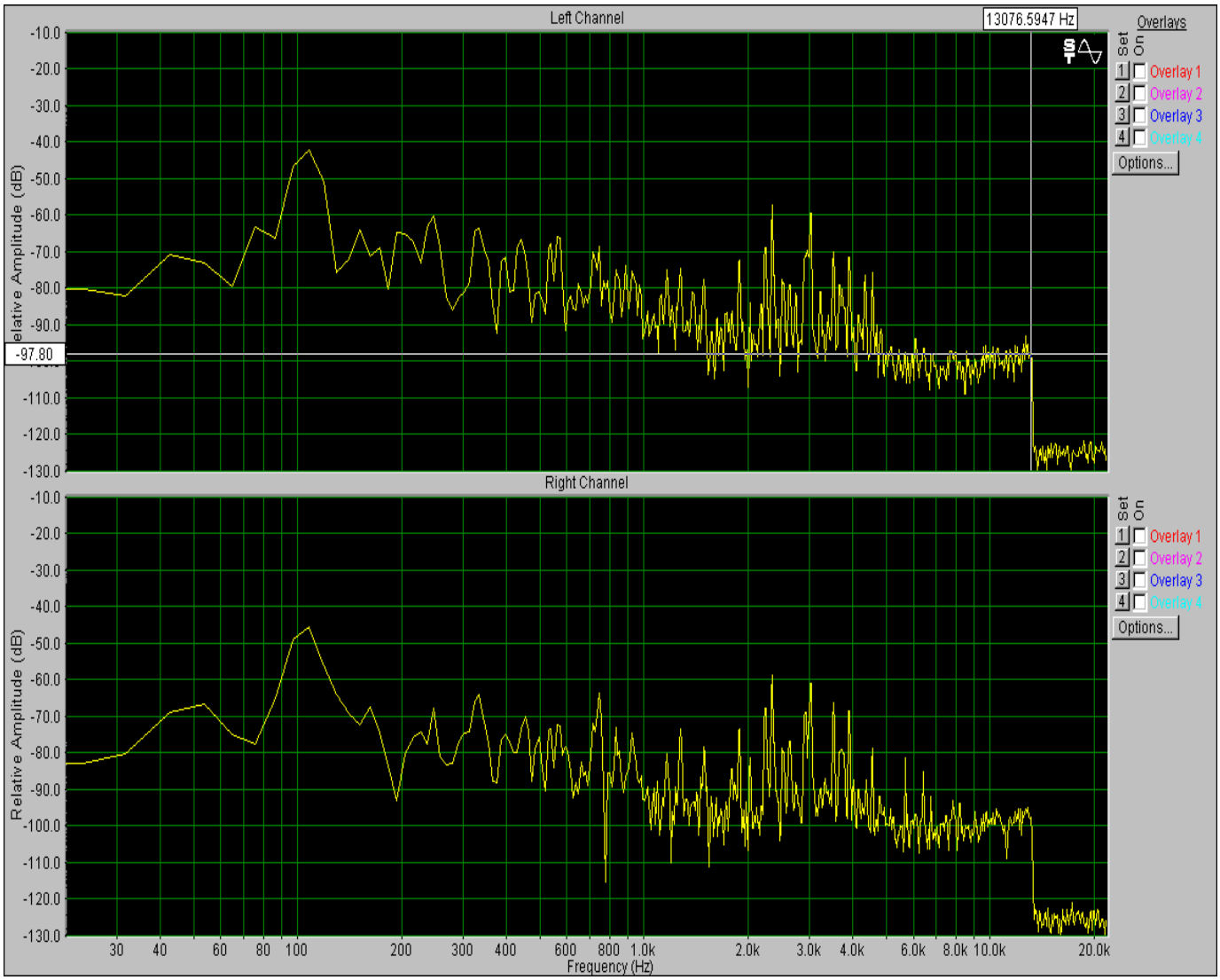


Figura 8.13 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido diez veces de un MP3 a 192 Kbps

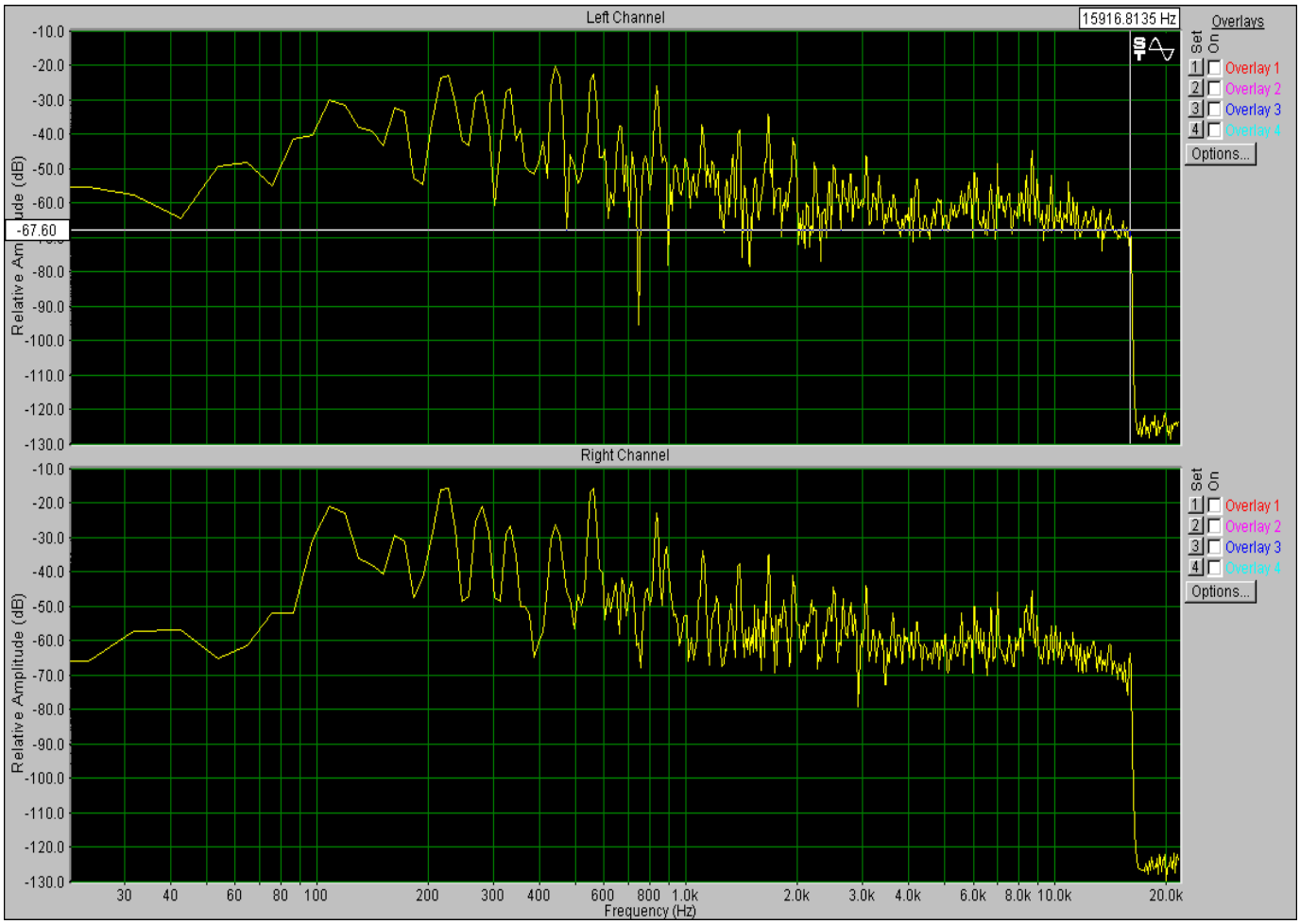
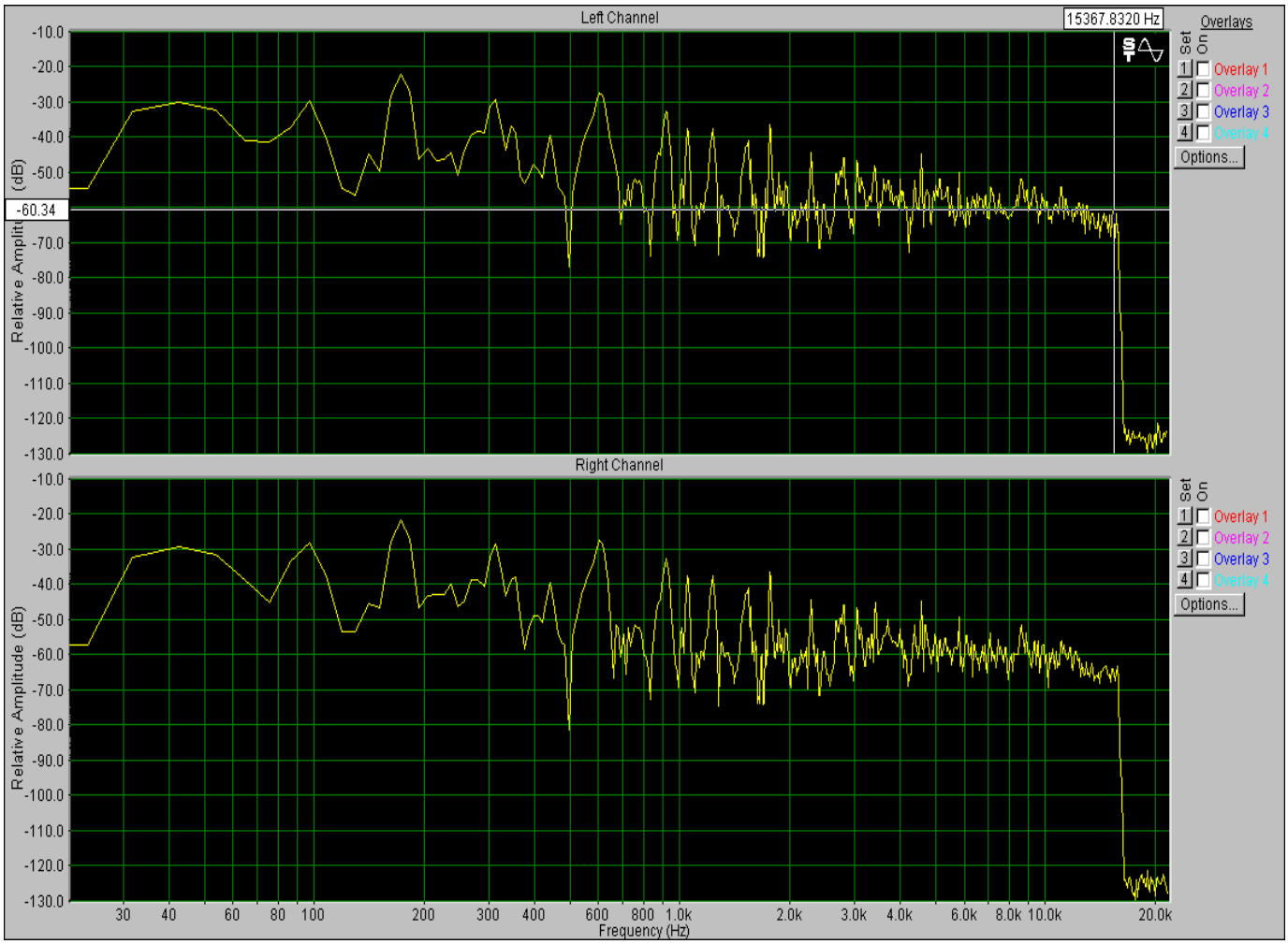


Figura 8.14 Gráfica del espectro de frecuencias de un archivo WAV(2) reconstruido diez veces de un MP3 a 128 Kbps



kbps	Track 1 MP3	Track 1 MPX
1=320	92728607.2	168645.506
2=256	83001157.5	170794.506
3=192	83395943.8	173182.506
4=160	82673964.1	175943.506
5=128	82933429.7	178929.506

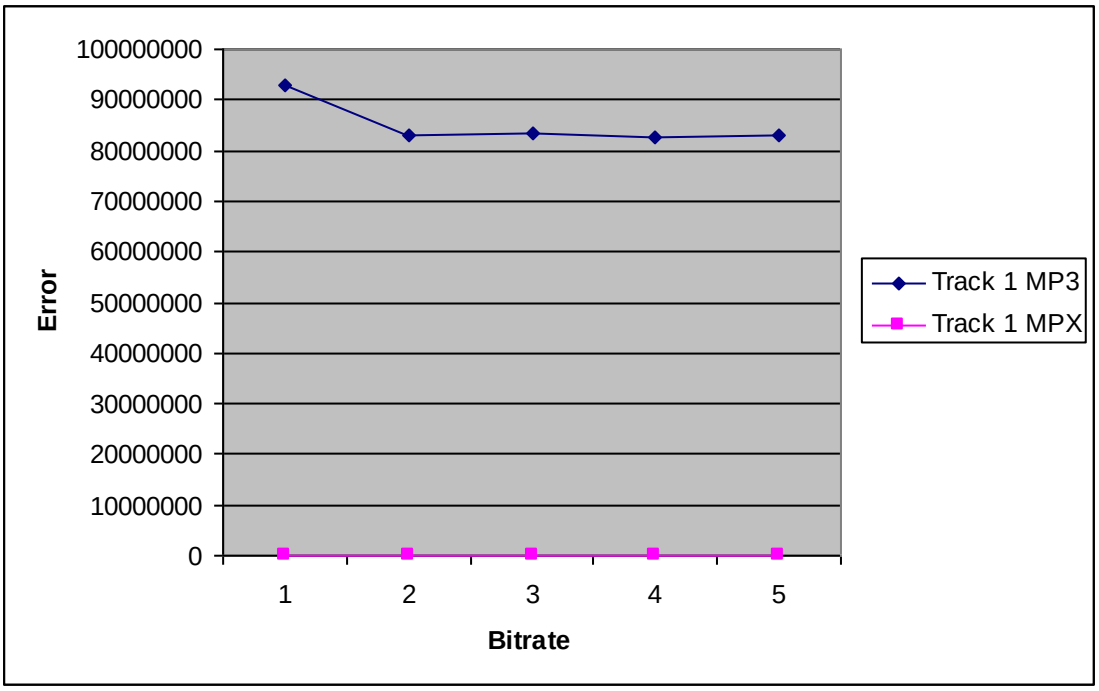


Tabla 8.5 comparación de reconstrucción 10 veces de MP3 y MPX a WAV Track1

kbps	Track 2 MP3	Track 2 MPX
1=320	157112351	171614.283
2=256	144278844	171614.283
3=192	154416954	174002.283
4=160	151007846	176988.283
5=128	149887528	180686.283

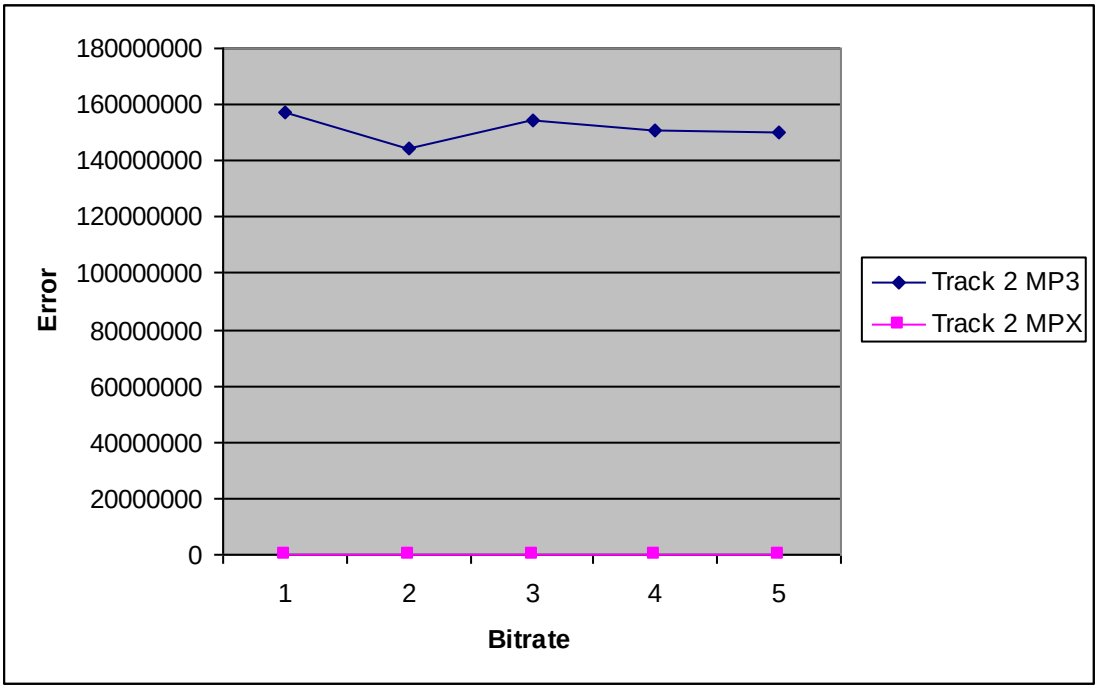


Tabla 8.6 comparación de reconstrucción 10 veces de MP3 y MPX a WAV Track2

kbps	Track 3 MP3	Track 3 MPX
1=320	189009909	183547.559
2=256	182057213	185935.559
3=192	182325848	188696.559
4=160	179493982	191682.559
5=128	179533101	195380.559

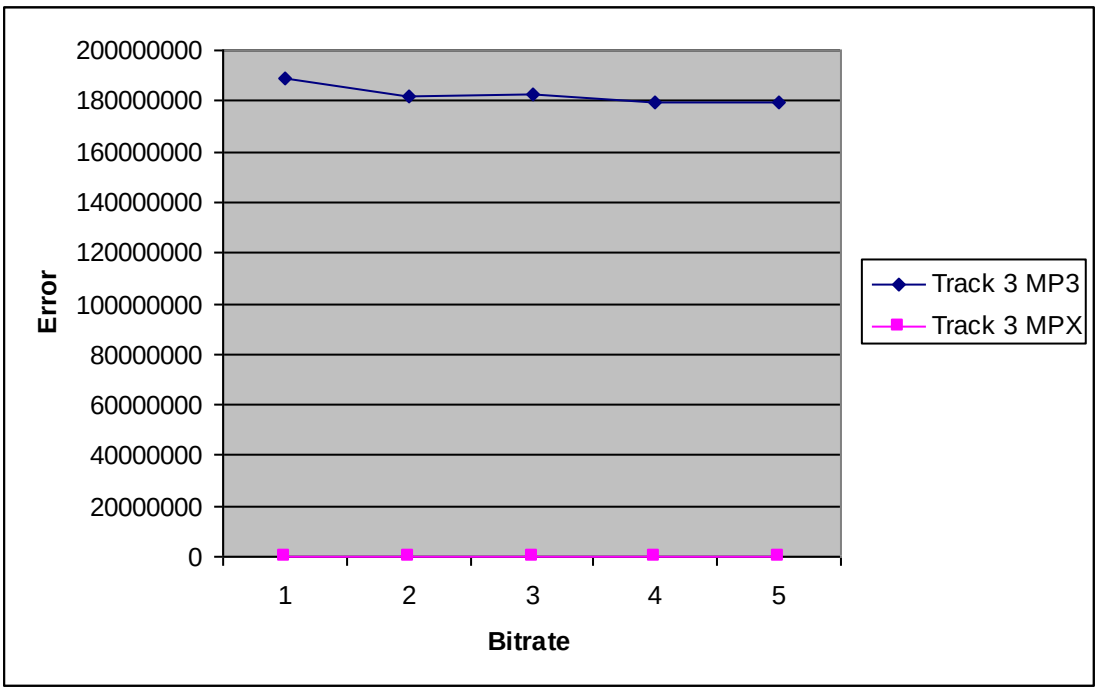


Tabla 8.7 comparación de reconstrucción 10 veces de MP3 y MPX a WAV Track3

kbps	Track 4 MP3	Track 4 MPX
1=320	96601229.7	173456.559
2=256	94871003.5	175844.559
3=192	93076033.7	178605.559
4=160	91934019.6	181591.559
5=128	91306024.5	185289.559

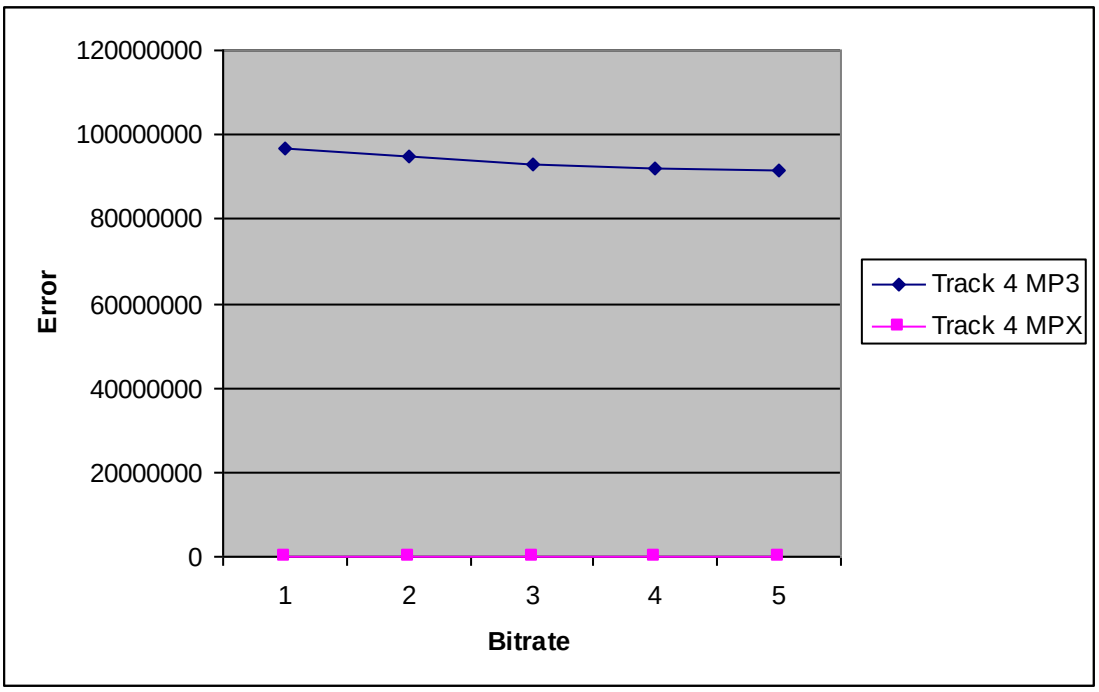


Tabla 8.8 comparación de reconstrucción 10 veces de MP3 y MPX a WAV Track4

kbps	Track 5 MP3	Track 5 MPX
1=320	130883917	176987.569
2=256	129401763	179375.569
3=192	126579089	182136.569
4=160	124718625	185122.569
5=128	124816269	188820.569

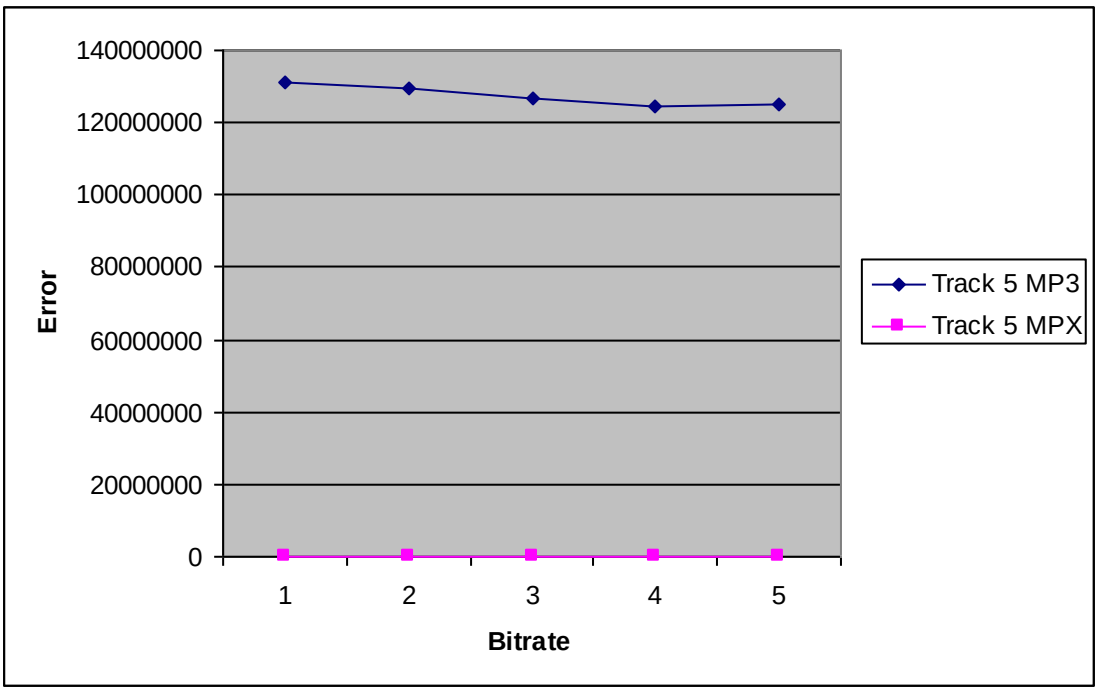


Tabla 8.9 comparación de reconstrucción 10 veces de MP3 y MPX a WAV Track5

9. Conclusiones

El desempeño del formato wavelet propuesto es suficientemente competitivo con el estándar MPEG con lo cual se muestra que el análisis de wavelet puede ser tan eficiente como lo ha sido el análisis de Fourier en la compresión de audio.

Se logró comprimir audio a través de wavelets reduciendo el error cuadrático medio en comparación con el error cuadrático medio de MP3 a distintas razones de compresión.

Con las anterior graficas de comparación entre MP3 y MPX se puede observar que el error cuadrático medio de MPX es menor en comparación con el de MP3 es decir MPX se acerca a la calidad de el CD y MP3 no, es decir tiene perdida de calidad y datos.

El formato MPX no realiza ningún análisis de frecuencias y con esto no perdemos ninguna frecuencia ni tenemos perdida de datos ya que nuestros algoritmos de procesamiento de datos como la transformada Wavelet Daubechies 4 (D4) tiene una inversa y sin perdida y el algoritmo de compresión Huffman no tiene perdida, es decir somos calidad de CD.

La calidad de un CD se mide por varios factores antes descritos entre ellos el bitrate y el de el CD es de 14Mbps y MP3 su máximo bitrate que alcanza es de 320kbps así pues MPX alcanza el mismo bitrate de un CD de 14Mbps que en comparación con el máximo bitrate de MP3 es bastante significativo.

De esto podemos concluir que MPX puede ser un sustituto del formato CDA y que pude ser una herramienta poderosa para las compañías disqueras ya que en un CD normal pueden caber hasta 20 tracks como máximo (dependiendo del tamaño un CD 80 minutos = 700 MB) y MPX alcanza su máxima eficiencia con una compresión de 3:1 con calidad CD, por esto en un CD pueden caber hasta 60 tracks con la misma calidad anterior 14Mbps, en una comparación con MP3 a una compañía disquera no le interesaría migrar sus tracks a MP3 ya que este formato nunca alcanza la calidad de CD por el análisis de frecuencias que realiza y por su baja calidad como máximo 320kbps también debemos tomar en cuenta que el factor de compresión de MP3 a 320kbps es de tan solo 4:1 y en comparación con MPX es de nuevo bastante significativo.

MPX es un formato de calidad total que servirá en un futuro para compañías disqueras así como procesamiento de audio digital partiendo de la calidad de un CD así como para los amantes de la música que prefieren la calidad del CD y no de otros formatos con perdida de calidad.

Con visto en los resultado anteriores podemos ver que reconstruir un archivo MP3 a un WAV 10 veces esto es de un WAV original a MP3 de ese MP3 a un WAV reconstruido y de ese WAV a un nuevo MP3 y asi sucesivamente 10 veces el error cuadrático medio de MP3 crece considerablemente lo cual en MPX no ocurre y MP3 tiene perdida de calidad conforme aumentan las reconstrucciones.

Con lo anterior se puede afirmar que se cumplieron los objetivos planteados en esta tesis.

10. LIMITACIONES

Debido a que la Transformada wavelet Daubechies D4 solo trabaja con potencias de dos, el archivo debe ser dividido en bloques de potencias de dos a menos que el tamaño del archivo sea una potencia de dos exacta.

Además si el último bloque del archivo no son cuatro muestras se tendrá una pérdida de a lo más 3 muestras.

Debido a que la transformada puede acabar con la memoria virtual de la computadora se toman bloques de potencia de dos máximos de 2^{23} (8MB), esto de acuerdo a estadísticas realizadas.

Referencias bibliográficas

- **Digital Signal Processing**
  Alan V. Oppenheim, Ronald W. Schafer
  Ed. Prentice Hall 1974

- **Tratamiento Digital De Señales**
  John G. Proakis, Dimitris G. Manolakis
  Ed. Prentice Hall

- **Señales Y Sistemas**
  Alan V. Oppenheim, Alan S. Willsky, S. Hamid Nawab
  Ed. Prentice Hall

- **El Gran Libro Del CD-ROM**
  Harald Hahn
  Ed. Computec – Marcombo

- **Analog and Digital Signal Processing**
  Ashak Ambardar
  PWS Publishing Company, (USA, 1995).

- **Introduction to Wavelets**
  Charles K. Chui
  Academic Press,San Diego CA., 1992.

- **Ten Lectures on Wavelets**
  Daubechies, Ingrid
  "SIAM Press" (Philadelphia PA, 1992).

- **A method for the construction of minimum-redundancy codes**
  Huffman, D. A.
  Proc. Inst. Radio Eng. 40, 9 (.), 1098–1101, Sept 1952.

Paginas WEB

- **http://www.pchardware.org/cdrom**
- **http://www.chipchapin.com/CDMedia**
- **http://www.howstuffworks.com**
- **http://www.epanorama.nel/documents/pc/cdaudio.html**
- **http://www.tardis.edu.ac.uk/~psyche/cdda**
- **http://www.mp3-tech.org**